שם: עומר אלכסנדר , ת"ז: 313703415

# Font recognition project

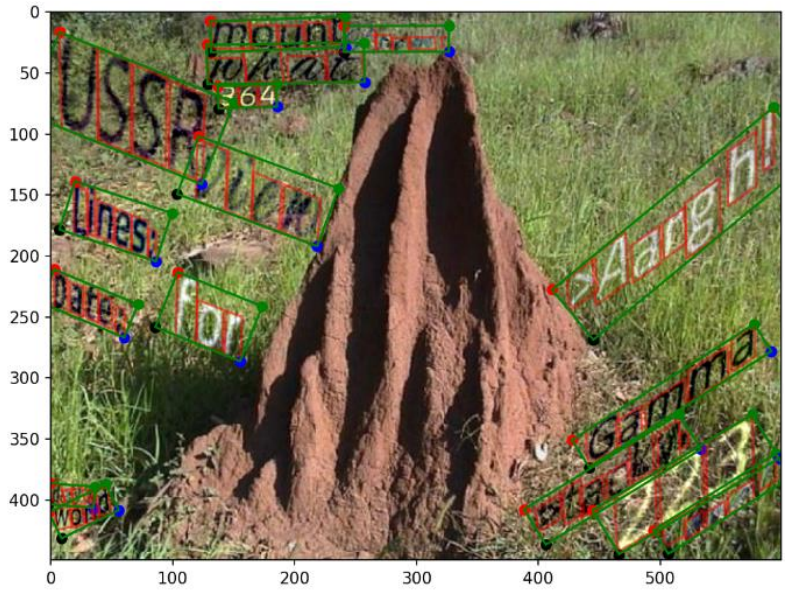
## רקע

הפרויקט הוא בעצם העבודת הגמר בקורס מבוא לראיה ממוחשבת באוניברסיטה הפתוחה- 22928.
בפרויקט נדרש לייצר מודל אשר יודע לקבל כקלט DATASET בפורמט HDF5 שמכיל את המידע הבא:

1. תמונות
2. שמות התמונות
3. טקסט – רשימה של מילים.
4. גבולות של המילים.
5. גבולות של כל אות.
6. פונט של כל מילה.


מטרת המודל היא לזהות את הפונט מבין 5 פונטים אפשריים:
את הזיהוי נצטרך לבצע על DATASET דומה למה שקיבלנו כקלט,
רק ללא ערכים של פונט.
התמונות עם המילים "מולבשות" בתוכם, נוצרו ע"י:
https://github.com/ankush-me/SynthText

ודוגמה לתמונה עם טקסט נראית כך:




ההישג הנדרש הוא לבסוף לייצר קובץ CSV המכיל את החלטת המודל בנוגע לזיהוי הפונט בכל מילה.

איך להשתמש בתוכנית – מפורט גם בקובץ READ.ME - קישור

בשביל להריץ את הפרויקט יש לפעול ע"פ השלבים הבאים:

1. לשמור את כל קבצי הפרויקט תחת אותה תיקייה ולשנות את PROJECT_PATH בmain.py :
   a. הקבצים שצריכים להיות:
      i. utils.py = תוכנית עם פרמטרים וקבועים לשאר התוכניות.
      ii. main.py = התוכנית הראשית את הפרויקט [אותה מריצים]
      iii. dataPrepper.py = תוכנית להכנת הDATA לאימון ולטסט.
      iv. train.py = תוכנית ליצירת ואימון המודלים.
      v. classify.py = תוכנית לסיווג הפונטים על הDATA המוכן.
      vi. visualizeResults.py = תוכנית להצגת התוצאות על הDATA עם הלייבלים.
      vii. קובץ אימון מסוג HDF5.
      viii. קובץ טסטס מסוג HDF5.

2. במידה ורוצים ניתן לשנות את השמות של קבצי האימון והטסט test_file_name /train_file_name בקובץ utils.py ע"פ הDATA שרוצים לאמן/לבדוק את המודל.
   a. ברירת המחדל הם הקבצים שקיבלנו:
      i. SynthText_train.h5 = קישור
      ii. SynthText_test.h5 = קישור
   b. מודל מאומן – קישור.

3. יש להריץ את התוכנית run_project() בעזרת התוכנית main.py.
   a. ברירת המחדל היא להשתמש במודל המאומן ללא הצגת התוצאות.
   b. במידה ורוצים לאמן מאפס יש להריץ: run_progect(need_train=True)
   c. במידה ורוצים להציג את התוצאות הבדיקה על סט האימון יש להריץ:
      run_progect(need_visualize=True)

4. את התוצאות נקבל בקובץ results.csv כפי שהתבקשנו בPROJECT_PATH שהגדרנו – קישור.

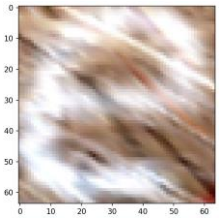
הרעיונות המרכזיים בתהליך העבודה

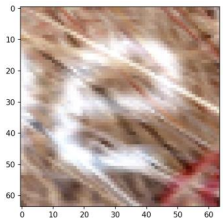
ממה שלמדנו במהלך הקורס ומה שחקרתי, כמובן שיש הרבה דרכים לבצע את המשימה ואת הפרויקט. אחרי שבדקתי המון אופציות ודרכים החלטתי על כמה עקרונות בשביל לייצר את המודל הכי טוב עבור המשימה הזו:

1. הכנת הDATA עליו נאמן את המודל.
2. שימוש במודל מבוסס רשת נוירונים – CNN.
3. פיצול המודל הקיים למודלים מותאמים אישית לכל אות בנפרד.
4. יצירת "משקול למודלים" אשר מתעדף את המודלים השונים ומכריע פונט נבחר לכל מילה
5. שיפור המודל בעזרת הרחבת הDATASET לאימון המודל שבניתי בעזרת gen.py שלקחתי מ:
   a. https://github.com/yuvalshi0/SynthText/blob/python3/gen.py

## הכנת הDATA עליו נאמן את המודל

ברור שיש חשיבות גדולה לDATA עליו נאמן את המודל. הדרך הנאיבית היא פשוט לחתוך את האותיות לפי הBB ולאמן את המודל על התמונות הללו עם הידע של איזה פונט מדובר, כמו שאפשר לראות בתמונה השמאלית.





אחרי שהסתכלתי על החיתוכים הבנתי שיותר קל להבין את האות בצורה טובה יותר כשמסתכלים קצת יותר בzoom out ולכן הוספתי לכל אות שוליים רחבים יותר בשביל להבהיר ולמקד את האות כפי שאפשר לראות בתמונה הימנית.

כעת חשוב לשמור את כל המידע שנרצה להשתמש בהמשך בצורה נוחה בקובץ readySynthText.h5:
פיצלתי את התמונות המוכנות ל3 קבוצות: tarin, test, val.
כל קבוצה מכילה 2 קבוצות: data, indexing:

1. data
   a. Images = תמונה מסודרת של האות.
   b. Font_index = האות שבתמונה מיוצג ע"י אינדקס של הפונט מfont_index_map.
   c. Chars = האות שבתמונה מיוצג ע"י ערך הASCII של האות.

בעזרת מידע זה אנחנו שומרים את המידע הרלוונטי שנרצה להשתמש בו בהמשך לאחר שטיפלנו במידע. המידע מכיל את התמונות ואת הלייבלים שאיתם נאמן בהמשך את המודל. הסיבה שאני שומר גם את הערך הASCII של כל אות היא בשביל בהמשך לחלק את השיפור המודל הכללי למודלים ספציפיים לפי כל אות [שאת החלוקה נבצע בהמשך לפי ASCII].

2. Indexing
   a. By_char = כל התמונות בעלי אותה אות [ע"פ ערך ASCII]
   b. By_word = כל התמונות בעלי אותה מילה
   c. By_img = כל התמונות שנלקחו מאותה תמונה מקורית, ע"פ סדר התמונות.

בעזרת מידע בהמשך יהיה לנו מאוד נוח לגשת למידע הרלוונטי שאנחנו רוצים ע"פ תמונה/מילה/אות.
את המוטיבציה לזה נראה בהמשך בפירוט, הרעיון הכללי בהתחלה היה לשמור את מירב המידע שנתנו לנו בהתחלה שנוכל להשתמש בו בהמשך.

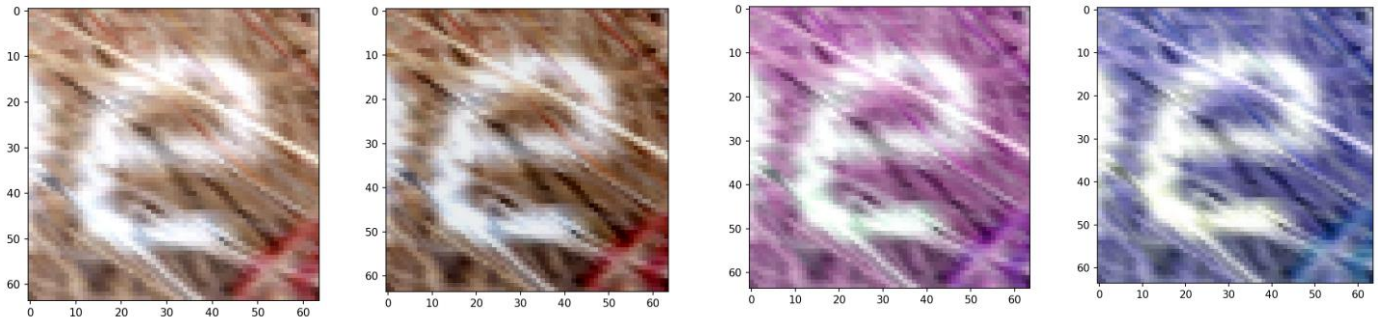
## שימוש במודל מבוסס רשת נוירונים – CNN

ממה שלמדנו בקורס ובחקר שעשיתי בנושא, הבנתי שאני רוצה להשתמש במודל מבוסס CNN משום שזוהי הדרך הכי מתקדמת [ולא מורכבת מידי עם זמני ריצה רלוונטים לפרויקט].

משיקולי זמנים לא הספקתי לנסות לייצר בעצמי DATASET נוסף רלוונטי, בשביל להגדיל את הDATA עליו אני אאמן את המודל שלי.
בגלל שהDATASET שלנו הוא מוגבל ניסיתי לחשוב איך אני מונע מצב של overfitting.

לאחר שהסתכלתי על התמונות שקיבלנו הבנתי שיש משמעות גדולה לרקע שמסביב לתמונות ושעליי לנסות למנוע מהמודל "ללמוד אותם" בשביל שעל DATA אחר עם סוג אחר של תמונות רקע המודל שלי יניב תוצאות טובות רק ביחס לאותיות עצמן.

לכן הבנתי שעליי להכניס למודל שכבה שתבצע מניפולציה אקראית על הצבע, בהירות, ניגודיות ורוויה וגוונים. בכך שהכנסתי שכבה כזאת במודל בעצם אני "מחדש" את סט האימון בכל פעם.



לאחר הרבה ניסיום ונסיונות שונים הגעתי למבנה מודל כזה:

| Layer (type) | Output Shape |
|---|---|
| randomcolordistortion (RandomColorDistortion) | (None, 64, 64, 3) |
| conv2d (Conv2D) | (None, 64, 64, 64) |
| max_pooling2d (MaxPooling2D) | (None, 32, 32, 64) |
| conv2d_1 (Conv2D) | (None, 27, 27, 32) |
| max_pooling2d_1 (MaxPooling2D) | (None, 13, 13, 32) |
| dropout (Dropout) | (None, 13, 13, 32) |
| conv2d_2 (Conv2D) | (None, 11, 11, 64) |
| max_pooling2d_2 (MaxPooling2D) | (None, 5, 5, 64) |
| dropout_1 (Dropout) | (None, 5, 5, 64) |
| flatten (Flatten) | (None, 1600) |
| dense (Dense) | (None, 128) |
| dropout_2 (Dropout) | (None, 128) |
| dense_1 (Dense) | (None, 128) |
| dropout_3 (Dropout) | (None, 128) |
| dense_2 (Dense) | (None, 5) |

השתמשתי בפונקציית אקטיבציה Leaky ReLU , שבניגוד לפונקציה ReLU הנפוצה ששם כאשר יש קלט שלילי הגרדיאנט של הפונקציה הוא 0 , בפונקציה שבחרתי הוא אינו מוריד לגמרי את הגרדיאנט עבור הקלטים השליליים ובכך בעצם עוזר בלהתגבר על הקושי עם קלטים שליליים.

בנוסף, השתמשתי בטכניקת learning rate decay אשר בעזרתה המודל מקטין את קצב הלמידה כאשר המודל מגיע ל"פסגה" מסויימת בביצועים שלו, דבר זה מאפשר לימוד בצעדים קטנים יותר אל עבר האופטימום.

אציין כי בהתחלה התחלתי לעבוד עם transform learning בשימוש במודלים קיימים כמו ResNet וVGG וקיבלתי תוצאות חיזוי לא מספיק גבוהות אז עברתי לאופציה של בניית מודל שהביא לבסוף לתוצאות הרבה יותר טובות.

<u>פיצול המודל הקיים למודלים מותאמים אישית לכל אות בנפרד</u>

שדרוג נוסף שחשבתי שניתן לעשות זה לפצל את המודל הקיים שאימנו כבר ולחלק אותו למודלים שונים ע"פ האותיות השונות. המוטיבציה לתוספת הזו נבעה מזה שהסתכלתי על התמונות של האותיות בנפרד וראיתי שלרוב האותיות יש מאפיינים שונים בולטים יותר בין הפונטים השונים מאשר ההכללה של כל האותיות כקבוצה אחת.
למשל האות 'a' בין הפונטים השונים יש מאפיינים ספציפיים שונים מאוד בין הפונטים לעומת אותיות אחרות ויהיה חבל לא להשתמש במידע הזה.

כשניסיתי להפריד את המודלים, בדקתי את המודל המקורי המאומן – global model. וראיתי שיש אותיות מסויימות שעדיין המודל הכללי מניב תוצאות טובות יותר ולכן הבנתי שיש מקרים בהם עדיף להשתמש במודל הכללי. לאחר ניסיונות ובדיקות החלטתי לייצר מודל ספציפי לאות מסויימת רק אם:

1. יש לפחות 20 תמונות של אותה אות עליה המודל יוכל להתאמן.
2. לאחר האימון לאות הספציפית דיוק החיזוי צריך להיות טוב יותר מהמודל הכללי.

את המודלים השונים שמרתי בנפרד בתיקייה models כאשר לכל אות ספציפית שעמדה בהגדרות שפירטתי למעלה יש תיקייה בשם הייצוג הASCII של האות עם המידע של המודל המשופר שלה.

<u>יצירת "משקול למודלים" אשר מתעדף את המודלים השונים ומכריע פונט נבחר לכל מילה</u>

המוטיבציה היא להשתמש בעובדה שבכל מילה יש פונט אחד. בכך בעצם במידה ויש חיזוי פונטים שונים בין אותיות שונות באותה המילה יהיה לנו כלי מתוחכם להכריע בצורה מיטבית מה הפונט הנבחר של המילה ובכך אנחנו משפרים את אחוזי הדיוק של התוכנית.

בזמן יצירת המודלים הספציפיים אני יצרתי גם קובץ weights.h5 אשר מכיל בתוכו DATASET ע"פ השמות של מודלים הנוספים שיצרתי [ע"פ ייצוג הASCII שלהם]. הערך שהוא מכיל הוא בעצם נקבע ע"י train.py בזמן האימון והוא בערך של (2 – accuracy) / accuracy.
ערך זה בעצם מייצר עבורנו אינדיקציה לכמה המודל איכותי ומביא תוצאות טובות, ובכך הוא יתעדף בהמשך את המדלים שהביאו תוצאות טובות יותר.

בשלב הסיווג, כל אות משתמשת במודל שהכי מתאים עבורה ולבסוף מתבצע נירמול של התוצאות ביחס למשקלים ששמרנו בקובץ weights.h5 עבור כל מילה ולבסוף נבחר פונט אחד למילה כולה.

שיפור המודל בעזרת הרחבת הDATASET לאימון המודל שבניתי בעזרת gen.py שלקחתי מ:

https://github.com/yuvalshi0/SynthText/blob/python3/gen.py

בעזרת התוכנית הזו המבוססת על קוד המקורי שקיבלנו, ייצרתי עוד DATASET בפורמט זהה למה שקיבלנו כקלט, בכך הצלחתי לאמן את המודל ושפר אותו ובכך גם להימנע מoverfitting עבור אותיות שלא היה עבורם מספיק DATA.

יש לציין כי עדיין לא הצלחתי להגיע למצב שבו יש מספיק DATA בשביל לייצר מודל ספציפי עבור כל אות אבל זה כן שיפר ותרם למודלים הקיימים והמעטים שנוספו.
בנוסף אציין שבמידה והייתי פחות בעומס במהלך הכנת הפרויקט הייתי מכין הרבה יותר DATA בשביל להגיע למצב שבו לכל אותו יש מודל אופטימלי נפרד, אך משום שלא הצלחתי להשקיע בזה מספיק זמן והייתי מרוצה מהתוצאות הנוכחיות הפסקתי עם הניסיונות הללו.

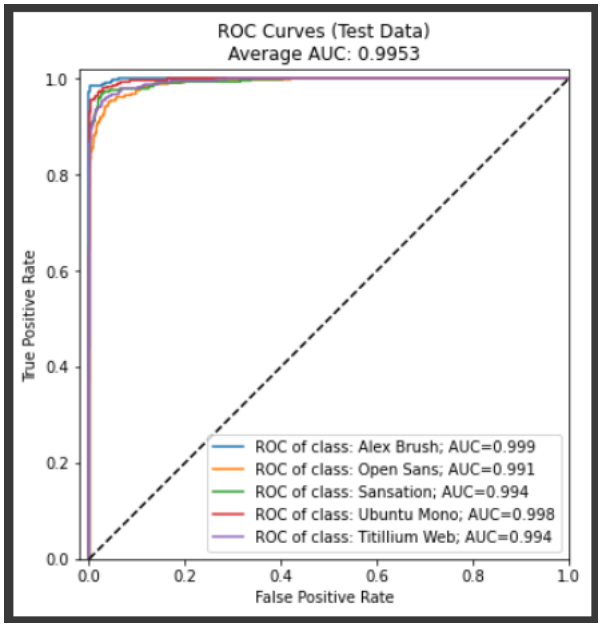
תוצאות המודל על DATASET שקיבלנו

כמו שהסברתי קודם פיצלתי את התמונות המוכנות ל3 קבוצות: tarin, test, val.
בעזרת visualResults.py , חישבתי לאורך העבודה כל הזמן את התוצאות אליהם הגעתי על קבוצת הtest.
(יש לציין כי החלוקה לקבוצות מתבצעת בצורה רנדומלית ולכן בכל הרצה התוצאות יהיו מעט שונות)

עם המודל הסופי והמאומן התוצאות היו **ACCURACY 0.951 על ה TEST**

Confusion Matrix (Test Data)

| | Alex Brush | Open Sans | Sansation | Ubuntu Mono | Titillium Web |
|---|---|---|---|---|---|
| Alex Brush | 1251 | 0 | 0 | 15 | 3 |
| Open Sans | 8 | 988 | 29 | 10 | 47 |
| Sansation | 10 | 49 | 1209 | 17 | 25 |
| Ubuntu Mono | 3 | 14 | 14 | 1408 | 12 |
| Titillium Web | 0 | 27 | 12 | 5 | 1079 |




| | |
|---|---|
| train accuracy | 0.999729 |
| test accuracy | 0.951885 |

Sensitivities (Test Data)

| | 0 |
|---|---|
| Alex Brush | 0.983491 |
| Ubuntu Mono | 0.967698 |
| Sansation | 0.956487 |
| Titillium Web | 0.925386 |
| Open Sans | 0.916512 |