

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

3מבוא
4פרטים כללים על התכנה
5מדריך למשתמש
5דרישות הפעלה
5הרצה תקינה של הפרויקט
5התחלת ההרצה
5תיקיות פלט
6פלטים אפשריים
13מדריך למפתח
13קבצים
14תיאור פנימי של הקוד
15משתנים גלובלים
16גוף הקוד הראשי
17מחלקות ופעולות עזר
20פעולות עזר לבדיקת הקבצים
22פעולות בדיקת המאגר וקביעתו
24פעולות שקשורות ליצירת המודל
26פעולות אימון, הערכה, ושמירת המודל
29מסקנות הרצת המודל
33סיכום אישי
35ביבליוגרפיה
36נספחים
36תרשים מסלול הרצה
37תרשים קריאות
38ייצוג ויזואלי של המודל

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 ממנה. 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. אני למען האמת 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.

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 בקוד בתבניות צבע (Color Formats/Spaces/Models), 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 Keras ו-TensorFlow, 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 כברירת 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:

```
(base) C:\WINDOWS\system32>python \path...\Project\Script.py
```

- דרך פלטפורמות המבוססות על Jupiter Notebook אפשר פשוט להריץ על ידי לחיצה על הכפתור הרלוונטי לאחר טעינת הקוד, במחברת 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

פלטנים אפשריים

התוכנית יוצרת מספר פלטנים 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.

```
Hello stranger, this is my python based deep learning project designed to learn how to color black and white images.
What would you like to do?
1 - model training
2 - predicting color of BW images with existing model
```

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

```
Use default dataset? [y/n]
y

Due to this deep learning project's purpose, any grayscale image will be excluded.
Any non-image format or image with broken data will also be excluded.
The program works with ONE color format (RGB/RGBA) per run for all the images you choose, based of your first image. Any other
color format will not be accepted.
...
Loaded 32657 files from dataset.
Checking dataset for incorrect files or images...
[ ] 0.14%
```

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

```
Use default dataset? [y/n]
n

Due to this deep learning project's purpose, any grayscale image will be excluded.
Any non-image format or image with broken data will also be excluded.
The program works with ONE color format (RGB/RGBA) per run for all the images you choose, based of your first image. Any other
color format will not be accepted.
Enter path to dataset. Can be zip or dir:
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/
...
Loaded 2128 files from dataset.
Checking dataset for incorrect files or images...
[=====] 10.57%
```

צביעת תמונות שחור-לבן

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

```
86 wrong format files or unreadable images.
Print paths? [y/n]
y
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/1000.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/1475.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/1375.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/0225.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/0825.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/1400.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/0650.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/1900.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/0900.jpg
../input/art-images-drawings-painting-sculpture-engraving/dataset/dataset_updated/training_set/painting/0400.jpg
```

אם אין, לא מבקש.

```
0 wrong color format excluded images.
```

בסוף מדפיס כמה תמונות נותרו.

```
2042 images are fine.
```

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

```
Enter how many images to take for training proccess, including validation in 2:8 ratio (hundreds~thousands):
2000
Enter how many images to take for testing proccess (singles):
20
Paths randomly assigned for training, validating and testing.

Enter how many epochs the training process will take (aprox. 1k images per epoch: GPU-1.5m | CPU-0.5h):
5
```

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.

```
Would you like to load an existing model or create a new one? [load/create]
create
...
Model created and compiled.
Would you like to:
1. save visualization of model as an image.
2. print summary of model.
3. do nothing.
```

אם בחרנו לייבא, מבקש דרך למודל ומייבא אותו.

```
Would you like to load an existing model or create a new one? [load/create]
load
Enter path for model. Can be file:
../input/flickr-image-dataset-30k/Colorization_Model.h5
Would you like to:
1. save visualiztion of model as an image.
2. print summary of model.
3. do nothing.
```

אם בחרנו במסלול 2: מבקש דרך למודל ומייבא אותו.

```
Enter path for model. Can be file:
../input/flickr-image-dataset-30k/Colorization_Model.h5
Would you like to:
1. save visualiztion of model as an image.
2. print summary of model.
3. do nothing.
```

אם לוחצים 1, התמונה נשמרת ומודפס המקום בו נשמרה.

```
1
Saved visualiztion at: /kaggle/working/models/Colorization_Model-visualization.png
```

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

```
2
-----
Layer (type)                Output Shape              Param #   Connected to
-----
input_9 (InputLayer)        (None, 256, 256, 1)      0
conv2d_252 (Conv2D)         (None, 256, 256, 128)    1280      input_9[0][0]
max_pooling2d_14 (MaxPooling2D) (None, 128, 128, 128)    0          conv2d_252[0][0]
conv2d_253 (Conv2D)         (None, 128, 128, 128)    262272    max_pooling2d_14[0][0]
conv2d_254 (Conv2D)         (None, 128, 128, 128)    147584    conv2d_253[0][0]
max_pooling2d_15 (MaxPooling2D) (None, 64, 64, 128)     0          conv2d_254[0][0]
conv2d_255 (Conv2D)         (None, 64, 64, 256)     524544    max_pooling2d_15[0][0]
-----
```


צביעת תמונות שחור-לבן

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

```
Save best model while runnig? [y/n]
y
By which monitor? [loss / acc / val_loss / val_acc]
val_loss
```

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

```
Beginning training...
Epoch 1/25
62/62 [=====] - 497s 8s/step - loss: 0.0138 - acc: 0.6018 - val_loss: 0.0127 - val_acc: 0.6245
Epoch 2/25
62/62 [=====] - 447s 7s/step - loss: 0.0137 - acc: 0.6022 - val_loss: 0.0127 - val_acc: 0.6246
Epoch 3/25
62/62 [=====] - 455s 7s/step - loss: 0.0136 - acc: 0.6022 - val_loss: 0.0127 - val_acc: 0.6245
Epoch 4/25
62/62 [=====] - 464s 7s/step - loss: 0.0136 - acc: 0.6022 - val_loss: 0.0126 - val_acc: 0.6244
```

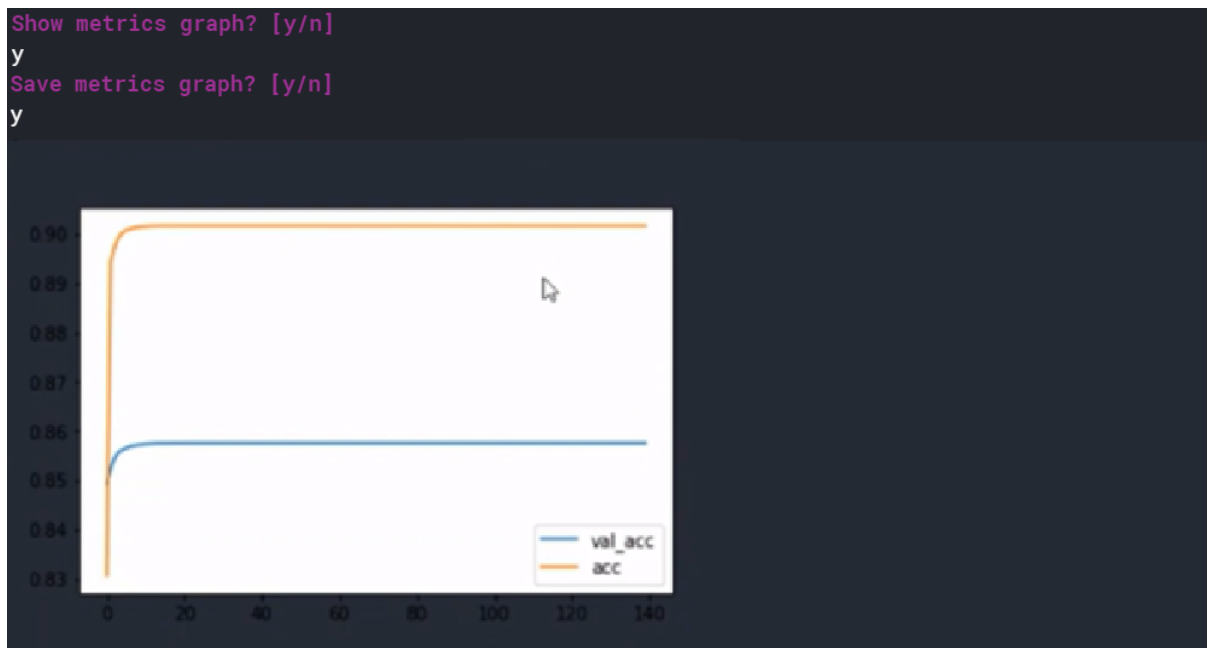
אם בחרנו במסלול 1: הערכה מתבצעת, מודפסים הערכים שיצאו בסוף.

```
5/5 [=====] - 5s 971ms/step
loss: 0.0036, accuracy: 0.8850
```

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

```
Save model after training? [y/n]
y
```

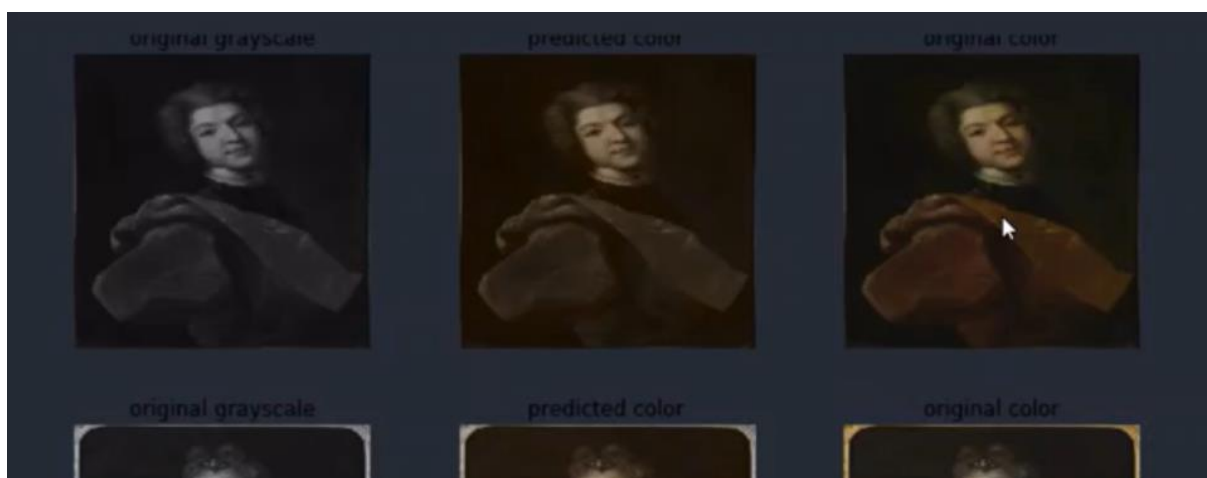
אם בחרנו במסלול 1: מציגה את הגרפים. (תמונה חלקית – תמונות מלאות במסקנות הפרויקט)



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

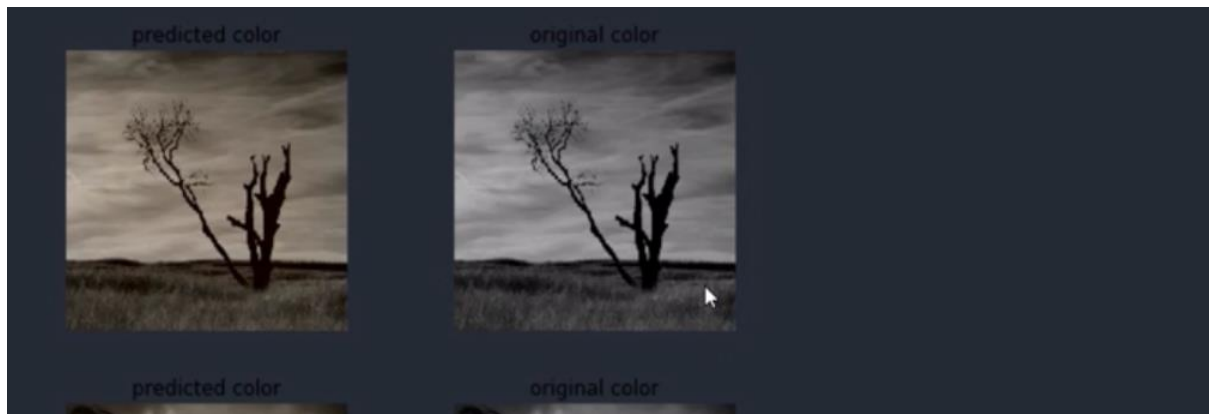
```
Show results of model colorization? [y/n]
y
Save results of model colorization? [y/n]
y
```

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



צביעת תמונות שחור-לבן

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:

```
Enter some path: Can be zip or file or dir:
acsjnlasc
Path is not zip/file/dir
```

במקרה של קלט תו שהוא לא מספר במתבקש:

```
Enter integer:
p
Input is not an integer.
```

במקרה של קלט מספר גדול מדי:

```
Enter integer:
200
Enter a smaller number.
```

במקרה של קלט מספר קטן מדי:

```
2
Enter a larger number.
Enter integer:
```

במקרה של קלט שלא מתאר אפשרות בשאלה מרובת אפשרויות:

```
yes or no? [y/n]
j
Not an option.
```

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

```
Default dataset wan't found.
```

במקרה של מאגר ריק:

```
0 images are fine.  
You have too little valid images in here, select another dataset.
```

במקרה של קלט כמויות גדולות מדי למאגר:

```
Enter how many images to take for training proccess, including validation in 2:8 ratio (hundreds~thousands):  
2500  
Enter how many images to take for testing proccess (singles):  
50  
You entered higher values than dataset provides (2042), try again.
```

במקרה של קלט דרך למודל שלא מובילה למודל שלם:

```
This model is weights only or incompatible.
```

מדריך למפתח

קבצים

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

Name	Size	Packed	Type
..			תיקיית קבצים
Dataset			תיקיית קבצים
Script.py	40,527	10,834	קובץ PY

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

- ✓ seed
- Ⓒ Color
- f myprint(message, mode, start_newline=True)
- f iskaggle()
- f path_verify(message, can_be)
- f int_verify(message, min_=1, max_=1000000)
- f option_verify(message, options)
- f draw_progress_bar(n, total, bar_len=50)
- f find_start_dataset(gray=False)
- f extraction(path)
- f scan_dataset(path)
- f remove_bads(unreadables, wrong_colors, min_)
- f normal_check_data(data_path)
- f gray_check_data(data_path)
- f normal_split_data()
- f gray_split_data()
- f colorize()
- f model_making(only_load=False)
- f train_gen()
- f validate_gen()
- f test_gen()
- f data_transformer_generator(num_of_batches, batch_size, paths)
- f callbacks_making()
- f training()
- f evaluating()
- f model_saving()
- f sampling_graphs()
- f sampling_images(grays=False)
- ✓ IMG_WIDTH
- ✓ IMG_HEIGHT
- ✓ BATCH_SIZE
- ✓ action

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.

שם	סוג	תפקיד
seed	Int	זרע לאקראיות
OUTPUT	String	דרך למיקום אליו נשמרים הקבצים
action	String	מחליט על המסלול בו התכנה תתקדם: האם לייבא מאגר ולאמן מודל לפיו, או לייבא מודל קיים ולצבוע תמונות ממאגר
IMG_WIDTH	Int	רוחב התמונות איתן יעבוד המודל, לא צריך להיות קשור לרוחב האמיתי שלהן
IMG_HEIGHT	Int	אורך התמונות איתן יעבוד המודל, לא צריך להיות קשור לאורך האמיתי שלהן
BATCH_SIZE	Int	גודל החלקים הנפרדים (batches, באצ'ים) עליהן מתבצעת הלמידה
IMAGES_PATHS	List	רשימה המכילה את הדרכים לכל הקבצים במאגר, לאחר מכן את כל הדרכים שמובילות לתמונות שאפשר לעבוד איתן
TEST_PATHS	List	הדרכים לתמונות שנועדו לבדיקה הסופית של ביצועי המודל
CHANNELS	Int	כמות הערוצים של התמונות שאיתן התכנה תעבוד (0/1/3/4) בהתאם לתבנית הצבע המדוברת
EPOCHS	Int	כמות הפעמים שתהליך הלימוד יעבור על כל המאגר
MODEL	Keras.Model	המודל עצמו
HIST	Dictionary	תיעוד הלמידה לאורך הזמן, מתקבל מפעולת הלימוד
start	Time.time	זמן ההתחלה בזימון פעולת draw_progress_bar

גוף הקוד הראשי

מבצע:

- מדפיס למשתמש הסברים.
- מגדיר את המשתמשים הגלובלים `BATCH_SIZE`, `IMG_HEIGHT`, `IMG_WIDTH`.
- מגדיר את המשתנה הגלובלי `OUTPUT` ומכניס לו ערך של תיקיית הפלט של התוכנית, בה ישמרו כל הקבצים. אם התוכנית 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

```
class Color:
```

- מחלקה שמכילה מחרוזות שמייצגות הדפסה בצבעים בפייתון.

```
def myprint(message, mode, start_newline=True):
```

- 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.

```
def path_verify(message, can_be):
```

- 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.

```
def int_verify(message, min_=1, max_=1000000):
```

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.
-

```
def option_verify(message, options):
```

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
```

```
def draw_progress_bar(n, total, bar_len=50):
```

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.
-

```
def iskaggle():
```

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.

`def find_start_dataset(gray=False):` 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, מחזרת ריקה
-

`def extraction(path):` extraction
מקבלת:

- path – דרך, כמחוזות.

מבצעת:

- אם הדרך מובילה לזיפ, מחלצת את הזיפ לדרך שתושג באמצעות path_verify.

מחזירה:

- הדרך הנוכחית למאגר המידע, כמחוזות.
-

`def scan_dataset(path):` scan_dataset
מקבלת:

- path – דרך, כמחוזות.

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.
-

```
def remove_bads(unreadables, wrong_colors):    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.

`def normal_check_data():` `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_PATHS` ואם 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_PATHS` באמצעות `remove_bads`.
- חוזרת על התהליך עד שישנו מאגר טוב שמכיל מספיק תמונות טובות.

מחזירה:

- כלום.

`def gray_check_data():` `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_PATHS` ואם 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_PATHS` באמצעות `remove_bads`.
- חוזרת על התהליך עד שישנו מאגר טוב שמכיל מספיק תמונות טובות.

מחזירה:

- כלום.

`def normal_split_data():`

`normal_split_data`

מקבלת:

- כלום.

מבצעת:

- קולטת מהמשתמש את כמות התמונות מהמאגר איתן ירצה לאמן את המודל בעזרת `int_verify` עם מינימום של `BATCH_SIZE*5`.
- 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_PATHS` או שווה לו.
- לפי כמויות אלה, מקצה דרכים אקראיות מ-`IMAGES_PATHS` ל-`TEST_PATHS`.
- קולטת מהמשתמש ערך ל-`EPOCHS`.

מחזירה:

- כלום.
-

`def gray_split_data():`

`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_PATHS` או 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.

def conv2d(layer_input, filters): 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

def deconv2d(layer_input, from_encode, filters):

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_encode לבאץ' אחד.

מחזירה:

- את המפענח לאחר הוספת השכבות.
-

```
def colorize():
```

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.
-

```
def model_making(only_load=False):
```

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 colorize ועובר 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 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. בחלק זה קיימת פעולת גנרטור (generator), שלא מחזירה (return), אלא מניבה (yield). כלומר, פעולות מסוג זה מחזירות ערכים בנפרד לאורך זמן – בכל פעם שפונים אליהן הן מניבות ערך אחד מתוך רבים מתוך איטרציה מסוימת.

`data_transformer_generator`

`def data_transformer_generator(paths, batch_size=BATCH_SIZE):`

מקבלת:

- `batch_size` – גודל הבאצ'ים שהגנרטור יעבור עליהם. ברירת מחדל: `BATCH_SIZE`
- `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).

`def training():`

`training`

מקבלת:

- כלום.

מבצעת:

- מאמנת את `MODEL` עם פעולת `fit` של `keras` בשיטת `cross-validation`.
- מייצרת את `HIST` כמילון (Dictionary) שמכיל את המידע של האימון שמתקבל מ-`fit`.

מחזירה:

- כלום.

```
def evaluating():
```

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.
-

```
def model_saving():
```

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.
-

```
def sampling_graph():
```

sampling_graph

מקבלת:

- כלום.

מבצעת:

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

מחזירה:

- כלום.
-

`def sampling_images(grays=False):` `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.
-

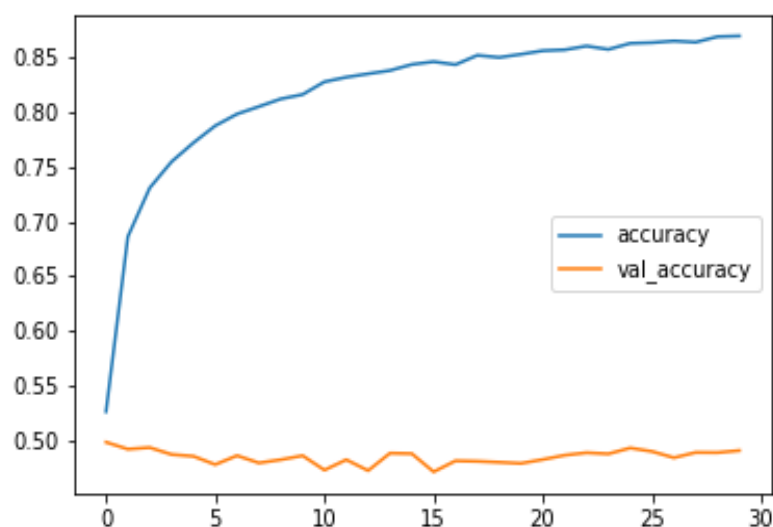
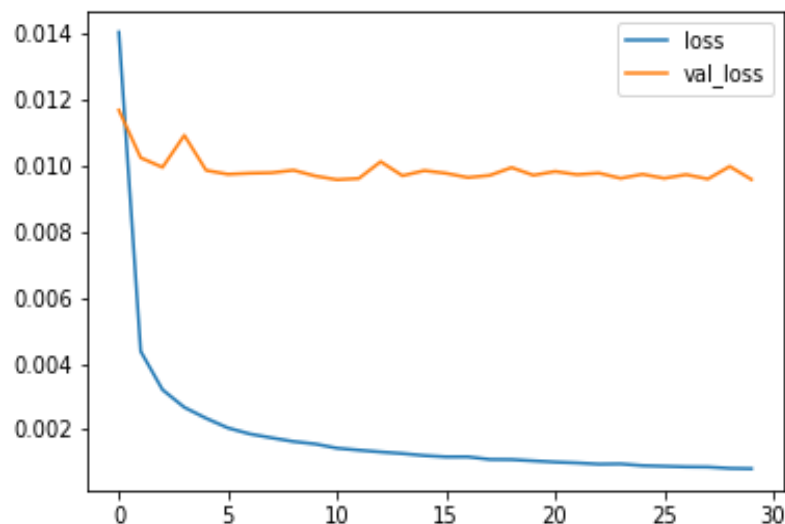
מסקנות הרצת המודל

ניסיון ראשון:

המודל רץ על 3000 תמונות כ-30 איפוצ'ים בזמן כולל של 18 דקות.

loss: 8.1659e-04, accuracy: 0.8694, val_loss:0.0096, val_acc: 0.4909

אפשר להבין מהתמונות בעמוד הבא 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.



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color

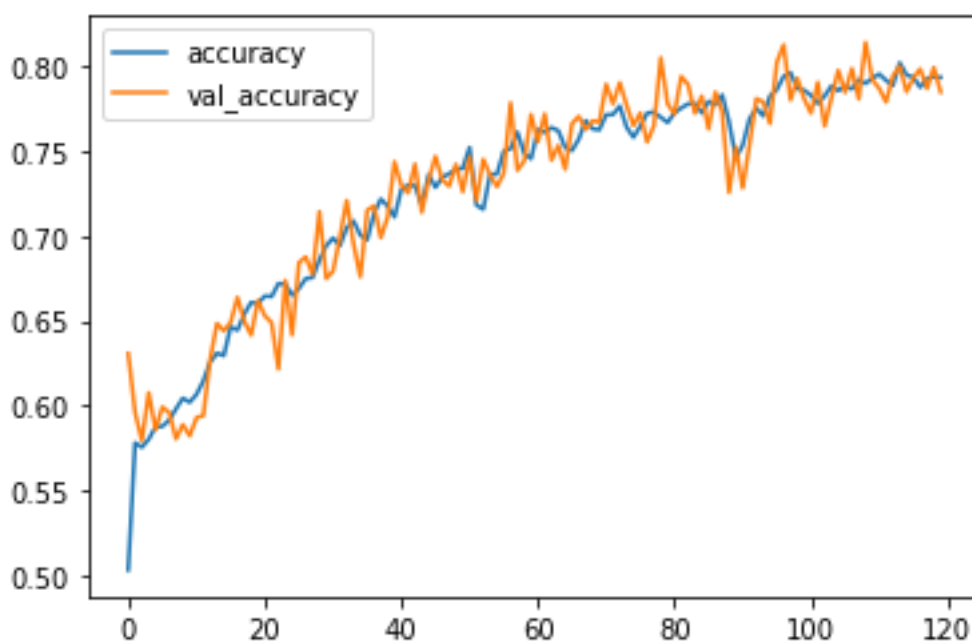
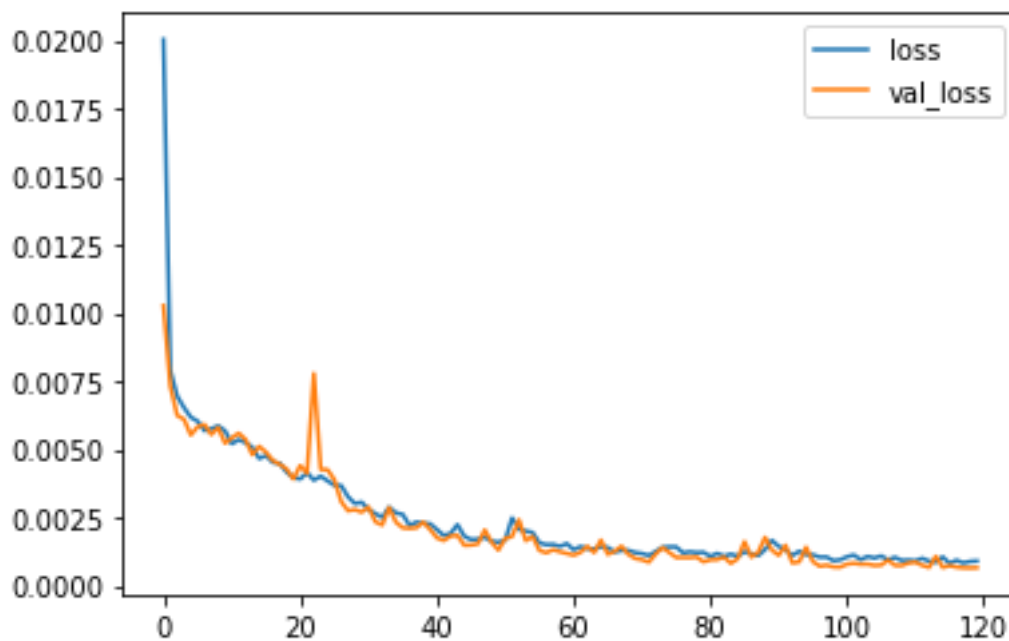


ניסיון שני:

המודל רץ על 3000 תמונות כ-120 איפוצ'ים בזמן כולל של 74 דקות.

loss: 9.0687e-04, accuracy: 0.7938, val_loss: 6.6080e-04, val_accuracy: 0.7851

הפעם השתמשתי בשיטת cross-validation כדי להימנע מהתאמת יתר באמצעות שינוי התמונות שמשמשות לאימון ולווידוא כל איפוף'. התוצאות נראות בגרפים ובתמונות, אך היה צורך במספר רב יחסית של איפוצ'ים כדי להגיע לנתונים האלה.



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



Input B/W



Original Color



Predicted Color



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 והתחוויר לי יותר מרגע לרגע כמה חסר מושג אני.

אני לא גאה בכך אבל ניסיתי להתעלם כמה שיותר מחוסר הידיעות שלי וניסיתי כמה שיותר פקודות ופעולות שמצאתי באינטרנט בניסיונות עוורים אולי להצליח במשימה שלי. לאחר חודש או שניים בעת ניסויי היותר-מוצלחים עם קוד תכנית 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 מכתובת סקריפט בפיתוח, 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.

לאחר סיום כתיבת הקוד וחצי מהספר, התמודדתי עם הבעיה האחרונה שלקחה לי הרבה מחשבה כדי לפתור – התאמת יתר. כפי שפורט בחלק על מסקנות המודל, הגעתי לתוצאות הסופיות לאחר מעבר בדרך חתחתים לא קצרה. הרעיון לפתרון הסופי הגיע כמובן מהאינטרנט, אך הביצוע היה מאתגר. לבסוף הגעתי לתוצאות מספקות, אפשר לראות שהמודל באמת יודע לעשות את העבודה שלו :)

אם הייתי מתחיל לעבוד על הפרויקט כיום, אני יודע שהייתי עושה עבודה הרבה יותר טובה. ראשית, אני מכיר את הנושא הרבה יותר טוב ואני מיועד 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 ועם הפנאי 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.

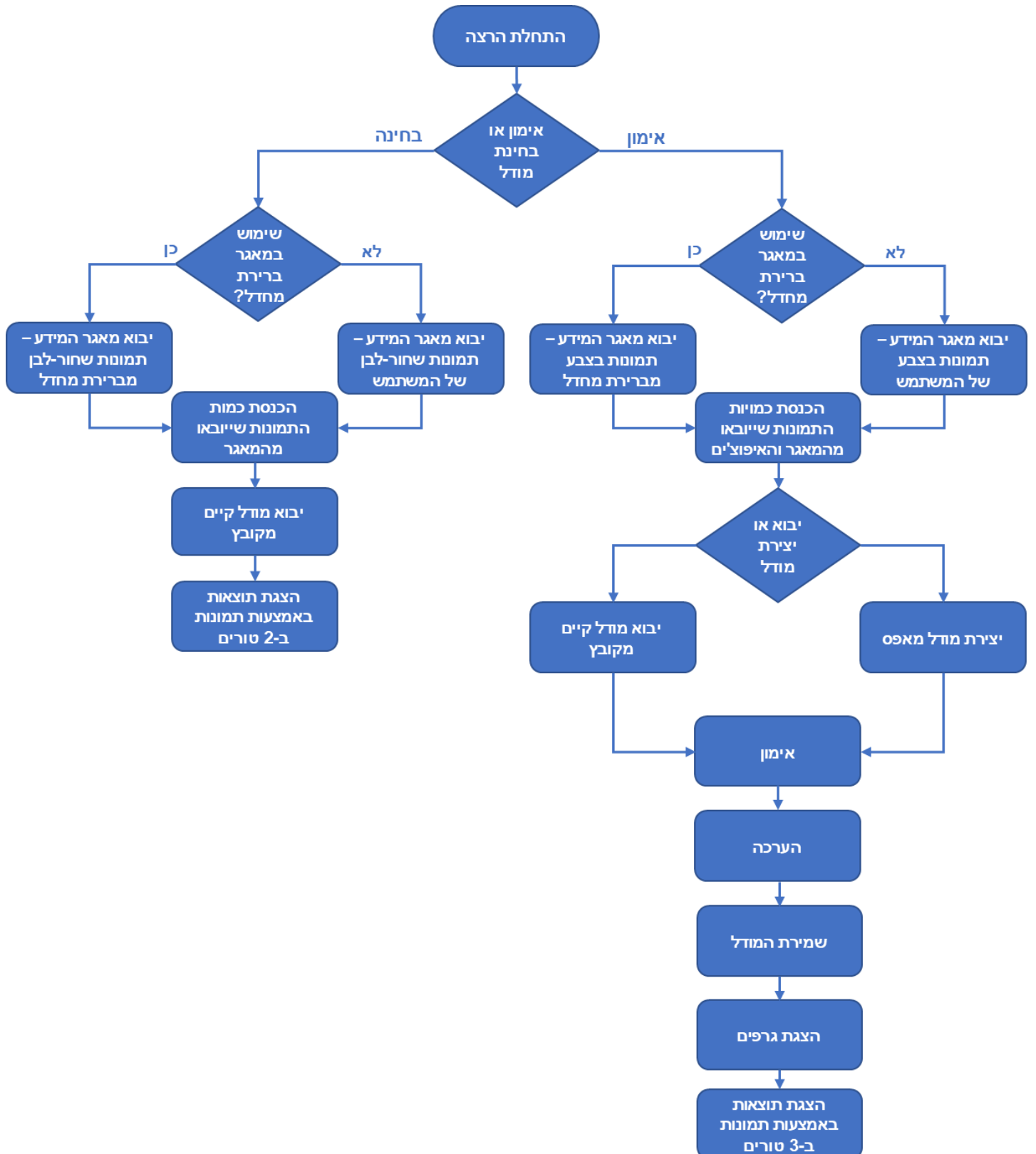
ביבליוגרפיה

- <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

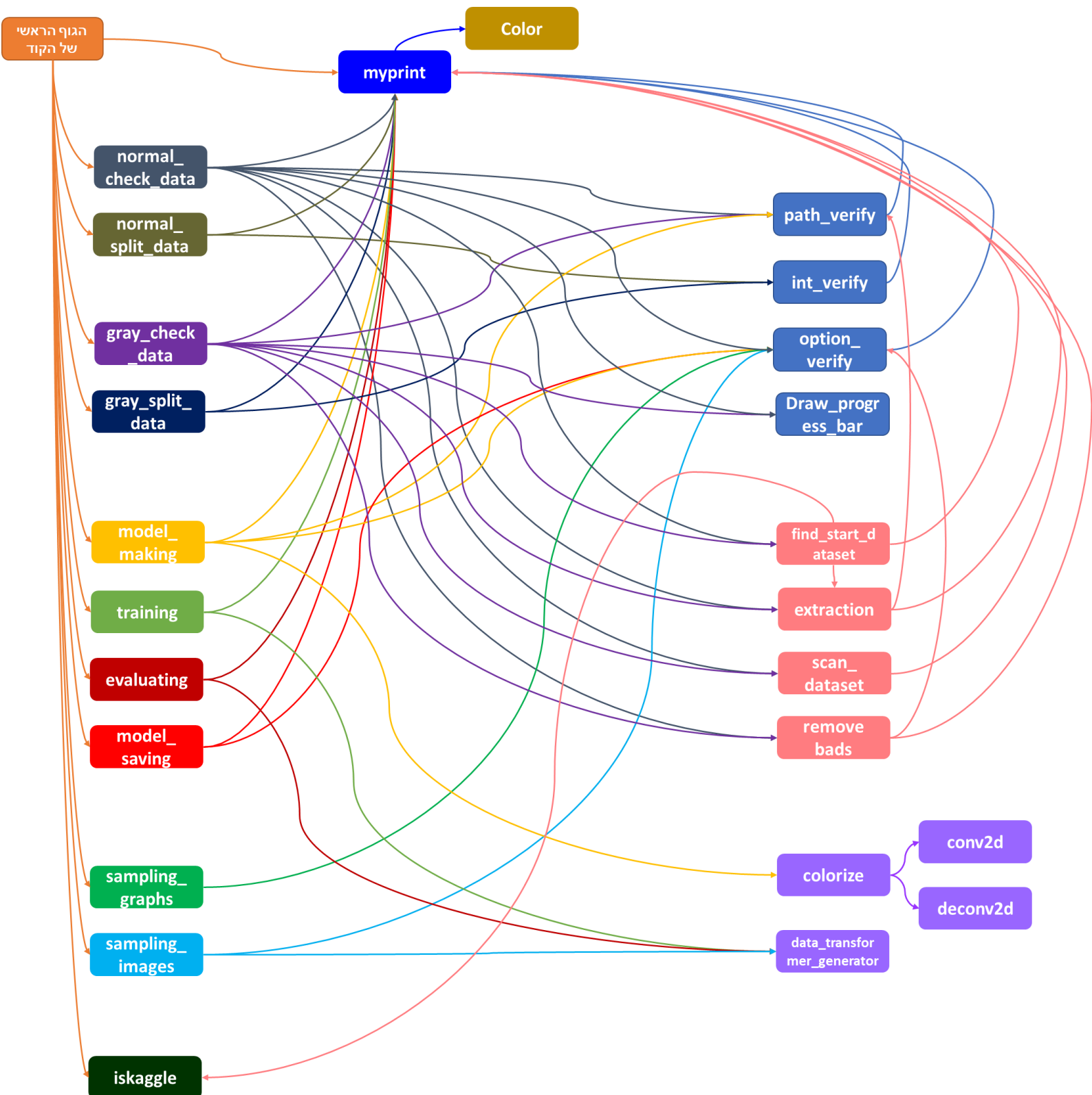
תרשים מסלול הרצה

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. שינויי הגודל מתרחשים כחלק מתהליך הקידוד האוטומטי ומתבצעים באמצעות השכבות conv2d, Concatenate, UpSampling2d.

