**פרויקט גמר 5 יחידות לימוד**

**התמחות – למידת מכונה**

**Deep Learning**

**נושא הפרויקט:**

**זיהוי אותיות קטנות באנגלית בכתב יד**

**Handwriting English letter identifier**

|  |  |  |
| --- | --- | --- |
| **שם התלמידה:** | מעיין אלייה |  |
| **שם המנחה:**  **תאריך הגשה:** | Alphabet GAN: AI Generates English Letters!דינה קראוס  18.6.2021 |
|  |  |

**תוכן עניינים**

**מבוא**............................................................................................................................................4

**מבנה / ארכיטקטורה של הפרויקט**...............................................................................................10

**מדריך למשתמש**.........................................................................................................................11

הוראות התקנה.....................................................................................................................11

הרצת התוכנית.....................................................................................................................13

האופציה הראשונה – העלאת הdata........................................................................................16

האופציה השנייה – Train אימון המודל...................................................................................21

האופציה השלישית – Test בחינת המודל.................................................................................25

האופציה הרביעית – חיזוי תמונה............................................................................................26

**מדריך למפתח**.............................................................................................................................30

קובץ ראשי – menu\_maayan.py............................................................................................31

קובץ – tk\_inter.py:..............................................................................................................31

Case Data....................................................................................................................38

extract\_zipfile...................................................................................................38

load\_data.py.....................................................................................................40

check\_input.py.................................................................................................43

Case Train...................................................................................................................45

the\_model.py....................................................................................................46

train\_model.py..................................................................................................50

Case last model............................................................................................................58

Case Test.....................................................................................................................58

test\_model.py...................................................................................................59

Case Predict Specific Image........................................................................................61

Case Predict Random Images......................................................................................61

…...classify.py..................................................................................................62

check\_directory.py.............................................................................................................69

prints\_types.py...................................................................................................................74

**מסקנות הרצת המודל**...................................................................................................................76

**רפלקציה / סיכום אישי**................................................................................................................79

**ביבליוגרפיה**................................................................................................................................80

**נספחים**......................................................................................................................................81

**מבוא**

פרויקט הגמר שלי ושל חבריי להתמחות בבית הספר עוסק בנושא "למידה עמוקה" (Deep Learning Computer Vision). תחום זה עוסק ביכולת לדמות באופן מלאכותי את הליך הלמידה האנושית. המכונה לומדת לנתח תהליכי לימוד כך שיהיו דומים ככל האפשר ליכולות החשיבה והניתוח של מוח אנושי. מוח האדם בנוי מרשת נוירונים בהם נקלט המידע. שיתוף הפעולה בין אלפי הנוירונים המורכב מפעולה קטנה של כל אחד מהם, מאפשר למוח לעשות פעולה גדולה ומורכבת. ובעזרת התקשורת שבין כל תא עצב לאילו שבקרבתו נוצרת היכולת ללמוד דברים חדשים גם בהתבסס על מידע קיים ולזכור את מה שלמדנו. על רעיון זה מתבססים הפרויקטים בהתמחות זו בכלל וכן הפרויקט שלי בפרט. רשת נוירונים של "למידה עמוקה" עובדת באופן דומה, שכן היא מורכבת ממשקלים ונוירונים שכל אחד מהם מבצע פעולה מתמטית פשוטה, ובסופו של דבר התוכנה מסוגלת ללמוד באופן דומה למוח האנושי. בתחוםDeep learning המתכנת אינו כותב את הקוד לזיהוי האובייקט הנלמד באופן ידני, אלא הוא כותב תוכנה שמסוגלת ללמוד בעצמה באמצעות התמונות שברשותה, באמצעות מודל.

בפרויקט זה בחרתי לבנות תוכנה **שמטרתה לזהות אותיות קטנות באנגלית הכתובות בכתב יד**. המודל לומד ולאחר תהליך למידה הוא יוכל לנתח תמונות שונות המייצגות אותיות בכתב יד ולזהות מהי האות המוצגת בתמונה. בחרתי בנושא זה מכיוון שראיתי בו כשימושי ביותר, בתור תלמידה אני כותבת הרבה ומסכמת חומר לימודי בכתב יד. כאר אני צריכה להשתמש בחומר שכתבתי במחשב, עליי להעתיק באופן ידני את הכתוב על ידי הקלדה במחשב. מכאן חשבתי על נושא זה ומצאתי שפרויקט שמזהה אותיות בכתב יד יוכל לסייע לי באופן אישי ולרבים אחרים. שכן, האפשרות לצלם את הכתוב בכתב היד ולהעלות את התוכן למחשב במהירות ובנוחות רבה יותר יכולה לסייע. לשם כך דרושה תוכנה אשר תוכל לזהות איזה אותיות ישנן בתמונה. בעתיד פיתוח תוכנה זו שכתבתי והרחבת האפשרות לזיהוי אותיות קטנות באנגלית לאפשרות לזיהוי כלל האותיות באנגלית ובשפות אחרות וכן לזיהוי מילים, תעצים ותחזק את הפוטנציאל הגלום תחת פרויקט זה.

בסקירה של המצב הקיים בשוק מצאתי כי ישנן תוכנות שונות המנסות לתת מענה לבעיה זו, ואכן ישנן תוכנות מבוססות machine learning בעלות היכולת לזהות אותיות מתמונות של אותיות בכתב יד באנגלית. עם זאת, כל אחת מהן כתובה אחרת ובעלת תוצאות שונות, ולעומתן הפרויקט שלי כולל תהליך שלם ומפורט של העלאת dataset, אימון המודל, בחינה של המודל, ואף זיהוי של תמונות ספציפיות.

במסגרת ביצוע פרויקט זה הרחבתי את הידע שלי בתחום הבינה המלאכותית AI בכלל ובתחוםdeep learning בפרט. למדתי מושגים תיאורטיים הקשורים לנושא, כמו רשת נויטרונית, משקלים, מודל, פונקציות פעולה, ועוד. כמו כן, הבנתי לעומק איך העולם הזה עובד ומהי החשיבות שלו.

עם זאת, במהלך ביצוע הפרויקט בהתמחות הנ"ל נתקלתי במספר אתגרים מרכזיים. ראשית כל, האתגר המשמעותי ביותר נבע מכך שהתחום הזה היה חדש לי ולא הכרתי אותו לפני תחילת העבודה. היה עליי ללמוד היטב את כל הרקע התיאורטי לנושא והרעיונות מאחוריי deep learning ובמקביל ללמוד את הדרך המעשית ליצירת פרויקט המתבסס על נושא זה. יתרה מכך, זהו נושא שיחסית חדש בלימודים התיכוניים בכלל ובבית הספר שלי בפרט. התמחות זו מתקיימת בבית הספר בשנה השנייה בלבד, לכן גם עבור המורה, גם עבור בית הספר, וגם עבור התלמידים הוא חדש יחסית. כלומר, התבקשנו לכתוב תוכנה המבצעת דבר שמלכתחילה לא ידענו כיצד לבצעו ועל כן למדנו תוך כדי פעולה. התגברתי על אתגר זה בכך שאני וחבריי להתמחות למדנו את הנושא תוך כדי ביצוע הפרויקט באמצעות מדריכים וספרים שונים, מורתנו דינה, האינטרנט ומקורות נוספים.

נוסף על כך, מחיפוש נרחב במקורות מידע ובאינטרנט הבנתי כי אין הרבה מידע בנושא deep learning בשפת האם שלי – עברית, על כן היה עליי לחפש מידע מורכב באנגלית ולהבין כיצד ליצור פרויקט מתוכן בשפה שאינה שפת האם שלי. אמנם העובדה שלמדתי ממקורות רבים באנגלית עזרה לי לשיפור אוצר המילים שלי וההבנה שלי באנגלית.

מאחר שביצעתי את הפרויקט ולמדתי תוך כדי כיצד לבצעו בדרך הטובה ביותר, התנסיתי במגוון גרסאות שונות שבכל פעם שיפרתי משהו מסוים עד לקבלת הגרסה הסופית. ההבדלים בין הגרסאות היו בעיקר במודל ממנו המכונה לומדת, וכן באופן מימוש הפעולות השונות שהמכלול שלהן מהווה את הפרויקט כולו. כנאמר לעיל, תוכנה זו ייצרה מוח מלאכותי המדמה מוח אנושי, ולמעשה האתגר הגדול היה לכתוב ולהגיע למודל, המדמה את שכבות הנוירונים במוח האנושי, אשר יניב את אחוז הצלחה הגבוה ביותר. המוטיבציה שלי נבעה מכך ששאפתי להגיע לפרויקט הטוב ביותר שאוכל לייצר, ועל כן תמיד היה מקום לשיפור.

דרישות להרצת התוכנה:

* **מערכת הפעלה Windows**
* **הורדת python:**

1) אם מדובר בהרצה ב-command line:

יש להוריד פייתון גרסה 3.7 או גרסה עדכנית יותר. שים לב שכשאתה מוריד את הפייתון למחשב אתה מסמן: Add python to path.

2) אם מדובר בהרצה באמצעות Anaconda prompt:

יש להשתמש בפייתון גרסה 3.6

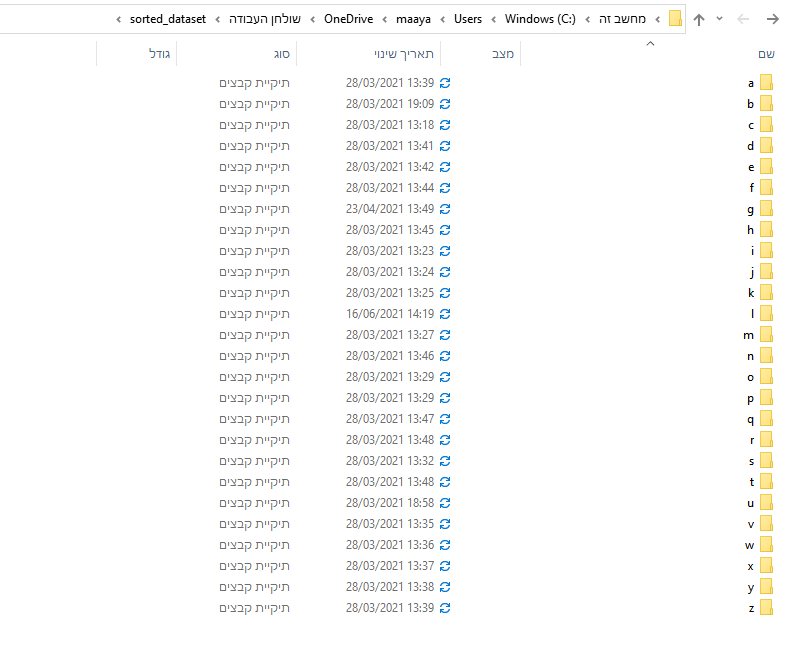
* **מאגרי הנתונים.**

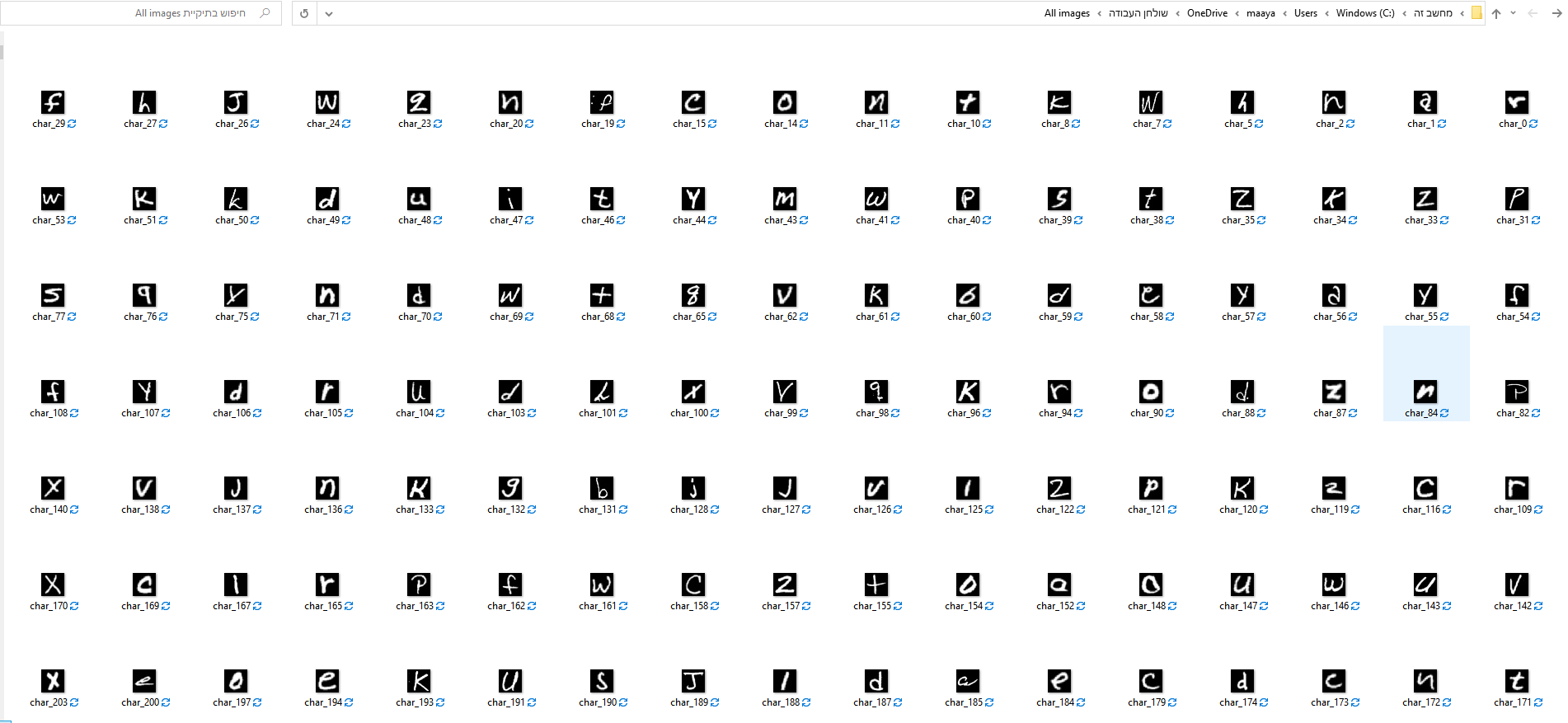
המודל משתמש ב-dataset המורכב מתמונות מתוך המאגר EMNIST, שהינו מאגר התמונות בכתב יד הגדול בעולם.

מאגר התמונות לאימון המודל שלי כולל תמונות מסוג pdf ומחולק לתיקיות, כאשר כל תיקייה הינה אות קטנה באנגלית והתמונות שבתוכה הן תמונות של אותה אות בכתב יד. זאת על מנת שהמחשב ידע איזה אות מוצגת לפניו וכך יבצע את תהליך הלמידה.

בנוסף, יש לי dataset של התמונות מEMNIST- מעורבבות, ללא חלוקה לאותיות. זאת על מנת להשתמש בו לאחר מכן כדי לחזות באקראיות תמונות.

כמו כן, יש לי dataset הכולל תמונות חדשות שהמודל לא למד, ומורכב מתמונות של אותיות באנגלית בכתב היד שלי, כדי לבחון את המודל לאחר האימון.

תמונה להמחשה של מאגר התמונות הממוין:

תמונה להמחשה של מאגר התמונות הלא ממוין:

תמונה להמחשה של מאגר התמונות הכולל תמונות בכתב היד שלי:



* **הספריות הבאות:**

|  |  |  |
| --- | --- | --- |
| **Library name** | **Installation command** | **Download link** |
| Tensorflow (2.1.0) | Pip install tensorflow | <https://pypi.org/project/tensorflow/> |
| Keras (2.3.1) | Pip install keras | <https://pypi.org/project/Keras/> |
| Numpy (1.19.2) | Pip install numpy | <https://pypi.org/project/numpy/> |
| Sklearn (0.24.1) | Pip install scikit-learn | <https://pypi.org/project/scikit-learn/> |
| PIL (8.2.0) | pip install Pillow | <https://pypi.org/project/Pillow/> |
| Matplotlib (3.3.4) | Pip install matplotlib | <https://pypi.org/project/matplotlib/> |
| Imutils (0.5.4) | Pip install imutils | <https://pypi.org/project/imutils/> |
| Opencv (3.3.1) | Pip install opencv-python | <https://pypi.org/project/opencv-python/> |
| Tkinter (8.6.10) | Pip install tk | <https://pypi.org/project/tk/> |
| Colorama (0.4.4) | Pip install colorama | <https://pypi.org/project/colorama/> |
| Pickle | Pip install pickle-mixin | <https://pypi.org/project/pickle-mixin/> |

הפרויקט שלי מחולק סך הכל ל-11 מודולים, ולמכוון מחלקות. החלוקה למודולים ומחלקות תרמה לי לנוחות בעת ביצוע הפרויקט ולארגון וסדר בפרויקט.

המודולים הקיימים בפרויקט (בהמשך אפקט על כל אחד מהם ותפקידו בהרחבה):

menu\_maayan.py

tk\_inter.py

load\_data.py

the\_model.py

train\_model.py

test\_model.py

classify.py

extract\_zipfile

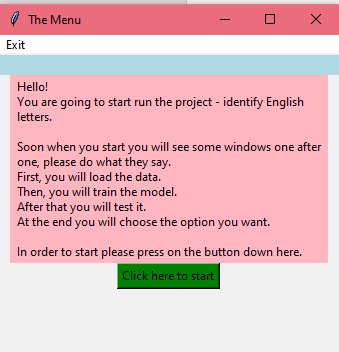
check\_input.py

check\_directory.py

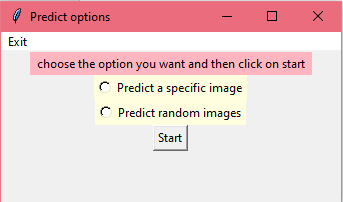
prints\_types.py

במדריך למשתמש אסביר על תהליך הרצת הסקריפט, שניתן להריץ אותו באמצעות ה-commend line או באמצעות סביבת עבודה – Anaconda prompt.

המסך הראשון שמופיע בתחילת הרצת הסקריפט הוא החלון הבא שנוצר על ידי שימוש בtk inter:



מריצים את הפרויקט בהתאם לשלבים המוצגים בחלון זה, האחד אחרי השני, ובסוף ההרצה החלון האחרון שמופיע הוא החלון הבא:



כדי להפעיל את הסקריפט ולאפשר למודל ללמוד, הוא משתמש במאגרי הנתונים שפורטו מעלה. לכן כל משתמש בסקריפט חייב שנתונים אלו יהיו על המחשב שלו.

בסוף תהליך האימון, הסקריפט שומר על המחשב של המשתמש, במיקומים שהמשתמש בחר, שלושה קבצים: המודל המאומן, קובץ התוויות הבינארי, תיקייה ובה תמונות המציגות גרפים מתהליך הלמידה (פירוט בהמשך).

**מושגים חשובים בנושא למידת מכונה – deep learning:**

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

**Accuracy** – אחוז ההצלחה של המודל בחיזוי הקטגוריות של התמונות (נכון או לא נכון).

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

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

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

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

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

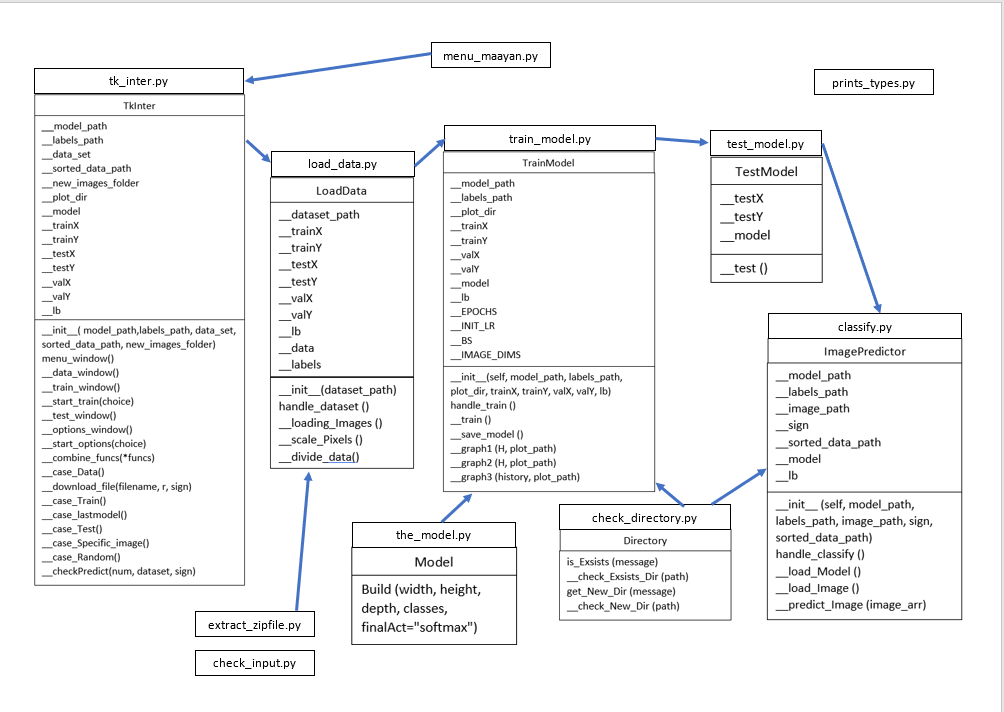
שתי בעיות שעלולות להיווצר הן: overfitting ו-under fitting:

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

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

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

**מבנה / ארכיטקטורה של הפרויקט**

UML class diagram

UML Actors diagram

**מדריך למשתמש**

להלן מדריך שמטרתו להנחות את המשתמש כיצד להשתמש בפרויקט ולהריץ אותו.

**הוראות התקנה:**

כדי להריץ את הפרויקט ישנן שתי דרכים:

1) באמצעות שורת הפקודה cmd:

יש להוריד Python 3.7 או גרסה עדכנית יותר - <https://www.python.org/downloads/>

אם יש לך Python על המחשב, בדוק מהי גרסתו באמצעות הפקודה:

py --version

2) באמצעות **סביבת עבודה**.

יש להתקין במחשב את סביבת העבודה **Anaconda** או כלIDE אחר.

לינק להורדת Anaconda - <https://www.anaconda.com/products/individual>

בתוך Anaconda יש ליצור Environment חדשה הכוללת python 3.6.

נריץ את התוכנית באמצעות ה- Anaconda prompt (python 3.6) של ה- Environmentשיצרנו (הסבר מפורט בהמשך).

1. יש להוריד מספר ספריות קוד אשר בהן הפרויקט משתמש:

|  |  |  |
| --- | --- | --- |
| **Library name** | **Installation command** | **Download link** |
| Tensorflow | Pip install tensorflow | <https://pypi.org/project/tensorflow/> |
| Keras | Pip install keras | <https://pypi.org/project/Keras/> |
| Numpy | Pip install numpy | <https://pypi.org/project/numpy/> |
| Sklearn | Pip install scikit-learn | <https://pypi.org/project/scikit-learn/> |
| PIL | pip install Pillow | <https://pypi.org/project/Pillow/> |
| Matplotlib | Pip install matplotlib | <https://pypi.org/project/matplotlib/> |
| Imutils | Pip install imutils | <https://pypi.org/project/imutils/> |
| Opencv | Pip install opencv-python | <https://pypi.org/project/opencv-python/> |
| Tkinter | Pip install tk | <https://pypi.org/project/tk/> |
| Pickle | Pip install pickle-mixin | <https://pypi.org/project/pickle-mixin/> |
| Colorama | Pip install colorama | <https://pypi.org/project/colorama/> |

1. יש להוריד את קבצי הפרויקט הכוללים קוד ודאטאסט.

יש להוריד מן חשבון ה-GitHub שלי את הקבצים הבאים:

1. קבצי ה- python אשר עליהם מתבסס המודל – יש לשמור את כל הקבצים בתיקייה אחת במחשב.
2. קובץ התמונות החדשות – בכתב היד שלי.
3. קובץ המודל המעודכן.
4. קובץ התוויות המעודכן.

את שני מאגרי התמונות - המאגר הממוין והמאגר הלא ממוין (קבצי הdataset) ניתן להוריד ישירות בהרצת הפרויקט, שכן ישנה אפשרות לבחור נתיב אליו התוכנית תוריד את הdatasets תוך כדי ההרצה. אם המשתמש בוחר להוריד את שני הקבצים מהחשבון שלי ב- GitHub לפני ההרצה, עליו להוריד בנוסף לאמור לעיל גם את שני הקבצים הבאים:

1. התיקייה המכילה את מאגר התמונות הלא מסודר dataset – כל התמונות מעורבבות.

ניתן לשמור את התיקייה כקובץ zip.

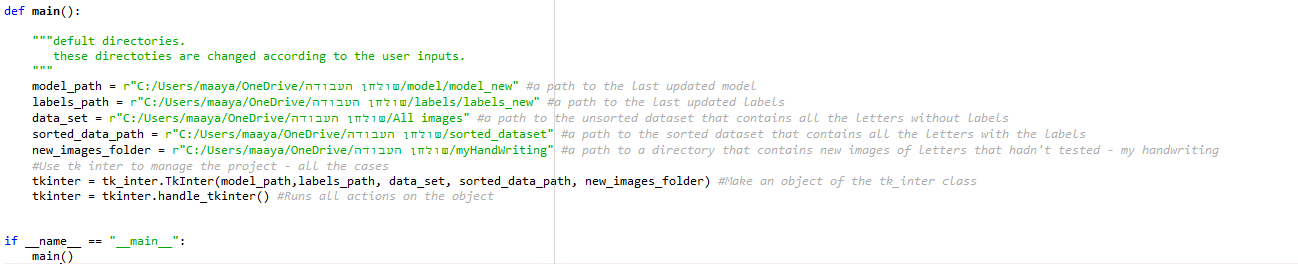
1. התיקייה המכילה את מאגר התמונות באופן הממוין לפי האותיות sorted dataset.

ניתן לשמור את התיקייה כקובץ zip.

**\*אין לשנות את תוכן הקבצים**

1. יש לארגן את הקבצים במחשב באופן הבא:
2. קבצי ה-python ירוכזו כולם בתיקייה אחת.
3. המודל השמור וקובץ התוויות יאוכלסו בתיקיות במחשב לפי רצון המשתמש.
4. התיקיות הבאות יאוכלסו בתיקיות במחשב לפי רצון המשתמש: התיקייה המכילה את מאגר התמונות בכתב היד שלי, ואם המשתמש הוריד גם את מאגר התמונות באופן ממוין ובאופן שאינו ממוין אז גם הן.
5. עדכון הקוד בהתאם ל directories החדשים:

בפונקציית ה main() בקובץmenu\_maayan.py יש לשנות את ערכי המשתנים הכוללים את הנתיבים.

model\_path – הנתיב למודל העדכני השמור.

labels\_path- הנתיב לקובץ העדכני הכולל את התוויות של התמונות.

data\_set – הנתיב למאגר התמונות הלא ממוין – במידה והמשתמש הוריד את הקובץ לפני ההרצה.

sorted\_data\_set – הנתיב למאגר התמונות הממוין לפי האותיות – במידה והמשתמש הוריד את הקובץ לפני ההרצה.

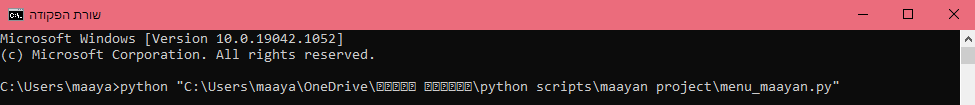
new\_images\_folder – הנתיב לתיקייה הכוללת תמונות חדשות בכתב היד שלי שלא נבדקו במהלך אימון המודל.

**הרצת התוכנית:**

כאמור, ניתן להריץ את המודל גם דרך שורת הפקודה וגם דרך סביבת עבודה.

אם גרסת הפייתון ה-default המותקנת על המחשב שלך הינה python 3.7 או עדכנית יותר וברצונך להריץ באמצעות שורת הפקודה cmd, לאחר שהורדת את כל הספריות הדרושות, יש לכתוב python רווח ואז את הpath למיקום בו שמרתם את קובץ ה Python: menu\_maayan.py בתוך מרכאות – ניתן לגרור את הקובץ ישירות לשורת הפקודה.

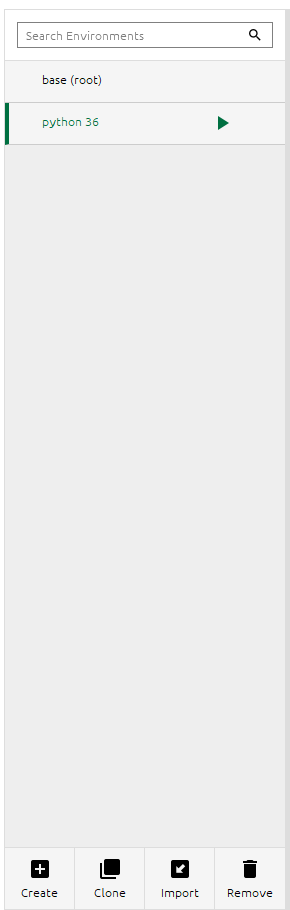
הרצת קובץ זה תריץ את כל התוכנית.



הדרך השנייה הינה על ידי שימוש בסביבת עבודה.

במהלך העבודה על הפרויקט השתמשתי בסביבת העבודה Anaconda באפליקצייה שנקראת spyder (גרסה 5.0.0).

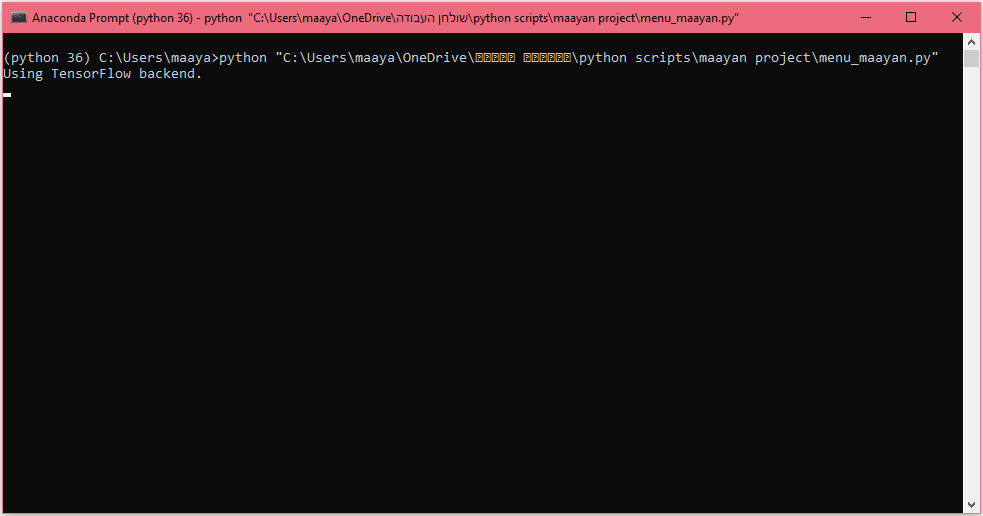
באנקונדה ניתן ליצור new Environment בהתאם לגרסת הפייתון שברצונך להשתמש, ולהריץ את הקוד בסביבה זו.



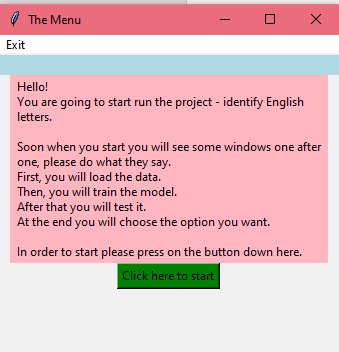
אם ברצונך להריץ באמצעות סביבת העבודה Anaconda, עליך להריץ ב-Command Line של סביבת עבודה זו, הנקראת Anaconda Prompt (python 36). יש לוודא כי הינך משתמש בprompt- של הסביבה שיצרת – הכוללת פייתון 3.6. יש להריץ את קובץ ה Python: menu\_maayan.py.

כדי לעשות זאת, יש לפתוח את שורת הפקודה של אנקונדה ולכתוב בהpython רווח ואז את הpath למיקום בו שמרתם את קובץ ה Python: menu\_maayan.py בתוך מרכאות – ניתן לגרור את הקובץ ישירות לשורת הפקודה.

הרצת קובץ זה תריץ את כל התוכנית.



לאחר הרצת התוכנית יוצג החלון הבא שנוצר על ידי שימוש בTkinter:



תפריט זה מציג למשתמש את האפשרויות העומדות לרשותו ואת הקלט שיש להזין עבור כל אפשרות.

כפי שניתן לראות מוצגות ארבע אפשרויות שיופיעו למשתמש בזה אחר זה:

**השלב הראשון:** **העלאת הדאטא**. בשלב זה התוכנית תעלה את מאגר התמונות הלא ממוין ואת מאגר תמונות הממוין. מאגר התמונות הממוין יחולק לתמונות לטריין (70%), לוולידיישן (10%) ולטסט (20%).

**השלב השני:** **הTrain-** – אימון המודל. בשלב זה המודל ילמד את התמונות שחולקו לטריין וכך יעלה את מידת ה-accuracy שלו ויפחית את מידת ה-loss שלו. המודל מתאמן ותוך כדי מציג את ערכי ה-validation שמדמים את ה-Test בזמן האימון.

בשלב זה המשתמש יכול לבחור אם ברצונו לבצע את תהליך ה-Train כעת או שהוא מעדיף להשתמש במודל האחרון שנשמר, שכבר עבר Train.

**השלב השלישי:** **הTest-** – חיזוי המודל. בשלב זה, לאחר שהמודל מאומן, הוא מבצע חיזוי. כעת הוא משתמש בכלים שצבר ובידע שהפיק ומנסה לזהות תמונות חדשות שלא התנסה בהן קודם לכן. בשלב זה רואים את אחוזי ההצלחה הסופיים של המודל – accuracy, loss.

כאמור, המשתמש יכול לבצע חיזוי ללא אימון, בכך שהתוכנה תשתמש בקובץ בו שמורים הנתונים מהאימון האחרון – המודל המאומן. את המודל הזה המשתמש התקין במחשבו לפי ההוראות שהצגתי קודם לכן ולכן יש ביכולתו האפשרות לבחור לבצע חיזוי ללא אימון.

**השלב הרביעי:** **Predictiom** – חיזוי המודל על תמונות ספציפיות. בשלב זה מופיעות בפני המשתמש שתי אופציות:

1) לבחור תמונה בעצמו ולהכניס כקלט את הנתיב לתמונה. המודל יזהה את התמונה בהתאם ללמידה שלו ויציג בפני המשתמש את החיזוי שלו לאות שהוא מזהה בתמונה.

2) שהתוכנית תבחור מספר תמונות שהמשתמש ירצה באקראיות, והמודל יזהה את כל אחת מהתמונות בהתאם ללמידה שלו ויציג בפני המשתמש את החיזוי שלו לאות שהוא מזהה בתמונה. כאן התמונות יהיו גם תמונות מהדאטאסט המקורי וגם תמונות חדשות בכתב היד שלי שהמודל לא מכיר.

ליציאה מן התוכנית על המשתמש ללחוץ על כפתור ה-Exit.

עבור כל אחד מהכפתורים יתנהל שיח של פלט וקלט עם המשתמש.

אחד היתרונות של התוכנה הוא שהיא יודעת להשתמש בקבצים העדכניים עבור כל הרצה חדשה. שכן אם המשתמש בחר לאמן את המודל ואחר כך לבצע חיזוי של תמונה חדשה, התוכנית תשתמש במודל שנשמר לאחר האימון שבוצע על מנת לחזות את התמונה. הרצה חדשה של התוכנית תחזור להשתמש בקובץ המודל האחרון שנשמר על המחשב במידה והמשתמש בחר שלא לאמן שוב את המודל.

כעת אפרט בהרחבה על ארבע האפשרויות השונות המוצגות בתפריט החלון הראשי.

בכל אפשרות אציג את המסך המוצג בפני המשתמש, קלטי המשתמש הנדרשים, את ההדפסות שנפלטות כפלט במהלך התהליך המתבצע, ואת הודעות השגיאה השונות אשר יציינו שגיאה וידרשו מהמשתמש קלט תקין בהתאם עד לקבלתו.

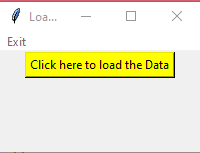
הדפסות בצבע ירוק – מנחות את המשתמש בבקשת קלט.

הדפסות בצבע כחול – מספקות אינפורמציה על התהליך המתרחש בכל רגע.

הדפסות בצבע אדום – מסמלות שגיאה של קלטיים אי תקינים.

**האופציה הראשונה – העלאת הdata:**

לאחר שהמשתמש ילחץ על start, יופיע החלון הבא:

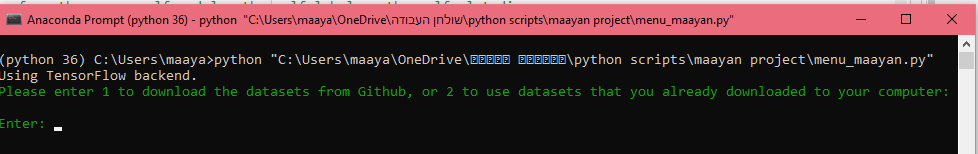


מטרת חלון זה היא להכניס לתכונה self.\_\_data\_set את מאגר התמונות הלא ממוין, ולתכונה self.\_\_sorted\_data\_path את מאגר תמונות הממוין.

כמו כן, התוכנית טוענת את התמונות מאגר התמונות הממוין, ומחלקת אותו לתמונות לטריין (70%), לוולידיישן (10%) ולטסט (20%).

תהליך זה לוקח מספר דקות.

לאחר שהמשתמש ילחץ על הכפתור הצהוב של העלאת הדאטא, יופיע הפלט הבא במסך ההרצה:

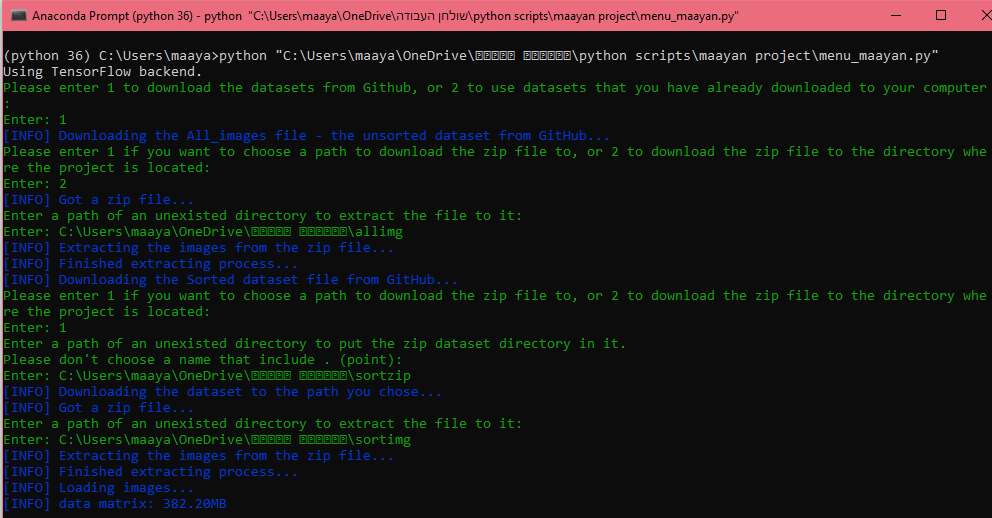


**כעת יש למשתמש 2 אפשרויות:**

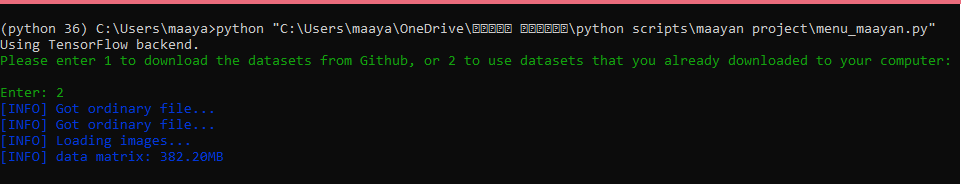
על המשתמש להקליד 1 או 2 בהתאם לכתוב, עבור כל קלט אחר התוכנית תמשיך לבקש קלט.

**1. העלאת התמונות - הdatasets מאתר Github** ישירות באמצעות הרצת הפרויקט, שכן ישנה אפשרות לבחור נתיב אליו התוכנית תוריד את הדאטאסט תוך כדי ההרצה.

אם בחר באופציה זו, עבור כל אחד משני המאגרים (הלא ממוין והממוין) תחילה יצטרך לבחור אם ברצונו להוריד את קובץ ה-zip למיקום של התיקייה בה נמצאים קבצי הפרויקט, או להכניס נתיב ל-directory חדש. לאחר מכן, יהיה על המשתמש להכניס את הנתיב למיקום שבו תשמר התיקייה המחולצת (לא דחוסה), בה יהיה שימוש כמאגר הנתונים במהלך הרצת הפרויקט.



**2. שימוש ב-datasets שהמשתמש כבר הוריד** מלכתחילה ושמר על המחשב שלו, מאתר גיטהאב מהחשבון שלי, לפני שהתחיל להריץ את הפרויקט. ה-datasets כוללים את התיקייה בה מאוכלסות התמונות באופן לא ממוין, וכן את התיקייה בה מאוכלסות התמונות באופן ממוין.  
כמו כן, המשתמש הוריד את התיקייה הכוללת תמונות של אותיות בכתב היד שלי כדי לבחון חיזויים של המודל בסוף.  
לאחר שהוא שמר את מאגרי התמונות על המחשב, הוא עדכן את הנתיבים בפונקציית הmain.



המשתמש הקיש 2 – כלומר העלאת הדאטא מתוך התיקיות שכבר שמורות על המחשב שלו.

תחילה מתקבלת הודעת אינפורמציה המודיעה על כך שהתוכנית טוענת את התמונות (תהליך זה יכול לקחת מספר דקות). לאחר סיום טעינת התמונות, מתקבלת הודעות אינפורמציה נוספת המעידה על הגודל במגה בתים של רשימת כל התמונות כמערכים.

עבור כל אחד מקבצי ה-datasets תודפס אחת מהודעות האינפורמציה הבאות:

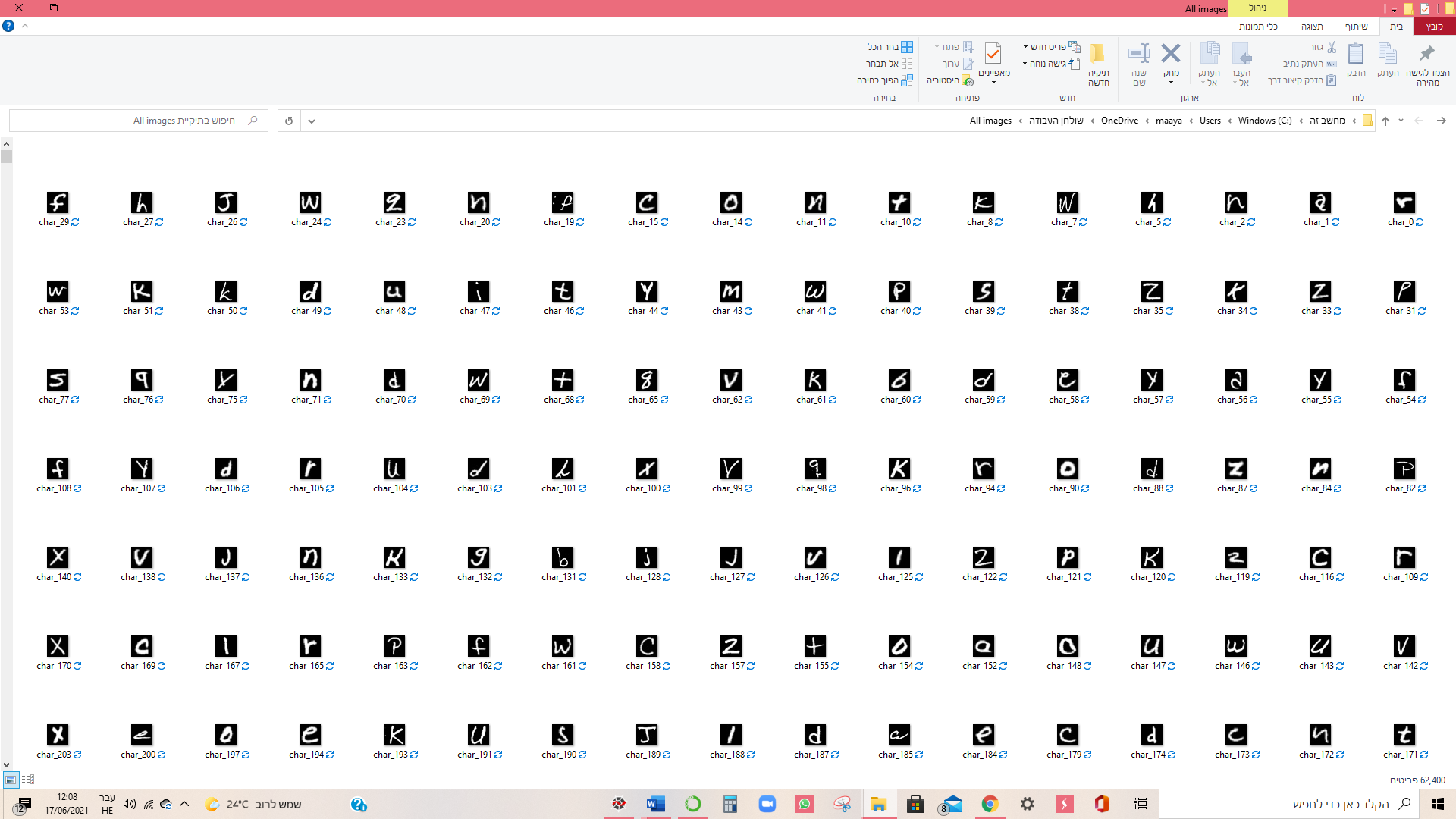
**1)** הודעת אינפורמציה בכחול לפיה התוכנה קיבלה קובץ רגיל – שאינו קובץ zip.

Got ordinary file…

במחשב שלי:

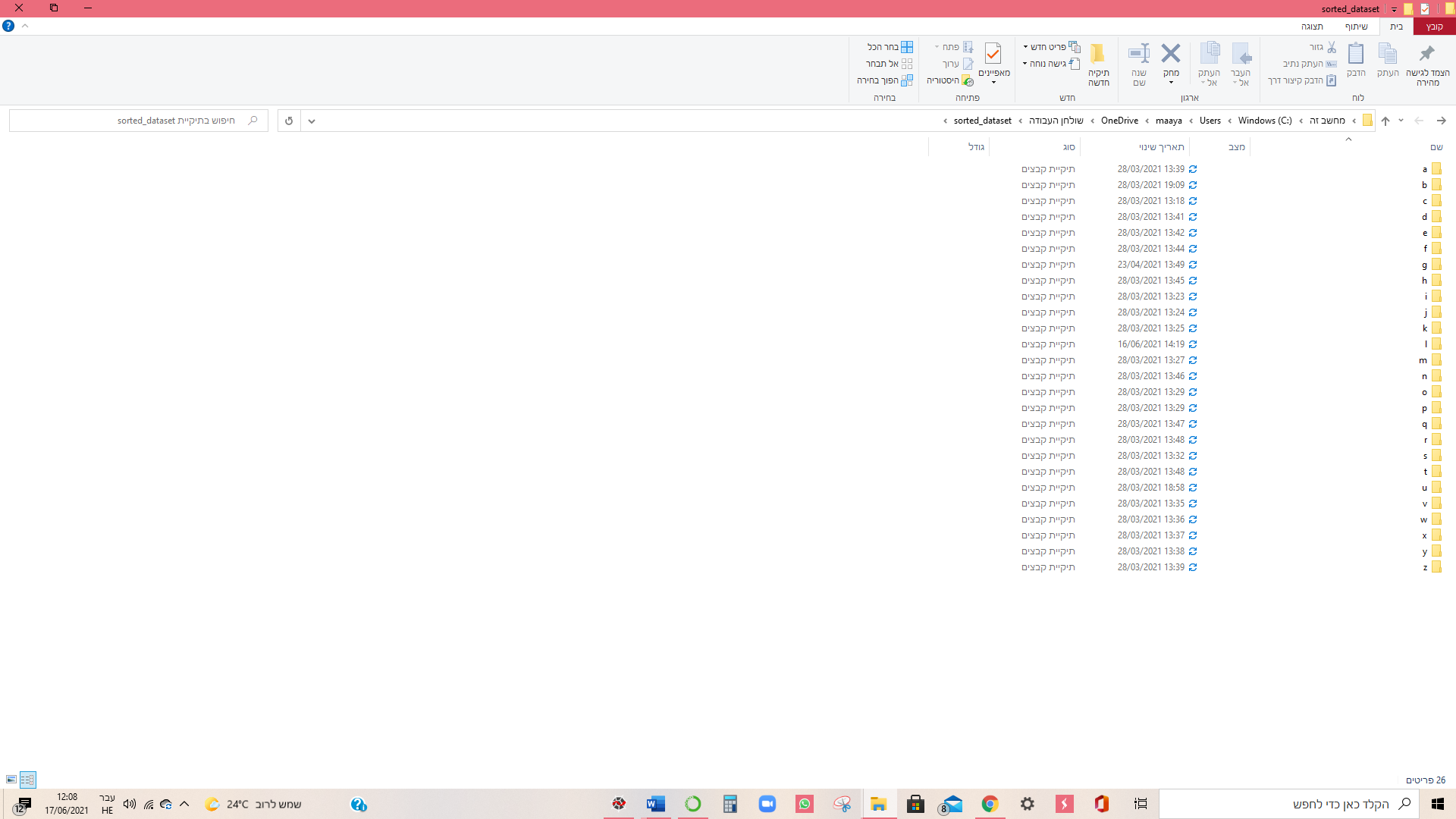
עבור קובץ התמונות שלא ממוין:

C:/Users/maaya/OneDrive/שולחן העבודה/ All images



עבור קובץ התמונות הממוין:

C:/Users/maaya/OneDrive/שולחן העבודה/sorted\_dataset



**2)** הודעת אינפורמציה בכחול לפיה התוכנה קיבלה קובץ דחוס - קובץ zip. במידה והתיקייה שהמשתמש הוריד למחשב שלה הינה מקובץ zip, הוא היה מקבל הודעת אינפורמציה לפיה התוכנה קיבלה קובץ zip.

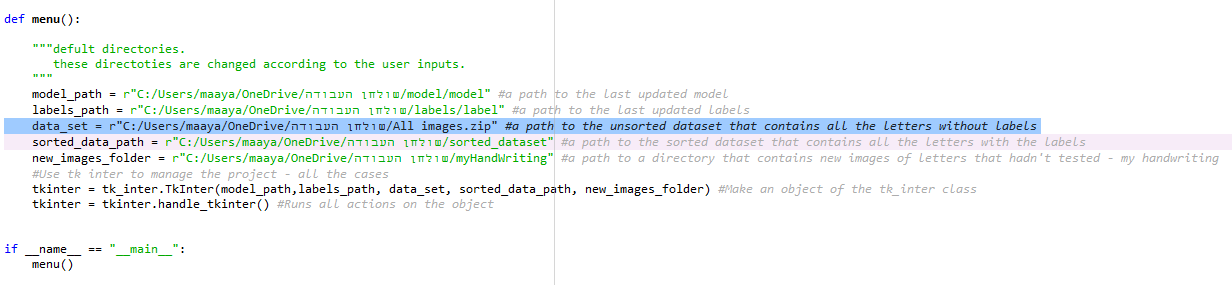
Got a zip file…

ולאחריה היה מתבצע תהליך של חילוץ הקבצים – המשתמש מכניס קלט של נתיב אליו הוא רוצה לחלץ את הקבצים.

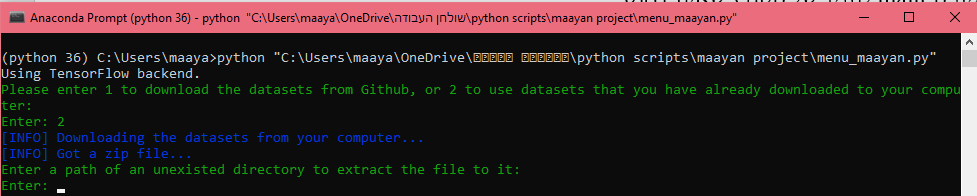
**דוגמה להרצה עם קובץ zip:**

כאמור, במידה ומאגר התמונות נשמר כקובץ zip, היה מתבצע תהליך של חילוץ קובץ הzip לתיקייה חדשה.

במחשב שלי ישנו קובץ zip המכיל בתוכו את כל התמונות באופן לא מסודר. אשנה בקוד את הנתיב הרלוונטי שמייצג את ה-dataset הלא ממוין לנתיב של קובץ ה-zip (שורה 107 בתמונה).



הרצה חדשה:

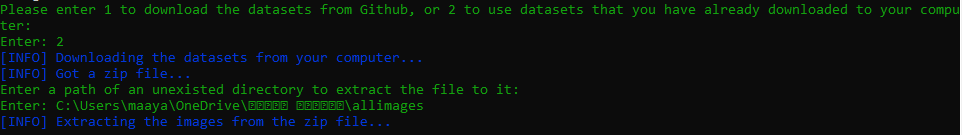


כפי שניתן לראות קיבלתי הודעת אינפורמציה שמאגר התמונות שמור בקובץ zip. בנוסף לכך, קיבלתי הודעה המבקשת directory שאליו יחולצו הקבצים. על המשתמש להקליד נתיב לקובץ לא קיים – שם ישמר הקובץ המחולץ.

directory זה ישמר במהלך התוכנית כ-directory בו שמור מאגר התמונות ולא ה-directory של קובץ ה-zip. (כמובן שבכל הרצה חדשה קובץ ה zip יחזור לשמש ככתובת דיפולטיבית, אם לא נשנה את הנתיב בפונקציית ה-main לנתיב של הקובץ שאינו דחוס.

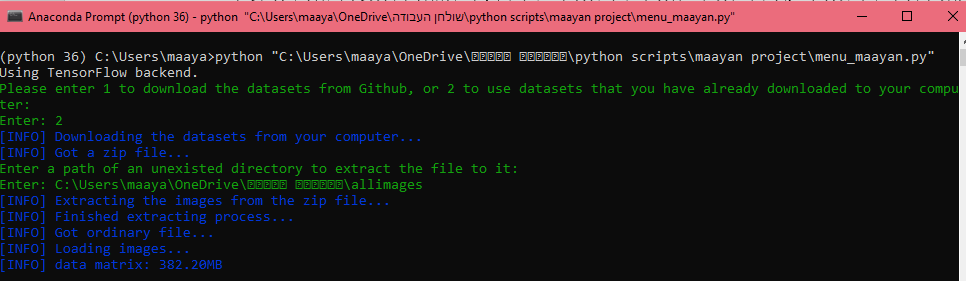
אכניס קלט תקין:

C:\Users\maaya\OneDrive\שולחן העבודה\allimages



הקלט תקין ולכן התוכנית מתחילה את תהליך יצירת התיקייה ואת אתחולה בתמונות – כלומר חילוץ קובץ התמונות מסוג zip לתיקייה זו.

נקבל הודעת אינפורמציה שהתהליך החל, ולאחריה הודעה שהתהליך החילוץ הסתיים.



לאחר סיום תת תהליך זה שאר השלבים זהים למקודם - מתקבלת הודעת אינפורמציה המודיעה על כך שהתוכנית טוענת את התמונות. לאחר סיום טעינת התמונות, מתקבלת הודעות אינפורמציה נוספת המעידה על הגודל במגה בתים של רשימת כל התמונות כמערכים.

**תקינות קלט:**

אם הנתיבים שהמשתמש הכניס כקלט לשים בהם את התיקיות הלא דחוסות אינם תקניים, יוצגו לפניו הודעות שגיאה.

נבחן את הודעות השגיאה השונות.

סוגי הודעות השגיאה:

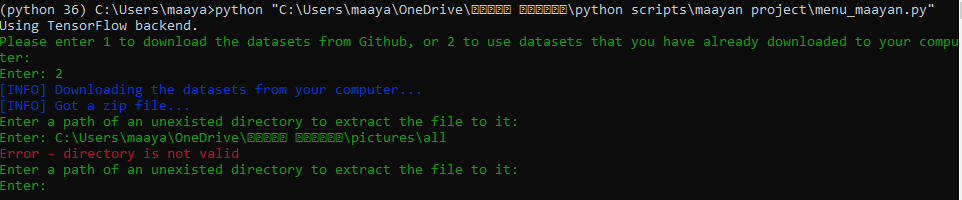
1) הודעת שגיאה Error - directory is not valid -

הודעת שגיאה שמשמעותה היא שהכתובת שנשלחה כקלט אינה תקינה והתוכנה מבקשת מאתנו קלט חדש.

דוגמה לקלט עבורו תודפס הודעת שגיאה זו:

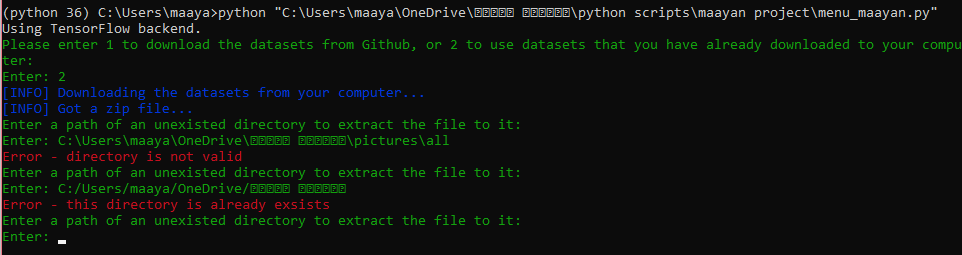
C:\Users\maaya\OneDrive\שולחן העבודה\pictures\all

(לא תקין כי התיקייה pictures לא קיימת ב directory זה).



2) הודעת שגיאה Error – this directory is already exists -

הודעת שגיאה שמשמעותה היא שהכתובת שנשלחה כקלט כבר קיימת, משמע ה- directoryאינו חדש – הוא אינו כולל את שם התיקייה החדשה ולכן התוכנה מבקשת קלט חדש.

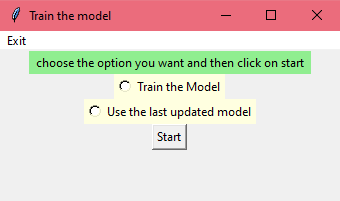
****

דוגמה לקלט עבורו תודפס הודעת שגיאה זו:

C:/Users/maaya/OneDrive/שולחן העבודה

נשלח כקלט directory שכבר קיים, ללא שם התיקייה החדשה אליה מעוניין המשתמש לאכלס את התמונות.

**האופציה השנייה – Train אימון המודל:**

לאחר שהמשתמש העלה את ה-data, יופיע החלון הבא:

מטרת חלון זה היא לאמן את המודל.

**בחלון זה יש למשתמש שתי אופציות:**

**1) שימוש במודל האחרון שנשמר**, שכבר עבר Train. המשתמש יכול לבחור שלא לעשות כעת את תהליך ה-Train הארוך, ובמקום זאת להשתמש במודל שכבר אומן קודם לכן ונשמר כקובץ במחשב – כאשר הנתיב אליו (ואל התוויות) מצויים בקובץ ה-main כברירת מחדל.  
אם יבחר באופציה זו תוצג לפניו הודעת האינפורמציה הבאה:



**2) ביצוע תהליך ה-Train כעת:** המודל לומד את התמונות שחולקו לטריין וכך מעלה את מידת ה-accuracy שלו ומפחית את מידת ה-loss שלו. תוך כדי האימון, המודל מציג כפלט בנוסף לערכי ה-accuracy,loss של ה-Train, גם את ערכים אלו מה-validation שמדמים את ה-Test בזמן האימון.

תהליך זה ארוך ויכול להימשך מספר שעות.

אם המשתמש בחר באופציה זו – ביצוע תהליך ה-Train, יוצגו בפניו בקשות קלט האחת אחרי השנייה. מטרתן לקבל נתיבים לכתובות שבהן המשתמש רוצה לשמור את המודל המאומן, את התוויות, ואת הגרפים שנוצרים במהלך תהליך האימון.

המשתמש יראה את הבקשה להזין את הקלט הבא רק לאחר שהזין קלט תקין.

בדיקות תקינות הקלט זהות לקודמות, ונוספה הודעת שגיאה נוספת:

הודעת שגיאה Error – file will be override

משמעות הודעת שגיאה זו היא שהמשתמש הכניס כקלט נתיב של תיקייה שהוא כבר הכניס קודם לכן עבור קובץ אחר.

תהלך הקלט לקראת ביצוע האימון:

1. קלט לנתיב שבו ישמר קובץ המודל המאומן – על המשתמש להכניס את ה-directory המלא עם שם הקובץ החדש שייווצר, בו הוא ירצה לשמור את המודל לאחר ביצוע האימון.

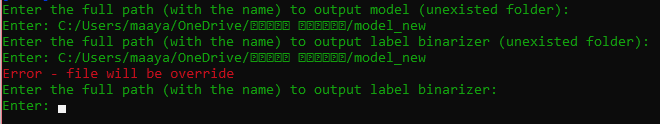
לשם הדוגמה אכניס את הנתיב הבא (directory שלא קיים לי במחשב):

C:/Users/maaya/OneDrive/שולחן העבודה/model\_new



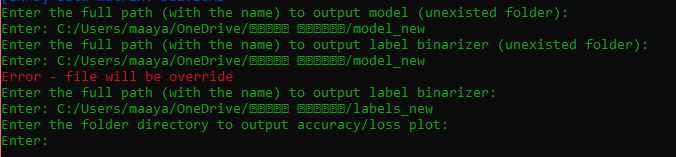
לאחר שהכנסתי את הנתיב לקובץ בו ישמר המודל, התוכנה מבקשת ממני directory חדש עם שם הקובץ שייווצר, בו נשמור את קובץ התוויות הבינארי (קובץ זה ישמש לביצוע הלמידה ולחיזוי קטגוריה של תמונה).

אם אכניס כעת את אותו ה-directory של הקובץ הקודם, קובץ אחד ידרוס את השני או שהתוכנית עלולה לקרוס, ולכן כדי למנוע זאת – מוצגת הודעת השגיאה ונמנעת מהמשתמש אפשרות זו. לדוגמה, אם כעת אנסה להכניס כקלט את ה directory הקודם שהזנו נקבל הודעת שגיאה.



אכניס את ה directory:

C:/Users/maaya/OneDrive/שולחן העבודה/labels\_new

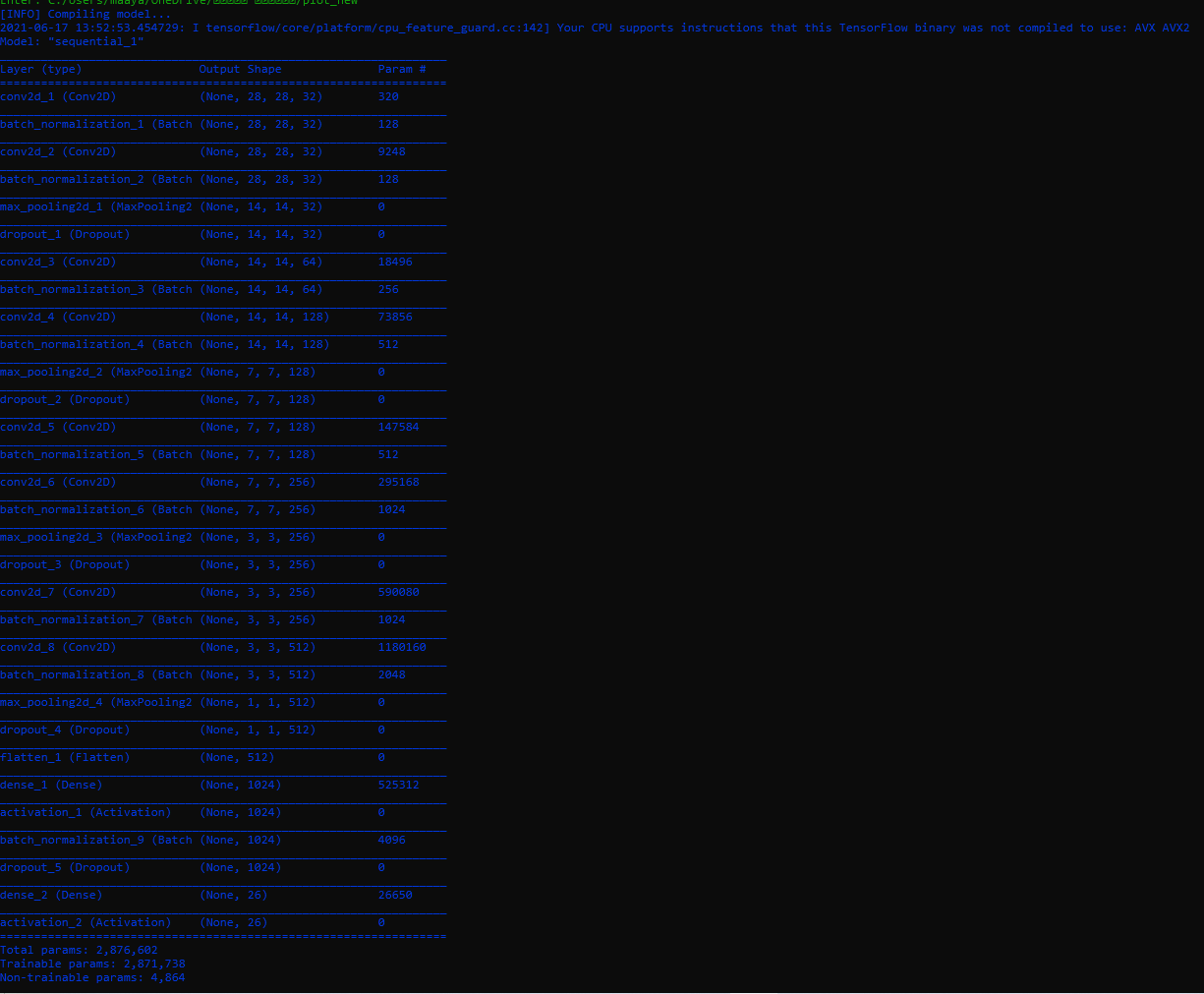


לאחר קלט זה התוכנית מבקשת נתיב לתיקייה חדשה שתיווצר, בה ישמרו גרפי תוצאות הלימוד (כקבצים מסוג png). גם כאן, אם נכניס נתיב לתיקייה שכבר הכנסנו למודל או לתוויות, תתקבל הודעת השגיאה.

אכניס את ה directory:

C:/Users/maaya/OneDrive/שולחן העבודה/plot\_new

לאחר הכנסת כל הקלטים, תהליך האימון מתחיל לרוץ ונקבל הודעת אינפורמציה על כך שהמודל התקמפל וישר לאחר מכן הודעה על כך שהחל האימון.



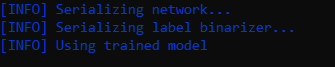


ניתן לראות שאחרי כל Epoch התחזיות משתפרות: ה-accuracy גדל, ה- loss קטן.

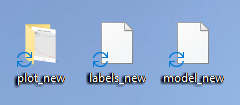
בסופו של דבר ערך ה-accuracy הוא 0.939 וערך ה-loss הוא 0.178 עבור ה-validation.

עבור ה-Train ערך ה-accuracy הוא 0.93 וערך ה-loss הוא 0.199

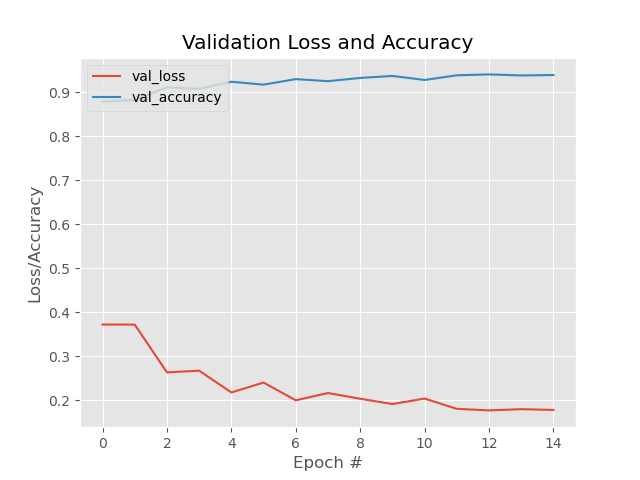
בסיום האימון אנו מקבלים הודעה כי כעת השימוש יתבצע בקובץ המודל החדש המאומן (ולא בדיפולטיבי)

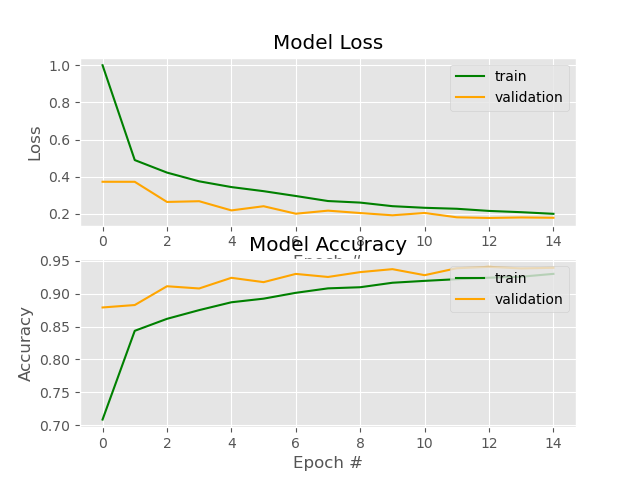


ניתן לראות כי ב-directories שהזנו אכן נוצרו הקבצים בהתאמה (צילום מסך של שולחן העבודה שלי):



קובץ המודל המאומן, קובץ התוויות הבינארי, התיקייה ובה שלושת קבצי תמונה שבהם גרפים מתהליך האימון:





תמונה הכוללת את הטריין והוולידיישן יחד

גרף של הטריין

גרף של הוולידיישן

**האופציה השלישית – Test בחינת המודל:**

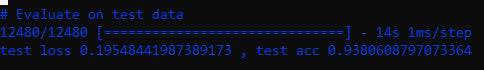
לאחר שהמשתמש אימן את המודל או בחר להשתמש במודל האחרון המאומן, יופיע החלון הבא:



מטרת חלון זה היא לבצע חיזוי של המודל. בשלב זה, לאחר שהמודל מאומן, הוא משתמש בלמידה שצבר ומנסה לזהות תמונות חדשות שלא התנסה בהן קודם לכן. בשלב זה רואים כפלט את אחוזי ההצלחה הסופיים של המודל – accuracy, loss.

אם המשתמש בחר להשתמש במודל הקודם שכבר אומן ולא לבצע את תהליך האימון מחדש, יוצגו ערכי ההצלחה של המודל האחרון המאומן.

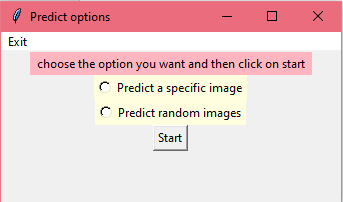
הפלט המתקבל מתהליך ה-Test:



ניתן לראות שתוצאת ה-accuracy הסופית הינה 0.938 ותוצאת ה-loss הסופית הינה 0.195.

**האופציה הרביעית – חיזוי תמונה:**

לאחר שהמשתמש ביצע Test בחינה של המודל, יופיע החלון הבא:



מטרת חלון זה היא לבצע חיזוי המודל על תמונות ספציפיות.

בשלב זה מופיעות בפני המשתמש שתי אופציות:

**1) לבחור תמונה בעצמו** ולהכניס כקלט את הנתיב לתמונה. המודל יזהה את התמונה בהתאם ללמידה שלו ויציג בפני המשתמש את החיזוי שלו לאות שהוא מזהה בתמונה.

2**) שהתוכנית תבחר מספר תמונות לפי הכמות שהמשתמש רצה באופן אקראי**, והמודל יזהה את כל אחת מהתמונות בהתאם ללמידה שלו ויציג בפני המשתמש את החיזוי שלו לאות שהוא מזהה בתמונה. כאן התמונות יהיו גם תמונות מהדאטאסט המקורי וגם תמונות חדשות בכתב היד שלי שהמודל לא מכיר.

\*יש לשים לב כי כל התמונות חייבות להיות מסוגJPG/JPEG (ולא png).

אם המשתמש בחר באופציה **הראשונה**, התוכנית מבקשת מן המשתמש את הנתיב לתמונה שאותה הוא רוצה שהמודל המאומן יזהה.

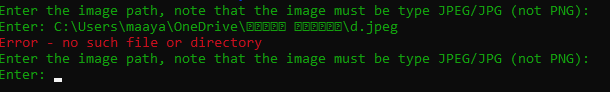
התמונה חייבת להיות כתובה באותו פורמט שהתמונות שהמודל למד מצויות בו – אות לבנה על גבי רקע שחור. לכן מודפסת כאן ההודעה הבאה, שכן יש להבחין בין אם מדובר בתמונה בפורמט הרצוי כמו אלו שב-dataset, אם כן על המשתמש להקליד 2. או שהתמונה בפורמט הפוך – אות שחורה על גבי רקע לבן ואז צריך לבצע היפוך צבעים של התמונה, במקרה זה על המשתמש להקליד 1. עבור כל פלט אחר שאינו 1/2 התוכנה תדרוש קלט חדש.

כמובן שהנתיב שניתן חייב להיות לקובץ קיים, ולכן אם יינתן directory שאינו קיים נקבל הודעת שגיאה.

לשם המחשת שגיאה זו אכניס כקלט את ה directory:

C:\Users\maaya\OneDrive\שולחן העבודה\d.jpeg

והנה קיבלנו הודעת שגיאה שמציינת בפנינו שאין קובץ או directory כמו זה שהזנו כקלט ואכן אין קובץ כזה בנתיב הנ"ל.



נבחר את התמונה אותה אנו רוצים לחזות מתוך מאגר התמונות ונשלח את ה directory שלה כקלט.

אכניס את ה directory:

C:\Users\maaya\OneDrive\שולחן העבודה\myHandWriting\d.jpeg

נקבל מיד הודעה על כך שהמודל השמור נטען בידי התוכנית (Loading network).

בכתובת זו נמצאת התמונה הבאה:

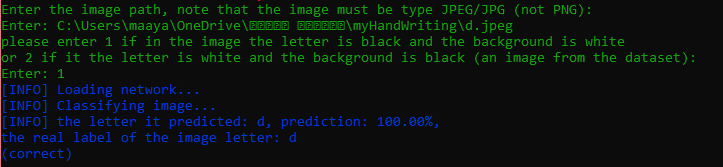
תמונה זו כפי שניתן לראות מסמלת את האות d בכתב היד שלי, בפורמט הפוך מהפורמט הרצוי – לכן אקליד 1.



לכן התוכנה תבצע היפוך צבעים ולאחר מכן תבצע חיזוי של התמונה.



החיזוי של המודל:



מתקבלת הודעת אינפורמציה שהתוכנית כעת חוזה את הקטגוריה שאליה שייכת התמונה שבחרנו (Classifying image).

לבסוף מודפס החיזוי של המודל, וניתן לראות שהמודל זיהה נכונה את התמונה שכן הוא חזה שהאות שבתמונה היא d ב-100%, ואכן צדק. התוכנית גם מדפיסה כפלט האם היא חזתה את הקטגוריה נכון או לא.

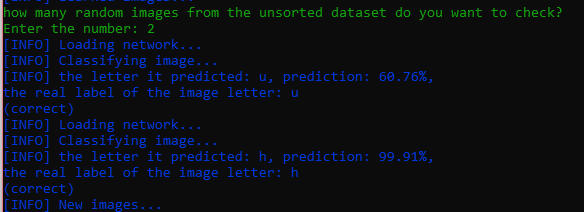
בנוסף להדפסה זו נפתח בפנינו מסך חדש ובו התמונה אותה שלחנו כקלט. רק לאחר שנסגור את החלון הנ"ל התוכנית תמשיך הלאה ותבקש מאיתנו לבחור אופציה חדשה.

אם המשתמש בחר באופציה **השנייה**, תחילה התוכנית מבקשת מן המשתמש להכניס כקלט מספר המייצג כמות של תמונות מה-dataset שאותן התוכנה תבחר באקראיות לזהות. לאחר מכן, התוכנית מבקשת מן המשתמש להכניס כפלט מספר המייצג כמות של תמונות חדשות שהמודל לא התנסה בהן – כתב היד שלי, אותן התוכנה תבחר באקראיות לזהות.

השלב הראשון – זיהוי תמונות מה-dataset:

בשלב זה התוכנה מבקשת מהמשתמש להכניס כקלט מספר שלם המייצג את כמות התמונות מתוך ה-dataset שהוא רוצה שהתוכנה תזהה.

לאחר שהוא מכניס מספר שלם (אם לא הכניס מספר שלם התוכנה דורשת קלט חדש), התוכנה בוחרת באקראיות מתוך מאגר התמונות מספר תמונות לפי המספר שהמשתמש בחר אותן היא תזהה.



עבור כל תמונה נקבל הודעה על כך שהמודל השמור נטען בידי התוכנית (Loading network). בנוסף, נקבל הודעת אינפורמציה המעידה על כך שהתוכנית כעת חוזה את הקטגוריה שאליה שייכת התמונה (Classifying image), ולבסוף ניתן לראות שהמודל זיהה את התמונה.

בנוסף להדפסה זו נפתח בפנינו מסך חדש ובו התמונה אותה שלחנו כקלט. רק לאחר שנסגור את החלון הנ"ל התוכנית תמשיך הלאה.

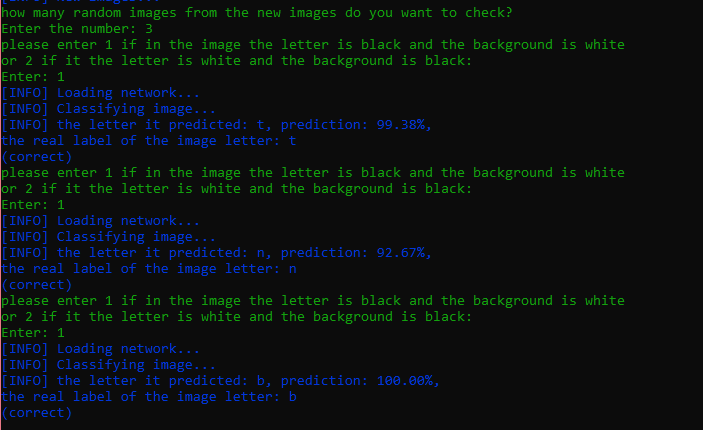
התמונות ב-dataset מצויות בפורמט הרצוי – אות לבנה על גבי רקע שחור, שכן אלו התמונות שהמודל למד מהן. לכן אין צורך להפוך את הצבעים של התמונה.

השלב השני – זיהוי תמונות מהמאגר שהמודל לא התנסה בהן -

בשלב זה התוכנה מבקשת מהמשתמש להכניס כקלט מספר שלם המייצג את כמות התמונות מתוך מאגר התמונות החדשות (בכתב היד שלי) שהוא רוצה שהתוכנה תזהה.

לאחר שהוא מכניס מספר שלם (אם לא הכניס מספר שלם התוכנה דורשת קלט חדש), התוכנה בוחרת באקראיות מתוך מאגר התמונות החדש מספר תמונות לפי המספר שהמשתמש בחר אותן היא תזהה.

עבור כל תמונה התוכנה תשאל אם מדובר בתמונה בפורמט הרצוי כמו אלו שב-dataset – אות לבנה על רקע שחור, אם כן על המשתמש להקליד 2. או שהתמונה בפורמט הפוך – אות שחורה על גבי רקע לבן ואז צריך לבצע היפוך צבעים של התמונה, במקרה זה על המשתמש להקליד 1. עבור כל פלט אחר שאינו 1/2 התוכנה תדרוש קלט חדש.



גם כאן, עבור כל תמונה נקבל הודעה על כך שהמודל השמור נטען בידי התוכנית (Loading network). בנוסף, נקבל הודעת אינפורמציה המעידה על כך שהתוכנית כעת חוזה את הקטגוריה שאליה שייכת התמונה (Classifying image), ולבסוף ניתן לראות שהמודל זיהה את התמונה.

בנוסף להדפסה זו נפתח בפנינו מסך חדש ובו התמונה אותה שלחנו כקלט. רק לאחר שנסגור את החלון הנ"ל התוכנית תמשיך הלאה.

למשל, התמונה הראשונה מתוך ה-3 שהתוכנית בחרה באקראיות הינה:



תמונה זו כפי שניתן לראות מסמלת את האות d בכתב היד שלי, בפורמט הפוך מהפורמט הרצוי – לכן הקלדתי 1.

לכן התוכנה תבצע היפוך צבעים ולאחר מכן תבצע חיזוי של התמונה.



כפי שניתן לראות למעלה, קיבלתי כפלט שהתמונה היא מייצגת את האות t ב-99.38%. התוכנית גם הדפיסה שהיא חזתה את הקטגוריה נכון.

**מדריך למפתח:**

הפרויקט שלי מחולק למספר קבצי קוד אשר לכל אחד ישנו תחום אחריות שונה. חלוקה זו בין חלקי הקוד השונים העוסקים בחלקים שונים בפרויקט מאפשרת ארגון קוד, ממזערת באגים ותקלות למיניהם ותרמה לי לנוחות בעת ביצוע הפרויקט.

כעת אציין את שמות הקבצים השונים ותחום האחריות שהוטל על כל אחד מהם:

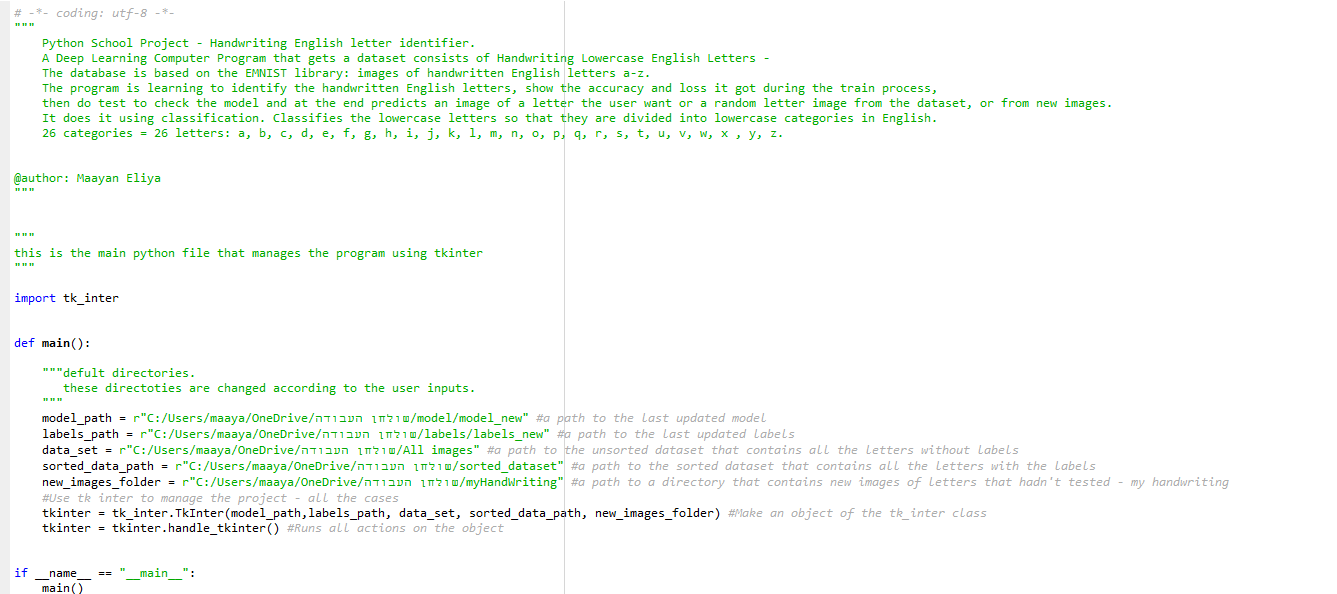
|  |  |
| --- | --- |
| **תוכן הקובץ:** | **שם הקובץ:** |
| קובץ זה מנהל את התוכנית – זהו הקובץ הראשי. בו נכללים הנתיבים לקבצים הדרושים להפעלת התוכנה וכן הוא יוצר אובייקט של המחלקה TkInter דרכו מתבצעות כל הפעולות. | menu\_maayan.py |
| קובץ זה יוצר את התקשורת עם המשתמש ומנהל את התוכנית על פי בחירות המשתמש. | tk\_inter.py |
| קובץ זה אחראי לטעינת הנתונים – לטעון את מאגר התמונות הממוין לקראת האימון וה-test (ולחלק אותו לטריין, טסט, וולידיישן). | load\_data.py |
| קובץ זה אחראי על הגדרת המודל ושכבותיו. | the\_model.py |
| קובץ זה אחראי על אימון המודל (משתמש במודל שהוגדר בקובץ the\_model.py ובמאגר התמונות הממוין). | train\_model.py |
| קובץ זה אחראי על בחינת המודל – הTest. מדפיס את ה-accuracy,loss הסופיים. | test\_model.py |
| קובץ זה אחראי על חיזוי הקטגוריה של תמונה שהתקבלה (או באקראיות על ידי התוכנה או כקלט מהמשתמש). | classify.py |
| אם התוכנית קיבלה כקלט את התמונות בקובץ zip, קובץ זה אחראי על חילוץ התמונות לתיקייה חדשה. | extract\_zipfile |
| קובץ זה אחראי על בדיקת תקינות קלט –  אם היה צריך להקליד 1/2 מוודא שזה מה שהוקלד, אם היה צריך מספר שלם מוודא שהקלט הוא מספר שלם. | check\_input.py |
| קובץ זה אחראי על תקינות קלטי המשתמש הנוגעים לנתיבים - המיקום של קבצים ותיקיות שונות במחשב. | check\_directory.py |
| קובץ זה אחראי על ההדפסות השונות המוצגות למשתמש בהתאם לסוג ההדפסה. | prints\_types.py |

כעת נעבור על הקבצים השונים ועל הפונקציונאליות הכלולה בהם:

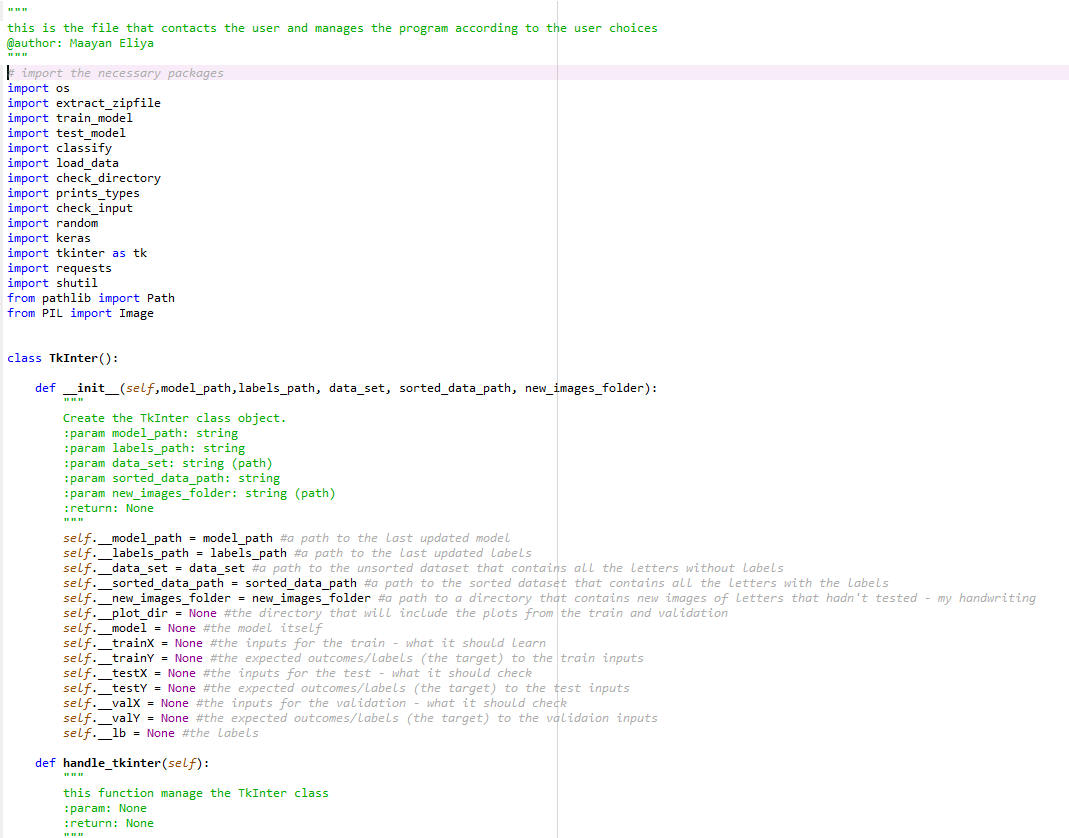
**קובץ ראשי – menu\_maayan.py :**

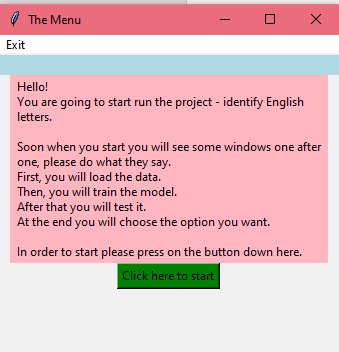
קובץ זה מנהל את התוכנית – זהו הקובץ הראשי. נכללים בו כל הנתיבים לקבצים הדרושים להפעלת התוכנה. כמו כן, הוא יוצר אובייקט של המחלקה TkInter שנמצאת בקובץ tk\_inter.py דרכו מתבצעות כל הפעולות.

בקובץ זה מצויה פונקציה אחת - הפונקציה הראשית, ששמה main()המהווה פונקציית שירות עבור המשתמש, שכן בה המשתמש שם את כל הנתיבים לקבצים הדרושים וכן היא יוצרת את אובייקט המחלקה TkInter. המשתמש מריץ קובץ פייתון זה ודרכו רצה כל התוכנית.



**קובץ – tk\_inter.py:**

קובץ פייתון זה יוצר חלונות של tkinter המציגים את האפשרויות שהמשתמש יכול לבצע. על מנת לבצע את הפונקציונאליות שבה המשתמש בחר, קובץ זה משתמש בשאר הקבצים.

זהו למשל החלון הראשון שנוצר באמצעות קובץ זה:

**כפי שכתוב חלון ה-menu, קובץ זה כולל בתוכו ארבע אפשרויות:**

1 – העלאת הדאטא

2 – אימון המודל – Train או שימוש במודל האחרון שאומן

3 – בחינת המודל - Test

4 – חיזוי קטגוריה של תמונה ספציפית (שהמשתמש שולח כקלט או שהתוכנה בוחרת באקראיות)

**האופן בו מומש קובץ זה:**

קובץ זה כולל מחלקה שנקראת TkInter, ובתוכה הפונקציות הבאות:

|  |  |
| --- | --- |
| תפקידה | הפונקציה |
| **תחום אחריות:**  פונקציה זו אחראית על יצירת אובייקט של המחלקה.  **אופן ביצוע:**  פונקציה זו יוצרת אובייקט של המחלקה בכך שהיא מכניסה לכל תכונה את הפרמטר המתאים. | \_\_init\_\_(self,model\_path,labels\_path, data\_set, sorted\_data\_path, new\_images\_folder) |
| **תחום אחריות:**  פונקציה זו יוצרת את החלון הראשון – התפריט.  **אופן ביצוע:**  פונקציה זו יוצרת חלון עם הסברים לגבי מהלך התוכנית והאפשרויות השונות, וכפתור start, שכאשר לוחצים עליו יש קריאה לפונקציה \_\_data\_window(self) ונפתח החלון הבא. | menu\_window(self) |
| **תחום אחריות:**  פונקציה זו יוצרת את החלון השני – של העלאת הדאטא וחלוקת התמונות לטריין, טסט, ולידיישן.  **אופן ביצוע:**  פונקציה זו יוצרת חלון שמטרתו להעלות את הדאטא. כשלוחצים על כפתור העלאת הדאטא יש קריאה לפונקציה \_\_case\_Data(self) וכן לפונקציה \_\_train\_window(self) ונפתח החלון הבא. | \_\_data\_window(self) |
| **תחום אחריות:**  פונקציה זו יוצרת את החלון השלישי – של אימון המודל (Train).  **אופן ביצוע:**  פונקציה זו יוצרת חלון שמטרתו לאמן את המודל. בחלון זה ישנן שתי אפשרויות – הראשונה אימון מודל חדש, השנייה העלאת המודל המאומן האחרון. כשלוחצים על אחד משני הכפתורים הללו, יש קריאה לפונקציה \_\_start\_train(self,choice) שמקבלת את ערך הכפתור שנלחץ. | \_\_train\_window(self) |
| **תחום אחריות:**  פונקציה זו אחראית להפעלת האפשרות לאימון שהמשתמש בחר מתוך 2 האפשרויות: אימון המודל או שימוש במודל המעודכן האחרון המאומן.  **אופן ביצוע:**  פונקציה זו מקבלת ערך של 1 או 2 שמייצג את הבחירה של המשתמש. בהתאם לערך שקיבלה היא קוראת לפונקציה המתאימה שאחראית לאותה פעולה – \_\_case\_Train(self) או \_\_case\_lastmodel(self). בכל מקרה היא גם קוראת לפונקציה \_\_test\_window(self) ונפתח החלון הבא. | \_\_start\_train(self,choice) |
| **תחום אחריות:**  פונקציה זו יוצרת את החלון הרביעי – של בחינת המודל (Test).  **אופן ביצוע:**  פונקציה זו יוצרת חלון שמטרתו לבחון את המודל. כשלוחצים על כפתור בחינת המודל יש קריאה לפונקציה \_\_case\_Test(self) וכן לפונקציה \_\_options\_window(self) ונפתח החלון הבא. | \_\_test\_window(self) |
| **תחום אחריות:**  פונקציה זו יוצרת את החלון החמישי והאחרון – של חיזוי המודל לגבי תמונה.  **אופן ביצוע:**  פונקציה זו יוצרת חלון שמטרתו לבצע חיזוי של המודל. בחלון זה ישנן שתי אפשרויות – הראשונה חיזוי של תמונה ספציפית שהמשתמש בחר, השנייה חיזוי של תמונות שנבחרות באופן אקראי לפי כמות שהמשתמש בחר. כשלוחצים על אחד משני הכפתורים הללו, יש קריאה לפונקציה \_\_start\_options(self,choice) שמקבלת את ערך הכפתור שנלחץ. | \_\_options\_window(self) |
| **תחום אחריות:**  פונקציה זו אחראית להפעלת האפשרות לחיזוי שהמשתמש בחר מתוך 2 האפשרויות: חיזוי של תמונה ספציפית שהמשתמש בחר או חיזוי של תמונות שנבחרות באופן אקראי לפי כמות שהמשתמש בחר.  **אופן ביצוע:**  פונקציה זו מקבלת ערך של 1 או 2 שמייצג את הבחירה של המשתמש. בהתאם לערך שקיבלה היא קוראת לפונקציה המתאימה שאחראית לאותה פעולה – \_\_case\_Specific\_image(self) או \_\_case\_Random(self). | \_\_start\_options(self,choice) |
| **תחום אחריות:**  פונקציה זו מאפשרת להשתמש בכפתור יחיד על מנת לזמן שתי פונקציות במקום רק אחת.  **אופן ביצוע:**  פונקציה זו מקבלת שמות של פונקציות כפרמטרים ומזמנת את הפונקציות**.** | \_\_combine\_funcs(self, \*funcs) |
| **תחום אחריות:**  כאשר המשתמש לחץ על כפתור העלאת הדאטא, יש זימון לפונקציה זו. פונקציה זו אחראית על העלאת ה-datasets ובדיקה אם הם קבצי zip, אם כן יש חילוץ של הקבצים. ניתן לבחור האם להעלות את המאגרים מתוך אתר גיטהב או להשתמש במאגרים שכבר הורדו למחשב לפני ההרצה. בנוסף הפונקציה אחראית לחלק את הדאטאסט הממוין לתמונות לטריין, לטסט, לוולידיישן.  **אופן ביצוע:**  פונקציה זו מבקשת תחילה לבחור האם להעלות את המאגרים מתוך אתר גיטהב או להשתמש במאגרים שכבר הורדו למחשב לפני ההרצה. במידה והמשתמש בחר באפשרות הראשונה, הפונקציה מקבלת כקלט את המיקום שבו המשתמש רוצה לשמור את תיקיות התמונות במחשב. מורידה את המאגרים באמצעות הפונקציה \_\_download\_file(self, filename, r). לאחר מכן היא מחלצת את התמונות מן הקובץ לתיקייה שהמשתמש הזין כקלט, במידה והתמונות שמורות בקובץ zip. אם המשתמש בחר להשתמש במאגרים שכבר הוריד קודם לכן למחשב, הפונקציה בודקת אם אלו תיקיות דחוסות ומחלצת אותן במידה וכן – על ידי שימוש בקובץ extract\_zipfile.  בנוסף, הפונקציה מבצעת שימוש בקובץ load\_data.py על מנת לחלק את הדאטאסט הממוין לתמונות לטריין, לטסט, לוולידיישן. | \_\_case\_Data(self) |
| **תחום אחריות:**  הורדת מאגר התמונות הממוין והלא ממוין מתוך אתר גיטהב.  **אופן ביצוע:**  פונקציה זו מקבלת את שם הקובץ שעליה להוריד, את המקור לקובץ מאתר גיטהב. באמצעות פרמטרים אלה היא מורידה את הנתונים בהתאם לרצון המשתמש – אם בחר להוריד לתיקייה שבה הפרויקט נמצא או אם שולח נתיב שהוא בוחר. | \_\_download\_file(self, filename, r) |
| **תחום אחריות:**  כאשר המשתמש לחץ על הכפתור של אימון המודל, יש זימון לפונקציה זו. פונקציה זו אחראית על אימון המודל על ידי שימוש בקובץ train\_model.py.  **אופן ביצוע:**  פונקציה זו קולטת מהמשתמש נתיבים שבהם הוא מעוניין לשמור את המודל המאומן, את קובץ התוויות הבינארי ואת הגרפים מתהליך האימון.  לאחר מכן הפונקציה משתמשת בפונקציונאליות שבקובץ train\_model.py ומחזירה משם את המודל המאומן. | \_\_case\_Train(self) |
| **תחום אחריות:**  כאשר המשתמש לחץ על הכפתור של שימוש במודל המעודכן האחרון המאומן, יש זימון לפונקציה זו. פונקציה זו אחראית על העלאת המודל המאומן.  **אופן ביצוע:**  פונקציה זו מעלה את המודל המאומן ששמור בנתיב שנמצא בתכונה self.\_\_model\_path על ידי שימוש בפקודה keras.models.load\_model. | \_\_case\_lastmodel(self) |
| **תחום אחריות:**  כאשר המשתמש לחץ על הכפתור של בחינת המודל (ביצוע Test), יש זימון לפונקציה זו. פונקציה זו אחראית על בחינת המודל – הדפסת ערכי ה-accracy,loss.  **אופן ביצוע:**  פונקציה זו בוחנת את המודל המאומן על ידי שימוש בקובץ test\_model. | \_\_case\_Test(self) |
| **תחום אחריות:**  כאשר המשתמש לחץ על הכפתור של חיזוי של תמונה ספציפית, יש זימון לפונקציה זו. פונקציה זו אחראית לחיזוי תמונה ספציפית שהמשתמש בחר.  **אופן ביצוע:**  פונקציה זו קולטת מהמשתמש נתיב לתמונה לה הוא רוצה לבצע חיזוי. הפונקציה בודקת שהנתיב אכן קיים באמצעות שימוש בקובץ check\_directory.  הפונקציה שואלת את המשתמש אם התמונה הינה בפורמט של הדאטאסט – כלומר אות לבנה על רקע שחור או להפך, על מנת לדעת אם לבצע היפוך צבעים.  הפונקציה מבצעת חיזוי של תמונה לפי למודל המאומן על ידי שימוש בפונקציה בקובץ classify. | \_\_case\_Specific\_image(self) |
| **תחום אחריות:**  כאשר המשתמש לחץ על הכפתור של חיזוי תמונות אקראיות. פונקציה זו אחראית על חיזוי של תמונות שנבחרות באופן אקראי לפי כמות שהמשתמש בחר.  ראשית, יש חיזוי של תמונות אקראיות ממערך הנתונים הלא ממוין. שנית, יש חיזוי של תמונות אקראיות מהספרייה המכילה תמונות חדשות שלא נלמדו - כתב היד שלי.  **אופן ביצוע:**  פונקציה זו קולטת מהמשתמש מספר המייצג את הכמות של תמונות מכל מאגר שהוא רוצה לבצע חיזוי. היא בודקת שאכן הוקלד מספר שלם על ידי הקובץ check\_input. הפונקציה מבצעת חיזוי של תמונה לפי למודל המאומן על ידי קריאה לפונקציה \_\_checkPredict(self, num, dataset, sign) שמשתמשת בקובץ classify. | \_\_case\_Random(self) |
| **תחום אחריות:**  פונקציה זו אחראית על ביצוע החיזוי של תמונות שנבחרות באופן אקראי לפי כמות שהמשתמש בחר.  **אופן ביצוע:**  פונקציה זו מקבלת כפרמטרים את המספר המייצג את כמות התמונות שיש לבחור, את המאגר שממנו יש לבצע חיזוי של תמונות אקראיות, וכן סימן שמשמעותו לדעת אם התמונות מהמאגר החדש או מהדאטאסט.  בפונקציה זו יש לולאה שחוזרת מספר פעמים לפי המספר שהפונקציה קיבלה כפרמטר, עבור כל פעם שהלולאה עוברת נבדקת תמונה אחרת. אם הסימן שהפונקציה קיבלה מראה שהתמונות הן מהמאגר החדש, הפונקציה שואלת את המשתמש לגבי הפורמט של התמונה כדי לדעת אם לבצע היפוך צבעים. פונקציה זו מבצעת את החיזוי בכך שהיא קוראת לפונקציה handle\_classify() שממוקמת בקובץ classify.py. כל תמונה שבוצע לה חיזוי יורדת מהרשימה של התמונות. | \_\_checkPredict(self, num, dataset, sign) |

כעת נעבור על האפשרויות השונות (cases) הנקראים לפי הסדר כאשר המשתמש לוחץ על הכפתורים השונים.

**Case Data**

כאמור, מקרה זה אחראי על העלאת הדאטא – מאגרי התמונות, וכן חלוקת המאגר הממוין ל-Train,Test,Validation. כחלק מכך, הפונקציה בודקת האם הקבצים הם קבצי zip, ואם כן מחלצת אותם לתיקייה חדשה.

**הדבר נעשה באמצעות קובץ חילוץ התמונות – extract\_zipfile.py:**

בקובץ זה מומשה הפונקציונאליות הנדרשת לחילוץ קבצים מתוך תיקייה דחוסה מסוג zip במידה ואכן התקבלה תיקייה מסוג זה. בקובץ זה ישנה פונקציה אחת בלבד בשם extract\_Zip(path) המקבלת נתיב כלשהוא כפרמטר. הפונקציה בודקת אם התקבל קובץ zip, ואם כן אז הפונקציה תודיע על כך למשתמש ותדרוש ממנו נתיב חדש בו תשמור את הקבצים שתחלץ מן הקובץ שנמצא בנתיב שקיבלה כפרמטר. אם הקובץ שבנתיב שהתקבל אינו קובץ ,zip יוחזר אותו נתיב. אחרת, יוחזר הנתיב החדש שהמשתמש הזין כקלט ובו התיקייה הלא דחוסה.

תדפיס של הקוד:

"""

this python file responsible for Extracting a zip file to an ordinary file

@author: Maayan Eliya

"""

# import the necessary package

from zipfile import ZipFile

import check\_directory

import prints\_types

import os

def extract\_Zip(zip\_path):

"""

This function checks if a file is a zip file - if so the in extracts the images from the zip file.

It returns an ordinary file - unzipped file

:param: None

:return: None

"""

path = zip\_path #the path to the file

ls = zip\_path.split(".") #a list of substrings from the path that split by '.'

split = os.path.split(ls[0]) #split the path

filename = split[-1] #the name of the file

#Checks if the file/folder is a zip file by its ending - the file type

if ls[len(ls)-1] == "zip": #if it is a zip file

prints\_types.printProcess("[INFO] Got a zip file...")

path = check\_directory.Directory.get\_New\_Dir("Enter a path of an unexisted directory to extract the file to it: ") #the path to the directory the user chose to download the images to it

#Extract the folder to the new location that the user chose

with ZipFile(zip\_path, 'r') as zipObj:

prints\_types.printProcess("[INFO] Extracting the images from the zip file...")

zipObj.extractall(path) #Extract all the contents of the zip file to a different directory

path = os.path.join(path, filename) #the path to the unzipped folder itself (with the images inside)

prints\_types.printProcess("[INFO] Finished extracting process...")

else: #if it is not a zip file

prints\_types.printProcess("[INFO] Got ordinary file...")

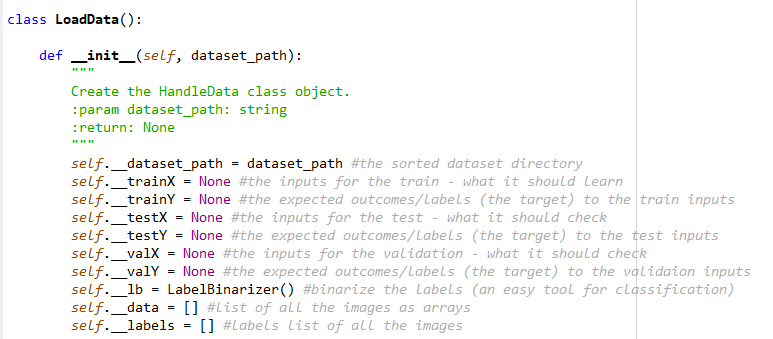
return path

**קובץ העלאת הנתונים – load\_data.py:**

קובץ זה כולל מחלקה ופעולות פנימיות של המחלקה.

ראשית נעבור על בנאי המחלקה:

כל ה-data members התכונות הם הרשאת private.



|  |  |  |
| --- | --- | --- |
| **תפקיד** | **ערך** | **המשתנה** |
| הכתובת שבה נמצא מאגר התמונות הממוין | Input | \_\_dataset\_path |
| התמונות שהמודל צריך ללמוד במהלך ה-Train | None | \_\_trainX |
| התוויות של התמונות שהמודל צריך ללמוד – המטרות המצופות | None | \_\_trainY |
| התמונות שהמודל צריך לחזות במהלך ה-Test | None | \_\_testX |
| התוויות של התמונות שהמודל צריך לחזות במהלך ה-Test – המטרות המצופות | None | \_\_testY |
| התמונות שהמודל צריך לחזות במהלך ה-Validation | None | \_\_valX |
| התוויות של התמונות שהמודל צריך לחזות במהלך ה-Validation – המטרות המצופות | None | \_\_valY |
| הקובץ הבינארי של התוויות | LabelBinarizer() | \_\_lb |
| רשימה שתחזיק את המערכים שייצגו את התמונות  **הסבר הצורך:**  הפונקציה loading\_Images(dataset\_path) שבה מתבצעת טעינת התמונות תמיר את התמונות למערכים ותשמור אותם ברשימה זו. (במקביל פונקציה זו תשמור את הקטגוריה של כל תמונה ברשימת ה labels. כך עבור כל תמונה נשמור את הקטגוריה שלה.) | [ ] | \_\_data |
| רשימה שתחזיק את התוויות של כל התמונות | [ ] | \_\_labels |

כעת נעבור על הפונקציות השונות במחלקה. במחלקה זו זה ישנה פונקציה פומבית אחת המהווה את הפונקציה הראשית בשם handle\_dataset(self). פונקציה זו אחראית על ניהול העלאת הנתונים באמצעות הפונקציות הפרטיות העומדות לרשותה.

כעת נעבור על הפונקציות השונות בקובץ זה:

|  |  |
| --- | --- |
| **תפקידה** | **הפונקציה** |
| הפונקציה הפומבית היחידה של המחלקה. ניתן לקרוא לפונקציה זו רק מתוך אובייקט מאותחל של המחלקה (לכן יש בנאי עם פרמטרים).  ניתן להפעיל מתוך אובייקט של המחלקה את הפונקציה הזו בלבד והיא זו שדואגת להעלאת הנתונים-> בעזרת הפונקציות הפרטיות. | handle\_dataset(self) |
| פונקציה פרטית זו אחראית על טעינת כל התמונות מן מאגר התמונות הממוין.  **אופן המימוש:**  פונקציה זו קובעת את גודלי ממדי התמונה לגודל אחיד שנקבע בתכונה IMAGE\_SIZE, ממירה את התמונות מ RGB ל GrayScale ואף ממירה אותן למערכים (המחזיקים את הפיקסלים של התמונות).  בנוסף פונקציה זו מחלצת מן הנתיב של כל תמונה את הקטגוריה שלה ומתאימה בין התמונה לקטגוריה באמצעות שתי רשימות המהוות חלק מתכונות המחלקה.  רשימה אחת מאכלסת את מערכי התמונות (\_\_data) והשנייה את התוויות (\_\_labels). מכיוון שההוספה לרשימות נעשית ביחד בלולאה ישנה התאמה במספר האינדקס בין שתי הרשימות ובכך ניתן לקשר בין כל תמונה לקטגוריה שלה.  פונקציה זו אינה מחזירה ערך שכן השינוי נעשה על ה- data members של המחלקה (על תכונות האובייקט). | \_\_loading\_Images(self) |
| פונקציה פרטית זו פונקציה זו שומרת את רשימת התוויות בקובץ בינארי אשר ישמש באימון המודל וכן בחיזוי קטגוריות של תמונות בהמשך. כמו כן, היא מקטינה את טווח הפיקסלים של המערכים השמורים ב \_\_data מן התחום [0,255] לתחום [0, 1]. | \_\_scale\_Pixels(self) |
| פונקציה זו מחלקת את ה-data set הממוין ל-Train, Test, Validation ביחס של 70:20:10 כלומר 70% מן התמונות ישמשו לאימון המודל, 10% מן התמונות ישמשו לבדיקת אחוז ההצלחה של המודל תוך כדי אימון המודל, 20% מן התמונות ישמשו לבדיקת אחוז ההצלחה של המודל בעת תהליך ה-Test. עושה את החלוקה באמצעות הפקודה train\_test\_split.  כמו כן, מבצעת טרנספורמציה של קובץ התוויות הבינארי \_\_labels. | \_\_divide\_data(self) |

**קובץ בדיקת הקלט –check\_input.py:**

קובץ זה כולל שתי פונקציות המשמשות לבדיקת תקינות הקלט בשני מקרים:

1) כאשר יש להקליד 1 או 2.

2) כאשר יש להקליד מספר שלם.

תדפיס של הקוד:

"""

this python file responsible for checking an input - if it valid according to the instructions.

@author: Maayan Eliya

"""

# import the necessary package

import prints\_types

def check\_1or2(num):

"""

This function checks if the input is 1 or 2. If not it asks for another input.

:param num: string

:return: num #a valid input

:rtype: int (1/2)

"""

while True:

if num == '1' or num == '2':

num = int(num)

return num

break

else:

prints\_types.printError("[Error] This is not a valid input.")

prints\_types.printInput("Please enter 1 or 2 according to what it said before: ")

num = input ("Enter: ")

continue

def check\_int(num):

"""

This function checks if the input is an integer number. If not it asks for another input.

:param num: string

:return: num #a valid input

:rtype: int

"""

while True:

if num.isnumeric() == True:

num = int(num)

return num

break

else:

prints\_types.printError("[Error] This is not a valid input.")

prints\_types.printInput("Please enter an integer number: ")

num = input ("Enter: ")

continue

**Case Train**

כאמור, מקרה זה אחראי על אימון המודל.

הפונקציה \_\_case\_Train(self) אשר בקובץ ה tk\_inter.py קולטת מן המשתמש:

1. The model directory – הנתיב למודל לאחר שיהיה מאומן
2. The binary labels file directory – הנתיב לקובץ התוויות הבינארי
3. A folder directory to save their graphs images – הנתיב לתיקייה שבה ישמרו קבצי הגרפים שיתקבלו מתהליך ה-Train.

הפונקציה תמשיך לדרוש קלט מן המשתמש במידה והזין directory הזהה לאחד מן ה- directories הקודמים שנתן כקלט.

לאחר מכן הפונקציה הזו יוצרת אובייקט מסוג TrainModel ומזמנת את המטודה handle\_train(), ושומרת את המודל המאומן בתכונה self.\_\_model של המחלקה TkInter.

**קובץ הגדרת מבנה המודל - the\_model.py:**

קובץ זה כולל מחלקה בשם Model אשר מגדירה את מבנה המודל שעל גביו נבצע את למידת התמונות.

במחלקה זו ישנה מטודה סטטית אחת בשם build.

מטודה זו מקבלת את ממדי התמונות, את מספר הקטגוריות ואת ה-activation function שתתבצע בשכבה האחרונה – (באופן דיפולטיבי softmax function) ובהתאם תבנה את שכבות המודל.

שכבות המודל מהוות יחדיו את המוח המלאכותי אותו יצרנו. בכל שכבה ישנו מספר שונה של נוירונים השקולים לתאי העצב במוח האנושי. במוח האדם תאי העצב מחוברים לתאי העצב הסמוכים אליהם באמצעות שלוחות התא. הקישוריות הזאת בין התאים יוצרת רשת אחת גדולה ומסועפת שמהווה את המוח האנושי. השלוחות שבין התאים מעבירות אותו חשמליים שמשמשים להעברת מידע בין התאים. תקשורת זו היא שמאפשרת לאדם לחשוב ולפעול. באופן דומה בנוי המודל המלאכותי. במודל ישנן מספר שכבות אשר בכל אחת מהן מספר שונה של נוירונים מלאכותיים. הנוירונים בכל שכבה יכולים "לתקשר" עם הנוירונים שבשכבות הסמוכות וכך נוצרת רשת נוירונית. אופן התקשורת בין השכבות נעשה על סמך משקלים שמהווים חישוב סטטיסטי אשר יניב את אחוז ההצלחה הגבוה ביותר.

בעת זימון פונקציה זו ערכי המשקלים הללו הינו רנדומלי ובעת אימון המודל הם משתנים בהתאם למערכי התמונות שרצים על גביהם.

בחרתי במודל זה לאחר ניסיונות רבים, כאשר לאחר הרצת התמונות על גבי המודלים השונים הגעתי למסקנה שמודל זה הניב את אחוז הצלחה הגבוה ביותר.

השכבה הראשונה היא שכבת הקלט המגדירה את ממדי התמונה- input\_shape בפרויקט שלי (28,28,1), והשכבה האחרונה היא שכבת הפלט. חשוב לציין שבשכבה אחת לפניה הגדרנו את מספר הקטגוריות האפשרויות – מספר נוירוניי הפלט.

השכבה האחרונה באופן דיפולטיבי (במידה ואין קלט אחר) משתמשת בפונקציית softmax.

כאשר מריצים תמונה על גבי המודל המאומן היא צפויה להגיע אל אחד מנוירוני הפלט המייצגים את הקטגוריות השונות. מכיוון שמדובר בחישובים כאלו ואחרים התשובה אינה חד משמעית, ולכן התוכנית תבחר באפשרות עבורה מתקבל הסיכוי הגבוה ביותר להיות נכון, כך היא בוחרת את נוירון הפלט אשר הינו המתאים ביותר עבור אותה התמונה.

המודל:

"""

this file contains a class with a static method

this method declared the model layers

@author: Maayan Eliya

"""

# import the necessary package

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 Model():

@staticmethod

def build(width, height, depth, classes, finalAct="softmax"):

"""

This function creates the Model.

# Get the imaged dims

:param width: int

:param height: int

:param depth: int

# Get the number of the categories

:param classes: int

# Get the final activation function - by defual "softmax"

:param finalAct: string

:return: None

"""

#It initialize the model with the input shape to be "channels last" -> update the input shape and channels dimension

model = Sequential()

inputShape = (height, width, depth)

chanDim = -1

# if we are using "channels first", update the input shape and channels dimension

if K.image\_data\_format() == "channels\_first":

inputShape = (depth, height, width)

chanDim = 1

# CONV => RELU => POOL

model.add(Conv2D(32, (3, 3), padding="same", input\_shape=inputShape, activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(32, (3, 3), padding="same", input\_shape=inputShape, activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.15))

# (CONV => RELU) \* 2 => POOL

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3, 3), padding="same", activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

# (CONV => RELU) \* 2 => POOL

model.add(Conv2D(128, (3, 3), padding="same", activation="relu"))

#model.add(Dropout(0.25))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(256, (3, 3), padding="same", activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

# (CONV => RELU) \* 2 => POOL

model.add(Conv2D(256, (3, 3), padding="same", activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(512, (3, 3), padding="same", activation="relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

# first (and only) set of FC => RELU layers

model.add(Flatten())

model.add(Dense(1024))

model.add(Activation("relu"))

model.add(BatchNormalization())

model.add(Dropout(0.5))

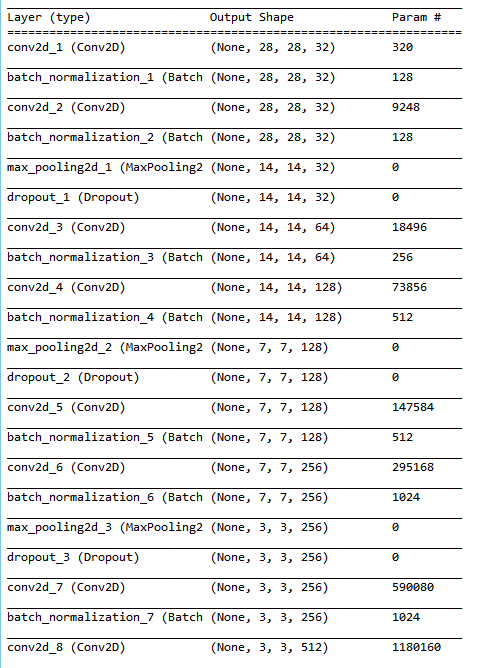
# softmax classifier

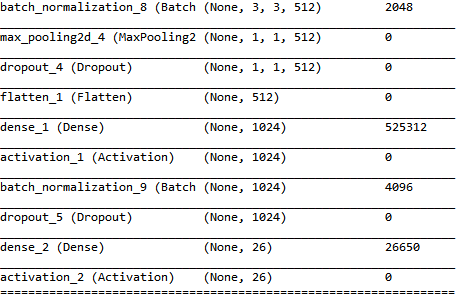
model.add(Dense(classes))

model.add(Activation(finalAct))

# return the constructed network architecture

return model

**שכבות המודל**

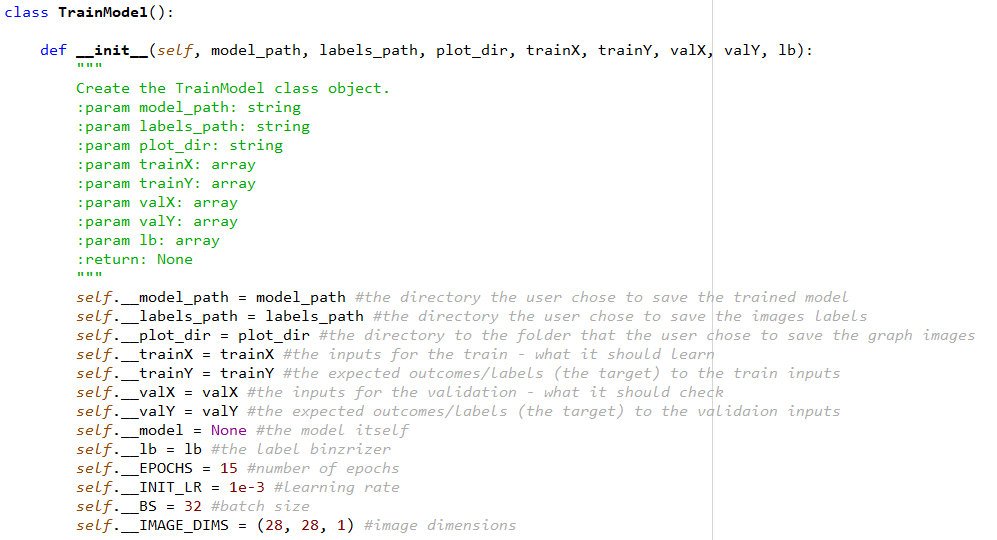


**קובץ אימון המודל – train\_model.py :**

כפי שציינתי קודם לכן קובץ זה אחראי על אימון המודל. בקובץ זה ישנה מחלקה בשם TrainModel אשר בה מטודה אחת פומבית המשתמשת במספר מטודות פרטיות במטרה לבצע את אימון המודל. הצורך במחלקה נבע מכך ששלל הפונקציות הנדרשות זקוקות לפרמטרים זהים - > דבר המוביל ליצירת מחלקה.

ראשית כל נעבור על בנאי המחלקה:

כל ה data members הם הרשאת private.



|  |  |  |
| --- | --- | --- |
| **תפקיד** | **ערך** | **המשתנה** |
| הנתיב למיקום שבו בחר המשתמש לשמור את המודל. | Input | \_\_model\_path |
| הנתיב למיקום שבו בחר המשתמש לשמור את תוויות התמונות. | Input | labels\_path\_\_ |
| התמונות שהמודל צריך ללמוד במהלך ה-Train | trainX | \_\_trainX |
| התוויות של התמונות שהמודל צריך ללמוד – המטרות המצופות | trainY | \_\_trainY |
| התמונות שהמודל צריך לחזות במהלך ה-Validation | valX | \_\_valX |
| התוויות של התמונות שהמודל צריך לחזות במהלך ה-Validation – המטרות המצופות | valY | \_\_valY |
| המודל עצמו | None | \_\_model |
| הקובץ הבינארי של התוויות | lb | \_\_lb |
| מספר הפעמים שנריץ את כל התמונות על גבי המודל לצורך ביצוע האימון (הלמידה) | 10 | \_\_EPOCHS |
| ה-learning rate | 1\*e - 3 | \_\_INIT\_LR |
| ה-batch size - מספר התמונות שבזמן אימון (לימוד) המודל יעבד במקביל | 32 | \_\_BS |
| ממדי התמונות שהמודל יריץ  (widght, height, color) | (28, 28, 1) | \_\_IMAGE\_DIMS |

כעת נעבור על הפונקציות השונות במחלקה. במחלקה זו זה ישנה פונקציה פומבית אחת המהווה את הפונקציה הראשית בשם handle\_ train(self). פונקציה זו אחראית על ניהול אימון המודל באמצעות הפונקציות הפרטיות העומדות לרשותה.

כעת נעבור על הפונקציות השונות בקובץ זה:

|  |  |
| --- | --- |
| **תפקידה** | **הפונקציה** |
| הפונקציה הפומבית היחידה של המחלקה. ניתן לקרוא לפונקציה זו רק מתוך אובייקט מאותחל של המחלקה (לכן יש בנאי עם פרמטרים).  ניתן להפעיל מתוך אובייקט של המחלקה את הפונקציה הזו בלבד והיא זו שדואגת לאימון המודל -> בעזרת הפונקציות הפרטיות. | handle\_train(self) |
| פונקציה פרטית זו מבצעת את אימון המודל בפועל. באמצעותה המודל לומד לזהות את האותיות הקטנות באנגלית בכתב יד ממערך התמונות ששייכות לאימון.  אופן המימוש:  ראשית, הפונקציה בונה את מחולל התמונות שמשתמש להגדלת נתונים - הוא מאפשר להגדיל את התמונות בזמן אמת תוך כדי שהמודל מתאמן.  שנית, הפונקציה בונה את המודל – יוצרת אובייקט של המחלקה Model שנמצאת בקובץ the\_model.py.  לאחר מכן הפונקציה מגדירה אופטימייזר – Adam שמיישם את האלגוריתם של Adam.  הפונקציה מגדירה את המודל לאימון על ידי הפקודה compile ולאחר מכן מדפיסה את ה-summery של הרשת.  לבסוף, הפונקציה מאמנת את המודל באמצעות הפקודה fit.  פונקציה זו מחזירה את ההיסטוריה של לימוד המודל. | \_\_train(self) |
| פונקציה זו שומרת את המודל המאומן בקובץ במיקום שהמשתמש בחר וכן שומרת את קובץ התוויות הבינארי במיקום שהמשתמש בחר. זאת כדי שיהיה ניתן להשתמש בלמידה הנוכחית גם בהרצות הבאות של התוכנית וכן בשביל שיהיה ניתן לבצעpredict מבלי להריץ את המודל מחדש. | \_\_save\_model |
| פונקציה פרטית זו מקבלת כפרמטר את ההיסטוריה והנתיב למיקום בו המשתמש בחר לשמור את פלט האימון.  plot\_path = \_\_plot\_dir+ r"\plot1.png"  פונקציה זו משתמשת בהיסטוריה ויוצרת קובץ תמונה מסוג png שבו היא משרטטת גרף שמתאר את השינוי ב-Loss,Accuracy במהלך אימון המודל. | \_\_graph1(self, history, plot\_path) |
| פונקציה פרטית זו מקבלת כפרמטר את ההיסטוריה והנתיב למיקום בו המשתמש בחר לשמור את פלט האימון.  plot\_path = \_\_plot\_dir+ r"\plot1.png"  פונקציה זו משתמשת בהיסטוריה ויוצרת קובץ תמונה מסוג png שבו היא משרטטת גרף שמתאר את השינוי ב-Loss,Accuracy מערכי ה-Validation מעת ביצוע אימון המודל. | \_\_graph2(self, history, plot\_path) |
| פונקציה פרטית זו מקבלת כפרמטר את ההיסטוריה והנתיב למיקום בו המשתמש בחר לשמור את פלט האימון.  plot\_path = \_\_plot\_dir+ r"\plot1.png"  פונקציה זו משתמשת בהיסטוריה ויוצרת קובץ תמונה מסוג png שבו היא משרטטת גרף שמתאר את השינוי ב-Loss,Accuracy גם מערכי ה-Train וגם מערכי ה-Validation מעת ביצוע אימון המודל. | \_\_graph3(self, history, plot\_path) |

"""

this python file handle the train section

@author: Maayan Eliya

"""

# import the necessary packages

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import Adam

from the\_model import Model

import matplotlib.pyplot as plt

import numpy as np

import pickle

import prints\_types

class TrainModel():

def \_\_init\_\_(self, model\_path, labels\_path, plot\_dir, trainX, trainY, valX, valY, lb):

"""

Create the TrainModel class object.

:param model\_path: string

:param labels\_path: string

:param plot\_dir: string

:param trainX: array

:param trainY: array

:param valX: array

:param valY: array

:param lb: array

:return: None

"""

self.\_\_model\_path = model\_path #the directory the user chose to save the trained model

self.\_\_labels\_path = labels\_path #the directory the user chose to save the images labels

self.\_\_plot\_dir = plot\_dir #the directory to the folder that the user chose to save the graph images

self.\_\_trainX = trainX #the inputs for the train - what it should learn

self.\_\_trainY = trainY #the expected outcomes/labels (the target) to the train inputs

self.\_\_valX = valX #the inputs for the validation - what it should check

self.\_\_valY = valY #the expected outcomes/labels (the target) to the validaion inputs

self.\_\_model = None #the model itself

self.\_\_lb = lb #the label binzrizer

self.\_\_EPOCHS = 15 #number of epochs

self.\_\_INIT\_LR = 1e-3 #learning rate

self.\_\_BS = 32 #batch size

self.\_\_IMAGE\_DIMS = (28, 28, 1) #image dimensions

def handle\_train(self):

"""

this function manage the TrainModel class - train section

return the train model path and the images labels path

:param: None

:return: self.\_\_model

:rtype: file

"""

history = self.\_\_train() #do the training process

self.\_\_save\_model() #save the model in the path the user chose

self.\_\_graph1(history, self.\_\_plot\_dir+ r"\plot1.png") #graph that shows the Training Loss and Accuracy

self.\_\_graph2(history, self.\_\_plot\_dir +r"\plot2.png") #graph that shows the Validation Loss and Accuracy

self.\_\_graph3(history, self.\_\_plot\_dir +r"\plot3.png") #graph that shows the Training and Validation togheter

return (self.\_\_model) #return the trained model

def \_\_train(self):

"""

This function responsible for training the model.

The model learn from the train dataset to identify the Handwriting English Lowercase Letters.

:param: None

:return: H #the history

:rtype: History object

"""

# construct the image generator for data augmentation - it allow to augment the images in real-time while the model is still training

aug = ImageDataGenerator(rotation\_range=25, width\_shift\_range=0.1,

height\_shift\_range=0.1, shear\_range=0.2, zoom\_range=0.2,

fill\_mode="nearest")

# initialize the model

prints\_types.printProcess("[INFO] Compiling model...")

self.\_\_model = Model.build(width=self.\_\_IMAGE\_DIMS[1], height=self.\_\_IMAGE\_DIMS[0],

depth= self.\_\_IMAGE\_DIMS[2], classes=len(self.\_\_lb.classes\_)) #Make an object of the Model class

# train the network

opt = Adam(lr=self.\_\_INIT\_LR, decay=self.\_\_INIT\_LR / self.\_\_EPOCHS) #the optimizer to the model - Adam

self.\_\_model.compile(loss="categorical\_crossentropy", optimizer=opt, metrics=["accuracy"]) #configures the model for the training

self.\_\_model.summary() #Prints a string summary of the network.

prints\_types.printProcess("[INFO] Training network...")

#Fit model on training data

H = self.\_\_model.fit(

aug.flow(self.\_\_trainX, self.\_\_trainY, batch\_size=self.\_\_BS),

validation\_data=(self.\_\_valX, self.\_\_valY),

steps\_per\_epoch=len(self.\_\_trainX) // self.\_\_BS,

epochs=self.\_\_EPOCHS, verbose=1)

return H #return the training process

def \_\_save\_model(self):

"""

This function responsible for saving the trained model and the label binarizer on the directories the user chose.

:param: None

:return: None

"""

prints\_types.printProcess("[INFO] Serializing network...")

self.\_\_model.save(self.\_\_model\_path) #save the model

prints\_types.printProcess("[INFO] Serializing label binarizer...")

f = open(self.\_\_labels\_path, "wb")

f.write(pickle.dumps(self.\_\_lb)) # save the label binarizer

f.close()

def \_\_graph1(self,H, plot\_path):

"""

This function creates a graph that represents the Training Loss and Accuracy,

and save an images that contains the graph on the path the user chose.

:param H: History object

:param plot\_path: string

:return: None

"""

plt.style.use("ggplot")

plt.figure()

N = self.\_\_EPOCHS

plt.plot(np.arange(0, N), H.history["loss"], label="train\_loss") #plot the training loss

plt.plot(np.arange(0, N), H.history["accuracy"], label="train\_accuracy") #plot the training accuracy

plt.title("Training Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="upper left")

plt.savefig(plot\_path) #save the praph to file on the path that the user chose to save the results

plt.show() #show the graph on the screen

plt.close()

def \_\_graph2(self,H, plot\_path):

"""

This function creates a graph that represents the Validation Loss and Accuracy,

and save an images that contains the graph on the path the user chose.

:param H: History object

:param plot\_path: string

:return: None

"""

plt.style.use("ggplot")

plt.figure()

N = self.\_\_EPOCHS

plt.plot(np.arange(0, N), H.history["val\_loss"], label="val\_loss") #plot the validation loss

plt.plot(np.arange(0, N), H.history["val\_accuracy"], label="val\_accuracy") #plot the validation accuracy

plt.title("Validation Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="upper left")

plt.savefig(plot\_path) #save the praph to file on the path that the user chose to save the results

plt.show()

plt.close()

def \_\_graph3(self, history, plot\_path):

"""

This function creates a graph that represents the Training and Validation Loss and Accuracy,

and save an images that contains the graph on the path the user chose.

:param H: History object

:param plot\_path: string

:return: None

"""

# plot loss

plt.subplot(211)

plt.title('Model Loss')

plt.plot(history.history['loss'], color='green', label='train') #plot the training loss

plt.plot(history.history['val\_loss'], color='orange', label='validation') #plot the validation loss

plt.xlabel("Epoch #")

plt.ylabel("Loss")

plt.legend(loc="upper right")

# plot accuracy

plt.subplot(212)

plt.title('Model Accuracy')

plt.plot(history.history['accuracy'], color='green', label='train') #plot the training accuracy

plt.plot(history.history['val\_accuracy'], color='orange', label='validation') #plot the validation accuracy

plt.xlabel("Epoch #")

plt.ylabel("Accuracy")

plt.legend(loc="upper right")

plt.savefig(plot\_path) #save the praph to file on the path that the user chose to save the results

plt.show() #show the graph on the screen

plt.close()

**Case last model**

כאמור, מקרה זה אחראי על העלאת המודל המאומן השמור במחשב.

אם המשתמש בחר להשתמש במודל המעודכן האחרון המאומן, יש זימון לפונקציה זו.

**אופן ביצוע:**

פונקציה זו מעלה את המודל המאומן ששמור בנתיב שנמצא בתכונה self.\_\_model\_path על ידי שימוש בפקודה keras.models.load\_model.

**Case Test**

כאמור, מקרה זה אחראי על בחינת המודל (ביצוע Test). פונקציה זו אחראית על הדפסת ערכי ה-accracy,loss.

**אופן ביצוע:**

פונקציה זו בוחנת את המודל המאומן על ידי שימוש בקובץ test\_model.

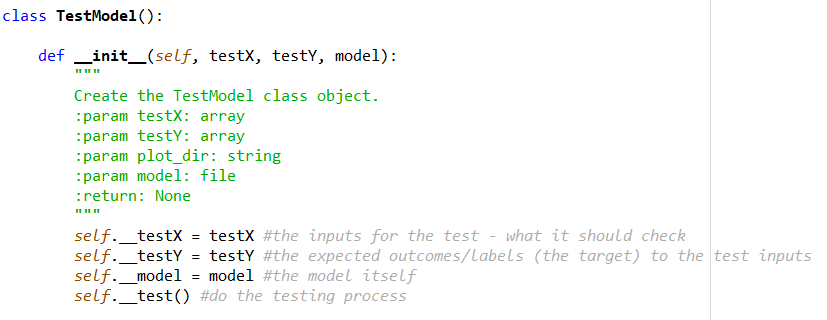
לאחר מכן הפונקציה הזו יוצרת אובייקט מסוג TestModel שדרכו מתבצע תהליך ה-Test.

**קובץ בחינת המודל –test\_model.py :**

כפי שציינתי קודם לכן קובץ זה אחראי על בחינת המודל. בקובץ זה ישנה מחלקה בשם TestModel.

ראשית כל נעבור על בנאי המחלקה:

כל ה data members הם הרשאת private.



|  |  |  |
| --- | --- | --- |
| **תפקיד** | **ערך** | **המשתנה** |
| התמונות שהמודל צריך לחזות במהלך ה-Test | testX | \_\_testX |
| התוויות של התמונות שהמודל צריך לחזות במהלך ה-Test – המטרות המצופות | testY | \_\_testY |
| המודל עצמו | model | \_\_model |

כעת נעבור על הפונקציה הקיימת במחלקה בקובץ זה:

|  |  |
| --- | --- |
| **תפקידה** | **הפונקציה** |
| פונקציה פרטית זו אחראית על בחינת המודל בפועל.  אופן המימוש:  הפונקציה מבצעת בחינה של המודל באמצעות הפקודה evaluate שבוחנת את אחוזי ההצלחה של המודל בכך שהוא מנסה לזהות תמונות חדשות שלא למד קודם לכן ומציג את אחוזי ההצלחה שלו.  בסופו של דבר הפונקציה מדפיסה את אחוזי ההצלחה של המודל – Loss, Accuracy. | \_\_test(self) |

"""

this python file handle the test section

@author: Maayan Eliya

"""

# import the necessary package

import prints\_types

class TestModel():

def \_\_init\_\_(self, testX, testY, model):

"""

Create the TestModel class object.

:param testX: array

:param testY: array

:param plot\_dir: string

:param model: file

:return: None

"""

self.\_\_testX = testX #the inputs for the test - what it should check

self.\_\_testY = testY #the expected outcomes/labels (the target) to the test inputs

self.\_\_model = model #the model itself

self.\_\_test() #do the testing process

def \_\_test(self):

"""

This function responsible for testing the model.

it evaluate the model on the test data using `evaluate` - check the trained model.

:param: None

:return: None

"""

prints\_types.printProcess('\n# Evaluate on test data')

results = self.\_\_model.evaluate(self.\_\_testX, self.\_\_testY, batch\_size=32) #the results of the testing

print('test loss ' + str(results[0]) + ' , test acc ' + str(results[1])) #print the results

**Case Predict Specific Image**

כאמור, מקרה זה אחראי על חיזוי של תמונה ספציפית. אם המשתמש בחר לחזות תמונה מסוימת שהוא בוחר, יש זימון לפונקציה זו.

אופן ביצוע:

פונקציה זו קולטת מהמשתמש נתיב לתמונה לה הוא רוצה לבצע חיזוי. הפונקציה בודקת שהנתיב אכן קיים באמצעות שימוש בקובץ check\_directory.

הפונקציה שואלת את המשתמש אם התמונה הינה בפורמט של הדאטאסט – כלומר אות לבנה על רקע שחור או להפך, על מנת לדעת אם לבצע היפוך צבעים.

הפונקציה מבצעת חיזוי של תמונה לפי למודל המאומן בכך שהיא יוצרת אובייקט מסוג ImagePredictor וממנו מזמנת את המטודה handle\_classify() מהקובץ classify.py.

**Case Predict Random Images**

כאמור, מקרה זה אחראי על חיזוי של תמונות אקראיות. אם המשתמש בחר לחזות תמונות שנבחרות באופן אקראי לפיה כמות שהוא בוחר, יש זימון לפונקציה זו.

אופן ביצוע:

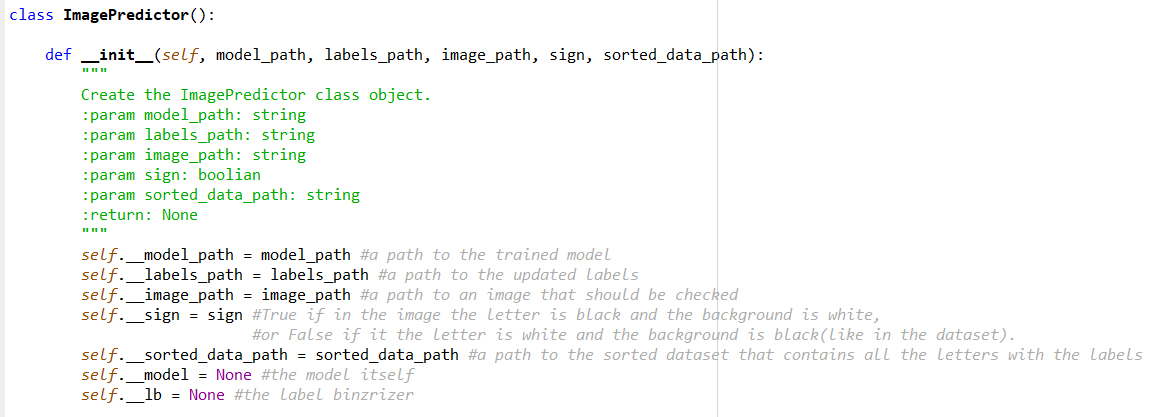
פונקציה זו קולטת מהמשתמש מספר המייצג את הכמות של תמונות מכל מאגר שהוא רוצה לבצע חיזוי. היא בודקת שאכן הוקלד מספר שלם על ידי הקובץ check\_input. הפונקציה מבצעת חיזוי של תמונה לפי למודל המאומן על ידי קריאה לפונקציה \_\_checkPredict(self, num, dataset, sign) שמבצעת חיזוי של תמונה לפי למודל המאומן בכך שהיא יוצרת אובייקט מסוג ImagePredictor וממנו מזמנת את המטודה handle\_classify() מהקובץ classify.py.

**קובץ ביצוע ה Predict – classify.py:**

כפי שציינתי קודם לכן קובץ זה אחראי על ביצוע חיזוי של תמונה – בהינתן תמונה הקובץ אחראי לבצע חיזוי של איזו קטגוריה התמונה שייכת. בקובץ זה ישנה מחלקה בשם ImagePredictor אשר בה מטודה אחת פומבית המשתמשת במספר מטודות פרטיות במטרה לבצע את חיזוי התמונה.

ראשית כל נעבור על בנאי המחלקה:

כל ה data members הם הרשאת private.



|  |  |  |
| --- | --- | --- |
| **תפקיד** | **ערך** | **המשתנה** |
| הנתיב למיקום שבו בחר המשתמש לשמור את המודל. | Input | \_\_model\_path |
| הנתיב למיקום שבו בחר המשתמש לשמור את תוויות התמונות. | Input | labels\_path\_\_ |
| מכיל את הנתיב למיקום של התמונה שיש לחזות. | image\_path | image\_path\_\_ |
| מכיל אמת אם בתמונה האות שחורה על רקע לבן.  מכיל שקר אם בתמונה האות לבנה על רקע שחור. | sign | \_\_sign |
| הנתיב למאגר התמונות הממוין עם התמונות לפי הקטגוריות. | sorted\_data\_path | \_\_sorted\_data\_path |
| המודל עצמו. | None | model\_\_ |
| הקובץ הבינארי השמור. | None | \_\_lb |

כעת נעבור על הפונקציות השונות במחלקה. במחלקה זו זה ישנה פונקציה פומבית אחת המהווה את הפונקציה הראשית בשם handle\_classify(self). פונקציה זו אחראית על ניהול חיזוי התמונה באמצעות הפונקציות הפרטיות העומדות לרשותה.

כעת נעבור על הפונקציות של המחלקה אשר ממשות יחדיו את תחום האחריות עליו ממונה הקובץ.

|  |  |
| --- | --- |
| **תפקידה** | **פונקציה** |
| הפונקציה הפומבית היחידה של המחלקה. . ניתן לקרוא לפונקציה זו רק מתוך אובייקט מאותחל של המחלקה (לכן יש בנאי עם פרמטרים).  ניתן להפעיל מתוך אובייקט של המחלקה את הפונקציה הזו בלבד והיא זו שדואגת לחיזוי התמונה -> בעזרת הפונקציות הפרטיות.  תחילה מזמנת את הפונקציה האחראית להעלאת המודל, לאחר מכן מזמנת את הפונקציה האחראית להעלאת התמונה, ולבסוף מזמנת את הפונקציה האחראית לזיהוי התמונה. | handle\_classify(self) |
| פונקציה פרטית שטוענת את המודל השמור המאומן וקוראת את הקובץ הבינארי – כך היא מאתחלת את תכונות המחלקה: \_\_model , \_\_lb. | \_\_load\_Model(self) |
| מטודה פרטית זו טוענת את התמונה שיש לחזות – היא מתאימה את התמונה שבנתיב שנשלח למחלקה ((\_\_image\_path לפורמט של התמונות שנלמדו על ידי המודל – כמערך, grayscale, בממד (28,28,1), מנרמלת את התמונה.  אם התמונה היא כשהאות שחורה והרקע לבן, הפונקציה מבצעת היפוך צבעים כדי להתאים את התמונה לפורמט של המאגר שהמודל למד ממנו.  הפונקציה פועלת באותו אופן שבוצע על התמונות לפני האימון:  ממירה את התמונה מ RGB ל GrayScale, משנה את ממדי התמונה לגודל אחיד שקבעתי (28,28,1), לאחר מכן ממירה את התמונה למערך ומבצעת נרמול - מקטינה את טווח הפיקסלים שלה מ [0,255] ל [0,1].  לבסוף המטודה מחזירה את המערך המייצג את התמונה. | load\_Image(self)\_\_ |
| פונקציה פרטית זו אחראית על ביצוע החיזוי עצמו. היא מקבלת כקלט את הפלט של הפונקציה הפרטית \_\_load\_image(): המערך שמייצג את התמונה  המטודה מבצעת חיזוי של האות שנמצאת בתמונה לפי המודל שנטען ומדפיסה את הזיהוי של המודל, את אחוז הסיכוי שהמודל זיהה נכון את התמונה לפי החישובים שלו והאם הזיהוי נכון או לא נכון  אופן הביצוע:  חיזוי התמונה נעשה על ידי הפקודה predict. פקודה זו מחזירה את ההסתברות של המודל לזיהוי של התמונה בהתאם לכל קטגוריה, והתווית שעבורה התקבל הערך המקסימלי היא התווית שהמודל מזהה לתמונה.  כדי לבדוק אם המודל צודק, נעשית השוואה בין התווית של התמונה שמצויה בשם הקובץ – האות האמיתית שמצויה בתמונה, לבין התווית שהמודל זיהה – האות שהמודל זיהה בתמונה. אם התוויות זהות המודל צדק, ואם לא הוא טעה. (מתבצע בעזרת הקובץ הבינארי של התוויות שנשמר בעת אימון המודל) | \_\_predict\_Image(image\_arr) |

"""

this python file handle the classify section

@author: Maayan Eliya

"""

# import the necessary packages

from keras.preprocessing.image import img\_to\_array

from keras.models import load\_model

import numpy as np

import pickle

import os

from PIL import Image

import PIL.ImageOps

from matplotlib import image

from imutils import paths

import prints\_types

IMAGE\_SIZE = (28,28)

class ImagePredictor():

def \_\_init\_\_(self, model\_path, labels\_path, image\_path, sign, sorted\_data\_path):

"""

Create the ImagePredictor class object.

:param model\_path: string

:param labels\_path: string

:param image\_path: string

:param sign: boolian

:param sorted\_data\_path: string

:return: None

"""

self.\_\_model\_path = model\_path #a path to the trained model

self.\_\_labels\_path = labels\_path #a path to the updated labels

self.\_\_image\_path = image\_path #a path to an image that should be checked

self.\_\_sign = sign #True if in the image the letter is black and the background is white,

#or False if it the letter is white and the background is black(like in the dataset).

self.\_\_sorted\_data\_path = sorted\_data\_path #a path to the sorted dataset that contains all the letters with the labels

self.\_\_model = None #the model itself

self.\_\_lb = None #the label binzrizer

def handle\_classify(self):

"""

this function manage the ImagePredictor (classify) class

:param: None

:return: None

"""

self.\_\_load\_Model() #loads the trained model

image\_arr = self.\_\_load\_Image() #load the image as array

self.\_\_predict\_Image(image\_arr) #predict the image

def \_\_load\_Model(self):

"""

This function loads the trained model and the label binarizer

:param: None

:return: None

"""

prints\_types.printProcess("[INFO] Loading network...")

self.\_\_model = load\_model(self.\_\_model\_path) #loads the trained model

self.\_\_lb = pickle.loads(open(self.\_\_labels\_path, "rb").read()) #load the label binzrizer

def \_\_load\_Image(self):

"""

this function load the image that should be checked - as an array

:param: None

:return: image\_arr

:rtype: numpy array

it returns the image as a numpy array (after fitting the image - normalization (pixels scale), #colors, #dims, )

"""

# pre-process the image for classification

image\_arr = Image.open (self.\_\_image\_path)

if self.\_\_sign == True: #if it is a new image - black letter and white background

image\_arr = PIL.ImageOps.invert(image\_arr) #make the letter white and the background black

image\_arr.save(self.\_\_image\_path) #save the inverted image

else:

image\_arr = Image.open (self.\_\_image\_path) #if the sign is false then the image is shown on the screen - if it's true then it was already shown

image\_arr.show() #show the image on the screen

image\_arr = image\_arr.convert('L') #make the image to be on grayscale

image\_arr.save(self.\_\_image\_path) #save the grayscaled image

image\_arr = image\_arr.resize((IMAGE\_SIZE)) #change the image's size to be (28,28)

image\_arr.save(self.\_\_image\_path) #save the image with the new size

image\_arr = image.imread(self.\_\_image\_path) #load the image

image\_arr = image\_arr.astype("float") / 255.0 #normalization

image\_arr = img\_to\_array(image\_arr) #convert the image to a numpy array

image\_arr = np.expand\_dims(image\_arr, axis=0) #expand the shape of the array

return image\_arr

def \_\_predict\_Image(self,image\_arr):

"""

This function predicts the letter of the image that the user pass.

It prints "correct" if the input image label is fit to the prediction label,

else we will print "incorrect".

It plots the image on the screen

:param image\_arr: numpy array

:return: None

"""

# classify the input image

prints\_types.printProcess("[INFO] Classifying image...")

Predict = self.\_\_model.predict(image\_arr)[0] #output predictions for the input image

idx = np.argmax(Predict) #the index of the max prediction value

label = self.\_\_lb.classes\_[idx] #the label of the prediction

#check if it predicted correctly

split\_pimg = os.path.split(self.\_\_image\_path) #split the path of image that should be predicted to head and tail

filename = split\_pimg[-1] #the name of image that should be predicted (the tail)

imagePaths = sorted(list(paths.list\_images(self.\_\_sorted\_data\_path))) #list of all the images paths

flag = False

for imagePath in imagePaths:

split = os.path.split(imagePath) #split the path of the image that is currently checked to head and tail

imgname = split[-1] #the name of the image that is currently checked (the tail)

if filename == imgname: # if the name of the image that should be predicted equals to the name of the image that is currently checked

flag = True #it found the image. this is an image from the dataset

split = os.path.split(split[0])

imglabel = split[-1] #the actuall label of the image

if label == imglabel: #if the label that it predicted equals to the actuall label of the image

correct = "correct"

else:

correct = "incorrect"

break

if flag == False: #if it is a new image

#check if it predicted correctly

imglabel = filename.split(".")

imglabel = imglabel[0] #imglabel = the new label of the image (the letter)

if imglabel == label: #if the label that it predicted equals to the actuall label of the image

correct = "correct"

else:

correct = "incorrect"

# print the results

result = "the letter it predicted: {}, prediction: {:.2f}%, \nthe real label of the image letter: {} \n({})".format(label, Predict[idx] \* 100, imglabel, correct)

prints\_types.printProcess("[INFO] {}".format(result))

**קובץ לבדיקת התקינות של הנתיב שהשתמש מקליד (directory) – check\_directory.py:**

מטרת קובץ זה היא לבדוק האם directory מסוים שהמשתמש הכניס הינו תקין לפי הדישה. בכל חלק בקוד שיש בו קלט שלdirectory מסוים מן המשתמש, יש שימוש בפונקציה שבקובץ זה. הקובץ כולל מחלקה בשם Directory שמכילה ארבע פונקציות סטטיות שבודקות את תקינות הקלט, ובמקרה של קלט לא תקני מודפסות הודעות שגיאה וכן יש דרישה מן המשתמש לקלט חדש עד שיוזן קלט תקין.

נעבור על הפונקציות של המחלקה המוגדרת בקובץ זה:

|  |  |
| --- | --- |
| **תפקידה** | **מטודה** |
| פונקציה פומבית וסטטית זו מבקשת מהמשתמש קלט שלdirectory שקיים במחשב.  הפונקציה מקבלת את ההודעה שיש להציג למשתמש בעת הבקשה לקלט.  אם הנתיב לא קיים, הפונקציה תבקש מהמשתמש קלט תקין כל עוד לא התקבל נתיב קיים, ולבסוף כשתקבל נתיב לקובץ קיים היא תחזיר את הקלט.  הפונקציה משתמשת בפונקציה \_\_check\_Exsists\_Dir(path) כדי לבדוק האם ה-directory קיים. | @staticmethod  is\_Exsists(message) |
| פונקציה סטטית ופרטית אשר בהינתן נתיב מסוים הפונקציה בודקת אם ה- directory קיים.  הפונקציה מחזירה True אם הכתובת הינה של נתיב קיים, אחרת תחזיר False. מבצעת את הבדיקה באמצעות הפקודה os.path.exists.  פונקציה פרטית זו מזומנת על ידי הפונקציה הפומבית is\_Exsists(message) שמזמנת אותה בלולאה עבור כל קלט של נתיב שהיא מקבלת מהמשתמש. | @staticmethod  \_\_check\_Exsists\_Dir(path) |
| פונקציה סטטית ופומבית זו משמשת לביצוע קלט מהמשתמש שלdirectory חדש תקין שאינו קיים כבר במחשב.  עבור פונקציה זו, נתיב תקין הוא כזה שמכיל רצף של תיקיות קיימות שבסופן תיקייה חדשה או שם קובץ חדש.  הפונקציה מקבלת את ההודעה שיש להציג למשתמש בעת הבקשה לקלט.  הפונקציה משתמשת בפונקציה \_\_check\_New\_Dir(path) כדי לבדוק האם ה-directory קיים. | @staticmethod  get\_New\_Dir(message) |
| פונקציה סטטית ופרטית אשר בהינתן נתיב מסוים הפונקציה בודקת אם ה- directoryהוא חדש שלא קיים כבר במחשב.  הפונקציה מחזירה True אם הכתובת הינה של נתיב שלא קיים, אחרת תחזיר False. מבצעת את הבדיקה באמצעות הפקודה os.path.exists.  פונקציה פרטית זו מזומנת על ידי הפונקציה הפומבית get\_New\_Dir(message) שמזמנת אותה בלולאה עבור כל קלט של נתיב שהיא מקבלת מהמשתמש. | @staticmethod  \_\_check\_New\_Dir(path) |

"""

this python file handle the input section - checks if it's okey

@author: Maayan Eliya

"""

# import the necessary package

import os

import prints\_types

class Directory():

@staticmethod

def is\_Exsists(message):

"""

This function request an exsisting directory from the user.

It returns the path only if cheak\_exsists\_dir() function returns true.

This function keeps request a directory path until it will be an existed path - valid according to the cheak function.

:param massage: string #A message to desplay for the user

:return: path

:rtype: string

"""

prints\_types.printInput(message)

path = input("Enter: ")

while not Directory.\_\_check\_Exsists\_Dir(path): #if the directory is not existed it request a new path

prints\_types.printInput(message)

path = input("Enter: ")

return path

@staticmethod

def \_\_check\_Exsists\_Dir(path):

"""

This function checks if the current directory exsists.

Returns true if the path is valid (exist), otherwise returns false .

:param path: string #A path that the user gave

:return: True/False

:rtype: boolian

"""

if not os.path.exists(path): #id the directory is not existed

prints\_types.printError("Error - no such file or directory")

return False

return True

@staticmethod

def get\_New\_Dir(message):

"""

This function request a new directory from the user.

It returns the path only if cheak\_new\_dir() function return true.

This function keeps request a directory until it will be not existed path - valid according to the cheak function.

:param massage: string #A message to desplay for the user

:return: path

:rtype: string

"""

path = ""

while not Directory.\_\_check\_New\_Dir(path): #if the directory is existed then it request a new path

prints\_types.printInput(message)

path = input("Enter: ")

return path

@staticmethod

def \_\_check\_New\_Dir(path):

"""

This function checks if the current directory is not exsists.

Returns true if the path is valid (new directory - not exist), otherwise returns false .

:param path: string #A path that the user gave

:return: True/False

:rtype: boolian

"""

#cheak that the path is not already exists

if(os.path.exists(path)): #if the path is already exists

prints\_types.printError("Error - this directory is already exsists")

return False

#if the path is not exists

try: #try to make a folder in the current path

#if its succeed the path is valid

os.mkdir(path) #Create a directory

os.rmdir(path) #Remove a directory

return True

except: #the path is not vaild and the function return false

if(path != ""):

prints\_types.printError("Error - directory is not valid")

return False

**קובץ הדפסת הפלטים למשתמש לפי צבעים – prints\_types.py**

קובץ זה אחראי על ביצוע ההדפסות למשתמש. ההדפסות מסווגות לשלושה סוגים וכל סוג מודפס בצבע אחר:

* הדפסות המבקשות קלט מהמשתמש – יודפסו בצבע **ירוק**
* הדפסות שמספקות אינפורמציה על המתרחש בזמן ריצה – יודפסו בצבע **כחול**
* הדפסות של הודעות שגיאה (קלטים לא תקניים) – יודפסו בצבע **אדום**

"""

this python file contains the function that prints messages for the user in colors.

@author: Maayan Eliya

"""

# import the necessary package

from colorama import init, Fore, Style

def printError(message):

"""

This function print the message in red color - represent Error.

:param massage: string #A message to desplay for the user

:return: None

"""

init(convert=True)

print(Fore.RED + message)

Style.RESET\_ALL

def printInput(message):

"""

This function print the message in green color - represent the options.

:param massage: string #A message to desplay for the user

:return: None

"""

init(convert=True)

print(Fore.GREEN + message)

Style.RESET\_ALL

def printProcess(message):

"""

This function print the message in blue color - represent the process.

:param massage: string #A message to desplay for the user

:return: None

"""

init(convert=True)

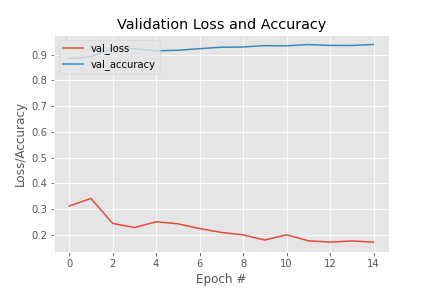
print(Fore.BLUE + message)

Style.RESET\_ALL

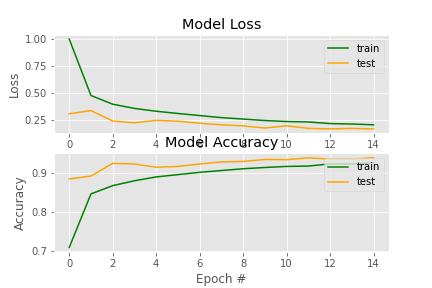
**מסקנות הרצת המודל:**

**גרפי התוצאות של הלמידה:**

גרף 1 (מתאר את תהליך הלמידה במהלך האימון)

גרף 2 (מתאר את אחוזי ההצלחה במהלך ה-validation שמדמה test במהלך האימון)

גרף 3 (מתאר את אחוזי ההצלחה בהמלך ה-train מול אחוזי ההצלחה במהלך ה-validation)

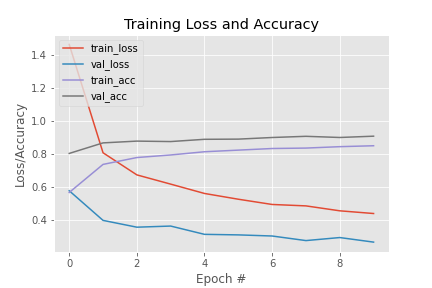


ניתן להבחין שבהרצה זו התבצעה למידה שכן בכל epoch אחוזי ההצלחה גדלים, וכן ה-validation accuracy וה-training accuracy מציגים ערכים קרובים לאורך כל התהליך, וגם ה-validation loss וה-training loss מציגים ערכים קרובים לאורך כל התהליך. הם התקדמו בכל epoch באופן דומה, כאשר אין הבדלים משמעותיים ביניהם. מכאן שתהליך הלמידה והחיזוי היו קרובים כלומר המכונה אכן למדה לזהות את הקטגוריות.

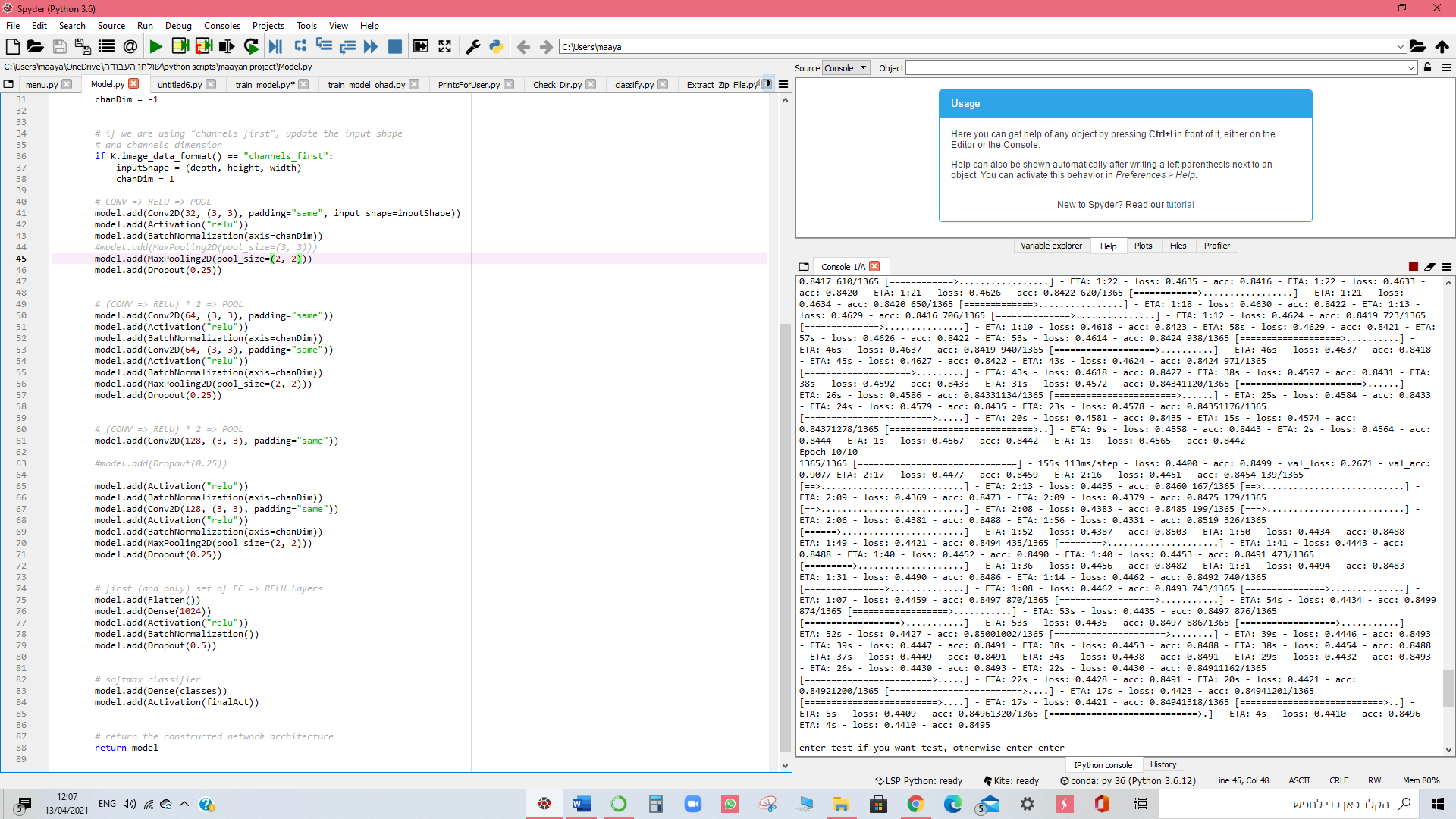
מהרצה להרצה ניסיתי לשפר את המודל באמצעים שונים, העלתי את מספר ה epoch, ניסיתי לשנות את השכבות במודל, שיניתי את ערכי ה-dropout, ניסיתי גם optimizers אחרים. בסופו של דבר הגעתי למסקנה שהמודל העדכני הוא המודל המוצלח ביותר, שמניב את תוצאות ה-accuracy הגבוהות ביותר וכן את ערכי ה-loss הנמוכים ביותר.

ההרצה המוצגת כאן הניבה תוצאות סופיות של: 93.9% validation accuracy, 93% training accuracy, 0.17 validation loss ו 0.19 training loss.



זהו למשל גרף ממודל ראשוני יותר שניסיתי להריץ (עם epochs=10):

המודל עליו מבוסס הגרף:



כפי שניתן לראות, ערכי ה-accuracy ,validation פה נמוכים ביחס למודל העדכני שלי.

ההרצה המוצגת כאן הניבה תוצאות סופיות של: 84.9% accuracy, 0.44 loss.

בכל פעם שיניתי משהו במודל ושיפרתי אותו עד שהגעתי למודל העדכני שמניב לי תוצאות טובות.

**רפלקציה/סיכום אישי:**

במהלך עשיית פרויקט הגמר שלי בהתמחות deep learning, למדתי דברים חדשים והרחבתי את הידע שלי רבות בשפת python בפרט ובמחשבים ולמידת מכונה בכלל.

לאורך כל תהליך העשייה שלי היה בי דחף ללמוד ולהגיע לתוצאה הטובה ביותר, לכן המשכתי לשפר את המודל ולנסות דברים חדשים ולבסוף הגעתי לתוצאות מעולות.

עם זאת, תהליך העבודה על הפרויקט לא היה פשוט. כפי שהסברתי במבוא, התחום הזה היה חדש לי ולא הכרתי אותו לפני תחילת העבודה. מאחורי התחום עומד רקע תיאורטי עצום וכן ידע מעשי שצריך לרכוש כדי ליצור פרויקט. כמו כן, כל חבריי להתמחות נחשפו לתחום החדש הזה השנה וזוהי השנה השנייה בלבד שבית הספר שלנו מעביר התמחות זו, לכן כל הנושא טרי אצל כולנו והיה לנו הרבה ללמוד לקראת הכנת הפרויקט. אמנם הצלחתי להתגבר על אתגרים אלו, בכך שחקרתי ולמדתי בעיקר באופן עצמאי על כל תחום למידת המכונה ו-deep learning, בעיקר ממקורות באינטרנט, ממקורות שהמורה שלחה ומסרטונים ביוטיוב. דבקתי במטרה ולמדתי רבות גם בכוחות עצמי, ואני שמחה על כך.

מעבר לכך, אני מרגישה כי העבודה על הפרויקט תרמה לי גם לשיפור אוצר המילים שלי באנגלית, שכן הרוב המוחלט של מקורות המידע בנושא זה הינם באנגלית בשפה גבוהה. העובדה שהיה עליי ללמוד את הנושא על מנת להכין פרויקט בתחום חייבה אותי ללמוד מאתרים אלו וכך לשפר את האנגלית שלי.

בעקבות חקר העבודה וכתיבתה, אני לוקחת איתי את המידע הרב שצברתי במהלך העבודה כולה, אשר נמשכה לאורך תקופה ארוכה. תחום הבינה המלאכותית ולמידת המכונה הינו תחום מרתק ומעניין מאוד, חדשני ואני מאמינה שיש עוד המון חידושים שהוא יכול לתרום בהם. על כן, אני שמחה שהייתה לי הזכות להיחשף אליו ולהכין פרויקט גמר בנושא.

כמו כן, משום שכל חבריי להתמחות נחשפו לתחום הזה לראשונה, למדנו רבות יחד ואף העברנו הרצאות במהלך השיעורים עם דינה מורתנו. אני חשה כי הדבר ליכד אותנו וגרם לכך שהייתה לנו מטרה משותפת. נוצרה עזרה ההדדית בין חברי הכיתה והיה לי כיף להיות נוכחת בכך.

לסיכום, מאחר ביצעתי את הפרויקט ולמדתי תוך כדי הכנתו, התנסיתי בגרסאות שונות כשבכל גרסה שיפרתי משהו לקראת הגרסה הסופית, אני חשה כי הצלחתי בסופו של דבר להגיע למטרה שלי ואני גאה בתוצר הסופי שלי.

**ביבליוגרפיה:**

אתרים שלמדתי מהם:

<https://www.analyticsvidhya.com/blog/2019/04/build-first-multi-label-image-classification-model-python/>

<https://www.studytonight.com/tkinter/introduction-to-python-tkinter-module>

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

<https://www.tensorflow.org/tutorials/keras/save_and_load>

<https://colab.research.google.com/drive/1NyYH1EPpaJlMBLK0fcKYz4icaD1SNSLK#scrollTo=DcyaMwrmWCk8&forceEdit=true&sandboxMode=true&uniqifier=1>

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

<https://www.pyimagesearch.com/2019/12/30/label-smoothing-with-keras-tensorflow-and-deep-learning/>

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

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

התמונות:

<https://www.kaggle.com/tomasramos21/emnist-jpeg>

ספרים:

Deep\_learning\_for\_computer\_vision

Keras tutorial

**נספחים:**

מודלים קודמים שניסיתי:

