

פרויקט גמר

ב-deep learning למידת צביעת תמונות שחור-לבן

מגיש: יהונתן טל

בית ספר: מקיף י"א ראשונים

ת.ז: 324825470

מנחה: דינה קראוס

08.06.2020

תוכן עניינים

[מבוא 3](#_Toc41584470)

[פרטים כללים על התכנה 4](#_Toc41584471)

[מדריך למשתמש 5](#_Toc41584472)

[דרישות הפעלה 5](#_Toc41584473)

[הרצה תקינה של הפרויקט 5](#_Toc41584474)

[התחלת ההרצה 5](#_Toc41584475)

[תיקיות פלט 5](#_Toc41584476)

[פלטים אפשריים 6](#_Toc41584477)

[מדריך למפתח 13](#_Toc41584478)

[קבצים 13](#_Toc41584479)

[תיאור פנימי של הקוד 14](#_Toc41584480)

[משתנים גלובלים 15](#_Toc41584481)

[גוף הקוד הראשי 16](#_Toc41584482)

[מחלקות ופעולות עזר 17](#_Toc41584483)

[פעולות עזר לבדיקת הקבצים 20](#_Toc41584484)

[פעולות בדיקת המאגר וקביעתו 22](#_Toc41584485)

[פעולות שקשורות ליצירת המודל 24](#_Toc41584486)

[פעולות וגנרטורי יבוא מידע למודל 26](#_Toc41584487)

[פעולות אימון, הערכה, ושמירת המודל 28](#_Toc41584488)

[מסקנות הרצת המודל 31](#_Toc41584489)

[סיכום אישי 33](#_Toc41584490)

[ביבליוגרפיה 35](#_Toc41584491)

[נספחים 36](#_Toc41584492)

[תרשים מסלול הרצה 36](#_Toc41584493)

[תרשים קריאות 37](#_Toc41584494)

[ייצוג ויזואלי של המודל 38](#_Toc41584495)

# מבוא

הסקריפט שפיתחתי הינו תכנה לצביעת תמונות בשחור לבן.

ספר זה הינו ספר הוראות למשתמש בתכנה. ישנו פירוט על המבנה של התכנה ומה שצריך כדי להפעיל אותה כמו שצריך. כמו גם, מה הן כל הפונקציות האפשריות בה ואיך אפשר להוציא ממנה את מלוא הפוטנציאל. בספר נמצא מדריך למשתמש וגם למפתח, לעיון של משתמשים המעוניינים להפעיל את התוכנית בהצלחה ולעיון מפתחים המעוניינים לשאוב השראה מהתוכנה. בספר נמצא גם פירוט על תהליך העבודה האישי שלי ומה הייתי צריך להשלים ולעבור כדי להגיע לתוצר הסופי, כל הקשיים וההצלחות הגדולות שלא צפיתי להן נכתבו בתמציתיות כדי לספק מענה לעוסקים בתחום המתמודדים עם אותן הבעיות.

תוכנית זו נכתבה כחלק מפרויקט הגמר במגמת מחשבים בבית הספר מקיף י"א ראשונים. נושא זה חדש בבית הספר שלנו ובכלל במערכת החינוך בארץ, לכן אני מעריך שהיה לנו קצת יותר קשה לעבוד על הפרויקט מאשר יורשינו בעתיד, שיוצעו להם מאגר עשיר יותר של חומרי לימוד נגישים ומנחים מומחים. התחלתי לכתוב את הסקריפט תחילה כתוכנה המסווגת בין תמונות לימונים על פי רמת הבשלות שלהן. בעת מחקר באינטרנט, נתקלתי בנושא צביעת התמונות והתעניינתי מאוד, החלטתי להחליף את הנושא שלי לזה ואינני מתחרט על כך כלל.

בעת המחקר שלי על הנושא, גיליתי שיש מספר תוכנות ושירותים אינטרנטיים חינמיים המציעים את השירות הזה ועושים עבודה טובה ביותר. מה גם שקיימת לאחרונה מגמת צביעת סרטים ישנים מתחילת המאה הקודמת באמצעות אותה הטכנולוגיה, וזאת בהצלחה רבה, ישנה אפילו פקודה בספרייה cv2 שעושה בדיוק את זה, היא מבוססת על אותם העקרונות של למידת מכונה.

הבעיה המרכזית שהייתה לי בעת כתיבת הפרויקט היא להתמודד עם העומס והכובד שיש להרצת התוכנית על המחשב. תהלך אימון מודל CNN הוא כבד ולוקח זמן רב, הוא דורש חישובים רבים מספור ומעמיס על המעבד (CPU) מאוד. פתרון אפשרי היה להשתמש במעבד גרפי (GPU), אך לא היה לי את הסוג המתאים לעבודה עם ספריית TensorFlow , עליה אפרט בהמשך, והדבר גלגל אותי לניסיונות שווא להתאים את המעבד הגרפי שברשותי לספרייה. לבסוף מצאתי פתרון באמצעות אתר Kaggle שמציע שירותי הרצה מרחוק לסקריפטים, בין היתר על מעבדים גרפים, בו כתבתי את רוב הסקריפט באמצעות עורך המבוסס על Jupiter Notebook, גם על כך אפרט בהמשך.

הרצת הסקריפט מתחילה בהסבר על מטרת התוכנית והפרטים הכלליים שלה, ממשיכה עם הבחירה בין מסלול הלמידה למסלול הצביעה בלבד, ונגמרת בעת סיום הצגת התוצאות. הסקריפט יכול להשתמש במאגר תמונות ברירת-מחדל שנלקח מאתר flickr ויפורט עליו בהמשך.

## פרטים כללים על התכנה

**מטרת התכנה**: ללמוד איך לצבוע תמונות שחור לבן (grayscale) באמצעות מאגר תמונות בצבע.

**בסיס תאורטי**:

* הלמידה נעשית באמצעות deep learning, שזהו סוג של Machine learning, למידת מכונה. זאת בעזרת מודלים המבוססים על רשתות עצביות מפותלות (CNN).
* נעשה שימוש רב בקוד בתבניות צבע (Color Formats/Spaces/Models) האלה RGB, RGBA, Grayscale:
  + **RGB** – red, green, blue – תבנית צבעונית המורכבת משלושה צבעים המרכיבים את שאר הצבעים. הערכים הם מספרים שלמים הנעים בין 0 ל-255.
  + **RGBA** – red, green, blue, alpha – תבנית צבעונית המורכבת משלושה צבעים ושקיפות, אלה מרכיבים את שאר הצבעים עם אפשרות לשקיפות ואטימות חלקית. הערכים הם מספרים שלמים הנעים בין 0 ל-255.
  + **Grayscale** – תבנית שחור-לבן שבה הערכים מייצגים רק בהירות. הערכים הם מספרים לא שלמים הנעים בין 0 ל-1 עם 256 ערכים אפשריים, או מספרים שלמים שנעים בין 0 ל-255.

**שפת התכנות והספריות:** שפת Python 3. הספריות אשר התבצעו בעזרתן הלמידה הן TensorFlow ו-Keras, ספריות קוד פתוח ללמידת מכונה, המפותחת על ידי חברת גוגל לבנייה ואימון רשתות עצביות (NN). נעשה שימוש במספר ספריות נוספות לעיבוד המידע והתמונות ולהצגתם, יפורט עליהן בהמשך.

**אופן ההפעלה:** המשתמש מריץ את התכנה והיא מדריכה אותו בין הפעולות השונות האפשריות באמצעות שאלות מרובות, עליהן המשתמש יכול לענות באמצעות הזנת תו או מילה.

# מדריך למשתמש

## דרישות הפעלה

* על המשתמש להשתמש בסביבת עבודה המבוססת על Python 3.
* על סביבת העבודה של המשתמש להכיל את הספריות הבאות שלא מותקנות בברירת המחדל על Python 3:
  + numpy
  + matplotlib
  + skimage
  + keras
  + tensorflow

התקנת הספריות יכולה להתבצע באמצעות pip (מתקין הספריות של פייתון) באופן הבא:

pip install library

או אם רוצים להוריד גרסה ספציפית (יתכן עם TensorFlow ו-Keras):

pip install library==version

אם מבצעים את ההתקנה דרך פלטפורמות המבוססות על Jupiter Notebook, מוסיפים סימן קריאה לפני:

!pip install library

## הרצה תקינה של הפרויקט

### התחלת ההרצה

* הרצה דרך פלטפורמות מבוססות על command line כמו command prompt תעשה בצורה הבאה:
* דרך פלטפורמות המבוססות עלJupiter Notebook אפשר פשוט להריץ על ידי לחיצה על הכפתור הרלוונטי לאחר טעינת הקוד, כמחברת או כסקריפט.

### תיקיות פלט

התוכנית שומרת את כל הקבצים שהמשתמש מעוניין בהם בתיקייה בשם OUTPUT בתוך התיקייה שבה נמצא הסקריפט.

* מודלים והצגה כתמונה של המודל, נשמרים בתת-תיקייה models
* תוצאות כתמונות, נשמרות בתת-תיקייה results

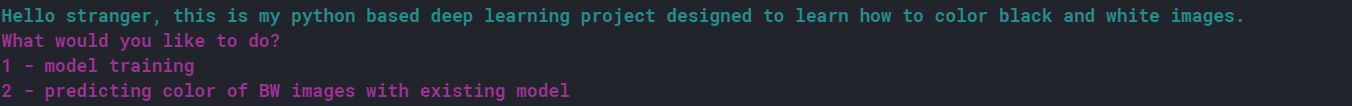
### פלטים אפשריים

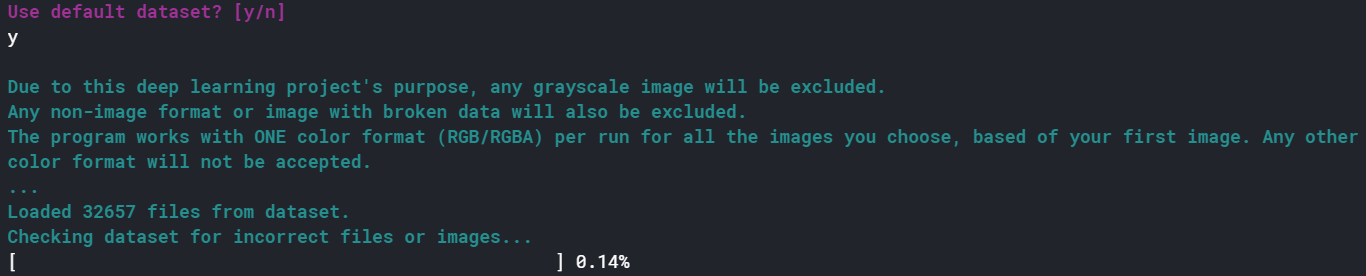
התוכנית יוצרת מספר פלטים בהתאם לתפקידם, כל אחד בצבע אחר:

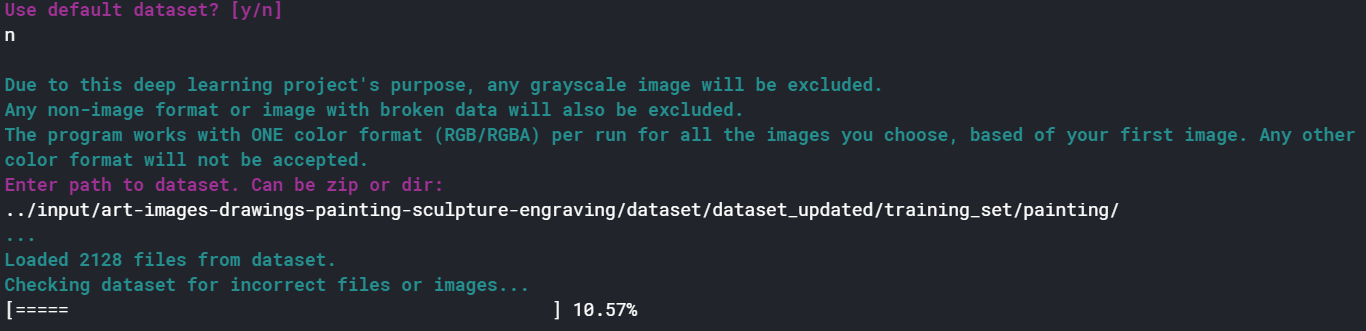
* סגול: בקשה מהמשתמש להכניס ערך מסוים.
* תכלת: מידע על תהליכי התוכנית. כלומר, יידוע המשתמש על תהליכים שהושלמו או דברים שהוא צריך לדעת לפני הפעלת פעולה כלשהי.
* אדום: אזהרה שנובעת מבעיה בקלט מהמשתמש.

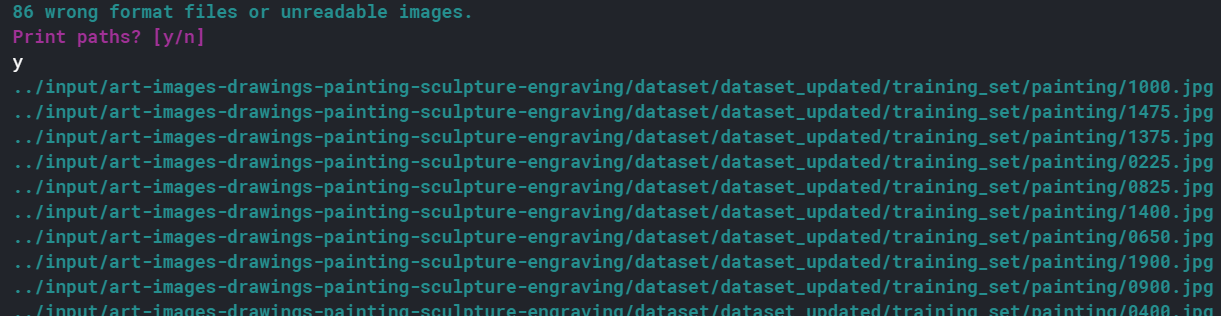
#### הרצה מבחינת המשתמש

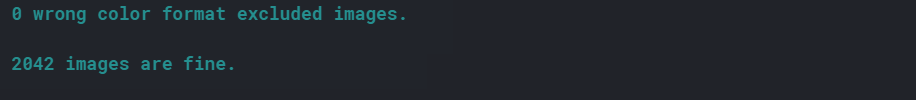
ראשית, התוכנית מבקשת לבחור מסלול הרצה:

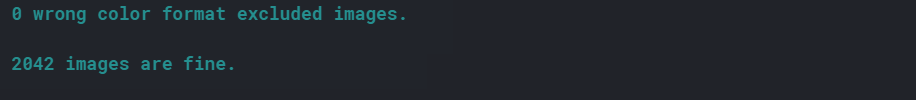
* **1 – לאימון מודל על מאגר תמונות.**
* **2 – לחזיית תמונות ממאגר לפי מודל קיים.**

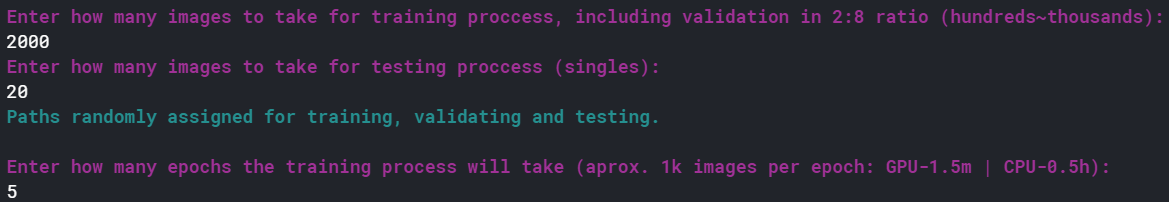
התוכנית שואלת האם להשתמש במאגר ברירת המחדל. אם כן, מציגה פרטים על דרישות התוכנית ומתחילה את הבדיקה.

אם לא, התוכנית מבקשת דרך חלופית למאגר, ומשם הכל זהה. **אם בחרנו במסלול 2:** חשוב שנכניס מאגר עם תמונות שחורות-לבנות.

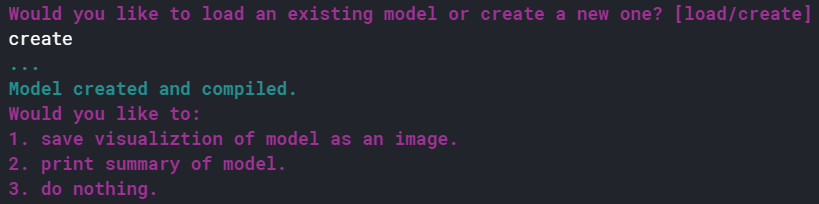
אם יש קבצים עם בעיות, לפי ההסבר בדרישות התוכנית, שואל אם להדפיס את הדרכים שלהם. אם כן, מדפיס. (תמונה חלקית)

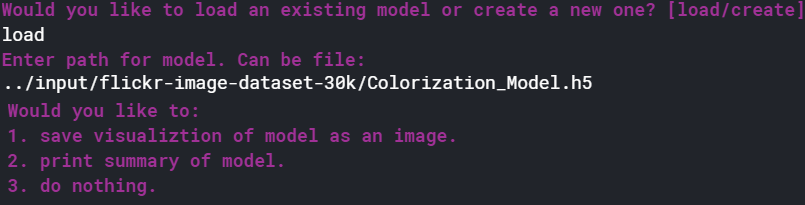
אם אין, לא מבקש.

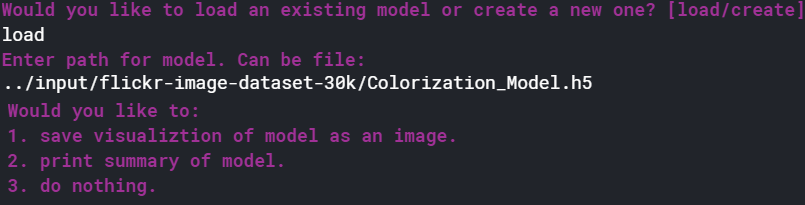
בסוף מדפיס כמה תמונות נותרו.

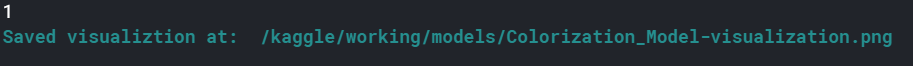
לאחר מכן, התוכנית מבקשת כמויות לתהליכי האימון והבחינה, ומספר האיפוצ'ים שיתרחשו.

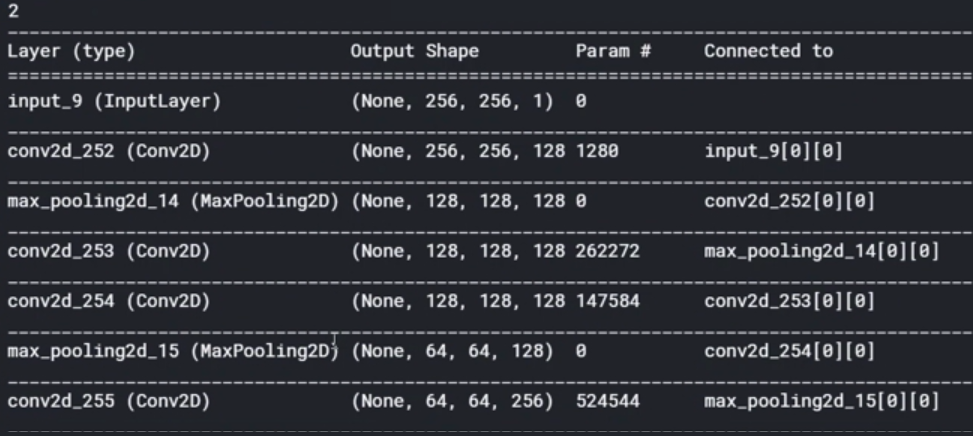
**אם בחרנו במסלול 1:** שואל האם ליצור או לייבא מודל קיים.

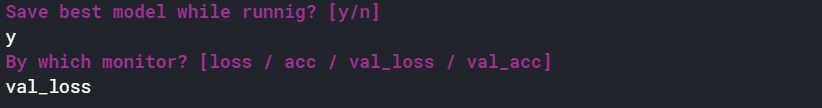
אם בחרנו ליצור, נוצר מודל ויש לנו אפשרות לשמור תצוגה ויזואלית שלו כתמונה (ישנו עותק בנספחים) /להדפיס את הפרטים של כל השכבות שלו.

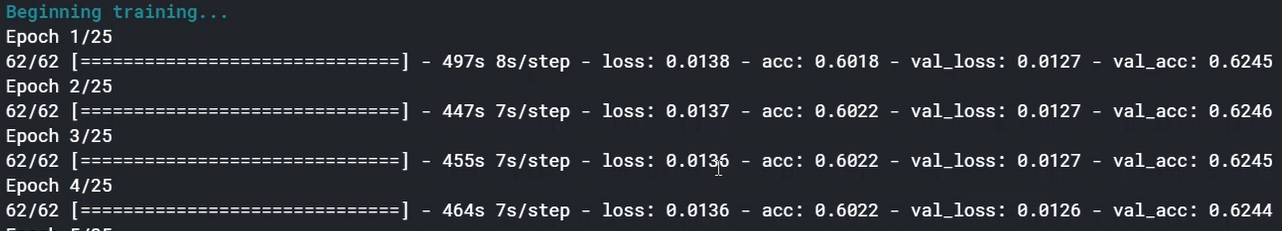
****אם בחרנו לייבא, מבקש דרך למודל ומייבא אותו.

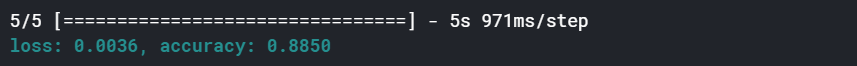
**אם בחרנו במסלול 2:** מבקש דרך למודל ומייבא אותו.

אם לוחצים 1, התמונה נשמרת ומודפס המקום בו נשמרה.

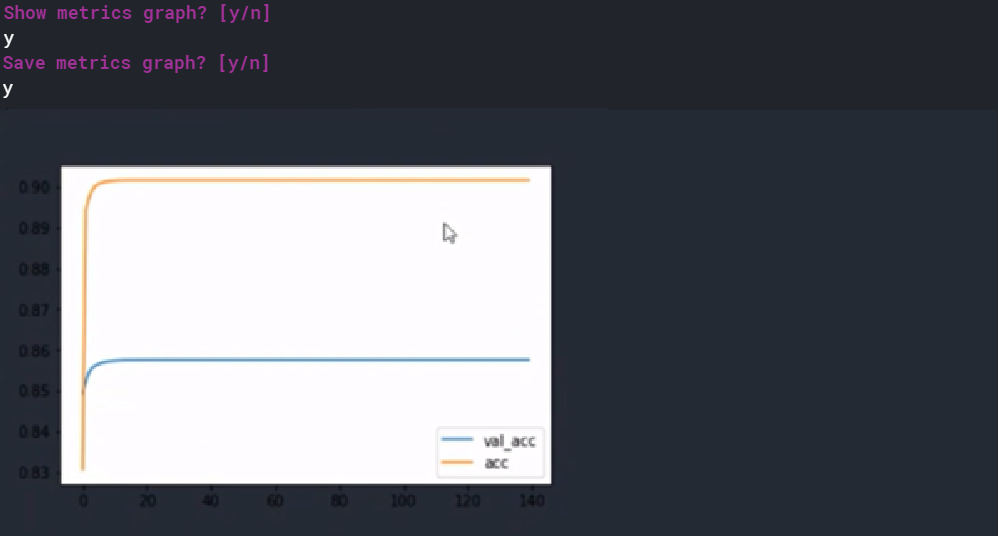
אם לוחצים 2, מודפס תקציר של המודל. (תמונה חלקית)

**אם בחרנו במסלול 1:** התוכנית שואלת האם לשמור את המודל בעת הריצה של האימון. אם כן, בוחרים לפי איזה קריטריון.

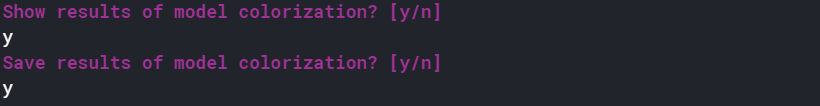
**אם בחרנו במסלול 1:** הלימוד מתחיל. (תמונה חלקית)

**אם בחרנו במסלול 1:** הערכה מתבצעת, מודפסים הערכים שיצאו בסוף.

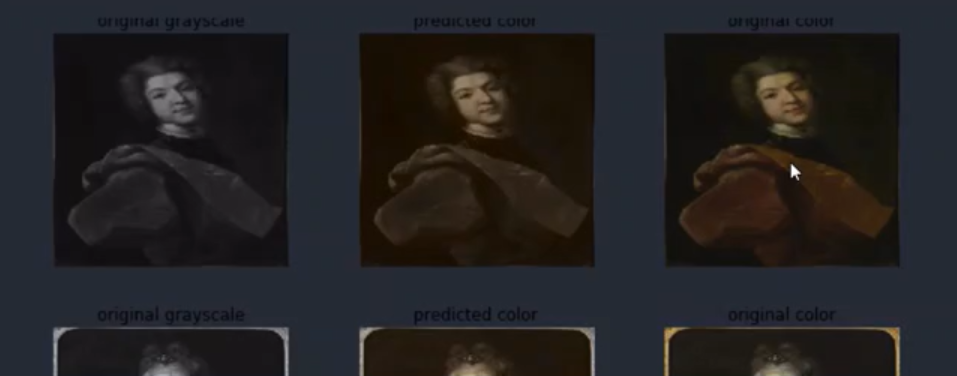
**אם בחרנו במסלול 1:** התכנית שואלת אם לשמור את המודל לאחר הלימוד. אם כן, שומרת.

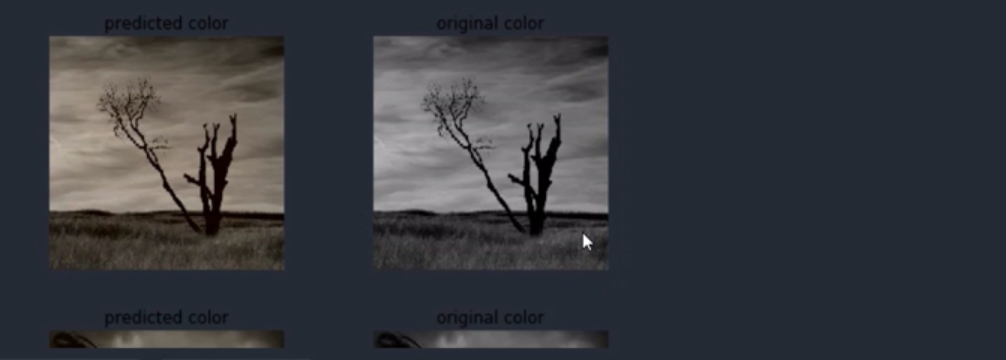
**אם בחרנו במסלול 1:** מציגה את הגרפים. (תמונה חלקית – תמונות מלאות במסקנות הפרויקט)

התוכנית שואלת את המשתמש אם הוא רוצה לראות את התוצאות ואם כן, האם הוא רוצה לשמור אותן. אם כן, התמונות עם הצבע החזוי, ישמרו.

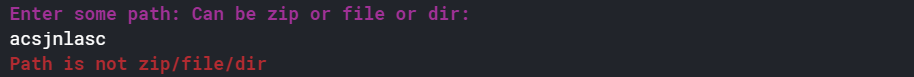


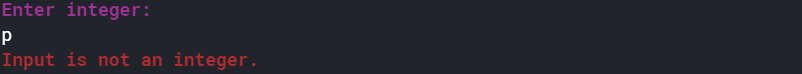
**אם בחרנו במסלול 1:** יודפסו 3 טורים של תמונות: צבע מקור, צבע חזוי, ושחור-לבן. (תמונה חלקית)

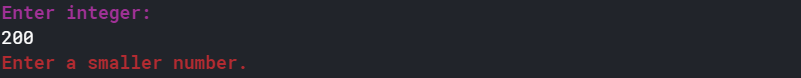


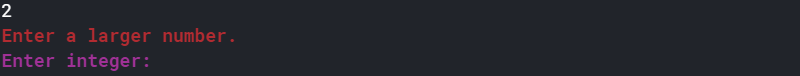
**אם בחרנו במסלול 2:** יודפסו 2 טורים של תמונות: צבע חזוי ושחור-לבן. (תמונה חלקית)

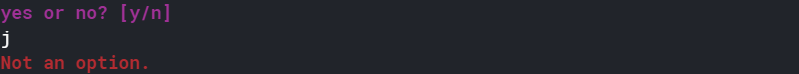
#### אזהרות אפשריות

במקרה של קלט דרך לא נכונה או מסוג לא נכון:

במקרה של קלט תו שהוא לא מספר כמתבקש:

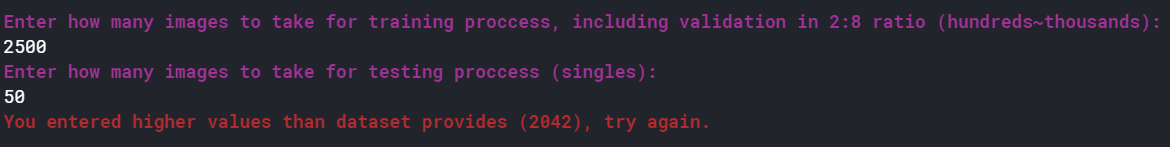
במקרה של קלט מספר גדול מדי:

במקרה של קלט מספר קטן מדי:

במקרה של קלט שלא מתאר אפשרות בשאלה מרובת אפשרויות:

במקרה שהתוכנית לא מוצאת את המאגר ברירת המחדל, למרות שבוקשה לייבא משם קבצים:

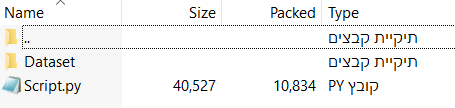
במקרה של מאגר ריק:

במקרה של קלט כמויות גדולות מדי למאגר:

במקרה של קלט דרך למודל שלא מובילה למודל שלם:

# מדריך למפתח

## קבצים

הפרויקט אמור להגיע כקובץ זיפ עם מאגר המידע והסקריפט כמו בתמונה:

**Script.py** – הסקריפט של הפרויקט מכיל את כל מה שנמצא בתיאור הפנימי. הקובץ מכיל את כל המשתנים הגלובלים, המחלקות, והפעולות. זאת מבלי להשתמש בקבצים אחרים, רק בספריות החיצוניות שאפשר להתקין על סביבת העבודה. כתבתי את הסקריפט דרך העורך של אתר Kaggle שמבוסס על Jupiter Notebook, לכן הסקריפט מכיל תאי קוד (code cells), שחלק מהעורכים לא תומכים בהם (למשל PyCharm Community). תאי קוד אלה מאפשרים הרצה של חלקי קוד בנפרד ואני ממליץ מאוד על שימוש בהם.

**Dataset** – תיקייה שמכילה את מאגר התמונות לשימוש הסקריפט, בה 3 תיקיות:

* **flickr-image-dataset-compressed** – 31,783 קבצי JPG. התמונות נלקחו מהאתר [flickr.com](https://www.flickr.com/) ומשמשות במקור למודלים שמתאימים משפטים באנגלית לתמונות כתיאור שלהן.
* **Art\_validation\_set** – 866 קבצים בתוך מספר תת-תיקיות ובסוגים מרובים. נלקח מהמאגר Art Images ב-Kaggle כדי לגוון את המאגר (להוסיף לו קבצים פגומים וסוגי תבניות צבע אחרים). יש קישור בביליוגרפיה.
* **BW** –7 תמונות שחור-לבן בתבניות Grayscale, RGB למטרת החיזוי.

## תיאור פנימי של הקוד

### משתנים גלובלים

אלו הם משתנים שהוגדרו בתוך פעולות או מחוץ להן כגלובלים. הדבר אומר שכל פעולה יכולה להשתמש בהם מבלי לייבא אותם. בהשראת סקריפטים רבים שראיתי באינטרנט החלטתי שאני מעדיף לעבוד עם משתנים מסוג זה. המשתנים הראשונים עד BATCH\_SIZE הוגדרו בגוף הראשי של הקוד, ומ- IMAGES\_PATHS המשתנים הוגדרו בתוך פעולות.

|  |  |  |
| --- | --- | --- |
| **תפקיד** | **סוג** | **שם** |
| זרע לאקראיות | Int | seed |
| דרך למיקום אליו נשמרים הקבצים | String | OUTPUT |
| מחליט על המסלול בו התכנה תתקדם: האם לייבא מאגר ולאמן מודל לפיו, או לייבא מודל קיים ולצבוע תמונות ממאגר | String | action |
| רוחב התמונות איתן יעבוד המודל, לא צריך להיות קשור לרוחב האמיתי שלהן | Int | IMG\_WIDTH |
| אורך התמונות איתן יעבוד המודל, לא צריך להיות קשור לאורך האמיתי שלהן | Int | IMG\_HEIGHT |
| גודל החלקים הנפרדים (באצ'ים, batches) עליהן מתבצעת הלמידה | Int | BATCH\_SIZE |
| רשימה המכילה את הדרכים לכל הקבצים במאגר, לאחר מכן את כל הדרכים שמובילות לתמונות שאפשר לעבוד איתן | List | IMAGES\_PATHS |
| הדרכים לתמונות שנועדו לתהליך האימון | List | TRAIN\_PATHS |
| הדרכים לתמונות שנועדו לחלק הווידוא בתהליך האימון | List | VALIDATE\_PATHS |
| הדרכים לתמונות שנועדו לבדיקה הסופית של ביצועי המודל | List | TEST\_PATHS |
| כמות הערוצים של התמונות שאיתן התכנה תעבוד (0/1/3/4) בהתאם לתבנית הצבע המדוברת | Int | CHANNELS |
| כמות הפעמים שתהליך הלימוד יעבור על כל המאגר | Int | EPOCHS |
| המודל עצמו | Model | MODEL |
| רשימה שבה מאוחסנים פעולות callbacks של Keras | List | CALLBACKS |
| תיעוד הלמידה לאורך הזמן, מתקבל מפעולת הלימוד | History | HIST |

### גוף הקוד הראשי

##### מבצע:

* מדפיס למשתמש הסברים.
* מגדיר את המשתמשים הגלובלים IMG\_WIDTH, IMG\_HEIGHT, BATCH\_SIZE.
* מגדיר את המשתנה הגלובלי OTPUT ומכניס לו ערך של תיקיית הפלט של התכנית, בה ישמרו כל הקבצים. אם התכנית רצה דרך Kaggle, התיקייה נמצאת בתוך תיקיית הפלט של הפלטפורמה, אחרת שם את OUTPUT בתיקייה שבה הסקריפט רץ.
* יוצר תיקיות results, models בתוך OUTPUT.
* שואל את המשתמש איזה מסלול ירצה לבחור:
  1. אימון מודל חדש או קיים
  2. חזיית תמונות על בסיס מודל קיים
* לפי המסלול שנבחר, מזמן את הפעולות הרלוונטיות בסדר הנכון כדי שהתוכנית תעבוד כמו שצריך:
  1. יבוא המידע -> בדיקת המידע -> חלוקת המידע -> יצירת/יבוא מודל -> אימון המודל -> הערכת המודל -> הצגת התוצאות.
  2. יבוא המידע -> בדיקת המידע -> חלוקת המידע -> יבוא מודל -> הצגת התוצאות.

### מחלקות ופעולות עזר

אלו הן פעולות שעוזרות לתפקוד התקין של הסקריפט, הן מטפלות בדברים קטנים שיהיה מיותר לכתוב כל פעם מחדש.

#### מחלקת Color

* מחלקה שמכילה מחרוזות שמייצגות הדפסה בצבעים בפייתון.

#### myprint

##### מקבלת:

* message – הודעה כמחרוזת.
* mode – מצב ההדפסה כתו.
* start\_newline – ערך בוליאני הקובע האם ליצור שורה חדשה בסוף ההדפסה.

##### מבצעת:

* מדפיסה את ההודעה בצבעים ש-Color מספקת, לפי mode.

##### מחזירה:

* כלום.

#### **path\_verify**

##### מקבלת:

* message – הודעה כמחרוזת.
* can\_be – רשימה המכילה מחרוזות של סוג הדרך (קובץ/תיקייה/זיפ).

##### מבצעת:

* מדפיסה את ההודעה באמצעות myprint.
* מקבלת מהמשתמש דרך כמחרוזת.
* מוודאת שהדרך תקינה ושהסוג שלה נמצא ב-can\_be.
* חוזרת על התהליך עד שיש דרך תקינה שאפשר להחזיר.

##### מחזירה:

* דרך כמחרוזת.

#### int\_verify

##### מקבלת:

* message – הודעה כמחרוזת.
* min\_ – מספר שלם מינימלי שאפשר להחזיר. ברירת מחדל: 1.
* max\_ – מספר שלם מקסימלי שאפשר להחזיר. ברירת מחדל: 1000000.

##### מבצעת:

* מדפיסה את ההודעה באמצעות myprint.
* מקבלת מהמשתמש קלט כמחרוזת.
* מוודאת שהקלט הוא מספר ושהוא בטווח בין min\_ ל-max\_.
* חוזרת על התהליך עד שיש מספר תקין שאפשר להחזיר.

##### מחזירה:

* מספר שלם.

#### option\_verify

##### מקבלת:

* message – הודעה כמחרוזת.
* options – רשימה של אפשרויות.

##### מבצעת:

* מדפיסה את ההודעה באמצעות myprint.
* מקבלת מהמשתמש קלט כמחרוזת.
* מוודאת שהקלט הוא ערך שקיים ב-options.
* חוזרת על התהליך עד שיש ערך תקין שאפשר להחזיר.

##### מחזירה:

* אפשרות כמחרוזת.

#### draw\_progress\_bar

##### מקבלת:

* n – מספר שלם, המיקום הנוכחי באיטרציה.
* total – מספר שלם, גודל האיטרציה.
* bar\_len – אורך ההדפסה.

##### מבצעת:

* מדפיסה באורך bar\_len סרגל התקדמות של איטרציה, כאשר n/total הוא האחוז הנוכחי.

##### מחזירה:

* כלום.

#### iskaggle

##### מקבלת:

* כלום.

##### מבצעת:

* מנסה להשתמש בפונקציה שקיימת רק בפלטפורמות שמשתמשות ב-IPython, כמו JupiterNotebook, ולא כמו cmd וכו'.
* בודקת אם הפלטפורמה היא מבוססת על JupiterNotebook

##### מחזירה:

* האם הקוד מורץ בפלטפורמה המבוססת על JupiterNotebook.

### פעולות עזר לבדיקת הקבצים

פעולות אלה הן פעולות שנעזרתי בהן כדי לבצע את בדיקת המאגר. הן מבצעות תהליכים טכניים שפישוטן סייע לי בכתיבת הקוד.

#### find\_start\_dataset

##### מקבלת:

* gray – האם מציאת המאגר ברירת המחדל היא רק לתחזית המבוססת על תמונות שחור-לבן.

##### מבצעת:

* אם המשתמש רוצה להשתמש במאגר ברירת המחדל, בודקת שהוא קיים בפלטפורמה.
* אם המאגר עדיין בזיפ, מחלצת אותו באמצעות extraction.

##### מחזירה:

* את הדרך למאגר ברירת מחדל אם קיים בפלטפורמה
* אחרת או אם המשתמש לא רוצה להשתמש במאגר ברירת המחדל, מחרוזת ריקה

#### extraction

##### מקבלת:

* path – דרך, כמחרוזת.

##### מבצעת:

* אם הדרך מובילה לזיפ, מחלצת את הזיפ לדרך שתושג באמצעות path\_verify.

##### מחזירה:

* הדרך הנוכחית למאגר המידע, כמחרוזת.

#### scan\_dataset

##### מקבלת:

* path – דרך, כמחרוזת.

##### מבצעת:

* עוברת כל הקבצים בתוך התיקייה של הדרך ומצרפת את הדרכים שלהם לרשימה.

##### מחזירה:

* רשימה של כל הדרכים בתוך התיקייה של הדרך.
* אם הדרך היא קובץ, אז רשימה שמכילה את הדרך.

#### remove\_bads

##### מקבלת:

* unreadables – רשימה של דרכים.
* wrong\_colors – רשימה של דרכים.
* min\_ - מספר שלם.

##### מבצעת:

* עוברת על כל הדרכים בשתי הרשימות ומסירה אותן מ- IMAGES\_PATHS.
* מציעה אפשרות להדפיס את הדרכים האלה באמצעות myprint.

##### מחזירה:

* ערך בוליאני המעיד האם נשארו יותר מ-min\_ דרכים ב- IMAGES\_PATHS.

### פעולות בדיקת המאגר וקביעתו

פעולות אלה ממלאו את תפקיד יבוא המאגר וחילוקו לשם ביצוע תפקידים שונים בתכנה. הן מייבאות רק את החלקים הטובים של המאגר וכך מונעות בעיות עתידיות בהרצת הסקריפט.

#### normal\_check\_data

##### מקבלת:

* כלום.

##### מבצעת:

* מגדירה דרך למאגר באמצעות path\_verify. מחלצת עם extraction.
* מכניסה ל-IMAGES\_PATHS את כל הדרכים במאגר עם scan\_dataset.
* עוברת על הדרכים ב- IMAGES\_PATHSואם יש שם קובץ שהוא לא תמונה או תמונה שאי אפשר לקרוא אותה כמו שצריך, או תמונה עם מספר ערוצים (תבנית צבע) לא נכונה, **כולל תמונות grayscale**, מכניסה אותה לרשימות.
* CHANNELS נקבע לפי התמונה הראשונה שנמצאת.
* מסירה את הדרכים שברשימות מ- IMAGES\_PATHSבאמצעות remove\_bads.
* חוזרת על התהליך עד שישנו מאגר טוב שמכיל מספיק תמונות טובות.

##### מחזירה:

* כלום.

#### gray\_check\_data

##### מקבלת:

* כלום.

##### מבצעת:

* מגדירה דרך למאגר באמצעות path\_verify. מחלצת עם extraction.
* מכניסה ל-IMAGES\_PATHS את כל הדרכים במאגר עם scan\_dataset.
* עוברת על הדרכים ב- IMAGES\_PATHSואם יש שם קובץ שהוא לא תמונה או תמונה שאי אפשר לקרוא אותה כמו שצריך, או תמונה עם מספר ערוצים (תבנית צבע) לא נכונה, מכניסה אותה לרשימות.
* CHANNELS נקבע לפי התמונה הראשונה שנמצאת.
* מסירה את הדרכים הנ"ל מ- IMAGES\_PATHSבאמצעות remove\_bads.
* חוזרת על התהליך עד שישנו מאגר טוב שמכיל מספיק תמונות טובות.

##### מחזירה:

* כלום.

#### normal\_split\_data

##### מקבלת:

* כלום.

##### מבצעת:

* קולטת מהמשתמש את כמות התמונות מהמאגר איתן ירצה לאמן את המודל בעזרת int\_verify עם מינימום של 5\*BATCH\_SIZE.
* קולטת מהמשתמש את כמות התמונות מהמאגר איתן ירצה לבחון את המודל בעזרת int\_verify עם מקסימום של BATCH\_SIZE.
* חוזרת על התהליך עד שסכום הכמויות קטן מסך הדרכים ב- IMAGES\_PATHSאו שווה לו.
* לפי כמויות אלה, מקצה דרכים אקראיות מ- IMAGES\_PATHSל-TRAIN\_PATHS, VALIDATE\_PATHS, TEST\_PATHS.
* קולטת מהמשתמש ערך ל- EPOCHS.

##### מחזירה:

* כלום.

##### **gray\_split\_data**

##### מקבלת:

* כלום.

##### מבצעת:

* אם IMAGES\_PATHS לא מכילה רק דרך אחת:
* קולטת מהמשתמש את כמות התמונות מהמאגר עליו ירצה לחזות את הצבע.
* חוזרת על התהליך עד שהכמות קטנה מסך הדרכים ב- IMAGES\_PATHSאו שווה לו.
* חותכת את IMAGES\_PATHS לגודל זה באקראיות.

##### מחזירה:

* כלום.

### פעולות שקשורות ליצירת המודל

פעולות אלה אחראיות לייצור/ייבוא המודל עצמו, איתו תתבצע הלמידה.

המודל הוא מקודד אוטומטי (autoencoder) שמורכב מרשתות עצביות מפותלות (CNN) ומכיל תהליך **קידוד** (encode) ותהליך **פיענוח** (decode) של התמונה: בקידוד השתמשתי בשכבות מפותלות (convolution) עם הפעלת (activation) LeakyReLU. הגדלתי את מספר המסננים (filters/kernels) עם כל שכבה. לשכבות הפענוח, השתמשתי בשכבת UpSampling ואחריה שכבה מפותלת עם מסננים שמספרם יורד בהדרגה, בעצם ההפך מתהליך הקידוד.

הרעיון הבסיסי הוא ששלב הקידוד לוקח קלט ודוחס אותו על ידי העברתו דרך שכבות מפותלות שונות. באמצעות הייצוג הדחוס הזה של הקלט, שלב הפענוח מייצר את הפלט הסופי. מכיוון ששלב הפענוח רואה רק את הייצוג הדחוס של הקלט המקורי, הוא עלול להחמיץ תכונות חשובות של התמונות שאבדו בשלב הקידוד. לכן הכנסתי לכל שכבה בפיענוח את הפלט של השלב המקביל בקידוד, כך שהמפענח יוכל לנצל גם את המידע זה.

#### conv2d

##### מקבלת:

* layer\_input – השכבה שעליה יתבצע הקידוד – המקודד עד כה.
* filters – מספר המסננים לייצר.

##### מבצעת:

* מבצעת קידוד באמצעות שכבה מפותלת עם הפעלת LeakyReLU.

##### מחזירה:

* את המקודד לאחר הוספת השכבות.

##### **deconv2d**

##### 

##### מקבלת:

* layer\_input – השכבה שעליה יתבצע הפיענוח – המפענח עד כה.
* from\_encode – הפלט של השכבה מתהליך הקידוד.
* filters – מספר המסננים לייצר.

##### מבצעת:

* מגדילה את התמונות עם UpSampling.
* מבצעת פענוח באמצעות שכבה מפותלת.
* מחברת את layer\_input ו- from\_encodeלבאץ' אחד.

##### מחזירה:

* את המפענח לאחר הוספת השכבות.

#### colorize

##### מקבלת:

* כלום.

##### מבצעת:

* יוצרת מקודד אוטומטי (autoencoder) באמצעות CNN:
* יוצרת מקודד (encoder) שמקבל תמונת Grayscale באמצעות conv2d.
* יוצרת מפענח (decoder) באמצעות deconv2d שמחזיר תמונת RGB.

##### מחזירה:

* את המקודד האוטומטי כמודל.

#### model\_making

##### מקבלת:

* only\_load – ערך בוליאני הקובע האם צריך רק לטעון מודל קיים. ברירת מחדל: False.

##### מבצעת:

* אם only\_load או אם משתמש בוחר לטעון מודל קיים, טוען מודל ל-MODEL מדרך שמתקבלת עם path\_verify.
* חוזרת על התהליך עד שההעלאה מצליחה.
* אם משתמש בוחר ליצור מודל, MODEL נוצר בעזרת colorizeועובר עיבוד.
* אם המשתמש רוצה, מודפס תקציר המודל או נשמרת תמונה ויזואלית של המודל בתיקיית models ב-OUTPUT.
* בסוף הפעולה, MODEL הוא מודל שמקבל תמונת Grayscale ומחזיר תמונת RGB צבועה.

##### מחזירה:

* כלום.

### פעולות וגנרטורי יבוא מידע למודל

אלה הן פעולות גנרטור שהשתמשתי בהם כדי לייבא מידע למודל בזמן האימון. פעולות גנרטור לא מחזירות (return), אלא מניבות (yield). כלומר, הן מחזירות ערכים בנפרד לאורך זמן. כל פעם שפונים אליהן, הן מניבות ערך אחד מתוך רבים באיטרציה מסוימת.

#### train\_gen

##### מקבלת:

* כלום.

##### מבצעת:

* מתאימה את המשתנים ש- data\_transformer\_generatorמקבל לפעולת האימון של המודל.

##### מניבה:

* את הערכים מ-data\_transformer\_generator עם המשתנים המותאמים.

#### validate\_gen

##### מקבלת:

* כלום.

##### מבצעת:

* מתאימה את המשתנים ש- data\_transformer\_generatorמקבל לפעולת הווידוא בתוך האימון של המודל.

##### מניבה:

* את הערכים מ-data\_transformer\_generator עם המשתנים המותאמים.

#### test\_gen

##### מקבלת:

* כלום.

##### מבצעת:

* מתאימה את המשתנים ש- data\_transformer\_generatorמקבל לפעולות הבחינה של המודל.

##### מניבה:

* את הערכים מ-data\_transformer\_generator עם המשתנים המותאמים.

#### data\_transformer\_generator

##### מקבלת:

* num\_of\_batchs – מספר הבאצ'ים שהגנרטור יעבור עליהם.
* batch\_size – גודל הבאצ'ים הנ"ל.
* paths – הדרכים לתמונות.

##### מבצעת:

* מרכיבה כל באץ' מהתמונות שב- path, בהסתמך על תבנית הצבע של התמונות.
* עושה סטנדרטיזציה (standardization) כדי להקל על החישובים במודל ולהפוך אותם ליותר מהירים.

##### מניבה:

* את הבאץ' כ-RGB בצבע.
* את הבאץ' כ-Grayscale (שחור-לבן).

### פעולות אימון, הערכה, ושמירת המודל

פעולות אלה יוצרות את התהליכים הנחוצים כדי לאמן את המודל ולשמור את התוצאות שלו בצורה טובה ואמינה. כמו גם להעריך את המודל באמצעות פונקציות ש-Keras מספקת.

#### callbacks\_making

##### מקבלת:

* כלום.

##### מבצעת:

* מייצרת callback שמנמיך את שיעור הלמידה (learning rate/lr) פי 5 אם היא לא משתפרת לאחר 5 איפוצ'ים רצופים.
* אם המשתמש רוצה, מייצרת callback ששומר את המודלים הטובים ביותר (לפי קריטריון שהמשתמש מזין) בעת האימון בתיקייה models ב-OUTPUT.
* מכניסה את הערכים האלה ל-CALLBACKS

##### מחזירה:

* כלום.

#### training

##### מקבלת:

* כלום.

##### מבצעת:

* מאמנת את MODEL עם fit\_generator עם כל גנרטורים והערכים המתאימים.
* מייצרת את HIST כמשתנה מסוג History שמכיל את המידע של האימון.

##### מחזירה:

* כלום.

#### evaluating

##### מקבלת:

* כלום.

##### מבצעת:

* מעריכה את MODEL עם evaluate\_generator עם כל גנרטורים והערכים המתאימים.
* מדפיסה את נתוני דיוק (accuracy) וההפסד (loss) הסופיים.

##### מחזירה:

* כלום.

#### model\_saving

##### מקבלת:

* כלום.

##### מבצעת:

* מוחקת את כל המודלים שנשמרו בתהליך האימון חוץ מהאחרון מביניהם, שהוא הטוב מכולם.
* אם המשתמש מעוניין, MODEL נשמר בתיקייה models בתוך OUTPUT. מודל ומשקלים בנפרד.

##### מחזירה:

* כלום.

#### sampling\_graph

##### מקבלת:

* כלום.

##### מבצעת:

* אם המשתמש מעוניין, מציגה גרף של פרטי הלימוד (דיוקים, הפסדים ושיעור הלמידה) לאיפוץ', באמצעות HIST.

##### מחזירה:

* כלום.

#### sampling\_images

##### מקבלת:

* grays – האם הצגת התוצאות היא רק לתחזית המבוססת על תמונות שחור-לבן.

##### מבצעת:

* אם המשתמש מעוניין:
* אם grays – מקבלת ערכים מגנרטור test\_gen ומציגה בעזרתם תמונות שחור לבן, ותמונות עם צבע חזוי לפי המודל, זו לצד זו לשם השוואה.
* אחרת, מציגה גם תמונות בצבע לצד שאר התמונות.
* אם המשתמש מעוניין, התמונות עם הצבע שנחזה נשמרות בתיקייה results ב-OUTPUT.

##### מחזירה:

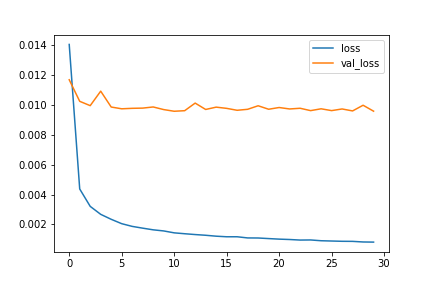
* כלום.

# מסקנות הרצת המודל

המסקנות בחלק זה מבוססות על הרצת המודל על 3000 תמונות כ-30 איפוצ'ים בזמן כולל של 41.5 דקות.

**loss: 8.1659e-04, accuracy: 0.8694, val\_loss:0.0096, val\_acc: 0.4909**

אפשר להבין מהתוצאה הסופית ומהגרפים שלא הייתה התאמה מספרית לכל אורך האימון בין התוצאות על תמונות האימון לבין התוצאות על תמונות הוולידציה. לאחר מחקר באינטרנט הבנתי שהסיבה לכך היא התאמת יתר (overfitting) לתמונות האימון. אז הוספתי שכבות Dropout, הוספתי פעולות רגולציה (regularization), והגדלתי את כמות התמונות. אלה אמורים לעזור במקרה של התאמת יתר, אך דבר לא עזר.





# סיכום אישי

העבודה על הפרויקט עבורי לא הייתה קלה. כן האמנתי בהתחלה שהיא תהיה נורא קלה ושאני אפילו אסיים אותה עד ינואר, אך יש לציין שטעיתי טעות מרה. תחילה הייתי צריך לחקור הרבה דברים באינטרנט כדי להבין מושג קלוש על מה מדברים במקומות אחרים באינטרנט שהסבירו איך מבצעים את המשימה שבחרתי לעצמי – לסווג לימונים לפי רמת הבשלות שלהם. התנסיתי עם כל מיני קודים באינטרנט שעשו את אותו והתחוור לי יותר מרגע לרגע כמה חסר מושג אני.

אני לא גאה בכך אבל ניסיתי להתעלם כמה שיותר מחוסר הידיעות שלי וניסיתי כמה שיותר פקודות, פעולות, גנרטורים ומה לא שמצאתי באינטרנט בניסיונות עוורים אולי להצליח במשימה שלי. לאחר חודש או שניים בעת ניסויי היותר-מוצלחים עם קוד תכנת סיווג הלימונים שלי, הגעתי לשתי מסקנות קריטיות – שהפרויקט שלי זז לאט מאוד, ושהפרויקט שלי לא מעניין אותי.

בראשונה טיפלתי מיד באופן בלתי מודע, בעת המחקר שלי על אחד הנושאים, נתקלתי בסקריפט שצבע תמונות שחור לבן, בהצלחה מועטה יחסית. הדבר פתח אותי לנושא ועניין אותי במיוחד. שם הוסברו הרבה נושאים ופקודות שלא הכרתי. התחלתי להתמקד בנושא הזה, ולאחר שבועיים של מחקר וניסיונות, הבנתי שזה הכיוון שאני מעוניין להתמקד בו וידעתי את המורה (דינה קראוס) בכך שאני רוצה להחליף נושא.

עכשיו נותרה לי בעיה אחת, המחשב שלי לא עמד בציפיות. לא היה לי RAM מספיק גדול ו-CPU מספיק חזק כדי להריץ את הפרויקט ביעילות. תחילה ניסיתי לצמצם את העומס על ה- RAM באמצעות כיווץ (compression) התמונות ומחיקת משתנים תוך כדי ההרצה, אך הגעתי למסקנה שזה לא מספיק. בין היתר השתמשתי בספריות שונות כדי לעקוב אחרי ניצול הזיכרון, ויתכן שלמדתי מיומנויות שונות שיעזרו לי בעתיד.

מצד שני, הבנתי מהר מאוד מה יוריד את העומס מהמעבד שלי, מעבד גרפי. ואיזה יופי, יש לי שניים! רק חבל ששניהם לא מהחברה המתאימה, ולכן אי אפשר לעבוד איתם... לאחר ניסיונות שווא רבים מספור לגרום לזה לעבוד איכשהו, הייתי נורא קרוב ללהתייאש לגמרי ואף הפסקתי לעבוד על הפרויקט לחודשיים. המזל שלי הוא שאני מכיר סטודנט אחד שבדיוק מסיים לימודי הנדסת מערכות מידע שהכיר לי את פונקציית העריכה וההרצה ב-Kaggle. שם יכולתי להריץ את הפרויקט אין-ספור פעמים לאורך זמן רב על מעבד גרפי, ביעילות ובמהירות.

התמודדתי עם בעיית הזיכרון בסופו של דבר לאחר שפתרתי את בעיית המעבד, עשיתי זאת עם שימוש בפעולות גנרטורים. אני חייב להביע צער על כך שלא מלמדים עליהן בבית הספר משום שכעת הן נראות לי חלק אינטגרלי מכתיבת סקריפט בפייתון, ואני בטוח שהיכולת שלי להשתמש בהן תעזור לי בעתיד. היכולת של גנרטורים להניב ערכים לאורך זמן בלי להעמיס על הזיכרון הוציאה אותי מהבעיה הזו והפכה את הפרויקט שלי למסוגל להתמודד עם כל כמות של תמונות, לא משנה כמה גדולה.

Kaggle מגדירים את עצמם כ-"בית שלך למדע נתונים". ואני לא יכול לחלוק עליהם כלל, רק בזכותם הצלחתי להתקדם עם הקוד ולכתוב הרבה דברים אחרים ששיפרו אותו בלי להתעסק במגבלות הטכניות של המחשב שלי. מה גם שהאתר מציע מאגרי מידע ותמונות רבים מספור שאפשרו לי להתנסות בהמון מצבים שונים ולעצב תוכנית יעילה במיוחד.

אם הייתי מתחיל לעבוד על הפרויקט כיום, אני יודע שהייתי עושה עבודה הרבה יותר טובה. ראשית, אני מכיר את הנושא הרבה יותר טוב ואני מיודע ברוב הנושאים של כתיבת תוכנית ללימוד רשת עצבית. הנושאים האלה הם נורא קשים ללמידה ללא מדריך ובשפה שהיא לא שפת אם. אני כן מאמין שהתמודדתי עם האתגר בהצלחה, אבל אין לדעת לאילו אופקים הייתי יכול להגיע אם הייתי מתחיל את העבודה עם הידע שיש לי כבר בנושא והפנאי ללמוד עוד.

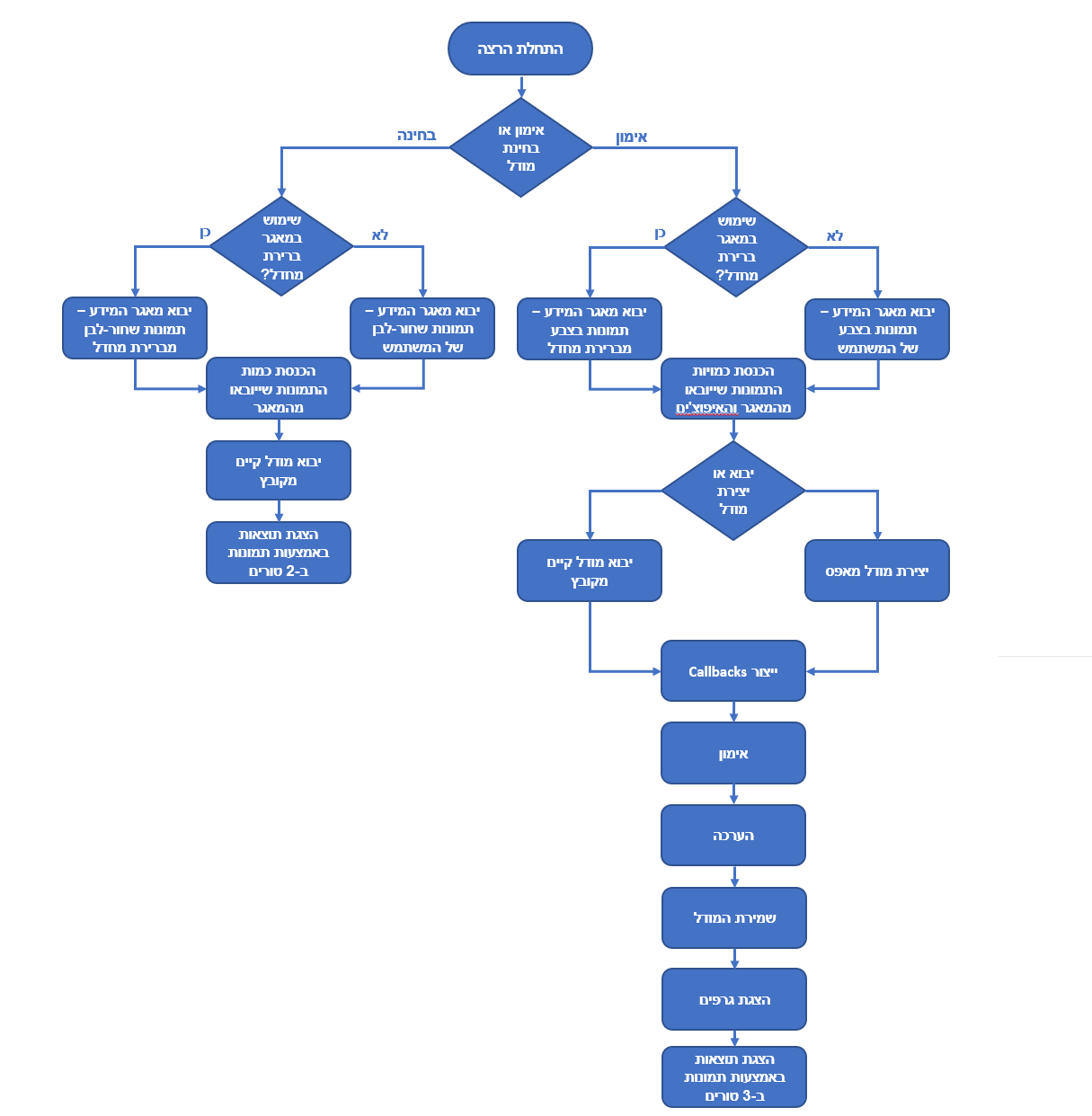
שנית, עכשיו אני מכיר את Kaggle שהייתה מקלה עליי מאוד בתחילת הדרך, משום שהיא גם מכילה מדריכים והמון המון דוגמות בנושא הפרויקט. ישנן רבבות של משתמשים באתר שיכולים לענות על שאלות ולהציע פתרונות ובכך להקל על תהליך העבודה של יחיד בהרבה. מיותר לציין שעם העורך המתקדם שלהם והיכולת להריץ תוכניות על חומרה חיצונית בחינם, הדבר מזרז בהרבה את הכתיבה עצמה ומקל על חוסר סבלנות ומשברים בפן הרגשי שיכולים להגרם בעת העבודה על נושא מורכב כמו זה.

# ביבליוגרפיה

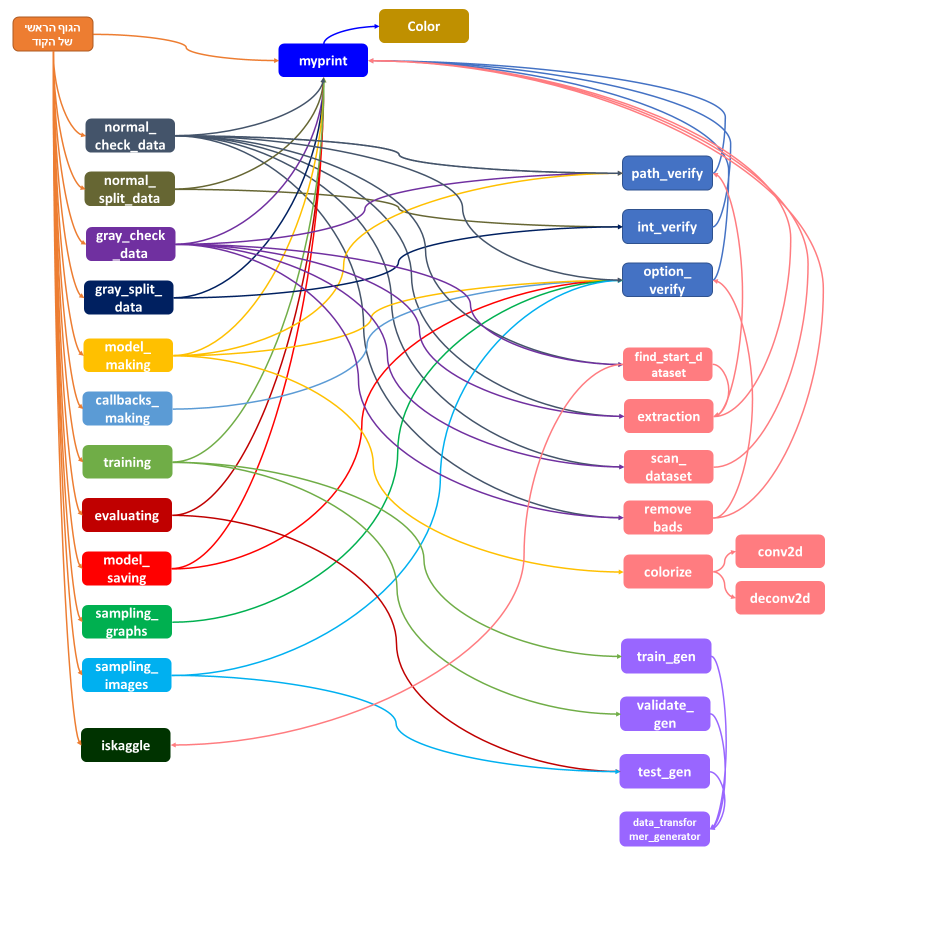
* <https://keras.io/api/applications/>
* <https://jupyter.org/>
* <https://github.com/BryanPlummer/flickr30k_entities>
* <https://fairyonice.github.io/Color-gray-scale-images-and-manga-using-deep-learning.html>
* <https://towardsdatascience.com/reduce-memory-usage-and-make-your-python-code-faster-using-generators-bd79dbfeb4c>
* <https://sanjayasubedi.com.np/deeplearning/black-and-white-to-color-using-deep-learning/>
* <https://becominghuman.ai/auto-colorization-of-black-and-white-images-using-machine-learning-auto-encoders-technique-a213b47f7339>
* <https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/>
* <https://keras.io/guides/functional_api/>
* <https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/>
* <https://www.pyimagesearch.com/2016/07/25/convolutions-with-opencv-and-python/>
* <https://developers.arcgis.com/python/guide/how-unet-works/>

# נספחים

## תרשים מסלול הרצה

זהו תרשים המתאר את מסלול ההרצה הכללי של התוכנית **מבחינת המשתמש**.

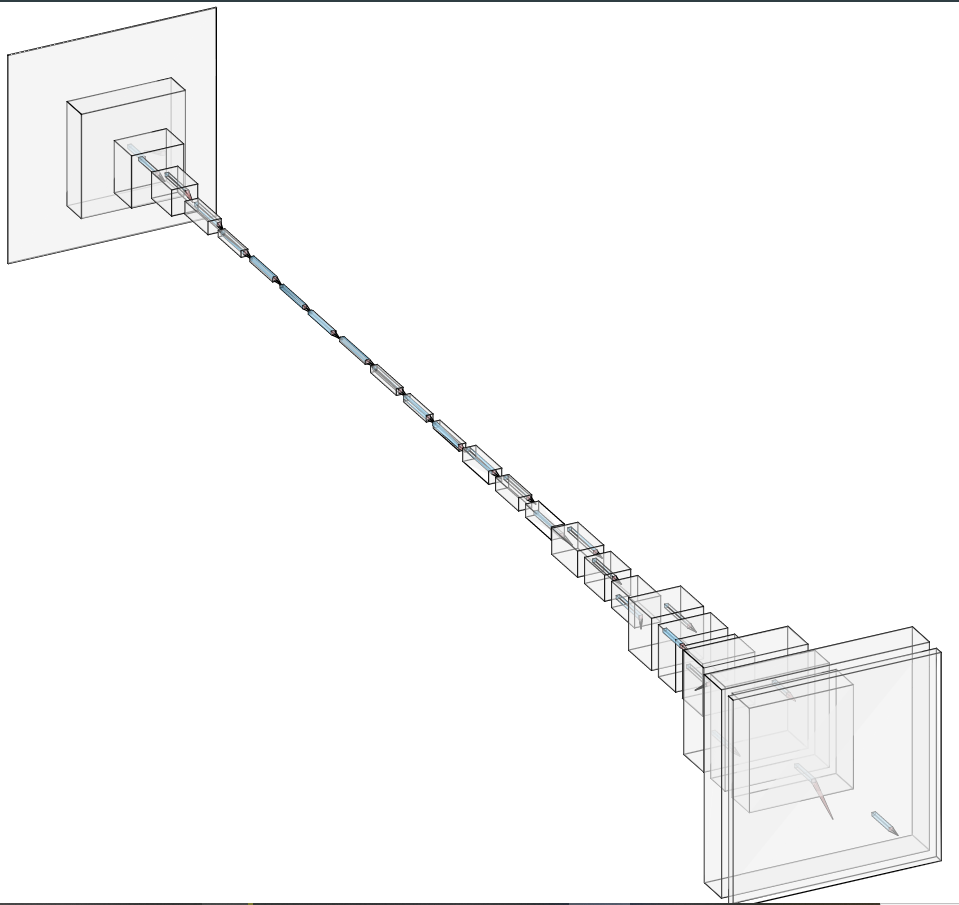
## תרשים קריאות

זהו תרשים קריאות (call graph) המציג את הקריאות של כל פעולה לכל פעולה.

## ייצוג ויזואלי של המודל

התמונה ש-Keras מפיק גדולה מדי למסמך אז היא זמינה בקישור:

<https://1drv.ms/u/s!AsNZggxhuzRzktk9Gc1pfs2v7kEPyg?e=AYO78I>

יש גם ייצוג תלת-מימדי מ-<http://alexlenail.me/NN-SVG/AlexNet.html> שמציג בצורה די טובה את 27 שינויי הגודל שעוברת תמונה במודל. כל תיבה מייצגת מערך ארבע-מימדי (טנסור tensor / באץ') שמועבר למודל, כשהגודל והרוחב מייצגים את עצמם ומספר הערוצים הוא העומק של התיבה. החיצים הכחולים-אדומים מייצגים את גודל ה-kernels, קטעי תמונה קטנים עליהם המודל מבצע חישובים.