פרויקט מסכם - זיהוי של כותב למידה עמוקה - שנת תשפ״ג

החוג למדעי המחשב
מכללת הדסה ירושלים

מרצה: ד״ר יורם יקותיאלי

דוד אלחנדרו ממן - 327437422
יונתן לייבל - 203654710

ספטמבר 2023

תוכן עניינים:

מבוא

משימת זיהוי כותב על פי כתב יד היא חלק מעבודת חוקרי הזיהוי הפלילי. מעבדת השוואת מסמכים וכתבי יד מבצעת מחקרים שונים כדי לשפר את יכולות הזיהוי ולתת תוקף מדעי לשיטות שונות. במסגרת מחקר כזה התבקשו כמה מאות אנשים לכתוב טקסט ידוע על דפי שורות שחולקו להם במעבדה. ההנחיות היו להעתיק את הטקסט הידוע בכתב ידם הרגיל.

עם ההתקדמות הדיגיטלית מעצבת במהירות טכניקות חקירה מודרניות, היכולת לנתח ולהבין כל צורה של ראיות הופכת מכרעת. בין שלל סוגי הראיות, מסמכים בכתב יד מופיעים לעתים קרובות כעדים אילמים המחזיקים במידע חיוני. הערך שלהם, לעומת זאת, מותנה בדייקנות שבה ניתן לזהות את המחבר. בסביבה ידנית, משימה זו דורשת מאמצים ממצים, מומחיות ורצופה באי ודאויות.

מתוך הכרה בצורך ההכרחי לשפר את הדיוק והמהירות של ניתוח כתב היד, שיתפנו פעולה עם מחלקת המשטרה כדי לרתום את הפוטנציאל של למידה עמוקה בתחום זה. המחקר סיפק מערך נתונים עשיר של למעלה מ-400 מסמכים בכתב יד שסופקו על ידי המחלקה. המשימה שלנו הייתה לפתח מודל למידה עמוקה חזק שיוכל לזהות את הכותב מאחורי פיסת טקסט ספציפית בכתב יד, ובכך להגדיל את היכולות של החוקרים.

מסמך זה מציע צלילה עמוקה לתוך הפרויקט שלנו. הוא מבהיר את המתודולוגיות שלנו, את תכנון הרשת העצבית, את האתגרים העומדים בפניהם, ההישגים שלנו וההשלכות על חקירות עתידיות. כשאתה מטייל בדפים אלה, תוכל לראות כיצד המיזוג של טכניקות חקירה מסורתיות עם גבול למידת המכונה יכול לחולל מהפכה בבדיקת מסמכים הכתובים בכתב יד.

זוכיות יוצרים

מטרת הפרויקט

במסגרת אקדמית, שבה השאיפה לידע מתכנסת עם יישום טכנולוגיות מתקדמות, המשימה שהוצבה לפנינו הייתה פשוטה ועמוקה כאחד: לפתח מערכת המסוגלת לזהות כותב מתוך מערך נתונים ספציפי שסופק.

האתגר הבסיסי לא היה רק על זיהוי דפוסים או עיבוד נתונים, אלא על ניצול הפוטנציאל העצום של יכולות למידה עמוקה כדי להבין את שלל הניואנסים והדקויות המגדירים סגנונות כתב יד בודדים.

מאגר הנתונים, המורכב מאוסף אצור של למעלה מ-400 מסמכים בכתב יד, הפך לסלע היסוד למאמץ זה.

המשימה שלנו הייתה להפוך את התמונות הסטטיות הללו לנקודות נתונים דינמיות, וליצור נוף עצבי שבו ניתן לנתח, להבין ולהשוות כל שבץ, עקומה ונקודת לחץ. על ידי השגת זאת, לא רק נצמד למפרטי הפרויקט אלא גם דוחף את הגבולות של מה שאפשר בתחום זיהוי כתב היד.

באמצעות תרגיל זה, המטרה שלנו הייתה למזג את הקפדנות האקדמית של הקורסים שלנו עם ההשלכות המעשיות של יישומים בעולם האמיתי, להציב רף לפרויקטים עתידיים בתחום זה.

נתונים

בסיס הנתונים:
נתון לנו גישה לבסיס נתונים שמכילה 6 תקיות:
1. תיקיה שבה התמונות המקוריות
2. תיקיה שבה התמונות לאחר יישור כיוון והמרה מצבע לרמות אפור
3. תיקיה שבה התמונות החלקה Median של הקודמת והמרה ל-BW
4. תקייה שבה התמונות אחרי הסרת קווים, התמונות BW.
5. תיקייה שבה התמונות שהן תוצאה מהעיבוד בתיקיה 4 ששימש כמסכה לדגימת פיקסלים מהמקור המיושר.
6. תיקייה שבה קבצים מסוג mat. שכל קובץ מכיל מידע עבור תמונה של כותב.

הנחות ידועות על התמונות:
- גודל התמונות הינו קבוע לכל תמונה שהוא בסביבות 7000 x 5000, סריקה 600 DPI באיכות טובה.

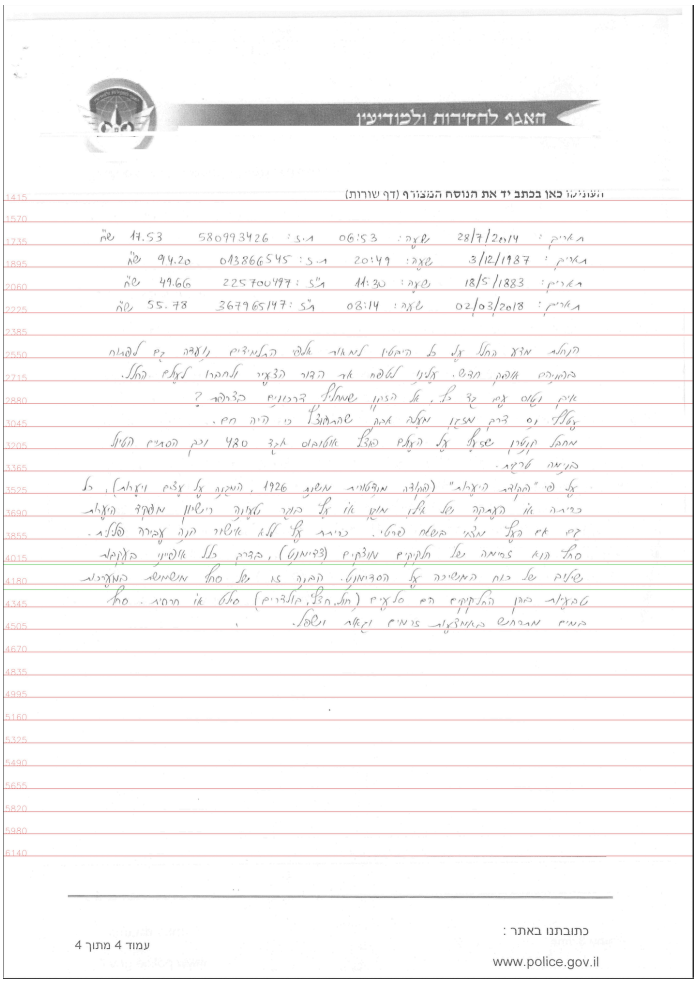
- כל תמונה היא בעל אותו תבנית:
  1. חלק עליון - Header - בגודל של בערך 1/8 הגובה הכולל.
  2. מרכז - בגודל של בערך 6/8, בתוכו שורות שהרווחים ביניהן קבועות ובקירוב 160 עד 200 פיקסלים.
  3. חלק תחתון - Footer - בגודל של בערך 1/8.

- הדף הנסרק הונח כיאות וזווית הסיבוב שלו לא עולה על 2 מעלות לכל כיוון.

- על הדף, על החלק המרכזי, יש או אין טקסט ידני.

- הדף לא כולל צביעה של שטחים גדולים או כל הפרעה אחרת גדולה.

- לכל תמונה מוגדרת שורת מבחן אשר מוגדרת בתקייה עם קבצי mat., שאר התמונה נתונה לשיקול דעתנו.



האגף לחקירות ולסודיות

הקליטו כאן בכתב יד את הנוסח המקורי (דף שורות)

תאריך:	28/7/2014	שעה:	06:53	מ.ט:	580993426	שח 17.53
תאריך:	3/12/1987	שעה:	20:49	מ.ט:	0438665 45	שח 94.20
תאריך:	18/5/1883	שעה:	11:30	מ"ל:	225700499	שח 49.66
תאריך:	02/03/2018	שעה:	08:14	מ"ל:	367965147	שח 55.78

הנושא מדגל בחלל על כל היבטיו למאזר אלפי הבעיות ולפיה יש לפעול בהתאם אופיה הדיס. עלינו לשים את הדגר הבאיר לתהרו לעלים הללו. איך נקיים יחס כך כל, אל הכיון שמולל דרכנים בלהלה?
עלול יש דרים שאינו מגלה אהב שהתמלל מי היה חיי.
מהחל חקירתן שעלל על העלים הללו אוקטובר מיכ 420 ובכן הסברים הנוגל הנוגאה לרגע.
על פי "הקודם הראשון" (הקודם מפקדה משנת 1926, המבוא על עצים ויצוא), כל בריאה אז הקמהה כל אלו אשר אז על בתר עצוב ראשון מלבדך הנעלה. יש אם בעל מלאי בשמש הרלי. בריאה על לא אישור הנה עבירה פלילה. סתל הנוא צמיאה של תקקיים אוכלים (דינוזאור), בדרך כלל אופיין בלקבל שינוי של כוח המשיכה על הסדומלל. הבנה זו על סתל משמשת המעגיבה לעבעלה בהן הלקקים הם סלעים (חול, חלל, חלדרים) סלע או חרסת. סתל בנוים מברתש בתמלעה צמים וקלה ושפל.

כתובתנו באתר:
www.police.gov.il

עמוד 4 מתוך 4

דוגמה של תמונה שבה נצבע:

- פסים אדומים: שורות חלק המרכזי.
- פסים ירוקים: טקסט בין פסים הללו הוא תוכן שורת המבחן.

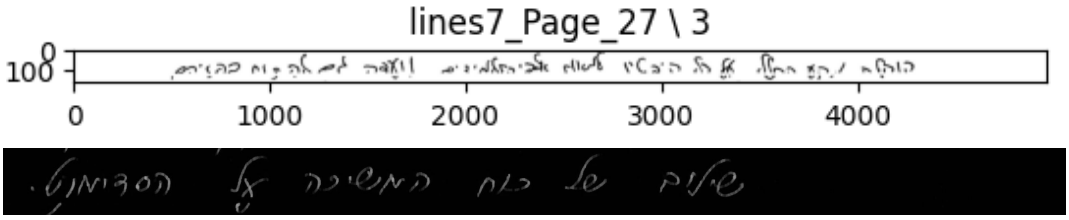
שטח מבחן ושטח אימון וולידציה

בחירת אסטרטגית הפיתוח

במהלך הפרויקט אחת ההתמודדויות המרכזיות הייתה על איזה סוג של תמונה לעבוד, אחרי איזה עיבוד, וכיצד לחלק אותה על מנת ליצור עוד מידע כך שנגיע לדיוק גבוה יותר בזיהוי.

ביצענו מספר שיטות של חלוקה על מנת ליצור מידע עבור כל כותב ולאחר מכן בחננו אותם במודל, בנוסף לכל שיטת חלוקה היינו צריכים לבחון אותה לכל סוג של עיבוד.

<u>ניסוי ראשון של חלוקה - שיטה איטרטיבית וחלוקה קבועה</u>:
במהלך ניסוי הזה מכיוון שנתן לנו מקומי השורות אז החלטתנו לעבור האופן איטרטיבי ובכך לבחור שורות שאינן ריקות (כלומר יש להן תוכן כלשהו), אך נתקלנו בבעיה אם קיים או לא קיים קווים אופקים של הדף.
במהלך הניסוי הסרנו את הקווים אופקים אך זה גרם לשאר התוכן לאבד פיקסלים, כלומר איבדו מהצבע שלהם.
בצורה שבו הסרנו את הקווים היה שם פיקסל לבן במקום הקו לכן בנוסף לאבד פיקסלים של תוכן היה מפריד את התוכן במיקום שבו עבר מעל הקו.



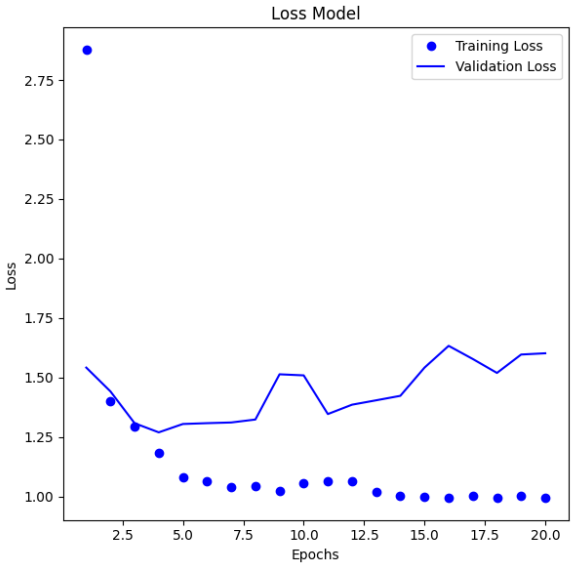
הסרת קווים השאיר "חור" מפריד את המילים.

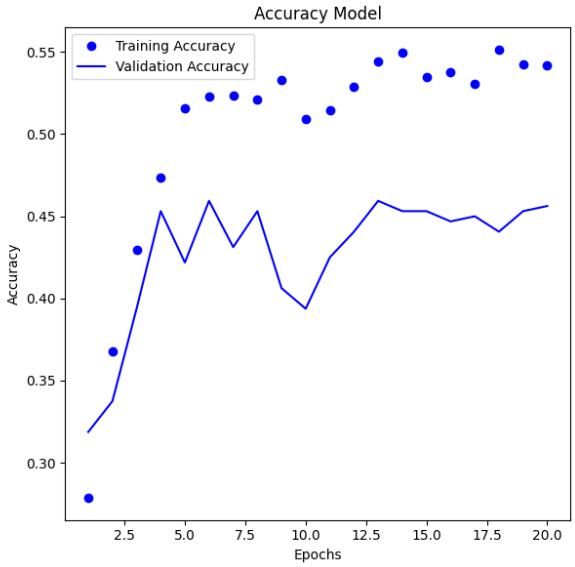
בניסוי הזה ניסינו שהמודל התמודד עם שורות שלמות אך המודל התקשה מאוד לעבוד עם גדלים כאלה וזה התבטה בתוצאות האימון ומבחן עם דיוק נמוך מדי.

<u>ניסוי שני של חלוקה - שיטה איטרטיבית חלוקה לתתי תמונות:</u>
ניסוי הזה הוא המשך של ניסוי הקודם אך הפעם כל שורה חולקה לתתי תמונות בגודל 292 x 180.

מכיוון שאנו עוברים בצורה איטרטיבית על השורות ולא כל השורות בעל אותו גובה, נזכיר שכל שורה בעל גובה 160 עד 200, אז היינו צריכים קודם לחתוך באורך 292 פיקסלים ואז להשלים עם הוספת פיקסלים לכל תת תמונה קטנה מ180 פיקסלים.

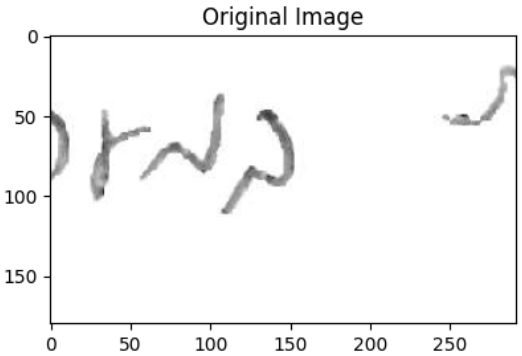
על ידי חלוקה כזאת הגענו לתוצאות יותר טובות באימון ובמבחן, במהלך אימון אחרי 20 תקופות הגיע לדיוק של 0.54 ובמבחן הגיע לדיוק של 0.5 על 6 מחלקות.
אמנם כמות קטנה של מחלקות ואי לכך גם כמות קטנה של מידע אך התקדמות בדיוק הלמידה, אך לאחר הצגת התקדמות הלמידה של המודל לטובת הבנה ומחקר להמשך ניסוי עם כמות יותר גדולה הגענו למסקנה שמשהו לא תקין בתהליך.

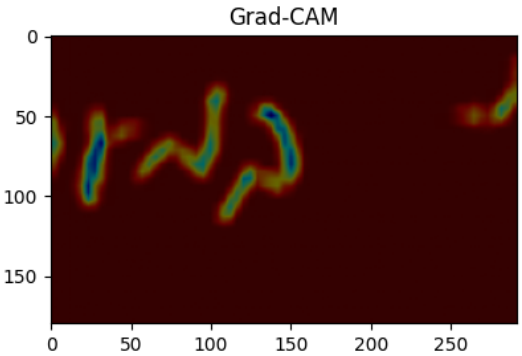




למרות זאת, החלטנו לבצע ניסוי נוסף עם יותר מחלקות, ביצענו עד 63 מחלקות והתוצאות דיוק המבחן היו כמעט זהות, עד קצת יותר מ0.6 בדיוק המבחן.

לטובת הבנת הלמידה של המודל רצינו לבדוק במה המודל מתעניין בכל תמונה ולאחר הצגת Heatmap הבננו שהמודל כנראה לא מבין במה להתבוננן בכל תמונה.





צד שמאל - תמונת האימון, צד ימין - נראה שהמודל מתעניין בהכל חוץ מהאותיות.

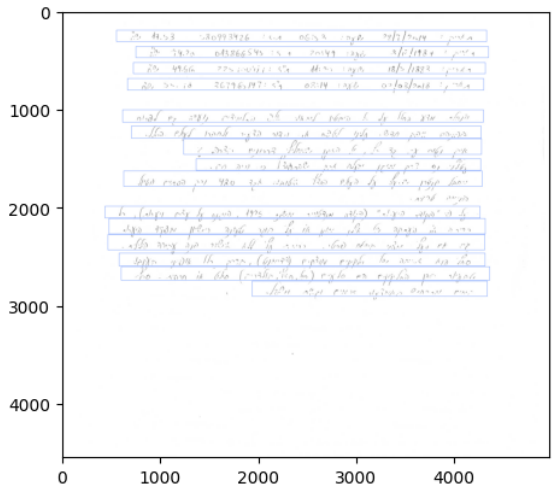
ניסוי שלישי - הצלחה באימון, כישלון במבחן:
בשלב זה היו לנו כמה לבטים כיצד להצליח לתוצאות יותר טובות.
האם הסרת הקווים משפיעים על הלמידה של המודל, אם התתי תמונות לא בעלות מספיק מידע שהמודל יבין במה להתבונן, האם יש דרך אחרת להגיע לחיתוכים יותר שלמים מבחינת מידע?

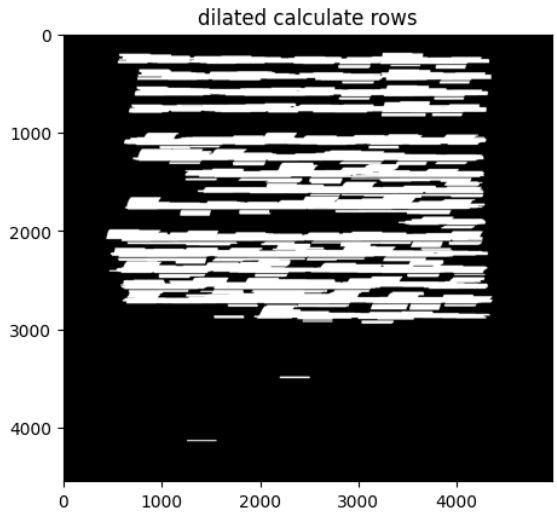
מכיוון שברצוננו היה להגיע למודל שאכן יודע לעבוד עם כל סוג של תמונה מכל עיבוד מצאנו כמה דרכים להתמודד עם הקשיים:

- מצאנו הודות למאמר של Victor Zaguskin שעל ידי התמרות פורייה ניתן לטשטש את הקווים אופקים של התמונות ללא להפריד את המילים.
על מנת להגיע לתוצאה טובה של הסרת קווים היה צורך לבצע התמרת פורייה מספר פעמים, לא ניתן להגיע למסקנה איזה מספר קבוע צריך לבצע, אך כל ביצוע כזה לוקח לא מעט זמן.

- מדוע היה חשוב לנו להסיר את הקווים? כי מצאנו דרך נוספת להפדרת תוכן.
הודות להסבר של Mohd Asif Momin שבו מדגים על ידי פעולות של טשטוש גס ולאחר מכן טשטוש עדין יותר יכול לדעת מה מיקום של כל מילה בטקסט.
אם לא היינו מסירים את הקווים אז בטשטוש הראשון, הטשטוש הגס, היה מתייחס עליו כשורה בעל תוכן.

מבחינת תהליך על התמונות היה נראה מאוד מבטיח.



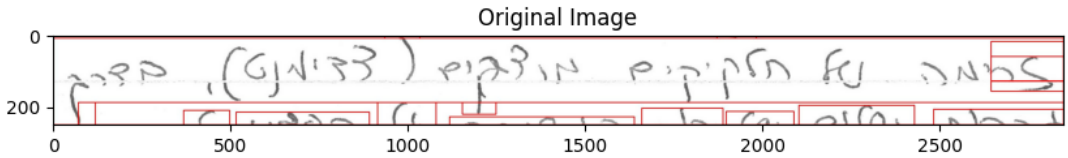


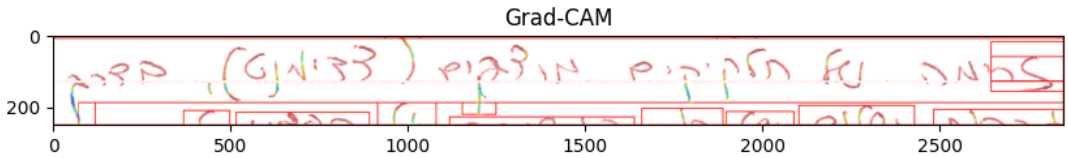
תוצאה של טשטוש גס וסימון שורות בעלות תוכן.

ביצענו ניסוי עם מודל שמקבל גדלים של תמונות לא ידוע:

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_45 (Conv2D)          (None, None, None, 32)    4736

 max_pooling2d_45 (MaxPoolin  (None, None, None, 32)   0
 g2D)

 conv2d_46 (Conv2D)          (None, None, None, 64)    18496

 max_pooling2d_46 (MaxPoolin  (None, None, None, 64)   0
 g2D)

 conv2d_47 (Conv2D)          (None, None, None, 128)   73856

 max_pooling2d_47 (MaxPoolin  (None, None, None, 128)  0
 g2D)

 conv2d_48 (Conv2D)          (None, None, None, 256)   295168

 max_pooling2d_48 (MaxPoolin  (None, None, None, 256)  0
 g2D)

 global_average_pooling2d_15  (None, 256)              0
  (GlobalAveragePooling2D)

 dense_30 (Dense)            (None, 256)               65792

 dropout_15 (Dropout)        (None, 256)               0

 dense_31 (Dense)            (None, 28)                7196

=================================================================
Total params: 465,244
Trainable params: 465,244
Non-trainable params: 0
_________________________________________________________________
```

הניסוי בוצע על 28 מחלקות ובמהלך האימון הגיע לדיוק של 0.44 אחרי 32 תקופות.





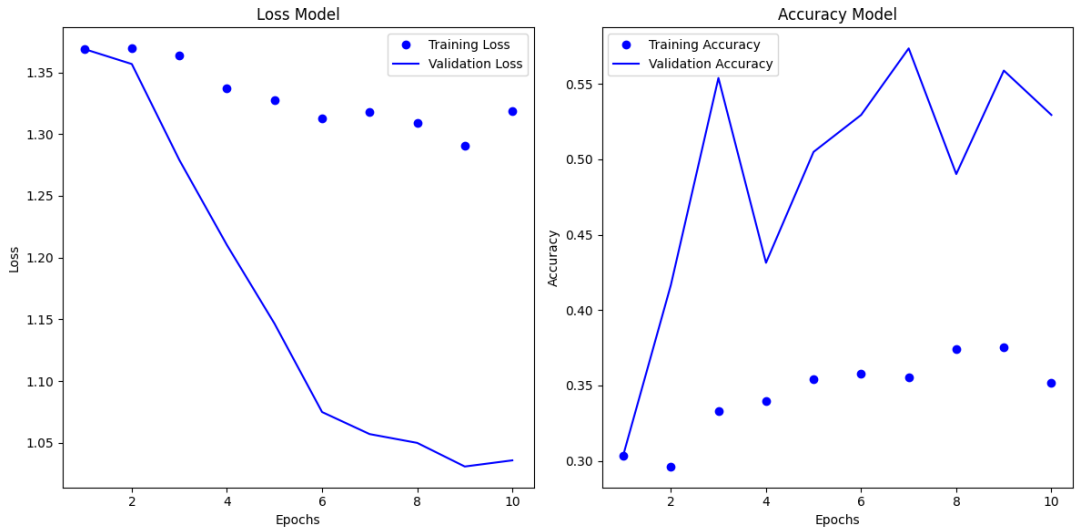
הפעם נראה שמודל הצליח להבין במה להתמקד.

כאשר הגענו לשלב של ביצוע מבחן המודל הגיע לדיוק של 0.07.

ביצענו ניסוי נוסף אבל הפעם אותו מודל מקבל קלט עם גדלים מוגדרים עם כמות קטנה של מחלקות, סך הכל 4 מחלקות:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 181, 4964, 32)     1600

 max_pooling2d (MaxPooling2  (None, 90, 2482, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 90, 2482, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 45, 1241, 64)      0
 g2D)

 conv2d_2 (Conv2D)           (None, 45, 1241, 128)     73856

 max_pooling2d_2 (MaxPoolin  (None, 22, 620, 128)      0
 g2D)

 conv2d_3 (Conv2D)           (None, 22, 620, 256)      295168

 max_pooling2d_3 (MaxPoolin  (None, 11, 310, 256)      0
 g2D)

 global_average_pooling2d (  (None, 256)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               65792

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 455940 (1.74 MB)
Trainable params: 455940 (1.74 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

גרף לא הצליחנו להגיע לתבונה מסוימת על ידי גרף הלמידה של המודל.



המודל הגיע לדיוק של 0.25 במבחן.

תובנות עד כה:
- צריך לחלק לתתי תמונות, המודל מתקשה לסווג לפי כל השורה.
- לא ידוע האם המודל אכן מספיק מורכב כדי להתמודד עם משימה כזאת.
- אמנם התמרת פורייה אכן בצע את הרצוי אך זה הופך ללא ריאלי עקב הזמן שידרש על 204 מחלקות.
- בוצע ניסוי על 204 כתובים ללא התמרת פורייה והתברר לנו שבחלק מתמונות לפי שיטת של Mohd Asif Momin לא מחלק נכון והדבר מוביל לתתי תמונות לא בעלות תוכן משמעותי. כנראה יש הבדל כלשהו אחרי 100 הכותבים הראשונים לכן לקח לנו זמן להבחין בכך.

מסקנות מהתובנות:
- עדיף לחתוך כפי שבוצע בניסוי הראשון אבל רק כהתחלה.
- שימוש בחלון שזז בגודל קבוע על מנת למנוע חפיפה בין התמונות ולפסול ריקות כפי שפסלנו שורות שלמות.
- עדיף לחתוך ולהזיז את החלון בגדלים יותר גדולים מ292 אך לא כל השורה, המודל לא מצליח ללמוד עם כל התמונה ועם חתכים קטנים לא מצליח להבין במה להתמקד. הוחלט לחתוך בגדלים של 600 פיקסלים אורך וגם זה הגודל שהחלון זז.
- הוחלט עקב אילוצי הזמן עם מודל מאומן, מודל VGG16, בהמשך הניסיונות.
- מכיוון שהמודל עובד עם תתי תמונות וצריך להכריע על התמונה גדולה אז הוחלט אחרי מבחן להכריע את הזיהוי של כותב על ידי מי שמופיע יותר פעמים על סט של חתכים עבור כותב מסוים.

ביצוע ניסוי על 100 מחלקות

עד כה הניסיונות היו על תמונות בכל סוגי העיבודים, ביצוע הנוכחי הוא על 100 תמונות הראשונות של התיקייה הראשונה.

כפי שנאמר, המודל הוא VGG16 אך הוספנו לו שכבה בת 256 וכיבינו את שאר השכבות של המודל.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 225, 600, 3)       0

 vgg16 (Functional)          (None, 7, 18, 512)        14714688

 global_average_pooling2d (  (None, 512)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               131328

 dense_1 (Dense)             (None, 100)               25700

=================================================================
Total params: 14871716 (56.73 MB)
Trainable params: 157028 (613.39 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

אמנם נראה שהמודל מקבל תמונות בצבע אך למעשה עובד עם תמונות אפורות.

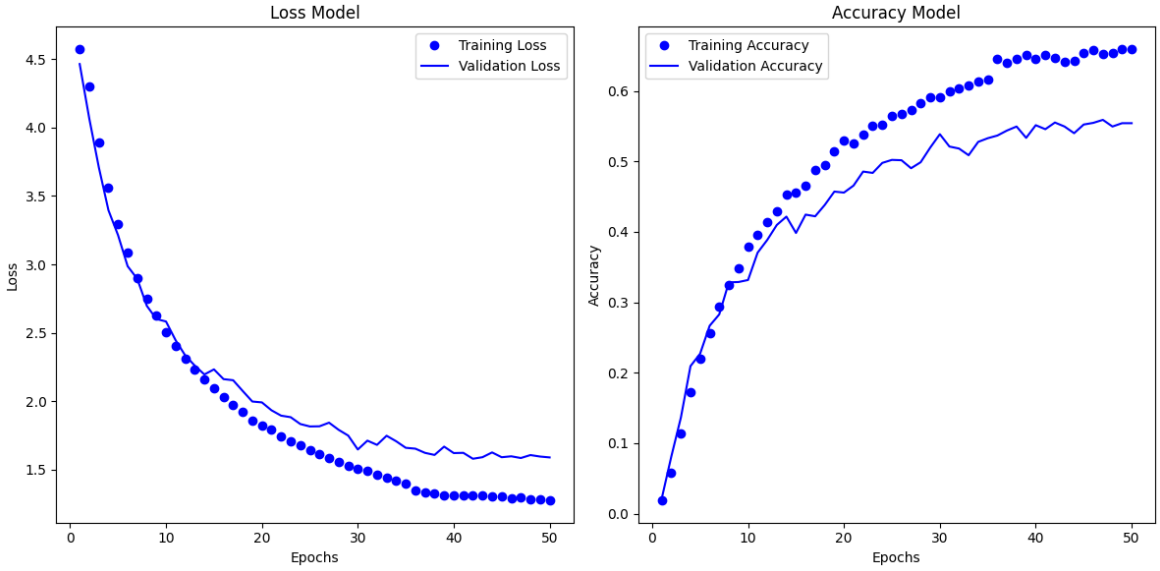
בנוסף הגדרנו:

```
opt = optimizers.Adam(learning_rate=0.0005)

model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

eraly_stop = EarlyStopping(monitor='val_accuracy', mode='max', patience=10,
verbose=1)

checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy',
mode='max', save_best_only=True, verbose=1)

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
verbose=1, min_delta=0.0001, min_lr=0.00001)
```

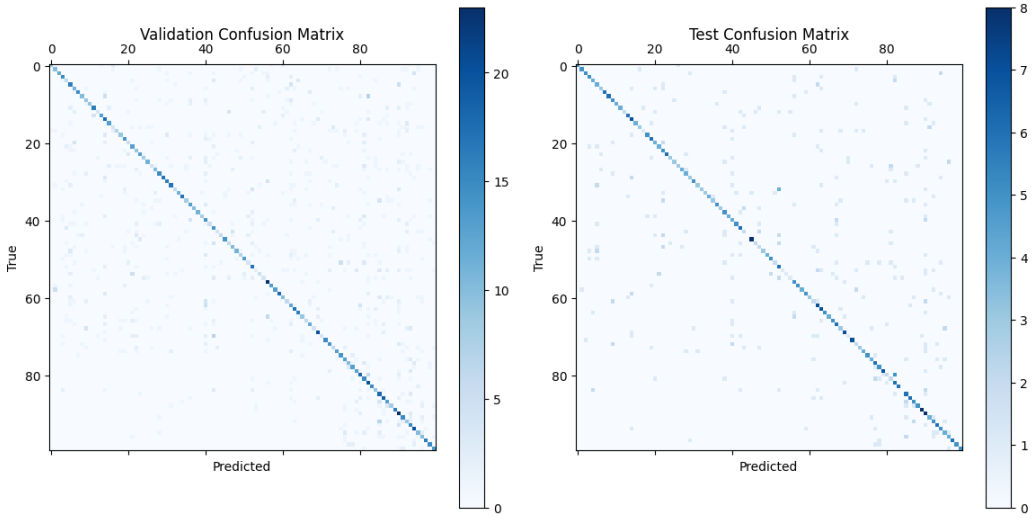
על מנת לשפר את יכולת הלמידה של המודל השתמשנו, לא רק בניסוי הזה,
בImageDataGenerator על מנת לטעון את התמונות וגם על מנת ליצור טרנפורמציות על תמונות
ובכך המודל יתאמן על מגוון צורות של תמונות.

```
train_datagen = image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    fill_mode='nearest'
    )

test_datagen = image.ImageDataGenerator(rescale=1./255)
```

מבחינת למידה של המודל קיבלנו את הגרף הבא:



מבחינת הצלחות של המודל עבור קבוצת הולידציה וקבוצת המבחן:



אמנם המודל התאמן במהלך 50 תקופות ולא נעצר באף שלב של הלמידה אז ניתן לומר שאם הגדלת התקופות יכל להגיע לדיוק יותר טוב.

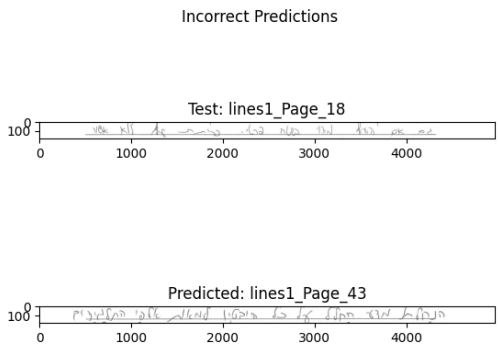
מבחינת דיוק המבחן המודל הגיע לדיוק של 0.59.

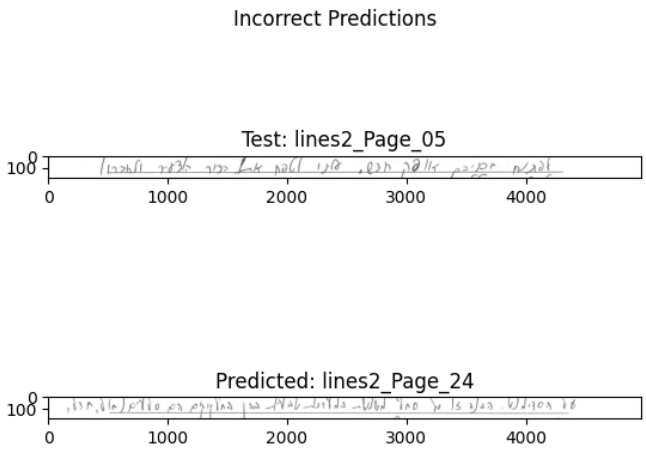
```
[77]  test_loss, test_acc = model.evaluate(test_data_gen, steps=steps, verbose=1)
                                                                                Python
...   722/722 [==============================] - 13s 18ms/step - loss: 1.5610 - accuracy: 0.5942

[78]  test_acc
                                                                                Python
...   0.5941828489303589
```

ולפי שיטת מי שמופיע יותר נבחר כי הכותב אז הגענו לדיוק של 60.92%.

```
[85]  author_to_final_preds, author_to_confidences = predict_using_majority_vote_by_author(loaded_model, test_data_gen)
                                                                                Python
...   722/722 [==============================] - 12s 17ms/step

[86]  average_confidence = sum(author_to_confidences.values()) / len(author_to_confidences)
      print(f"Average Confidence: {average_confidence * 100:.2f}%")
                                                                                Python
...   Average Confidence: 60.92%
```

מבחינת הרצה הנ"ל במודל סיווג לא נכון סך הכל 13 כותבים, נציג 2 מתוכן:





ביצוע ניסיון 203 מחלקות ושינוי חלוקה

החלטנו הפעם להסיר את הרווחים בין כל מילה ובכך להגיע לחיתוכים עם יותר תוכן.

בשלב הכנת התמונות לאימון (pre-processing) ראינו שאנחנו מייצרים מספר לא קטן של תתי-תמונות (segments) בהן חלק גדול מהשטח שלהן מכיל אזור ריק (כלומר ללא מילה או חלקי מילה). קיבלנו החלטה שהצורך ברווחים ה"מתים" בין מילים פחות רלוונטי לאימון ושעדיף להביא למודל כמה שיותר מידע משמעותי ממנו יוכל ללמוד, שכן רווחים בין מילים כנראה לא יביא ערך ללמידה של כתב היד.

תחילה, חילקנו כל שורה תת-תמונות באורך קבוע. עבור כל תת-תמונה בדקנו מה אחוז הפיקסלים הלבנים בתוכה, ומצאנו ערך threshold מולו בדקנו האם אחוז הפיקסלים הלבנים גבוה מערך זה. אם כן, לא היינו משתמשים בתת-תמונה זו.

שיטה זו לא עבדה מספיק טוב שכן איבדנו חלק מאוד גדול מהשורות.

בשלב זה ניסינו להשתמש בשיטת קונטור (contour) בה טשטשנו את המילים בשורה באמצעות dilate וכך זיהינו איפה המילים נמצאות בתוך התמונה. לפי המיקומים של המילים יצרנו תתי-תמונות במיקומים האלו וצמצמנו את הרווחים המתים בין המילים. אך כעת הבעיה הייתה שתתי-התמונות היו בגודל לא אחיד והיינו צריכים להשתמש ב padding של פיקסלים לבנים, מה שגרם לתתי-התמונות שלנו להכיל שוב, מידע לא רלוונטי.

לבסוף החלטנו לנקוט בשיטה יותר דרסטית. כל שורה מתוך התמונה המקורית עברה תהליך של pre-processing שכלל טשטוש של התמונה באמצעות gaussian blur של ספריית cv2.
באמצעות הטשטוש, זיהינו אזורים רצופים של שטח מת (בעיקר פיקסלים לבנים) ואותם חתכנו לגמרי מהשורה ובעצם דחסנו את כל הטקסט לרצף אחיד של כתב. התהליך כלל מציאת הערכים המתאימים עבור הפילטר כך שלא שמילים ואותיות לא ייחתכו מהשורה הסופית. לאחר לא מעט ניסויים הגענו לערכים הסופיים שיצרו טשטוש קל לרוחב (כדי לקבל רציפות של מילים, אבל לא יותר מדי כדי לא לגשר בין מילים יחסית קרובות) וטשטוש מסיבי לאורך כדי לזהות בצורה וודאית את האזורים בהם קיים מידע אותו אנו רוצים להשאיר.

לאחר חיתוך החלקים הלבנים מתוך התמונה, קיבלנו תמונות באורכים משתנים (כפי שקיבלנו בניסוי הקודם). כדי לפתור את הבעיה הזו, שוב השתמשנו בpadding מצד שמאל של התמונה (שכן כל הטקסט הוצמד לימין). אך כעת, בשלב בו אנו מחלקים את התמונה לתתי-תמונות, נזרוק לכל היותר את התת-תמונה האחרונה, אשר (אולי) מכילה יותר מדי פיקסלים לבנים. שאר תתי התמונות מעבר לתמונה זו יכילו אך ורק פיקסלים לבנים, ולכן גם בהם לא נשתמש.

התמונה המקורית:

הפעלת הפילטר:

זיהוי שטחים מתים:

חיתוך השטחים המתים לכדי תמונה אחת:

לטובת ניסוי הזה נלקחו תמונות מהתיקייה הרביעית.
שינוי נוסף בניסוי, גובה השורות יהיו בגובה המקסימלי מבין השורות המבחן. לכן הפעם המודל הוא:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 220, 600, 3)       0

 vgg16 (Functional)          (None, 6, 18, 512)        14714688

 global_average_pooling2d (  (None, 512)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               131328

 dense_1 (Dense)             (None, 203)               52171

=================================================================
Total params: 14898187 (56.83 MB)
Trainable params: 183499 (716.79 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

```
opt = optimizers.Adam(learning_rate=0.001)

model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

גודל ה-batch_size במהלך האימון הוא 8, תוצאות במהלך האימון:

```
ETA: 0s - loss: 4.3974 - accuracy: 0.0583 - [==============================] 1339/1339
Epoch 2: val_accuracy improved from 0.04575 to 0.07097, saving model to best_model.h5
205s 153ms/step - loss: 4.3974 - accuracy: 0.0583 - - [==============================] 1339/1339
val_loss: 4.1918 - val_accuracy: 0.0710 - lr: 0.0010
Epoch 3/203
ETA: 0s - loss: 4.0389 - accuracy: 0.0966 - [==============================] 1339/1339
Epoch 3: val_accuracy improved from 0.07097 to 0.10663, saving model to best_model.h5
205s 153ms/step - loss: 4.0389 - accuracy: 0.0966 - - [==============================] 1339/1339
val_loss: 3.9593 - val_accuracy: 0.1066 - lr: 0.0010
Epoch 4/203
```

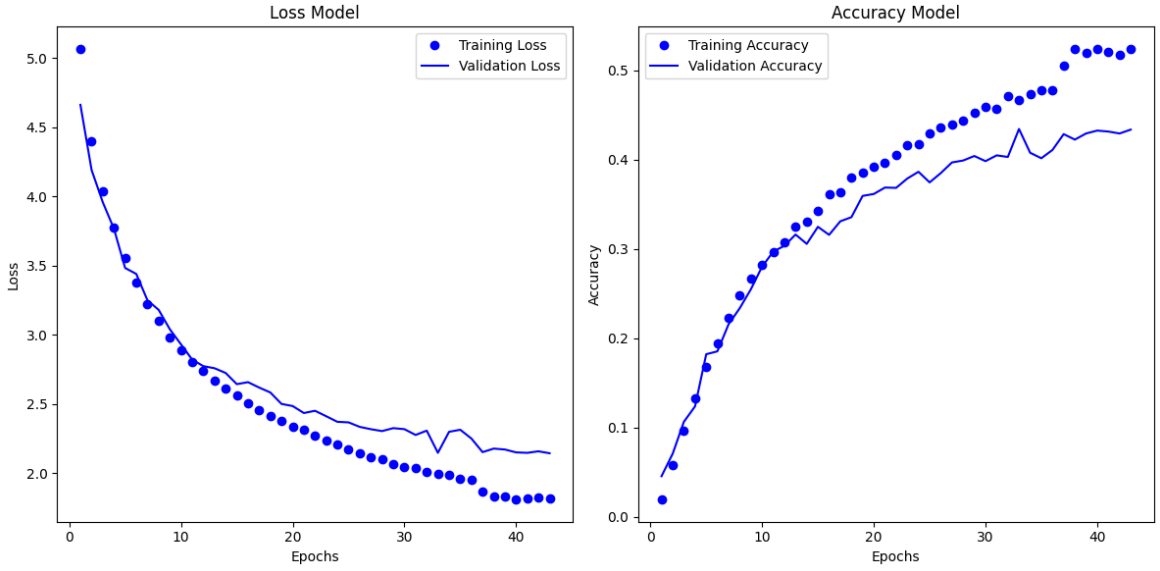
```
ETA: 0s - loss: 3.7726 - accuracy: 0.1325 - [==============================] 1339/1339
Epoch 4: val_accuracy improved from 0.10663 to 0.12392, saving model to best_model.h5
205s 153ms/step - loss: 3.7726 - accuracy: 0.1325 - - [==============================] 1339/1339
val_loss: 3.7676 - val_accuracy: 0.1239 - lr: 0.0010
Epoch 5/203
ETA: 0s - loss: 3.5545 - accuracy: 0.1680 - [==============================] 1339/1339
Epoch 5: val_accuracy improved from 0.12392 to 0.18228, saving model to best_model.h5
205s 153ms/step - loss: 3.5545 - accuracy: 0.1680 - - [==============================] 1339/1339
val_loss: 3.4841 - val_accuracy: 0.1823 - lr: 0.0010
Epoch 6/203
ETA: 0s - loss: 3.3795 - accuracy: 0.1948 - [==============================] 1339/1339
Epoch 6: val_accuracy improved from 0.18228 to 0.18552, saving model to best_model.h5
205s 153ms/step - loss: 3.3795 - accuracy: 0.1948 - - [==============================] 1339/1339
val_loss: 3.4409 - val_accuracy: 0.1855 - lr: 0.0010
Epoch 7/203
ETA: 0s - loss: 3.2239 - accuracy: 0.2226 - [==============================] 1339/1339
Epoch 7: val_accuracy improved from 0.18552 to 0.21578, saving model to best_model.h5
205s 153ms/step - loss: 3.2239 - accuracy: 0.2226 - - [==============================] 1339/1339
val_loss: 3.2504 - val_accuracy: 0.2158 - lr: 0.0010
Epoch 8/203
ETA: 0s - loss: 3.0993 - accuracy: 0.2482 - [==============================] 1339/1339
Epoch 8: val_accuracy improved from 0.21578 to 0.23379, saving model to best_model.h5
205s 153ms/step - loss: 3.0993 - accuracy: 0.2482 - - [==============================] 1339/1339
val_loss: 3.1814 - val_accuracy: 0.2338 - lr: 0.0010
Epoch 9/203
ETA: 0s - loss: 2.9786 - accuracy: 0.2667 - [==============================] 1339/1339
Epoch 9: val_accuracy improved from 0.23379 to 0.25504, saving model to best_model.h5
205s 153ms/step - loss: 2.9786 - accuracy: 0.2667 - - [==============================] 1339/1339
val_loss: 3.0414 - val_accuracy: 0.2550 - lr: 0.0010
Epoch 10/203
ETA: 0s - loss: 2.8869 - accuracy: 0.2826 - [==============================] 1339/1339
Epoch 10: val_accuracy improved from 0.25504 to 0.28026, saving model to best_model.h5
205s 153ms/step - loss: 2.8869 - accuracy: 0.2826 - - [==============================] 1339/1339
val_loss: 2.9322 - val_accuracy: 0.2803 - lr: 0.0010
Epoch 11/203
ETA: 0s - loss: 2.8036 - accuracy: 0.2968 - [==============================] 1339/1339
Epoch 11: val_accuracy improved from 0.28026 to 0.29719, saving model to best_model.h5
205s 153ms/step - loss: 2.8036 - accuracy: 0.2968 - - [==============================] 1339/1339
val_loss: 2.8221 - val_accuracy: 0.2972 - lr: 0.0010
Epoch 12/203
ETA: 0s - loss: 2.7422 - accuracy: 0.3071 - [==============================] 1339/1339
Epoch 12: val_accuracy improved from 0.29719 to 0.30367, saving model to best_model.h5
205s 153ms/step - loss: 2.7422 - accuracy: 0.3071 - - [==============================] 1339/1339
val_loss: 2.7745 - val_accuracy: 0.3037 - lr: 0.0010
Epoch 13/203
ETA: 0s - loss: 2.6685 - accuracy: 0.3251 - [==============================] 1339/1339
Epoch 13: val_accuracy improved from 0.30367 to 0.31628, saving model to best_model.h5
205s 153ms/step - loss: 2.6685 - accuracy: 0.3251 - - [==============================] 1339/1339
val_loss: 2.7594 - val_accuracy: 0.3163 - lr: 0.0010
Epoch 14/203
ETA: 0s - loss: 2.6158 - accuracy: 0.3311 - [==============================] 1339/1339
Epoch 14: val_accuracy did not improve from 0.31628
205s 153ms/step - loss: 2.6158 - accuracy: 0.3311 - - [==============================] 1339/1339
val_loss: 2.7252 - val_accuracy: 0.3058 - lr: 0.0010
Epoch 15/203
ETA: 0s - loss: 2.5628 - accuracy: 0.3426 - [==============================] 1339/1339
Epoch 15: val_accuracy improved from 0.31628 to 0.32493, saving model to best_model.h5
205s 153ms/step - loss: 2.5628 - accuracy: 0.3426 - - [==============================] 1339/1339
val_loss: 2.6442 - val_accuracy: 0.3249 - lr: 0.0010
Epoch 16/203
ETA: 0s - loss: 2.5048 - accuracy: 0.3609 - [==============================] 1339/1339
Epoch 16: val_accuracy did not improve from 0.32493
205s 153ms/step - loss: 2.5048 - accuracy: 0.3609 - - [==============================] 1339/1339
val_loss: 2.6588 - val_accuracy: 0.3159 - lr: 0.0010
Epoch 17/203
ETA: 0s - loss: 2.4587 - accuracy: 0.3636 - [==============================] 1339/1339
```

```
Epoch 17: val_accuracy improved from 0.32493 to 0.33105, saving model to best_model.h5
205s 153ms/step - loss: 2.4587 - accuracy: 0.3636 - - [==============================] 1339/1339
val_loss: 2.6197 - val_accuracy: 0.3311 - lr: 0.0010
Epoch 18/203
ETA: 0s - loss: 2.4141 - accuracy: 0.3803 - [==============================] 1339/1339
Epoch 18: val_accuracy improved from 0.33105 to 0.33573, saving model to best_model.h5
205s 153ms/step - loss: 2.4141 - accuracy: 0.3803 - - [==============================] 1339/1339
val_loss: 2.5842 - val_accuracy: 0.3357 - lr: 0.0010
Epoch 19/203
ETA: 0s - loss: 2.3784 - accuracy: 0.3855 - [==============================] 1339/1339
Epoch 19: val_accuracy improved from 0.33573 to 0.35951, saving model to best_model.h5
205s 153ms/step - loss: 2.3784 - accuracy: 0.3855 - - [==============================] 1339/1339
val_loss: 2.5014 - val_accuracy: 0.3595 - lr: 0.0010
Epoch 20/203
ETA: 0s - loss: 2.3360 - accuracy: 0.3918 - [==============================] 1339/1339
Epoch 20: val_accuracy improved from 0.35951 to 0.36167, saving model to best_model.h5
205s 153ms/step - loss: 2.3360 - accuracy: 0.3918 - - [==============================] 1339/1339
val_loss: 2.4861 - val_accuracy: 0.3617 - lr: 0.0010
Epoch 21/203
ETA: 0s - loss: 2.3160 - accuracy: 0.3964 - [==============================] 1339/1339
Epoch 21: val_accuracy improved from 0.36167 to 0.36888, saving model to best_model.h5
205s 153ms/step - loss: 2.3160 - accuracy: 0.3964 - - [==============================] 1339/1339
val_loss: 2.4356 - val_accuracy: 0.3689 - lr: 0.0010
Epoch 22/203
ETA: 0s - loss: 2.2743 - accuracy: 0.4049 - [==============================] 1339/1339
Epoch 22: val_accuracy did not improve from 0.36888
205s 153ms/step - loss: 2.2743 - accuracy: 0.4049 - - [==============================] 1339/1339
val_loss: 2.4519 - val_accuracy: 0.3685 - lr: 0.0010
Epoch 23/203
ETA: 0s - loss: 2.2389 - accuracy: 0.4161 - [==============================] 1339/1339
Epoch 23: val_accuracy improved from 0.36888 to 0.37896, saving model to best_model.h5
205s 153ms/step - loss: 2.2389 - accuracy: 0.4161 - - [==============================] 1339/1339
val_loss: 2.4125 - val_accuracy: 0.3790 - lr: 0.0010
Epoch 24/203
ETA: 0s - loss: 2.2089 - accuracy: 0.4171 - [==============================] 1339/1339
Epoch 24: val_accuracy improved from 0.37896 to 0.38653, saving model to best_model.h5
205s 153ms/step - loss: 2.2089 - accuracy: 0.4171 - - [==============================] 1339/1339
val_loss: 2.3721 - val_accuracy: 0.3865 - lr: 0.0010
Epoch 25/203
ETA: 0s - loss: 2.1763 - accuracy: 0.4296 - [==============================] 1339/1339
Epoch 25: val_accuracy did not improve from 0.38653
205s 153ms/step - loss: 2.1763 - accuracy: 0.4296 - - [==============================] 1339/1339
val_loss: 2.3681 - val_accuracy: 0.3746 - lr: 0.0010
Epoch 26/203
ETA: 0s - loss: 2.1427 - accuracy: 0.4356 - [==============================] 1339/1339
Epoch 26: val_accuracy did not improve from 0.38653
205s 153ms/step - loss: 2.1427 - accuracy: 0.4356 - - [==============================] 1339/1339
val_loss: 2.3356 - val_accuracy: 0.3851 - lr: 0.0010
Epoch 27/203
ETA: 0s - loss: 2.1175 - accuracy: 0.4399 - [==============================] 1339/1339
Epoch 27: val_accuracy improved from 0.38653 to 0.39697, saving model to best_model.h5
205s 153ms/step - loss: 2.1175 - accuracy: 0.4399 - - [==============================] 1339/1339
val_loss: 2.3188 - val_accuracy: 0.3970 - lr: 0.0010
Epoch 28/203
ETA: 0s - loss: 2.1015 - accuracy: 0.4433 - [==============================] 1339/1339
Epoch 28: val_accuracy improved from 0.39697 to 0.39914, saving model to best_model.h5
205s 153ms/step - loss: 2.1015 - accuracy: 0.4433 - - [==============================] 1339/1339
val_loss: 2.3051 - val_accuracy: 0.3991 - lr: 0.0010
Epoch 29/203
ETA: 0s - loss: 2.0689 - accuracy: 0.4525 - [==============================] 1339/1339
Epoch 29: val_accuracy improved from 0.39914 to 0.40418, saving model to best_model.h5
207s 154ms/step - loss: 2.0689 - accuracy: 0.4525 - - [==============================] 1339/1339
val_loss: 2.3263 - val_accuracy: 0.4042 - lr: 0.0010
Epoch 30/203
ETA: 0s - loss: 2.0455 - accuracy: 0.4588 - [==============================] 1339/1339
Epoch 30: val_accuracy did not improve from 0.40418
```

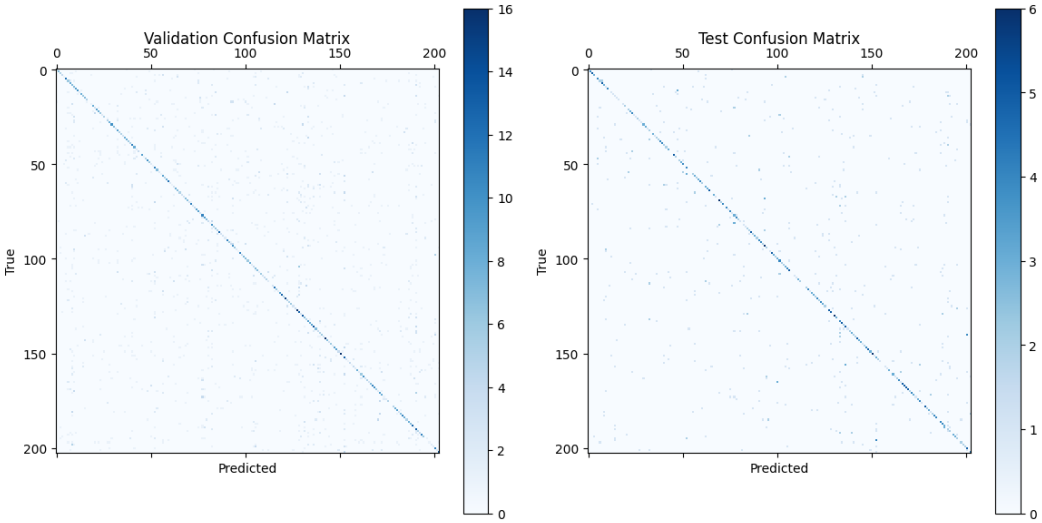
```
205s 153ms/step - loss: 2.0455 - accuracy: 0.4588 - - [==============================] 1339/1339
val_loss: 2.3191 - val_accuracy: 0.3984 - lr: 0.0010
Epoch 31/203
ETA: 0s - loss: 2.0368 - accuracy: 0.4575 - [==============================] 1339/1339
Epoch 31: val_accuracy improved from 0.40418 to 0.40490, saving model to best_model.h5
205s 153ms/step - loss: 2.0368 - accuracy: 0.4575 - - [==============================] 1339/1339
val_loss: 2.2772 - val_accuracy: 0.4049 - lr: 0.0010
Epoch 32/203
ETA: 0s - loss: 2.0076 - accuracy: 0.4710 - [==============================] 1339/1339
Epoch 32: val_accuracy did not improve from 0.40490
205s 153ms/step - loss: 2.0076 - accuracy: 0.4710 - - [==============================] 1339/1339
val_loss: 2.3083 - val_accuracy: 0.4031 - lr: 0.0010
Epoch 33/203
ETA: 0s - loss: 1.9980 - accuracy: 0.4668 - [==============================] 1339/1339
Epoch 33: val_accuracy improved from 0.40490 to 0.43444, saving model to best_model.h5
205s 153ms/step - loss: 1.9980 - accuracy: 0.4668 - - [==============================] 1339/1339
val_loss: 2.1481 - val_accuracy: 0.4344 - lr: 0.0010
Epoch 34/203
ETA: 0s - loss: 1.9858 - accuracy: 0.4734 - [==============================] 1339/1339
Epoch 34: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.9858 - accuracy: 0.4734 - - [==============================] 1339/1339
val_loss: 2.3000 - val_accuracy: 0.4078 - lr: 0.0010
Epoch 35/203
ETA: 0s - loss: 1.9609 - accuracy: 0.4778 - [==============================] 1339/1339
Epoch 35: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.9609 - accuracy: 0.4778 - - [==============================] 1339/1339
val_loss: 2.3148 - val_accuracy: 0.4017 - lr: 0.0010
Epoch 36/203
ETA: 0s - loss: 1.9501 - accuracy: 0.4774 - [==============================] 1339/1339
Epoch 36: val_accuracy did not improve from 0.43444

.Epoch 36: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026
205s 153ms/step - loss: 1.9501 - accuracy: 0.4774 - - [==============================] 1339/1339
val_loss: 2.2513 - val_accuracy: 0.4110 - lr: 0.0010
Epoch 37/203
ETA: 0s - loss: 1.8646 - accuracy: 0.5054 - [==============================] 1339/1339
Epoch 37: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8646 - accuracy: 0.5054 - - [==============================] 1339/1339
val_loss: 2.1530 - val_accuracy: 0.4287 - lr: 2.0000e-04
Epoch 38/203
ETA: 0s - loss: 1.8300 - accuracy: 0.5237 - [==============================] 1339/1339
Epoch 38: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8300 - accuracy: 0.5237 - - [==============================] 1339/1339
val_loss: 2.1787 - val_accuracy: 0.4226 - lr: 2.0000e-04
Epoch 39/203
ETA: 0s - loss: 1.8318 - accuracy: 0.5200 - [==============================] 1339/1339
Epoch 39: val_accuracy did not improve from 0.43444

.Epoch 39: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05
205s 153ms/step - loss: 1.8318 - accuracy: 0.5200 - - [==============================] 1339/1339
val_loss: 2.1719 - val_accuracy: 0.4294 - lr: 2.0000e-04
Epoch 40/203
ETA: 0s - loss: 1.8127 - accuracy: 0.5236 - [==============================] 1339/1339
Epoch 40: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8127 - accuracy: 0.5236 - - [==============================] 1339/1339
val_loss: 2.1512 - val_accuracy: 0.4326 - lr: 4.0000e-05
Epoch 41/203
ETA: 0s - loss: 1.8162 - accuracy: 0.5211 - [==============================] 1339/1339
Epoch 41: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8162 - accuracy: 0.5211 - - [==============================] 1339/1339
val_loss: 2.1479 - val_accuracy: 0.4316 - lr: 4.0000e-05
Epoch 42/203
ETA: 0s - loss: 1.8266 - accuracy: 0.5177 - [==============================] 1339/1339
Epoch 42: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8266 - accuracy: 0.5177 - - [==============================] 1339/1339
val_loss: 2.1590 - val_accuracy: 0.4294 - lr: 4.0000e-05
```

```
Epoch 43/203
ETA: 0s - loss: 1.8189 - accuracy: 0.5235 - [==============================] 1339/1339
Epoch 43: val_accuracy did not improve from 0.43444
205s 153ms/step - loss: 1.8189 - accuracy: 0.5235 - - [==============================] 1339/1339
val_loss: 2.1450 - val_accuracy: 0.4337 - lr: 4.0000e-05
Epoch 43: early stopping
```

גרף האימון:



גרף ה-Confusion Matrix עבור קבוצת המבחן וולידציה:



תוצאת המודל על קבוצת המבחן הוא 49.376%, אך לפי שיטת מי שמופיע יותר נבחר הגענו לדיוק של 56.96% דיוק.

Use Majority vote for final predictions

```python
author_to_final_preds, author_to_confidences = predict_using_majority_vote_by_author(loaded_model, test_data_gen)
```

[86] Python

```
121/121 [==============================] - 15s 121ms/step
```

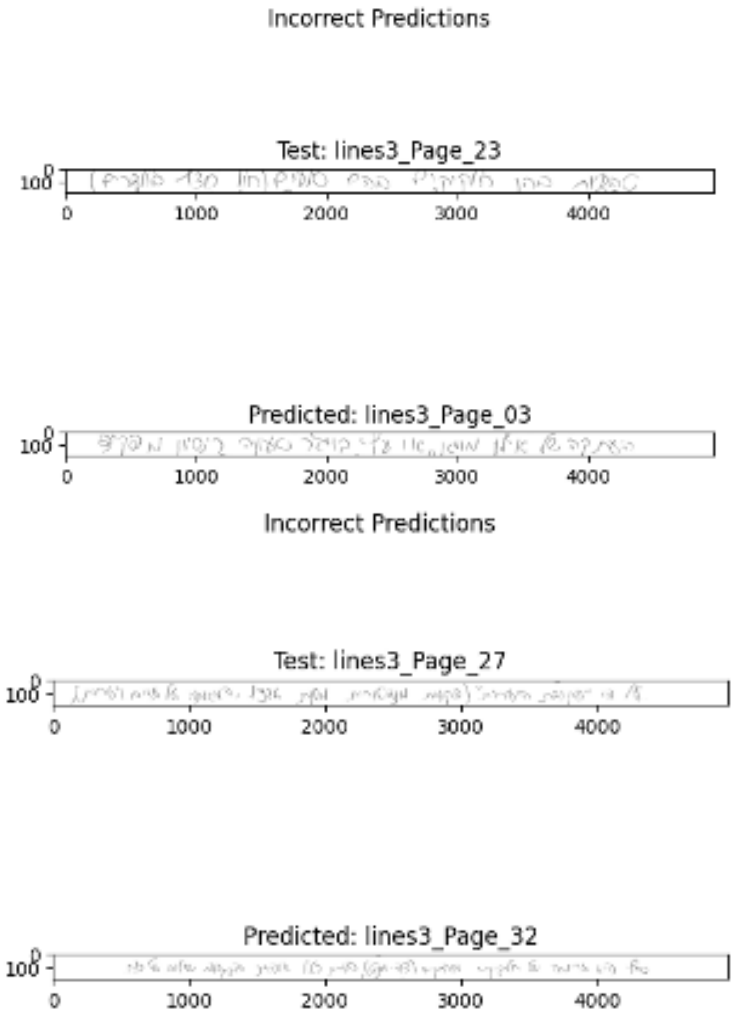
Calculate average confidence

```python
average_confidence = sum(author_to_confidences.values()) / len(author_to_confidences)
print(f"Average Confidence: {average_confidence * 100:.2f}%")
```

[87] Python

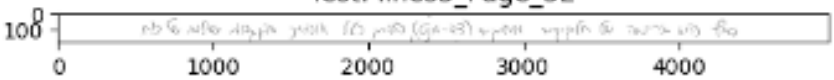
```
Average Confidence: 56.96%
```

בניסוי הנ״ל מתוך 203 כותבים המודל סיווג נכון 139 כותבים וסיווג לא נכון 64 כותבים. להלן 10 תוצאות שגויות:

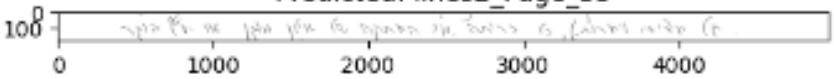


Incorrect Predictions

Test: lines3_Page_32

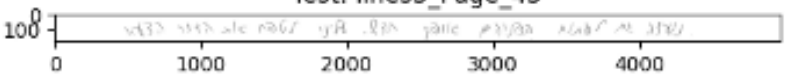


Predicted: lines2_Page_35

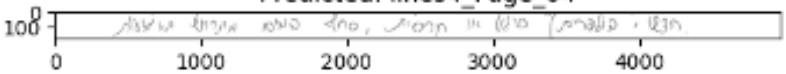


Incorrect Predictions

Test: lines3_Page_45

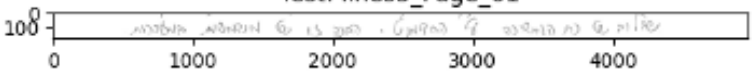


Predicted: lines4_Page_04

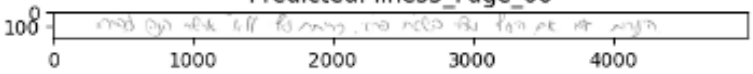


Incorrect Predictions

Test: lines3_Page_61

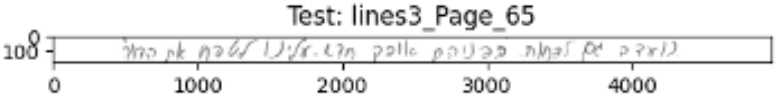


Predicted: lines3_Page_06

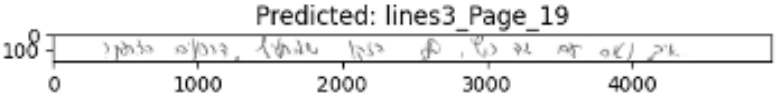


Incorrect Predictions

Test: lines3_Page_65

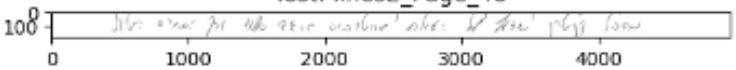


Predicted: lines3_Page_19

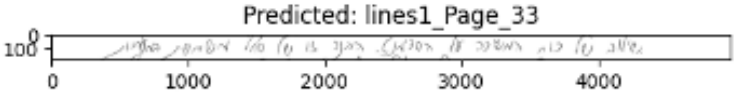


Incorrect Predictions

Test: lines2_Page_48

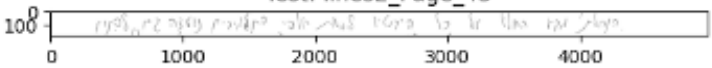


Predicted: lines1_Page_33

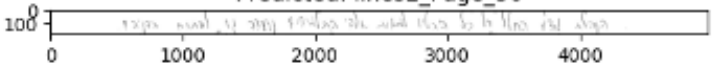


Incorrect Predictions

Test: lines2_Page_43

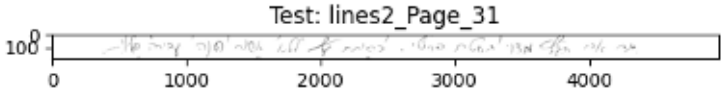


Predicted: lines2_Page_36

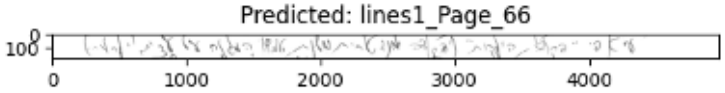


Incorrect Predictions

Test: lines2_Page_31

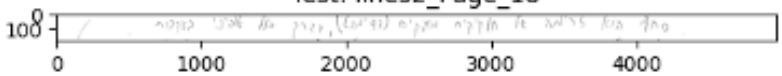


Predicted: lines1_Page_66

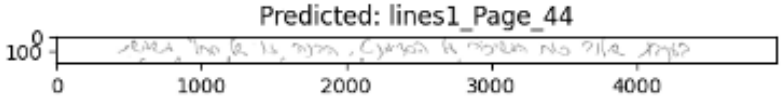


Incorrect Predictions

Test: lines2_Page_18



Predicted: lines1_Page_44



סיכום ומסקנות:

אמנם לא הגענו לדיוק מספק מבחינתנו אכן הצלחנו להגיע לנקודה שבו אם היה לנו יותר זמן אולי היינו ממשיכים לנסות עם עוד כמה שינוים, בעיקר עם הפרמטרים ב-ImageDataGenerator ואולי היינו מגיעים לדיוק יותר מספק.

ביצענו מספר ניסיונות כיצד להגיע למידע בעל משמעות, ביצענו ניסיונות שלא נכנסנו במסמך הזה כי לא היו בעל משמעות עם מודלים מאומנים אחרים כגון AlexNet ו-ResNet אך לא הגיעו לתוצאה אפילו קרובה למחשבה לבצע שיפורים.

פיתוח מערכת מסוג זה, למרות האתגר, הוא דורש המון ניסיונות וכל ניסוי שאולי עובד על קבוצה קטנה לא בהכרח יעבוד על קבוצה גדולה.

הפיתוח בוצע ב-Google Colab על ידי שימוש מנוי פרו כדי לקבל יכולות עיבוד חזקים כגון זיכרון ראם של 51 GB ו-GPU T4. לכן אימון על 203 כותבים לקח כ3 שעות וחצי.

בסופו של דבר, אנו עומדים מאחורי התוצאה ומאחורי הקוד ולדרך שבה בחרנו לנתח ולפתח את המידע. להגיע לתוצאה הנ״ל לא מספק אבל הדרך שעברנו אכן מספק.

הצעות להמשך המחקר

בפרויקט מסדר גודל הזה קודם כל ממליצים לעבוד עם קבוצות גדולות יותר על מנת להריץ יותר ניסיונות במקביל ולקבל יותר אינדיקציות כיצד לשפר ולהבין יותר חולשות.

למרות ששימוש במודל כמו VGG16, שהוא קצת מיושן, שווה לחקור על מודלים יותר חדישים, אפילו הבנו שקיים מודל שיודע להתמודד עם קלטים בגדלים שונים.

בסופו של דבר, אנו בטוחים שאם מחליטים להמשיך מהנקודה שבה הגענו ניתן לשפר פלאים רק אם יודעים מה צריך לשפר.

מקורות

1. ספריית TensorFlow
2. Deep Learning with Python, Francois Chollet, Second edition.
3. Line Text Segmentation
4. Removing lines from reeled paper with Fourier transform.
5. Chat-GPT 4

נספח הרצת הקוד

ראשית צריך להגדיר את מיקום התמונות ומיקום קבצי mat., במידה ומריצים דרך גוגל קולב אז למרות שהתיקיות נמצאות יחד עם הקובץ ipynb הדרך גישה עליהם היא:
./drive/MyDrive/DL_PROJECT

אצלנו מוגדר בשם data_path.

בכתובת הנ״ל הגדרנו תיקיות בשפה משותפת:

```
author_images_dir = f'{data_path}/author_images' # where the images are kept inside the Google Drive project folder
author_mat_dir = f'{data_path}/author_mat' # where the .mat files are kept inside the Google Drive project folder
```

בנוסף הגדרנו את התיקיות הבאות:

```
orig_images = "images" # storing original images
authors_dir = 'authors' # storing the author's .pkl files
base_dir = 'dataset_directory' # storing the data sets (after pre-processing)
train_dir = os.path.join(base_dir, 'train') # train data set directory
val_dir = os.path.join(base_dir, 'validation') # validation data set directory
test_dir = os.path.join(base_dir, 'test') # test data set directory
```

השימוש בקבצי pkl. היה עקב גדלים של האובייקטים, כלל שהגדלנו את הכמות הכותבים הגענו למצב שמיגיעים לאימון כמעט ללא זיכרון ראם, למרות שניינו מנוי פרו.

על מנת להריץ על כל כמות קטנה מ-407 כותבים:

```
# select what percentage from the 407 authors we want to use
fraction_of_authors = 0.5
selected_authors = author_names[:int(len(author_names) * fraction_of_authors)]
```