Font Recognition

מגיש - עמית אהרוני

בפרויקט זה אני אחקור את הנתונים של SynthText Dataset שיוצרו בהשראת המאמר Synthetic Data for Text Localisation in Natural Images. במקור מאגר הנתונים מכיל מעל 800 אלף תמונות עם מעל 8 מיליון מילים שהוכנסו באופן סינטטי לתמונה. המילים מופיעות בצבעים שונים, גדלים שונים, זוויות שונות ופונטים שונים.
כמה תמונות לדוגמה:



מטרת הפרויקט:
בהינתן קובץ SynthText.h5 שמכיל

- 980 תמונות עם מילים בפונטים שונים שהוכנסו אליה באופן סינטטי. בסך הכל יש לנו 980 תמונות שמכילות 7432 מילים ו- 30520 תווים.
- קואורדינטות מסגרת (Bounding Boxes) של כל תו (ציר x וגם ציר y)
- קואורדינטות מסגרת (Bounding Boxes) של כל מילה (ציר x וגם ציר y)
- הטקסט שמופיע בתמונה במסגרת
- הפונט של כל תו

מטרתנו היא לסווג מהו הפונט של כל תו בתמונה.
נציג דוגמאות של ה-Bounding Boxes:

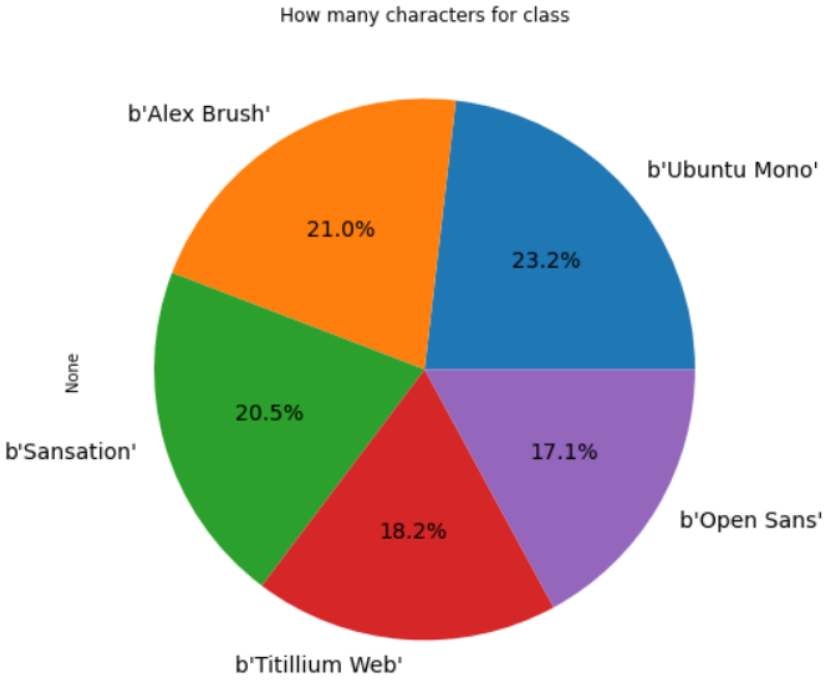


במאגר הנתונים שלנו יש חמישה סוגים שונים של פונטים:

את חלק מהתווים מאוד קשה לראות ככה שאפשר להניח שלאלגוריתם למידה יהיה מאוד קשה ללמוד אותם. התווים בצבעים שונים, בגדלים שונים, בפונטים שונים ובזוויות שונות. נצטרך שהמודל שלנו יהיה רובסטי ככל שאפשר.

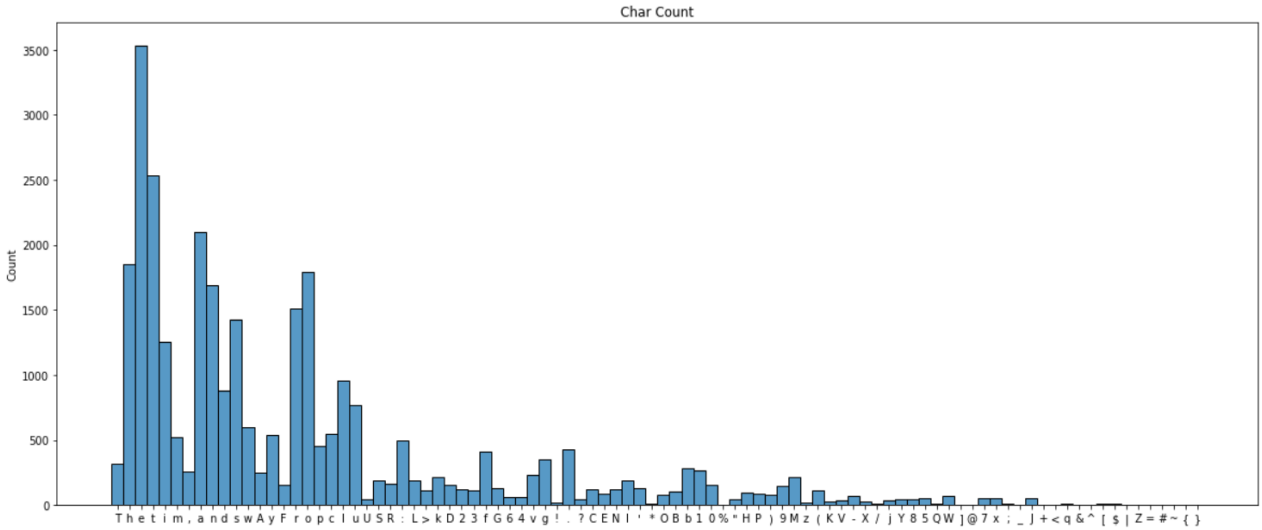
<u>חקירה של הנתונים (Exploratory data analysis):</u>

תחילה נסתכל התפלגות הפונטים לכל תו במאגר הנתונים שלנו:



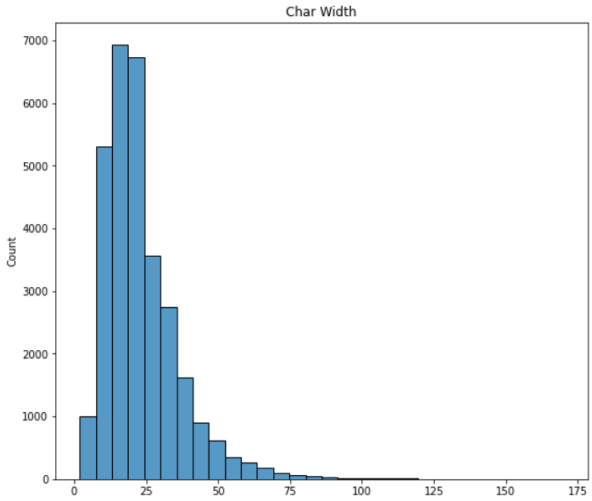
נשים לב שיש קצת חוסר איזון בין הפונטים השונים אך אפשר להניח שחוסר האיזון זניח כי לכל הפונטים יש סביב ה-20% דוגמאות (חלוקה שווה).

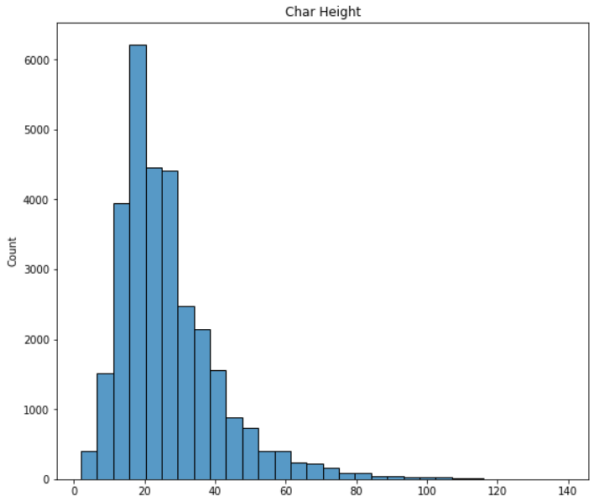
נסתכל על התפלגות התווים שלנו:



התפלגות התווים שלנו לא מאוזנת כלל. יש תווים שמופיעים מספר בודד של פעמים ויש תווים שמופיעים הרבה (למשל e מופיע הכי הרבה במאגר הנתונים ו-Z הופיע פעמים בודדות). דבר זה יכול להקשות מאוד על המודל כי יש תווים שהוא ראה פעמים בודדות ולכן כשנריץ על ה-test set יתכן שנקבל תוצאות לא טובות.

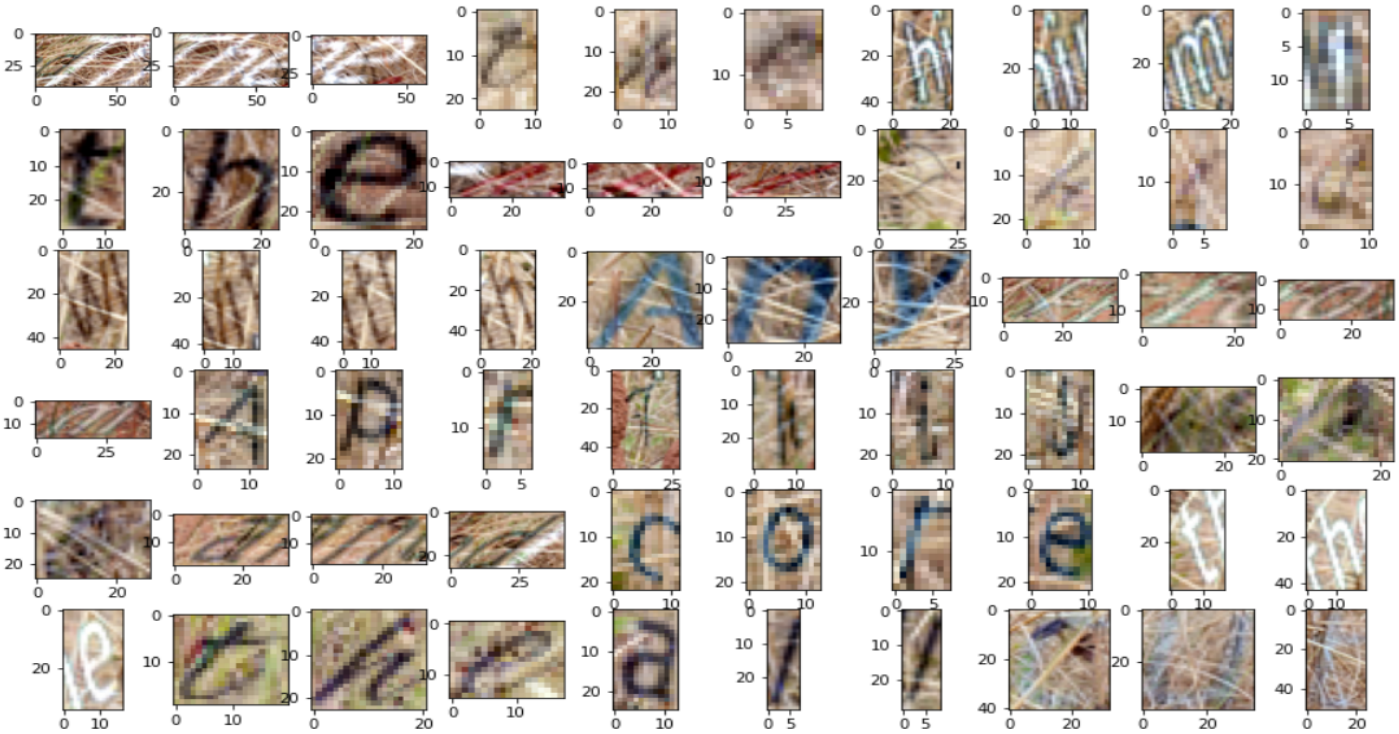
נסתכל על התפלגות גדלי התווים:





ניראה שהפיק הוא סביב גודל (20,25) לתו. מידע זה יכול להיות שימושי כשנרצה לבנות מודל.

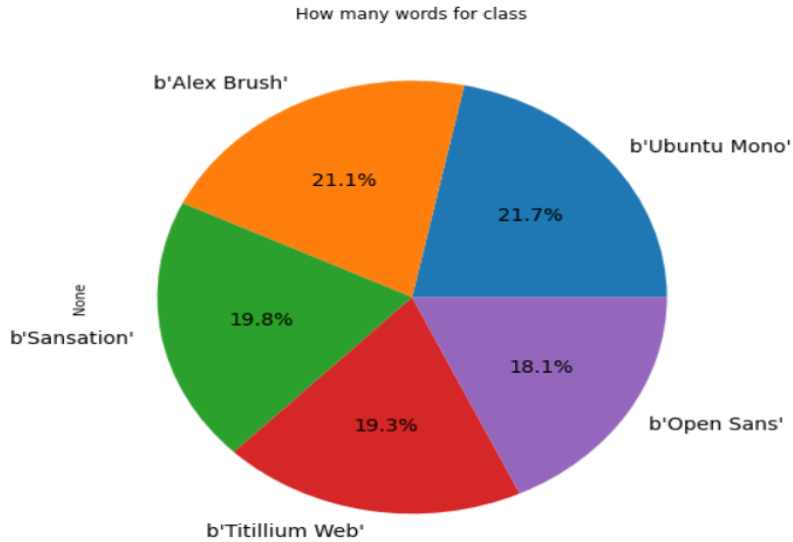
לבסוף נסתכל על התווים החתוכים שלנו:



התווים בזוויות שונות אחת מהשנייה, דבר זה מאוד מקשה על הלמידה. חלקם ממש מטושטשים ובקושי אפשר לראות את התו עצמו. בחלק מהתמונות החתוכות יש יותר מתו אחד בגלל הזווית שהתו נמצא בו.

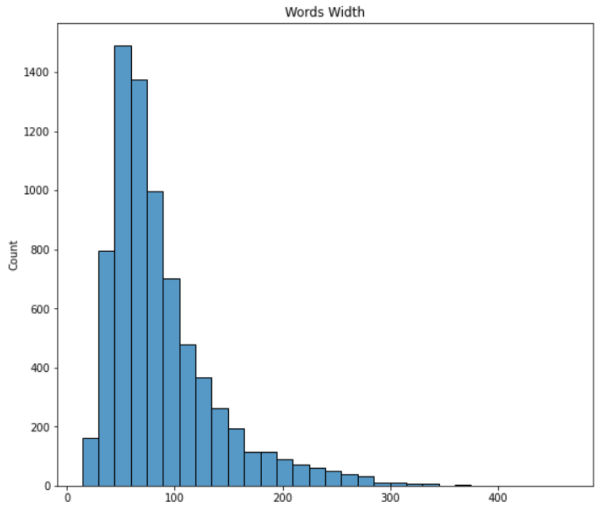
מכיוון שאפשר להניח שכל המילים הן באותו הפונט אז שווה גם לחקור את המילים ולא רק את התווים.

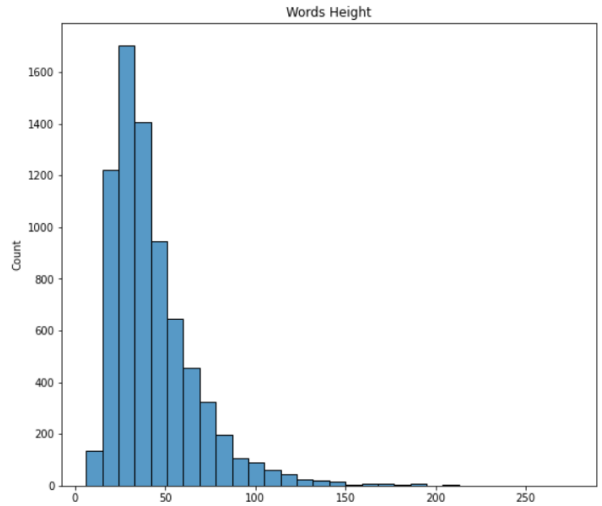
נסתכל על התפלגות הפונטים לכל מילה:



התוצאות דומות מאוד לתווים. גם פה נוכל להניח שחוסר האיזון הוא זניח.

נסתכל על גדלי תמונות המילים החתוכות.





לבסוף נסתכל על תמונות המילים החתוכות:

המודל הסופי:

השתמשתי במודל מאומן שנקרא EfficientNet B0. מודל זה התאמן על מאגר הנתונים ImageNet. בחרתי במודל זה מכיוון שיש לו תוצאות מאוד גבוהות על מאגר הנתונים ImageNet (כ- 77.69%) ביחס לגודל שלו (רק כ-5.3 מיליון פרמטרים!). אם ניקח לדוגמה את המודל ResNet50 (מודל פורץ דרך בשנת 2015) נשים לב שיש לו פי 5 יותר פרמטרים מהמודל שלנו ובנוסף לזה הוא מגיע לאחוזי ניבוי נמוכים יותר על ImageNet (כ- 76.13%). אם הייתה לי גישה לכוח חישוב חזק יותר הייתי מנסה להשתמש במודלים State Of The Art של EfficientNetV2.

דבר נוסף שמאוד עזר לביצועים של המודל זה שימוש בפונקציית המחיר CircleLoss. פונקציה זו יוצרת feature space חדש שבו היא שואפת למקסם את הדמיון בין דוגמאות מאותה מחלקה s_p ולצמצם את הדמיון בין דוגמאות ממחלקות שונות s_n. היא עושה זאת על ידי הבאה למינימום של הביטוי $(\alpha_n s_n - \alpha_p s_p)$ ככה שכל similarity score לומד בקצב שלו (בשונה מפונקציות מחיר שונות כמו Triplet Loss ו-Cross Entropy Loss).
המאמר מראה שפונקציה זו שימושית בעיקר למשימות של Face Recognition אך מסתבר שגם במקרה שלנו פונקציית המחיר משפרת את הביצועים.

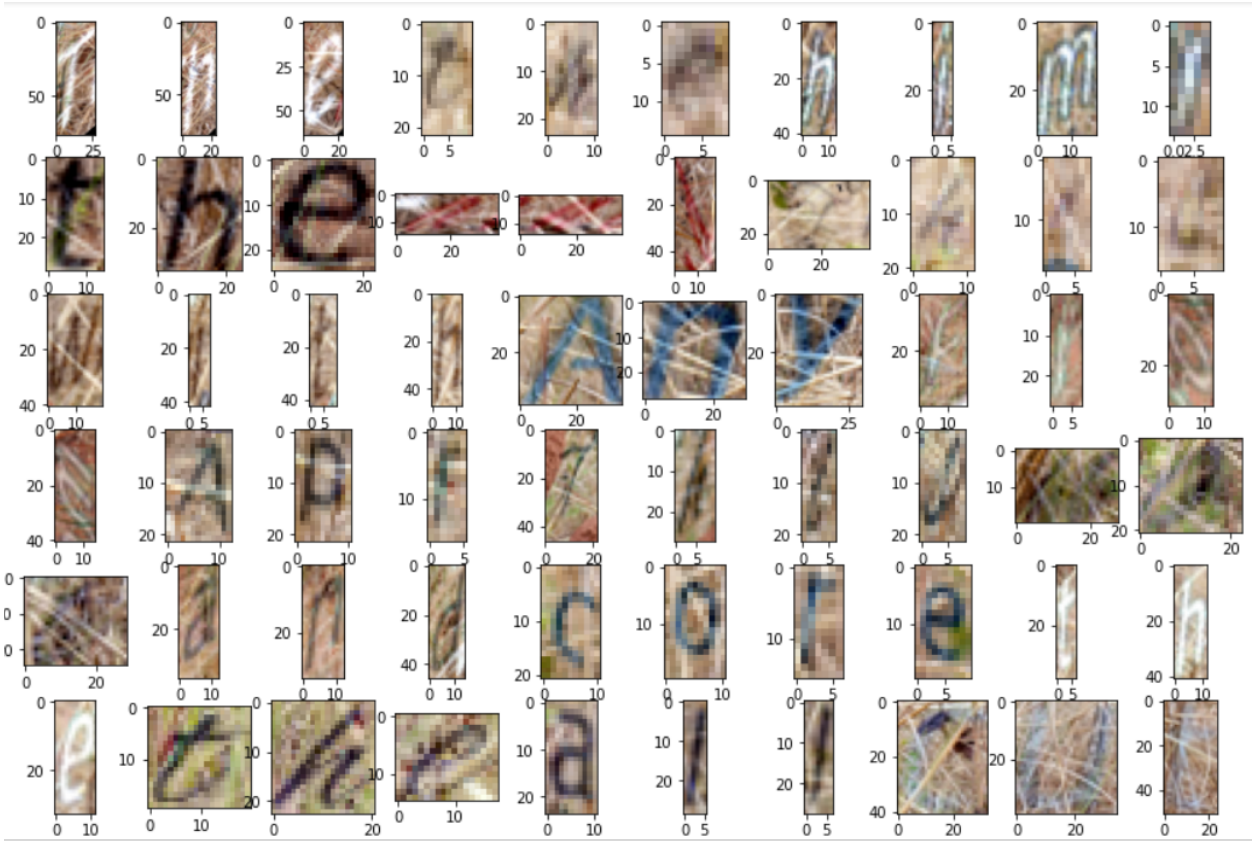
במודל הסופי בחרתי לאמן את המודל שלי על תווים (בהמשך אסביר על מודל שהתאמן על מילים). אחד הדברים שהעלו לי משמעותית את הביצועים היה להשתמש בהנחה שכל התווים במילה הם מאותו הפונט אז אפשר לחזות את הפונט בעזרת הצבעה (שיפור של 8%~):

1. נשמור את ערכי הניבויים והקטגוריה עבור כל אחת מהדוגמאות. עבור דוגמאות שיש יותר מניבוי אחד לקטגוריה מסויימת אז נחבר בין ערכי הניבויים.
2. ניקח את הפונט בעל סכום ערכי הניבויים הגבוה ביותר ונחזה שכל המילה שייכת למחלקה הזו (שכל המילה שייכת לפונט בעל סכום ערכי הניבויים הגבוה ביותר)

האינטואיציה הראשונית הייתה לקחת את הפונט בעל המספר המקסימלי של ההצבעות (במקרה של תיקו לקחת את הקטגוריה עם ערכי הניבוי הגבוהים ביותר). אך מכיוון שחלק מהתווים מאוד מטושטשים קיים מקרה קצה שהמודל מנחש את רוב התווים. נניח שקיימת מילה שתו אחד ממנה ברור והשאר מטושטשים. לכן השיטה שהצגתי דואגת למקרה הקצה מכיוון שהעוצמה של הניחושים תהיה נמוכה (cosine similarity נמוך) והעוצמה של התו הברור תהיה גבוהה וככה נצביע שכל המילה שייכת למחלקה שהתו הברור הצביע.

<u>עיבוד נתונים מקדים (Preprocessing):</u>

כחלק מתהליך ה-preprocessing החלטתי לישר את התמונות ולחתוך אותן כך שתישאר כמות מינימלית של רקע (הרקע מפריע למודל ללמוד ולהתרכז במה שחשוב).



דבר זה נעשה בעזרת Warping שלמדנו בהרצאות (ספריית OpenCV)

בנוסף להומוגרפיה הוספתי גם אוגמנטציות לתמונות. אוגמנטציות עוזרות למודל להיות רובסטי לדוגמאות חדשות על ידי הוספת רעש, בהירות, סיבוב של דוגמאות האימון. אחרי הרבה ניסויים בחרתי להשתמש באוגמנטציות הבאות:

- Resize to (224,224)
- Affine(shear=[-10,10], rotate=[-10,10])
- Random Brightness Contrast
- Gaussian Blur
- Motion Blur
- Normalize

גודל התמונה נבחר מכיוון שהמודל שלנו (EfficientNetB0) אומן על תמונות בגודל (224,224). בחירת גודל זה משפרת **מאוד** את הביצועים שלנו.

שימוש באוגמנטציות עזר מאוד לאימון המודל שלי אך גם היו גם הרבה אוגמנטציות שלא עזרו למודל ללמוד (Horizontal flip, Vertical flip וכו').

אימון:

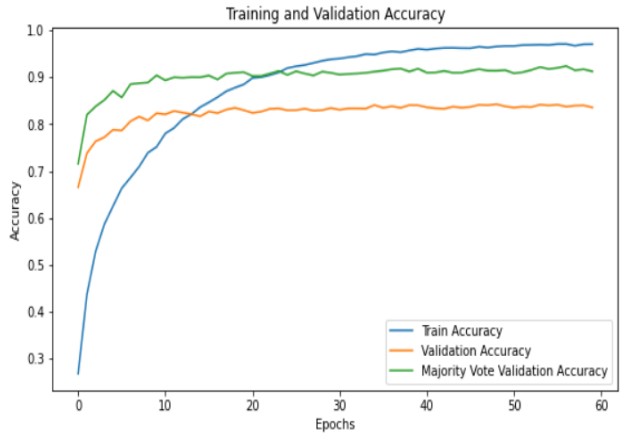
פיצלתי את הנתונים ל-80% אימונים ו-20% ולידציה.

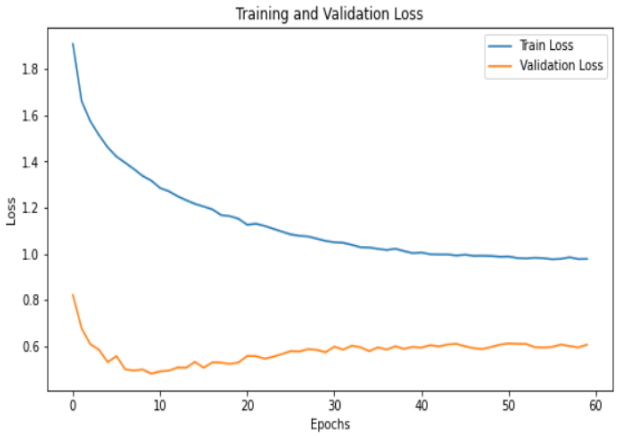
ההייפרפרמטרים שהשתמשתי:

- Adam Optimizer
- Epochs = 60
- Learning Rate = 3e-5
- Weight Decay = 2e-5
- Batch Size = 10 (השאיפה שלי הייתה להריץ את ה-Batch הגדול ביותר בהשראת [Googles Tuning Playbook])
- הפרמטרים של Circle Loss (יש לציין שבמאמר הערכים הדיפולטיביים הם s=256, m=0.25 אך במאמר אימנו רשתות על דאטהסטים עם המון קטגוריות לכן ערכים אלה לא מתאימים למקרה שלנו):
  - s = 4
  - m = 0.4
  - Embedding Size = 512

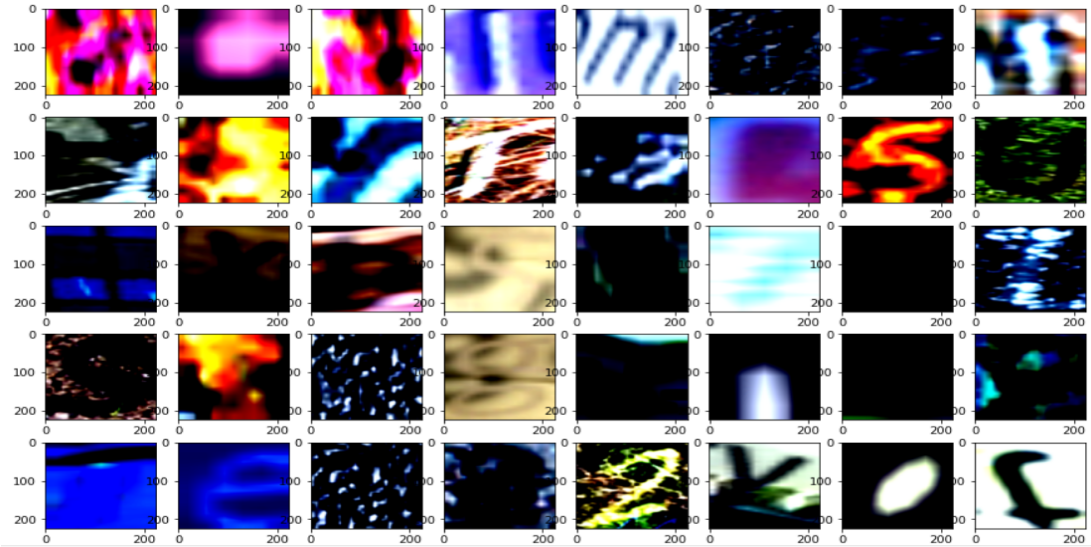
תוצאות:

לאחר 60 אפוקים הגעתי ל-91.4% ניבוי על הולידציה בעזרת הצבעה ו-84.2% ללא הצבעה





נסתכל על התמונות בעלות ה-loss הגבוה ביותר:



ונשים לב שהפונטים Titillium Web, Open Sans, Sansation מופיעים הכי הרבה פעמים.

```
{"b'Sansation'": 61,
 "b'Alex Brush'": 19,
 "b'Open Sans'": 54,
 "b'Titillium Web'": 52,
 "b'Ubuntu Mono'": 14}
```

<u>דברים שניסיתי ולא צלחו:</u>

- מודל על המילים במקום על התווים. לאחר שרוב הפרויקט עבדתי על המילים הבנתי לבסוף שאחוזי הדיוק של התווים גבוהים יותר. המודל של המילים הרבה יותר מהיר וקל לאימון כי יש **משמעותית** פחות דוגמאות (גם עבור תווים וגם עבור מילים אנחנו מגדילים את התמונה לגודל של (224,224) כי זה הגודל שהמודל אומן עליו. לכן יש פי 4.1~ דוגמאות וכתוצאה מכך זמן האימון של המודל שאומן על התווים לוקח משמעותית יותר זמן וככה גם זמן הניבוי). אך מפאת ההבדל המשמעותי בתוצאות על ה-Validation set נאלצתי לעבור למודל שמתאמן על תווים.

- מודל שמכיל בתוכו שני מודלים. המודל הראשון מתאמן על תווים והמודל השני מתאמן על מילים. כל מודל מוציא feature vector מנורמל באורך 512 ולאחר החיבור של הוקטורים האלה נקבל וקטור באורך 1024. וקטור זה יעבור בשכבות Batch Normalization, Dropout, Fully Connected, Batch Normalization, (בהשראת המאמר של ArcFace) ולאחר מכן לפונקצית המחיר ArcFace.
בעיה שקיימת היא שאורך המילה אינו קבוע (אורך מילה נע בין 3 ל-19 תווים) וזה אומר שהמודל של התווים יקבל מספר שונה של דוגמאות וככה מספר ה-feature vector ישתנה בין מילה למילה. הפתרון שלי לבעיה היה לעשות ממוצע (ניסיתי גם לקחת את המקסימום בדומה למאמר PointNet) לכל ה-feature vectors של התווים ולבסוף לקבל מכל מודל feature vector באורך 512.

המודל הצליח ללמוד והגיע לאחוזי ניבוי טובים (באיזור ה83% עם הצבעה על ה-Validation Set עם המודל EfficientNet_B0) אך פחות טובים מהמצופה ולא טובים כמו המודל הכי טוב שלי. בנוסף לזאת הלמידה של המודל לקחה המון זמן ומכיוון שאני מוגבל בכוח חישוב נאלצתי לוותר על הרעיון.

- פונקציות מחיר שונות - ניסיתי להשתמש ב- PolyLoss, Cross Entropy Loss, ArcFace, CircleLoss. הטובים ביותר היו ArcFace ו-CircleLoss אך לפונקציית המחיר CircleLoss היו ביצועים יותר טובים.

- מודל קטן שבניתי ואימנתי (ללא שימוש במודל מאומן). בתחילת הפרויקט האינטואיציה שלי הייתה שאנחנו מתאמנים על תמונות קטנות לכן אין סיבה לאמן רשת ענקית ועדיף יהיה לאמן רשת קטנה. אך לאחר שנכנסתי לאתר Kaggle והסתכלתי על מחברות של MNIST ו-CIFAR (בעיה דומה לשלנו מבחינת גודל תמונה) הבנתי שפתרון לבעיה הוא להגדיל את גודל התמונה לגודל שהמודל אומן עליו ב-ImageNet (כתוב מהו הגודל בדוקומנטציה של Pytorch). כשהכנסתי לרשת המאומנת הבנתי שמודל קטן ולא מאומן לא יעבוד (קפיצה של כ-13% ב-Validation).

- ניסיתי לעשות Ensemble של 2 מודלים שהתאמנו על מילים ומודל אחד שהתאמן על תווים (במקרה של תיקו לקחתי את הניבוי של המודל שהתאמן על תווים מכיוון שהיה לו את אחוזי הניבוי הגבוהים ביותר). קיבלתי שיפור של כחצי אחוז בניבוי על ה-Validation עם הצבעה. החלטתי לוותר על מודל זה כי השיפור לא היה מספיק משמעותי ביחס לגודלו (אחת הדרישות לציון גבוה הייתה מודל מהיר). הוא היה מאוד כבד (בנוי משלושה מודלים שונים) ולקח זמן רב להריץ אותו על דוגמאות חדשות.

- ניסיתי להוריד דוגמאות עם גודל תמונה קטן (גובה * אורך = שטח) אך בניגוד לאינטואיציה שלי התוצאות ירדו.

- במאמר ArcFace בעמוד 8 (Experimental Settings) הכותבים בנו רשת שבנויה מחילוץ פיצ'רים על ידי מודל Resnet50/Resnet100 ולאחר מכן להעביר את ה-Feature Vector בשכבות:
  - Batch Normalization
  - Dropout
  - Fully Connected
  - Batch Normalization

  ניסיתי לממש את הרעיון עם Circle Loss במקום ArcFace Loss אך זה פגע בביצועים וגם האריך את זמן הריצה.