ASLie

ASL made simple

מרצה

ד"ר משה בוטמן

מגישים

ספיר מוסאצ'ו	318654001
תומר יעקב	316593722
עדי קנפו	207040783
עמית נאמן	315856468
דניאל אידליס	205907298

תוכן עניינים

תוכן עניינים **2**

תקציר פרויקט **3**

הגדרת עולם הבעיה **4**

מה נעשה בתחום **5**

הצדקה אקדמית **7**

מהלך העבודה **8**

חלקי המודל 10

Detection 10

Keypoints 11

Classification 12

אימון 22

מבנה המודל 24

תוצאות 28

צד הלקוח 31

תרשים מודולים כללי של המערכת **32**

כלי הפיתוח, תוכנה וחומרה **33**

תוצר סופי **34**

קישורים לקוד **35**

תקציר פרויקט

בעולם בו אנו חיים, רובנו רואים את היכולת לתקשר עם הסביבה כמובנת מאליה. רוב בני האדם הבריאים מגלים את היכולת לתקשר עם הסביבה בצורה כזאת או אחרת כבר בגיל 7-8 חודשים. עד גיל שנתיים מרביתם כבר מבינים כמה עשרות מילים ובד"כ גם מסוגלים כבר לדבר בכוחות עצמם.

לעומת זאת, בני אדם אשר נולדים עם קשיי שמיעה בדרגות שונות מתקשים לעשות זאת מכיוון שלעיתים בעיות שמיעה לא מתגלות בשלב מוקדם מספיק ועל ההורים ללמוד דרך חלופית לתקשר עם ילדיהם – בדרך כלל שפת סימנים. פער זה מצטמצם עם השנים ועד גיל 6 מרבית כבדי השמיעה מסוגלים לנהל שיחות מלאות בשפת הסימנים בדומה לבני גילם הבריאים. חלקם גם מסוגל לדבר בצורה רגילה לחלוטין בהתאם לדרגת איבוד השמיעה.

מטרת הפרויקט שלנו היא לאפשר עוד דרך תקשורת עבור אנשים עם בעיות שמיעה.

ASLie מאפשרת לאנשים שדוברים את שפת הסימנים האמריקאית לדבר מול מצלמה ולקבל כפלט טקסט שמהווה תמלול של השיחה.

ASLie משתמשת במספר יכולות ML ע"מ לעשות את התרגום משפת סימנים לטקסט. יכולות אלה באות לידי ביטוי בשלושת הרכיבים העיקריים במערכת:

1. **מודל Detection:** זיהוי כף היד בתמונה (או סרטון) ע"מ לבצע חיתוך של התמונה באזור כף היד.
2. **מודל Keypoints:** הוצאת קווי מתאר של היד כדי להקטין את מימד הבעיה ולייעל את תהליך הפענוח.
3. **מודל Classification:** תרגום מנח כף היד לאות המתאימה ב ASL.

הגדרת עולם הבעיה

ASL - American **S**ign **L**anguage היא שפת הסימנים הנפוצה ביותר בקרב אנשים עם בעיות שמיעה בארה"ב ובמדינות דוברות אנגלית אחרות. ASL מורכבת מאוסף אותיות התואם את זה של השפה האנגלית כך שלכל אות בשפה האנגלית יש סימן מתאים לה ב ASL – כמתואר בתרשים:

1 מתוך 20 אנשים בארה"ב סובלים מאובדן שמיעה כאשר חצי מהם מעל גיל 65 - מדובר בכ 15 מיליון איש.

יתרה מכך, בארה"ב יש רק 500,000 דוברי ASL שאינם חרשים וקשישים אמריקאיים רבים הסובלים מבעיות שמיעה לא מסוגלים לתקשר עם בני המשפחה שלהם כמו שצריך.

כאמור, ASLie שואף לתווך בין דוברי ASL לדוברי האנגלית, ע"י כך שהיא מאפשרת לדוברי ASL לתקשר בצורה נוחה להם ולדוברי אנגלית להבין אותם.

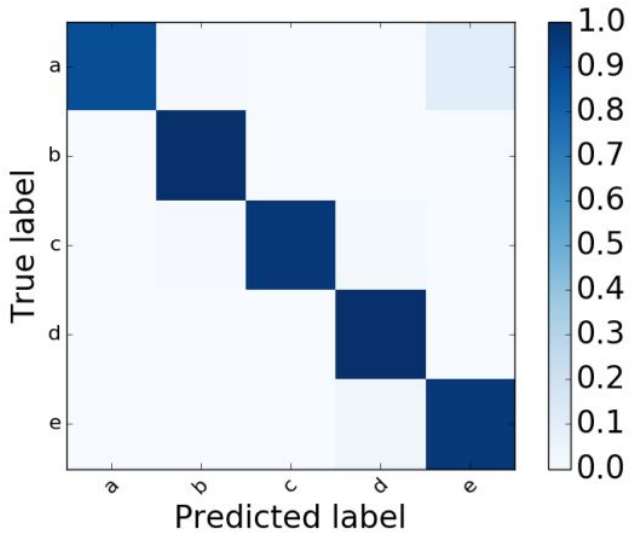
מה נעשה בתחום

רוב התוצרים שמצאנו בתחום, מנסים לפתור את הבעיה מהכיוון ההפוך, כלומר תרגום מטקסט לתמונה. דוגמאות בולטות לפרויקטים כאלה הן:

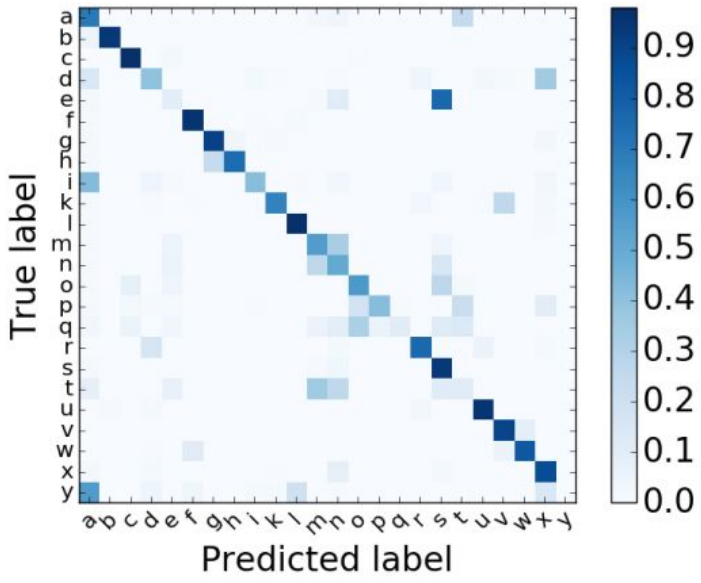
1. **Fun Translations / WeCapable:** אתרים שממירים טקסט באנגלית לתמונות מצוירות של שפת סימנים. מדובר בפרויקטים פשוטים שממפים אותיות טקסטואליות באנגלית לתמונות של כפות ידיים המציינות את האותיות ב ASL.

2. **Hand Talk Translator / Mimix:** אפליקציות שממירות טקסט או שמע באנגלית לסרטון של דמות תלת מימדית ש"אומרת" את המשפט ב ASL. הפרויקטים משלבים יכולת טכנולוגית מעניינת יותר מאלה בסעיף הקודם כי במקום מיפוי פשוט של אות לתמונה, מצוירת אנימציה חלקה וטבעית של הדמות כאשר היא עוברת בין התנועות השונות.

בנוסף, מצאנו מאמר של סטנפורד על ניסוי מעניין עם CNN שמטרתו לפתור בעיה זהה. מחברי המאמר השתמשו ב Transfer Learning מרשת GoogLeNet שאומנה על Dataset בשם ILSVRC 2012. כפי שניתן לראות, על Dataset מצומצם של חמשת האותיות A עד E הם הגיעו לתוצאות טובות:



אך כאשר מתבוננים ב Confusion Matrix המלא, ניתן לראות כי הביצועים הכוללים של הרשת אינם מספקים:



בחלק המסכם ניתן לראות את ההסבר שלהם לחוסר ההצלחה - חוסר גיוון ב Dataset:

“Because of the lack of variation in our datasets, the validation accuracies we observed during training were not directly reproducible upon testing on the web application.”

לסיכום, לא הצלחנו למצוא דוגמאות לפתרונות מוצלחים לבעיה אותה אנו מנסים לפתור ולכן החלטנו לבחור להתעמק בה כפרויקט הגמר שלנו.

הצדקה אקדמית

כאמור, ASLie בנויה מ 3 רכיבים עיקריים:

1. **מודל Detection:** זיהוי כף היד בתמונה (או סרטון) ע"מ לבצע חיתוך של התמונה באזור כף היד.
2. **מודל Keypoints:** הוצאת קווי מתאר של היד כדי להקטין את ממדי הבעיה ולייעל את תהליך הזיהוי.
3. **מודל Classification:** זיהוי מנח כף היד לאות המתאימה ב ASL.

שלושת הרכיבים הנ"ל מאפשרים ל ASLie לתרגם שפת סימנים לטקסט בזמן אמת ע"י שימוש במצלמות הנמצאות היום בשפע מכשירים שאנו משתמשים בהם ביום-יום (כדוגמת סמארטפון או לפטופ).

ASLie בעלת ממשק Web-י אשר ניתן להרצה ע"ג פלטפורמות ניידות ונייחות כאחד במטרה לאפשר גמישות מקסימלית עבור המשתמשים. בממשק הנ"ל ניתן לצפות ב Live Feed מהמצלמה שמותקנת על המכשיר שבו משתמשים (במידה והיא קיימת) וכאשר המשתמש מתקשר ב ASL מול המצלמה, המערכת תדפיס לו את התמליל ע"ג המסך עצמו.

ניתן לראות כי בכל חלקי הפרויקט השתמשנו במגוון רחב של כלים שלמדנו לאורך התואר ובפרט מיומנויות שרכשנו במסגרת קורסי ההתמחות בנושא Deep Learning.

מהלך העבודה

על מנת לממש את ASLie השתמשנו בכמה מודלים של DL מסוגים שונים.

ביצענו מחקרים רבים וניסויים שונים על המודלים על מנת לקבל את התוצאה הטובה ביותר.

לאחר מחשב רב בו חשבנו על איך לממש את הרעיון הגענו לשני כיווני חשיבה שונים:

1. <u>קלאסיפקציה מבוססת Keypoints -</u> הוצאת Keypoints מהתמונה ולאחר מכן סיווג התמונה על בסיס ה Keypoints.
2. <u>קלאסיפקציה מבוססת תמונה -</u> קבלת תמונה ממצלמת הלקוח ושליחה אל מודל CNN אשר יחזיר את סיווג האות.

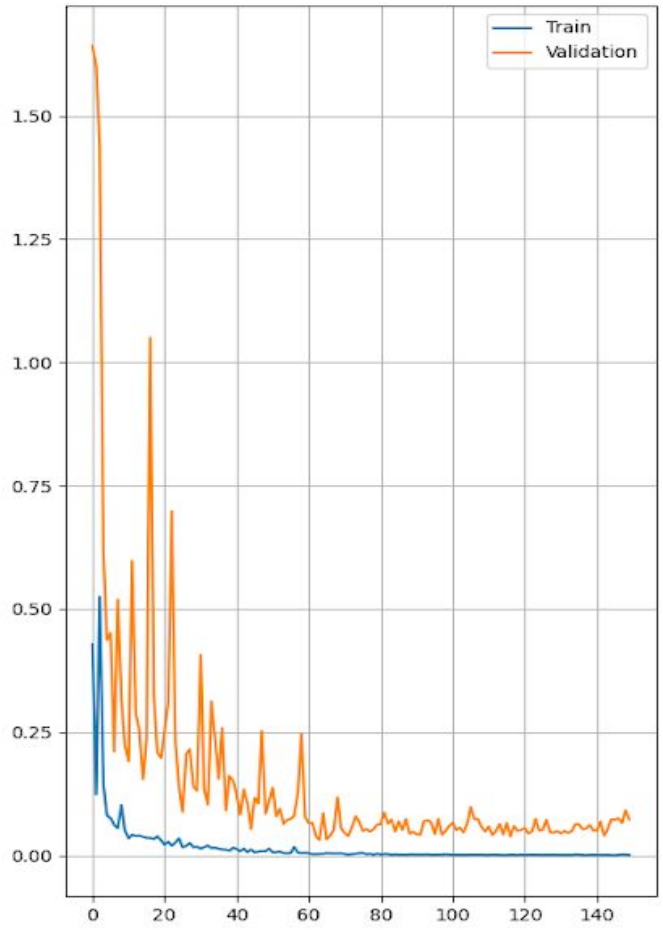
לאחר דיון נוסף וכמה ניסויים שיפורטו בהמשך, בחרנו להמשיך עם כיוון המחשבה הראשון מכיוון שסיווג Keypoints יצמצם את מרחב הבעיה ויתאים יותר כפתרון ע"ג פלטפורמה Webית.

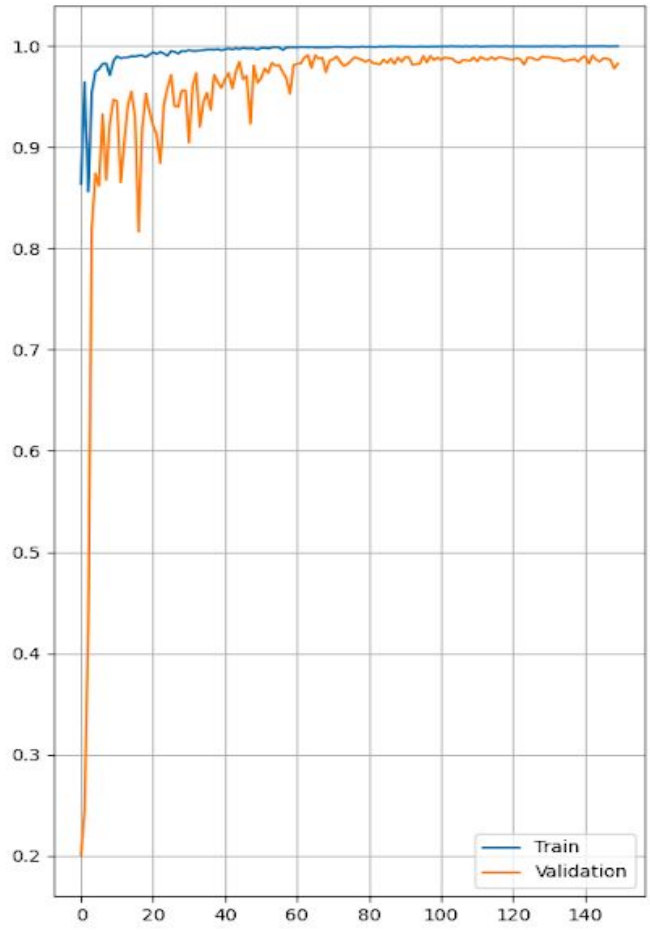
השיקולים ללכת לכיוון של Keypoints:

- <u>איכות הסיווג -</u> אימון רשת CNN על גבי מספר גבוה יחסית של class-ים, שחלקם מאוד דומים האחד לשני, מתמונות באיכות ירודה יחסית, הוא תהליך ארוך וקשה יותר וכנראה שבסופו לא נצליח לתת תוצאות מדויקות כמו שרצינו כפי שראינו בניסויים שערכנו.
- <u>משאבים -</u> רשת CNN למקרה הזה הרבה יותר יקרה מבחינת משאבים וזמן אימון. אימון Epoch אחד של מסווג תמונות שקול מבחינת זמן לאימון מודל שלם של מסווג Keypoints.
- <u>זמן חישוב -</u> אחת מדרישות המערכת זה ביצועי Realtime. על מנת להגיע לביצועי Realtime יש לתכנן רשת שתוכל לעבד מספר גבוה של פריימים (FPS גבוה), כלומר כל שנייה תוכל לתת מספר גבוה של פרדיקציות. בנוסף, התעבורה על גבי הרשת גם היא גורם המוריד את ה FPS ומוסיף Latency למערכת. למשל, אם נרצה להגיע ל 30 FPS, נצטרך לשלוח 30 תמונות בשנייה לשרת, לקבל פרדיקציה מהרשת, ולהחזיר אותה ללקוח. מבחינת תעבורת רשת, עצם שליחת 30 התמונות יכולה להוות בעיה, אל מול שליחת Keypoints שמיוצגים כוקטור בגודל קבוע.

כחלק מהעובדה, ביצענו הרבה ניסויים על התמונות עצמן, חלקם אף הגיעו לתוצאות דיי טובות.

בדוגמה רואים תוצאות (Loss ו Accuracy) של רשת שאומנה על תמונות A-E.





לאחר מחקר נרחב הגענו למסקנה כי למודלים של MediaPipe[1] (פרויקט רחב היקף של חברת גוגל הנותן מענה לפרויקטי ML על גבי ה-Mobile ו-Web) יש את הפוטנציאל הרב ביותר.

למסקנה זאת הגענו משתי סיבות עיקריות:

1. <u>ביצועים -</u> איכות הפרדקציות שהמודלים נותנים גבוהה מאוד.
2. <u>תשתית -</u> כיוון ש-MediaPipe מכוונים לעולם ה-Web, ניתן להריץ את המודלים שלהם בצורה מאוד נוחה וקלה בצד לקוח ללא התפשרות בביצועים. הדבר מתאפשר ע"י שימוש בספריית TFLite.

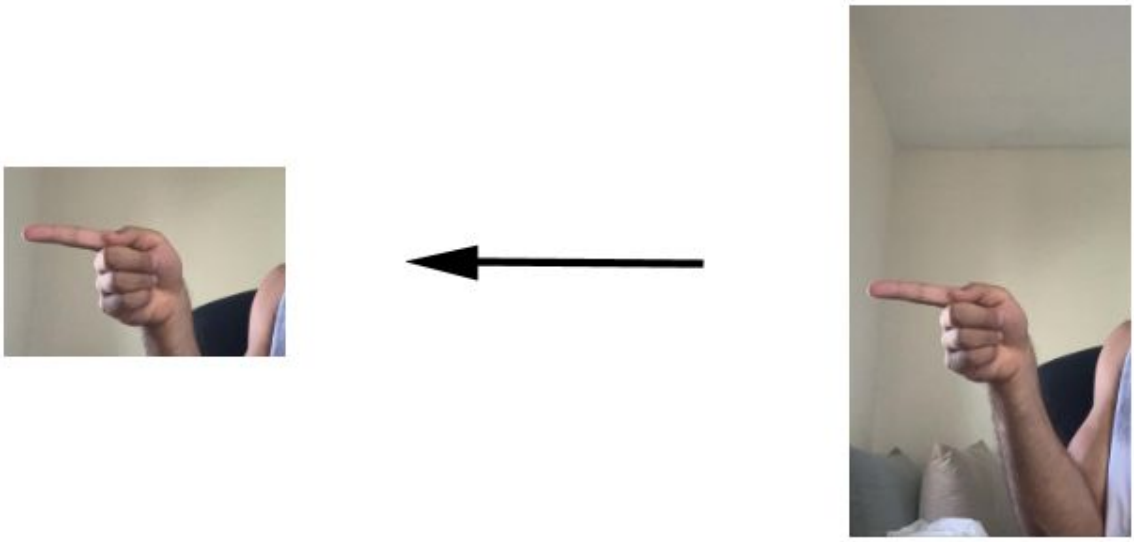
[1] https://github.com/google/mediapipe

חלקי המודל

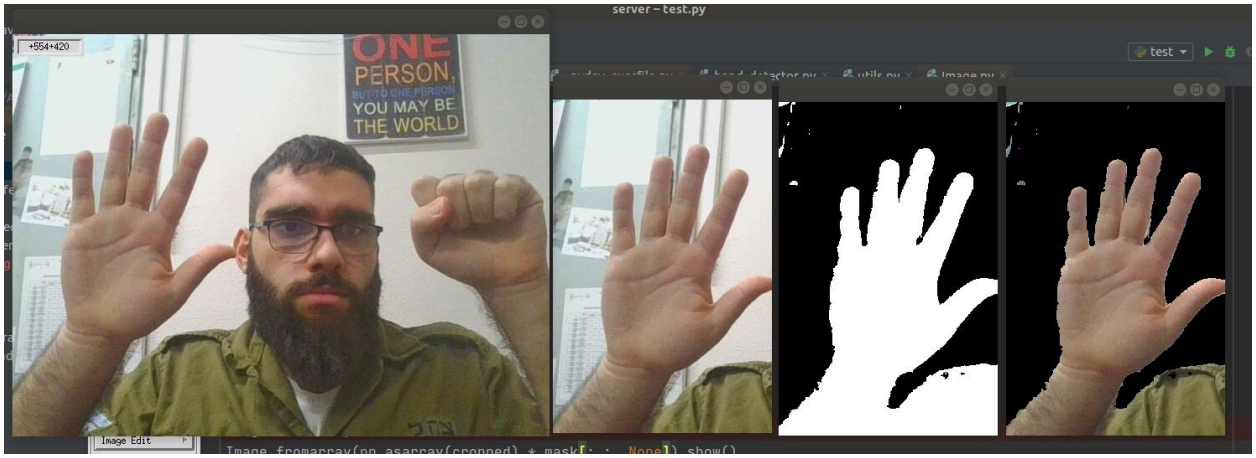
1. Detection

<u>תהליך בחירת המודל</u>

מודל זה מוציא לנו פרדיקציה שאומרת היכן הוא חושב שהיד נמצאת על גבי הFrame. לאחר קבלת הפרדיקציה נקבל Bounding Box אשר איתו נוכל לעשות לבצע חיתוך ל Frame ובכך לקבל תמונה ממוקדת יותר על היד. ה Bounding box הוא vector מהצורה (x1, y1, x2,y2) כך שנוצר המלבן בו נמצאת היד. החיתוך מתבצע לגודל של 240X320 פיקסלים.



כחלק מהניסויים והמחקר שערכנו, בדקנו עוד מודלים נוספים אשר לא הגיעו לתוצאות מספקות. לדוגמא: אחד המודלים לא צלח כלל כאשר הרקע היה חום או מורכב מכמה צבעים. על מנת לשפר את המודל Detection, ניסינו שיטות של עיבוד תמונה על ידי OpenCV כדי לעשות Background Subtraction. ראינו, כי במקרים מסויימים ה Background Subtraction שביצענו שיפר מאוד את ביצועי ה Detector.



בדוגמה רואים פילטר פשוט שמחיל Threshold על channel ה-S בפורמט HSV של תמונה. ניתן לראות שבמקרה הנ"ל בוצע Background Subtraction מושלם.

למרות זאת, השיטה מאוד רגישה לשינויי צבעים בתמונה ולהבדלים ביניהם. כתוצאה מכך, על רקעים מסוימים הפילטר עובד מושלם, אך עבור מצבים אחרים, הפילטר משבש לגמרי את התמונה עד כדי שלא ניתן ללמוד ממנה יותר. לכן, הוחלט שלא להשתמש בשיטה "שאינה נלמדת", כלומר היא מעוצבת על ידי אדם, ולא חכמה או מתאימה את עצמה למודל.

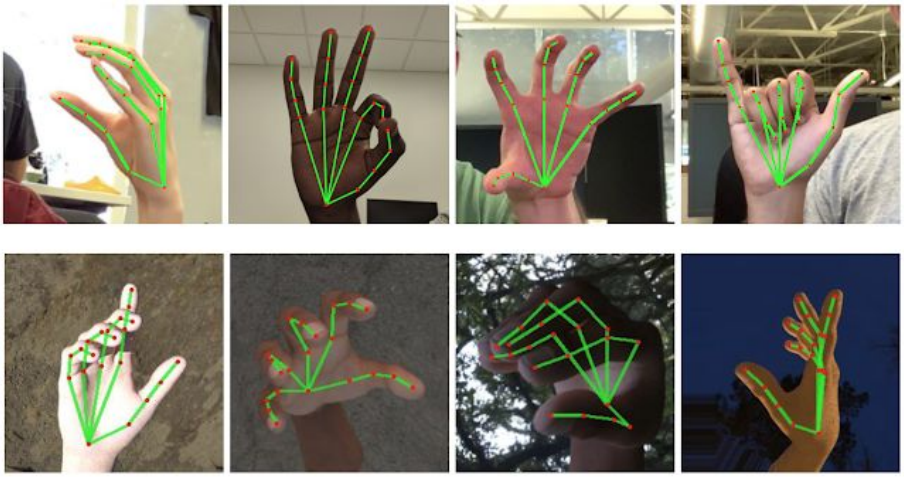
הבחירה הסופית -

מודל ה-Detection הסופי שנלקח הוא מודל ה Palm Detection של MediaPipe. המודל מחזיר, בנוסף ל-Bounding Box, גם את ה Score שהוא נותן לאותה box, כלומר את רמת הביטחון שלו בין 0 ל 1.

2. Keypoints

מודל זה מוצא על גבי הFrame החתוך שהוציא מודל ה-Detection את ה Keypoints.
ה-Keypoints, או Landmarks הם 21 נקודות (x,y) על כף היד, המייצגים את מנחי האצבעות כך שבעצם כל מפרק בכף היד הוא Keypoint. בעזרת ה Keypoints הללו אפשר בעצם לשרטט את התנועה אותה מבצעת כף היד. למודל אנו מגדירים ערך סף מסוים וכאשר המודל עובר אותו נקבל חזרה את 21 הצמדים של ערכי x,y המרכיבים את היד כאשר סדר הנקודות קבוע. לדוגמא: נקודת ה-0 תמיד תהיה שורש כף היד. את 21 הנקודות אנו הופכים לווקטור חד מימדי מגודל 42.

בדומה למודל ה-Palm Detection, גם מודל ה Hand Landmark מחזיר את רמת הביטחון שלו עבור כל



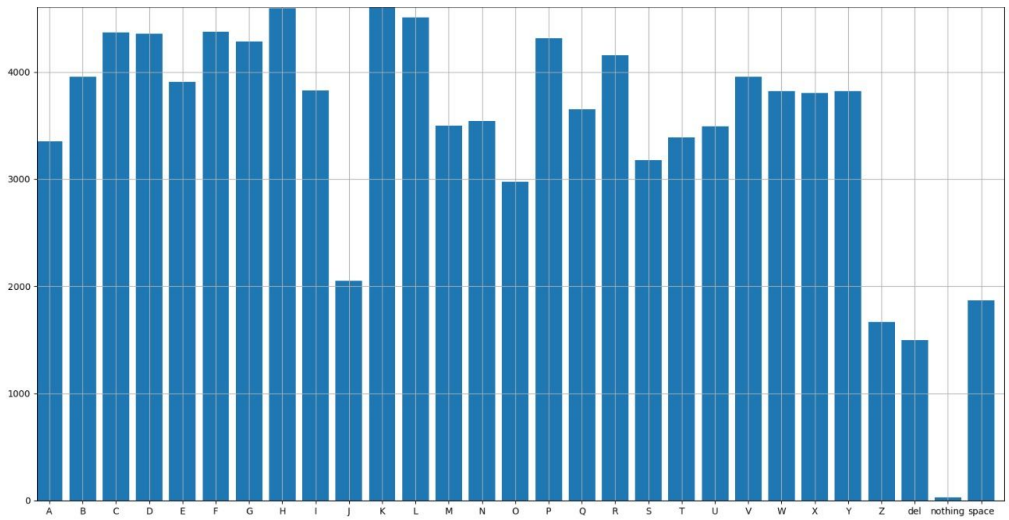
Input.

3. Classification

מודל זה מקבל את הווקטור ממודל ה Keypoints המוזכר לעיל ומחזיר לנו פרדיקציה שאומרת איזה אות הווקטור מייצג.

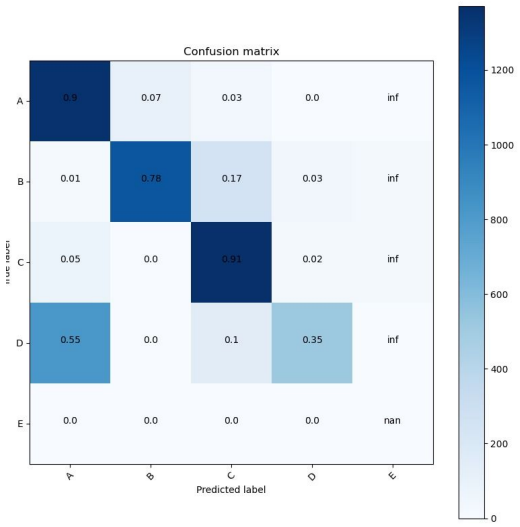
על מנת לבנות את הרשת התחלנו בלאסוף תמונות רבות של אותיות מה ASL ולבסוף אספנו כ-170 אלף תמונות מ Datasetים שונים שמצאנו ב-Kaggle ומקומות נוספים באינטרנט. לאחר איסוף התמונות התחלנו בתהליך Pre Process ידני בו וידאנו מדגמית כי התמונות אכן באיכות טובה ומסווגות נכון. לאחר מכן, המשכנו לתהליך אוטומטי שמייצר לנו מכל תמונה וקטור של 42 על מנת לדמות את ה Input שהמודל מקבל מהלקוח. לכן, כל תמונה עברה דרך שני המודלים המוזכרים לעיל על מנת לבנות לנו dataset מתאים.

לאחר תהליך זה נשארו לנו כ-90 אלף תמונות. התמונות התפלגו בצורה הבאה:

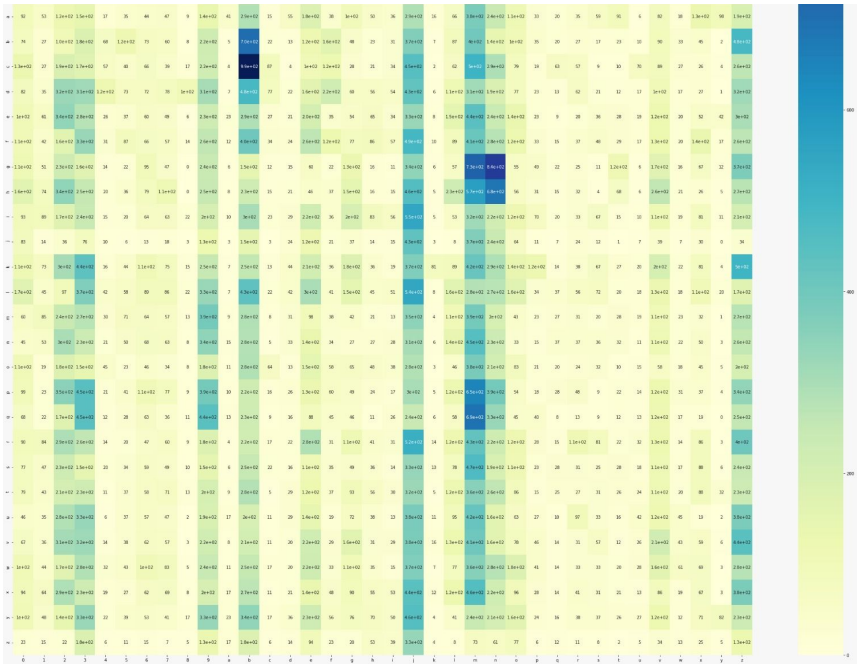


בנוסף, חתכנו חלק מה-Dataset לכמות מצומצמת יותר של אותיות על מנת לקבל זמני אימון קצרים יותר. המטרה היא לפתור קודם את הבעיה ב-Scale קטן יותר, להוכיח היתכנות ולבצע Hyperparameters Tuning על המודל, ורק לאחר מכן להרחיב ל-Dataset המלא. בצורה שרירותית, בחרנו לבדוק את המודל על חמשת האותיות הראשונות, A-E.

כאשר התחלנו לאמן את הרשת על גבי ה Dataset הקטן ובדקנו את התוצאות, קיבלנו את ה Confusion Matrix הבאה:



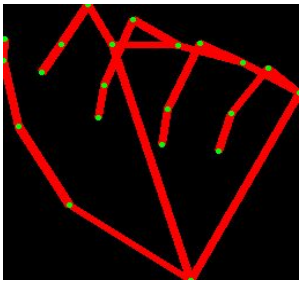
ניתן לראות כי התוצאות לא מספקות והרשת לא חזתה בצורה מספקת את האותיות. בשלב זה ניסינו בכל זאת לאמן גם על ה Dataset המלא וקיבלנו את התוצאות הבאות:



קל לראות כי בתמונה לעיל הרשת אינה מצליחה לחזות אותיות בצורה נכונה. יתרה מכך, לא הצלחנו להגיע לתוצאות קרובות לטובות אפילו על ה-Dataset המצומצם של A-E.

בשלב הזה עלו כמה כיווני מחשבה:

- הרשת\המודל שבחרנו לא חזק או גדול מספיק. לכן, ניסינו להרחיב את הרשת, ולאחר מכן גם להעמיק את הרשת. בנוסף ניסינו לבחון ארכיטקטורות שונות ומגוונות כמו הוספת Residual Blocks. לצערנו השיטות הללו לא הצליחו לפתור את הבעיה.
- עלתה ההשערה כי למודל FC פשוט יחסית אין די כוח על מנת לפתור את הבעיה, ועדיין יש להשתמש באיזושהי רשת CNN. ניסינו לשלב את ה Keypoints עם תמונת המקור, במודל שמשתמש ב Feature Extraction מהתמונה, FC כדי להוציא מידע מה-Keypoints, ולאחר מכן לעשות Concat ולנסות



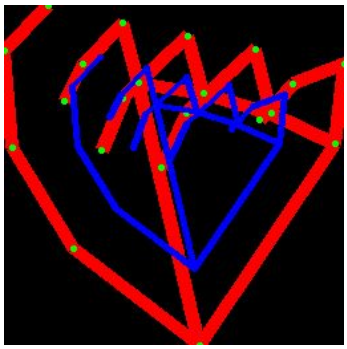
להכריע על פי השילוב.

בנוסף, ניסינו לקחת את ה-Keypoints, להפוך אותם לתמונה שבה רואים את ה-Keypoints על רקע שחור (משמאל), שוב על מנת לעשות שימוש ביכולות Feature Extraction של CNN.

- מכיוון ש"המרנו" את הבעיה לבעיה שיותר דומה לעולמות ה Data Science (כלומר, ה-Input הוא לא תמונה אלא אוסף מיקומים) חשבנו לנסות לעשות כל מיני Feature Extraction ידניים, לפי שיטות שנראות לנו הגיוניות ויכולות לסווג בצורה טובה יותר את ה-Keypoints. למשל, הרבה Classים הם כמעט זהים למעט מנח שונה של האגודל, לכן, זווית האצבעות ואגודל בפרט יכול להיות Feature מאוד חזק להזריק למודל.
- ל-MediaPipe יש מודל נוסף של Hand Landmarks שנותן Keypoints בתלת מימד, כלומר ה-Output שלו הוא נקודות (x,y,z). ערך ה z יכול להיות מאוד מאוד משמעותי למודל, בייחוד כי ישנם Classים שחלק מהאצבעות נמצאות מאחור.

מניסויים שערכנו איתו, נראה כי המודל התלת מימדי אינו מהימן וכי ערך ה-Z לעיתים תכופות לא מייצג טוב את המרחב.

לבסוף לאחר בדיקה של ה Dataset, הגענו למסקנה שה-Data לא מספיק איכותי.



ראשית, ראינו כי תמונות בגדלים שונים מביאות ל-Keypoints בסקאלות שונות מאוד, ומיקום היד בתמונה משפיע מאוד על הפרדיקציה (מה ש CNN תיאורטית אמור לפתור). לכן, בדקנו מה השיטה הכי נכונה וטובה לנרמל את ה-Data.

תחילה, חשבנו על לקבוע "נקודת ציר" שהיא (0,0), למשל פרק כף היד, ולקבוע את כל הנקודות לפיה. *(בכחול: היד המקורית, באדום: היד המנורמלת)*

עדיין נותרנו עם בעיית הגודל, ולכן לבסוף, על מנת לנרמל את הערכים אנו לוקחים את ערך ה-X וה- Y הגדולים והקטנים ביותר (לאו דווקא מאותה נקודה) ומנרמלים על פיהם כך שערכי המינימום והמקסימום לאחר הנירמול יהיו 0 ו- 1 בהתאמה. בכך, אין קשר לגודל היד או מיקומה בתמונה.

דוגמא לקוד שמבצע את הנירמול:

```
normalized = data.reshape(-1, 21, 2)

maximum = np.max(normalized , axis=1)

minimum = np.min(normalized , axis=1)
```

ranges = maximum - minimum

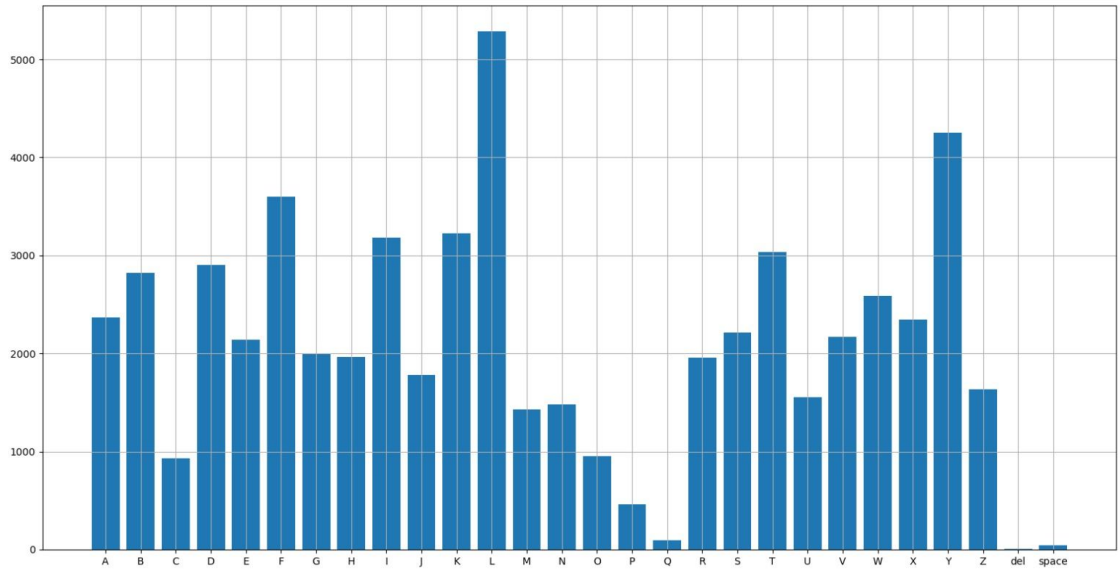
normalized = (normalized - minimum[:,None,:]) / ranges[:,None,:]

בנוסף, הבחנו על ידי התאמה בין ה-Keypoints ל- Label ולתמונה המקורית כי חלק ניכר מן ה Keypoints אינם מייצגים את האות אותה הם אמורים לייצג. על מנת לשפר את איכות ה- Dataset החלטנו להעביר את כל התמונות מחדש דרך שני המודלים כאשר החלטנו להעלות את ערך הסף איתו אנו מחליטים האם להשתמש בתמונה או לא.

הבחנו כי חלק מהתמונות ב-Dataset היו כבר חתוכות, ולכן הרצת מודל ה Palm Detection עליהן הייתה מיותר ואף בעייתית. לכן, העברנו את ה Dataset דרך Pipeline שמנסה להעביר את התמונה ב-Detector אך גם מנסה להעביר אותה ישירות ל Keypoints, ולקחנו עבור כל תמונה את התוצאה הגבוהה ביותר מבחינת רמת הביטחון של המודל. בנוסף, החלנו Threshold של 0.9 על המודל Keypoints, כלומר לקחנו רק את הדוגמאות עבורן המודל היה בטוח בסבירות של 90 אחוזים.

ה Threshold נקבע אחרי תהליך ארוך של מחקר על המודל של MediaPipe שכלל קריאת דוקומנטציות והרבה ניסוי וטעייה. ראינו כי לעיתים, אפילו במצב שהמודל בטוח בסבירות של 80, התוצאות לא נראות כמו היד ולכן הן יכולות להוביל למצב שבו המודל השלישי לא מסוגל ללמוד כלום מהמידע. לכן, לקחנו מקדם ביטחון מסויים והחלטנו על 0.9.

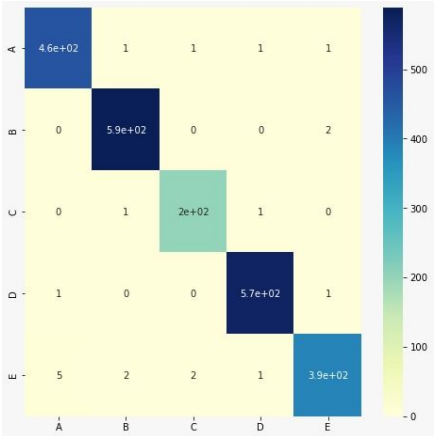
הרצת התמונות בצורה הנ"ל הניבה כ-50 אלף תמונות על פי ההתפלגות הבאה:



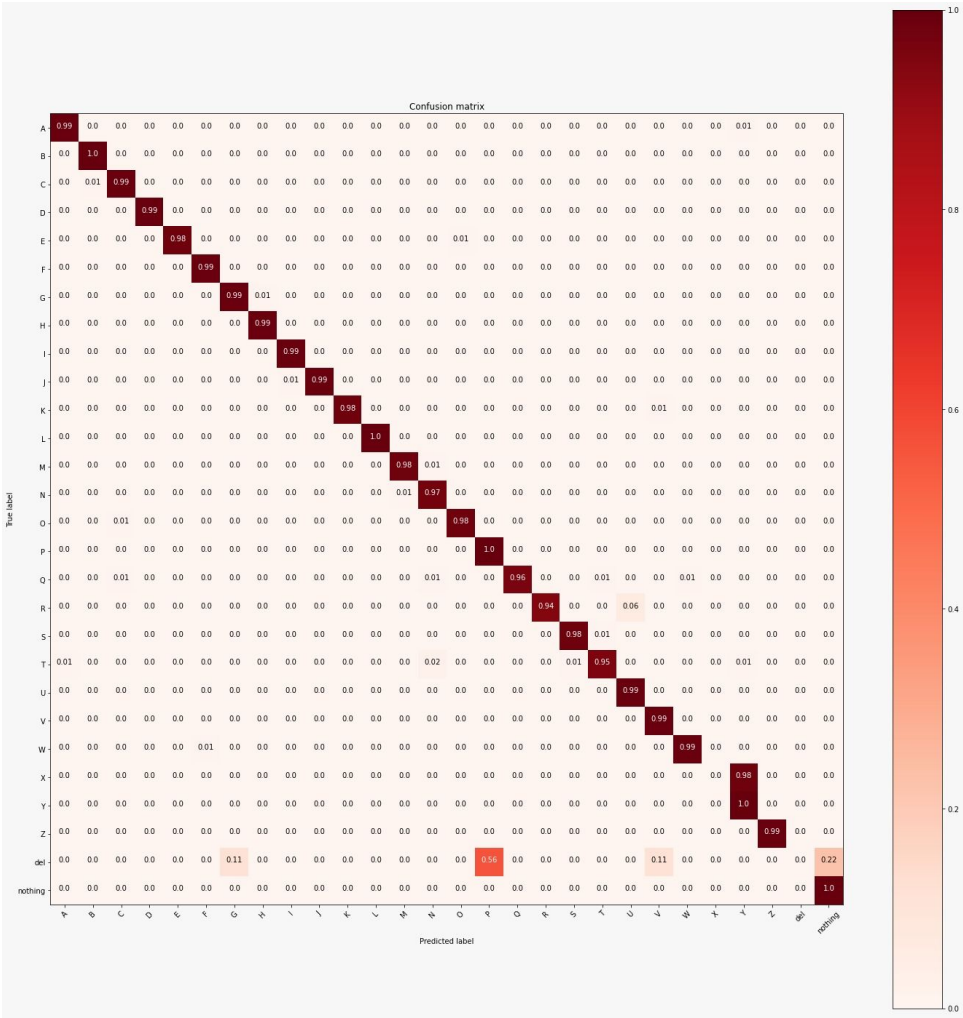
ניתן להבחין כי ישנן אותיות שבהן כמות קטנה מאוד של דוגמאות, לעומת אותיות שלהן כמות רבה מאוד של דוגמאות, כלומר ה Dataset אינו מאוזן. למשל לאות Q יש כמה מאות דוגמאות, לעומת האות L עם יותר מ 5000 דוגמאות. הסיבה לכך היא ששני המודלים הראשונים מזהים טוב יותר מנחי יד בהם כל האצבעות חשופות. באופן הגיוני, כאשר מודל ה Keypoints מנסה לחזות תמונות בהן לא רואים את כל האצבעות באופן ברור, או בכלל, הוא יתן Score נמוך לתמונה. לכן, לאחר החלת Threshold גבוה, איבדנו כמות גדולה של תמונות מהסוג הנ"ל.

Q - Bad L - Good L - Bad

שוב, גם לאחר תיקון ה-Dataset, ביצענו קודם כל ניסוי על ה Dataset הקטן:



ניתן לראות כי התוצאות הנ"ל מראות שהמודל מצליח לחזות בצורה טובה מאוד את האותיות:



בניגוד ל Datasetים של CV קלאסי כמו ImageNet או CIFAR שבהם ניתן לייצר הפרדה אבסולוטית בין הקלאסים (למשל כלב ומטוס במקרה של CIFAR), חלק מהקלאסים במקרה שלנו קרובים מאוד אחד לשני (לדוגמא U, V, K שונות אחת מהשנייה במנח של אצבע אחת בלבד).

U V K

כדי להתגבר על הבעיה הזאת ועל מנת לשפר את הביצועים עוד יותר, החלטנו לבנות ארכיטקטורה של מודל היררכי בהמלצתו של המנחה. ההיגיון מאחורי ההחלטה הוא פשוט: אותיות דומות יהיו בעלות Feature-ים דומים וכך נוכל לבצע חלוקה לוגית של האותיות לקבוצות ע"פ הדמיון ביניהם. המודל יחזה לאיזה קבוצת אותיות הווקטור שייך ובהתאם לשיוך הווקטור לקבוצה נזין את הווקטור למודל נוסף המאומן רק על אותה קבוצת אותיות. בעצם אנחנו לוקחים בעיה קשה, הפרדת U, V ו K בנוסף לשאר האותיות לבעיה פשוטה בהרבה: האם התמונה היא U V או K.

נשאלת השאלה כיצד לחלק את הקבוצות, ועלו מספר אפשרויות:

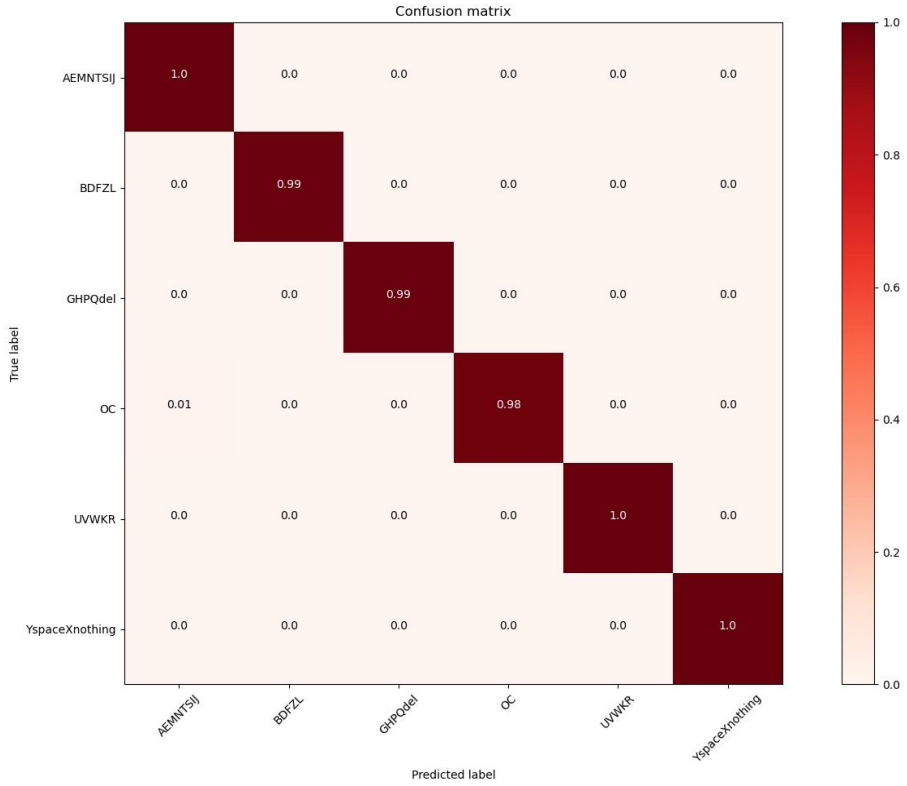
- <u>מיפוי לפי מרחב פיצ'רים -</u> להריץ CNN לקלסיפיקציה על התמונות ולחתוך אותו לאחר ה Feature Extraction. כעת, ניתן לחפש Clusterים קרובים במרחב הפיצ'רים ע"י חישוב מרחקים או ע"י שיטות Unsupervised Learning למציאת Clusterים.
  החלטנו לזנוח את הרעיון בעיקר כי הלכנו לפתרון שבו לא מריצים את התמונות, אז לא השקענו בבניית Classifier שכזה.
- <u>לפי Confusion Matrix -</u> להריץ מודל על כל ה Data ולאחד כל פעם קבוצות שבהן הוא לא בטוח. למשל אם המודל "מתבלבל" הרבה בין V ל U, נאחד אותן לקבוצה. בעצם בתהליך איטרטיבי אנחנו מגיעים למצב בו יש לנו כמות מסוימת של קבוצות שהחלוקה ביניהן מאוד איכותית.
- <u>אינטואיציה -</u> כמו שאמרנו והראנו, ישנן אותיות מאוד דומות ויזואלית, לכן סביר להניח שאם ננקוט בשיטה אחרת, נגיע לתוצאות קרובות לפחות לאינטואיציה.

לבסוף, ביצענו את החלוקה הראשית לפי אינטואיציה, ולאחר מכן שיפרנו על פי Confusion Matrix.

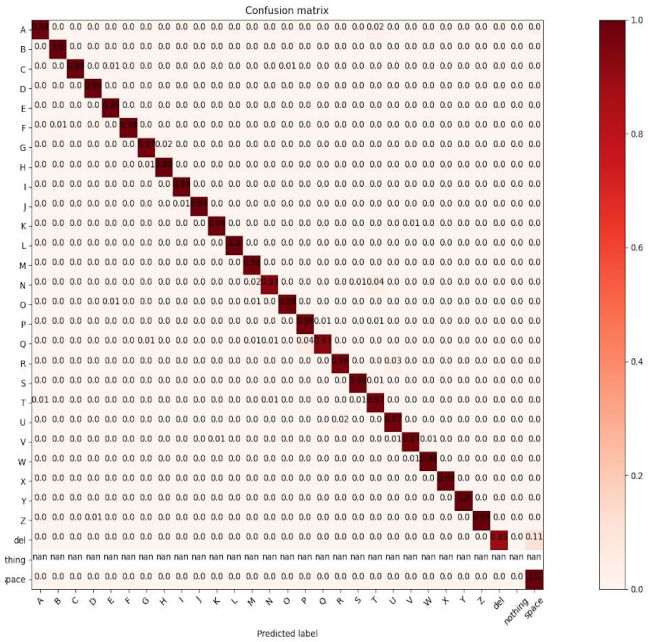
הגענו לקבוצות הבאות:

```
["Y", "space", "nothing", "X"]

["A", "E", "M", "N", "T", "S", "I", "J"]

["B", "D", "F", "Z", "L"]

["U", "V", "W", "K", "R"]

["G", "H", "P", "Q", "del"]

["O", "C"]
```

התוצאות של המודל ההיררכי:



תוצאות לסיווג ברמת האות:



ניתן לראות כי יש שיפור ניכר, בייחוד עבור קבוצות אותיות דומות. למשל, שיפור של כמה אחוזים בין האותיות R ו U, ושיפור גדול עבור X ו Del.

אימון

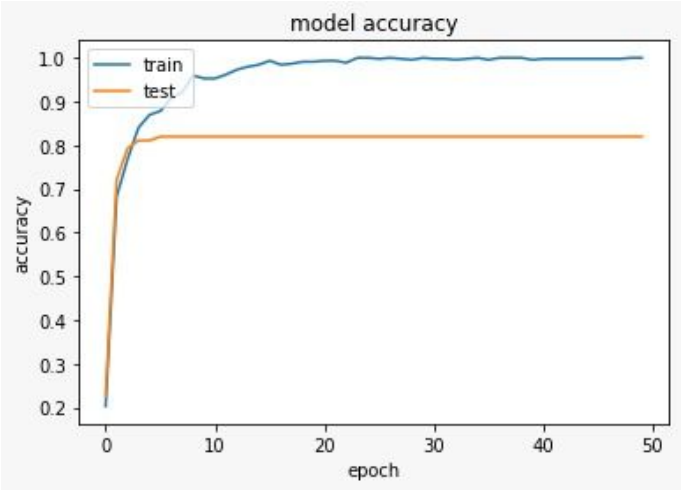
במהלך האימון, השתמשנו במספר טכניקות על מנת להפוך את התהליך ליותר מהיר ואפקטיבי, אבל גם פחות רגיש ל Overfitting:

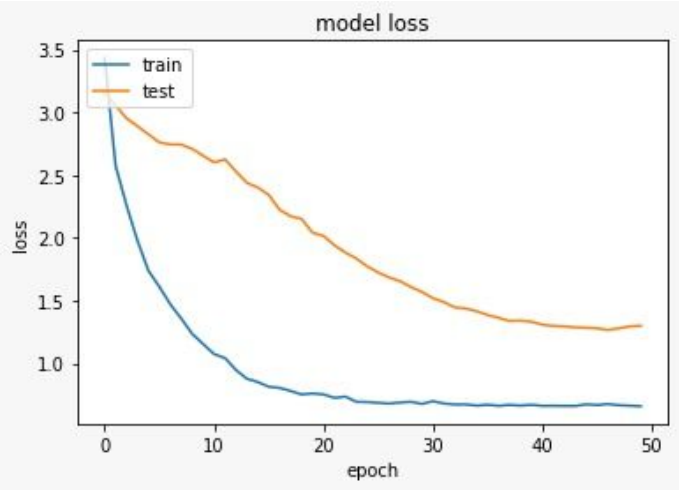
- Early Stopping - אחד המנגנונים הכי חשובים על מנת למנוע Overfitting. הגדרנו כי אם אין שיפור ב Validation Loss במהלך 5 epochים, האימון יופסק.
- שימוש נרחב ב- Dropout, אחרי כל שכבה כמעט.
- שימוש ב Batch Normalization, מקטין Overfitting ומאיץ משמעותית את זמני האימון.
- Label Smoothing - מנגנון נוסף לצמצום Overfitting. למעשה, זה הופך את ה-Labelים ל 0.9 או 0.1 במקום 1 או 0, בהתאמה. התוצאה היא שהמודל מוענש על ביטחון יתר.
- אתחול משתנים - השתמשנו ב-Xavier בהתפלגות נורמלית, במקום Xavier יוניפורמי הדיפולטי ב-Keras.
- שימוש ב Gaussian Noise בשלב האימון.
- Reduce Learning Rate on Plateau - מנגנון ב-Keras שעוקב אחר התקדמות Validation Loss, ויודע להוריד את ה-Learning Rate ב-Factor מסויים אם מגיעים ל-Plateau. הגעה ל-Plateau בערך ה-Validation Loss יכולה להעיד על הפסקת השתפרות, אך גרוע מכך על Overfitting. הורדת ה-Learning Rate יכול להועיל במקרים הללו. כמובן שמקרה שהורדת מקדם הלמידה לא עוזרת, ה-Early Stopping יעצור את האימון. אנחנו השתמשנו בהורדת ה-Learning Rate כל 3 Epochים ללא התקדמות ב Validation Loss, ב-Factor של 0.9.

  היתרון של השיטה, הוא שהיא מאפשרת להתחיל עם מקדם למידה גבוה יחסית, בלי לחשוש שהמודל יתבדר ויגיע מהר מאוד ל-Overfitting, בייחוד כאשר משתמשים בשיטות אופטימיזציה כמו Adam, שנוטות לעדכונים גדולים במשקולות. מקדם למידה גבוה בהתחלה מאיץ את זמן האימון כאשר הרשת לומדת למידה "גסה" יותר, וכדי לעשות את הצעדים הקטנים יותר לקראת הסוף, ה-Learning Rate יורד.

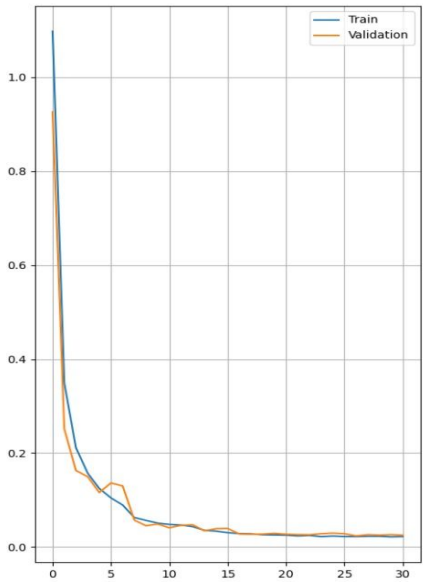
  ניתן להגדיר גם Learning Rate Decay רגיל, כלומר ירידה ב-Factor מסויים כל X Epochים, אך השיטה היתה פחות יעילה בשבילנו, בייחוד כי אנחנו צריכים להגדיר בעצמנו כמה לרדת, בניגוד לשיטה שהשתמשנו בה.

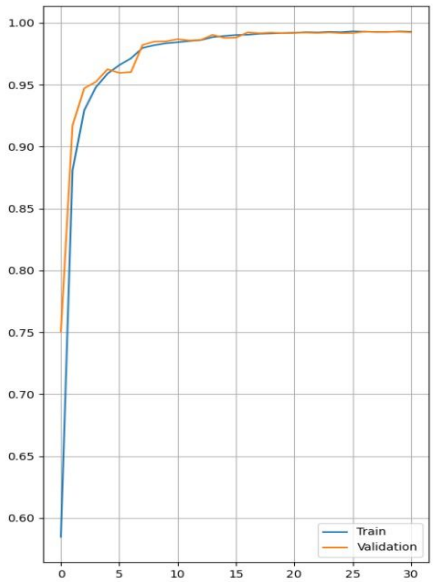
דוגמא לניסוי ללא Dropout ו- Early Stopping:

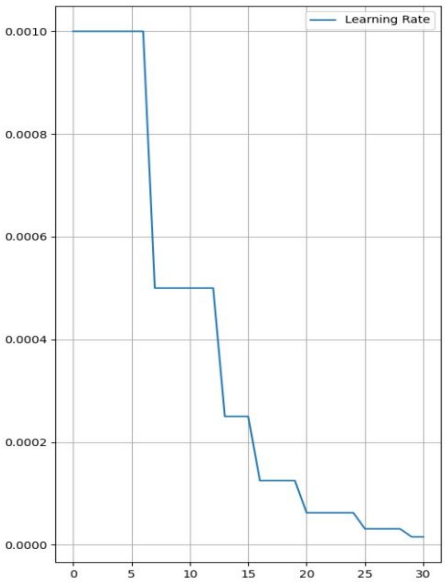




דוגמא לניסוי עם Dropout, Early Stopping ו- Reduce on Plateau.







בגרף הימני, ניתן לראות איך ה-Learning Rate יורד כאשר מגיעים למישור validation loss.

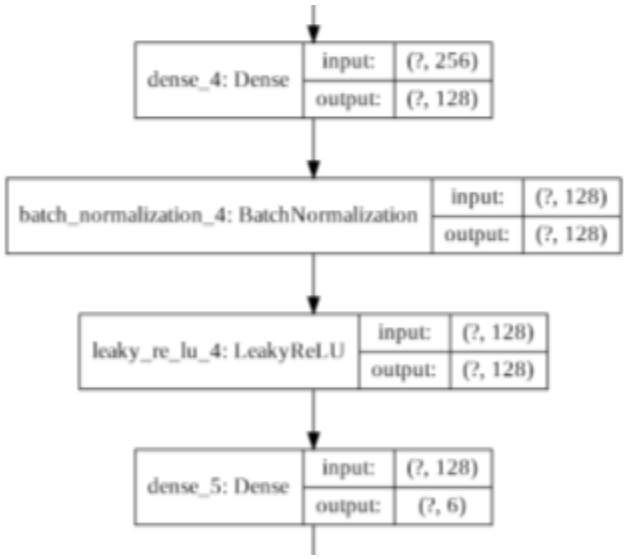
מבנה המודל

מכיוון שהמודל היררכי, בנינו 2 סוגי מודלים: את המודל הגבוה, ה-High model, שתפקידו לסווג בין ה Clusterים השונים, ועוד 5 מודלים מאותו סוג, אחד עבור כל Cluster.

בשני הסוגים השתמשנו בקונפיגורציות הבאות:

- אופטימייזר Adam עם מקדם למידה התחלתי של $5e-4$, בדעיכה של 0.9 אם אין שיפור כל 3 Epochים.
- Categorical Cross Entropy כפונקצית Loss, עם Label Smoothing של 0.1.
- גודל Batch של 64.
- הרצה של 150 Epochים עם Early Stopping אם אין שיפור ב-5 Epochים.
- פונקצית אקטיבציה Leaky Relu.
- Dropout Rate 0.2.
- אתחול משקולות Xavier Normal.
- רעש גאוסיאני בשונות של 0.02.

בנוסף, המודלים מורכבים מבלוקים הבנויים משכבות Dense, BN, Activation ו Dropout, כראה בשרטוט הבא.

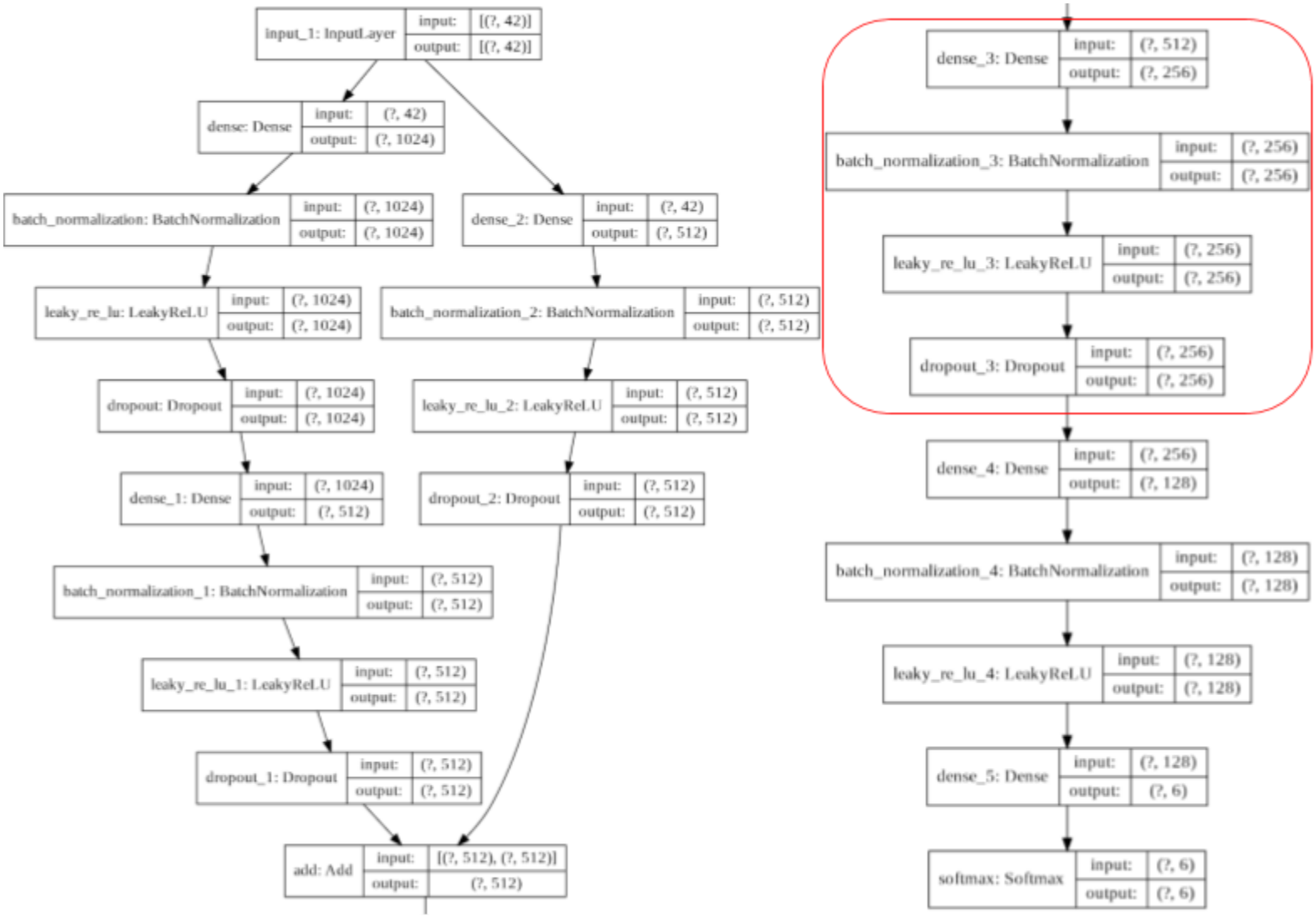


1. High model: המודל הגבוה צריך להיות יותר חזק ועמוק, מכיוון שהוא רואה את כל הדוגמאות בניגוד למודלים הקטנים. בנוסף, השונות של הדוגמאות שהוא רואה גבוהה יותר, כי הוא רואה את כל האותיות לעומת המודלים הקטנים שרואים כמות קטנה של אותיות הדומות אחת לשנייה.

במודל זה, השתמשנו ב Residual Block אחד כפי שניתן לראות בשרטוט בעמוד הבא. לפי ניסויים שערכנו, אותו Residual block שיפר את הביצועים של המודל (השתמשנו ב-Residual Blocks גם במודלים ה"שטוחים" שניסינו טרם המודל היררכי).

Residual Block, או Skip Connection, הוא בעצם חיבור המאפשר לדלג על חלקים ברשת.

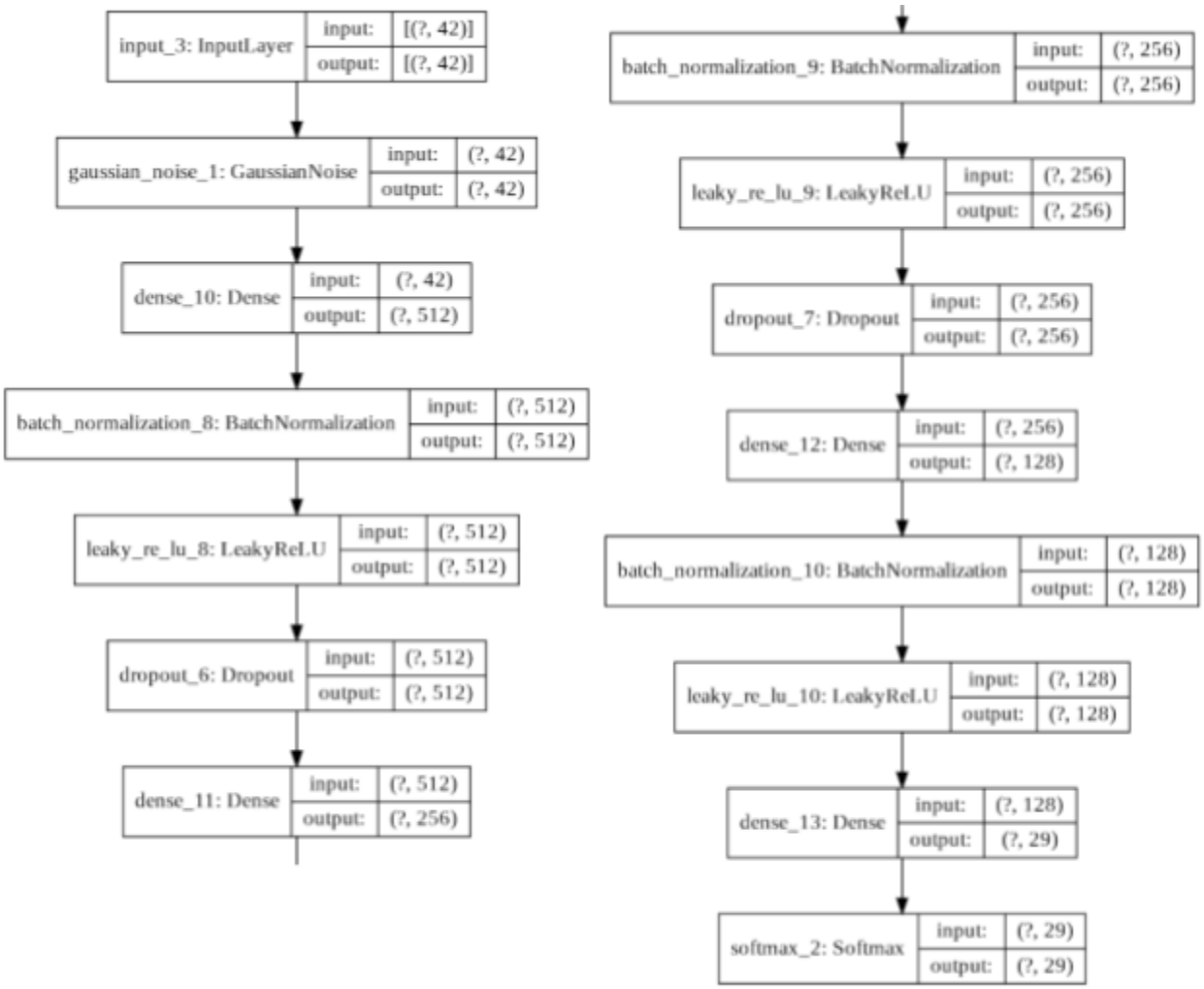
ארכיטקטורות State-of-the-Art רבות משתמשות בו, כמו ResNet, ResNext, Xception ועוד. הוא שימושי בעיקר ברשתות מאוד עמוקות, ומטרתו העיקרית היא למנוע מצב בו חלקים מהרשת הם למעשה פונקצית זהות ולכן לא לומדים כלום, וגם למנוע Vanishing Gradients. אנחנו עשינו בו שימוש בצורת ה FC שלו.



ארכיטקטורת המודל הגבוה. באדום: ה-Block הסטנדרטי בו השתמשנו בכל המודל.
החלק השמאלי למעשה מכיל את שני המסלולים, השמאלי הוא הרגיל והימני הוא Skip Connection עם שכבה על מנת להתאים גדלים.

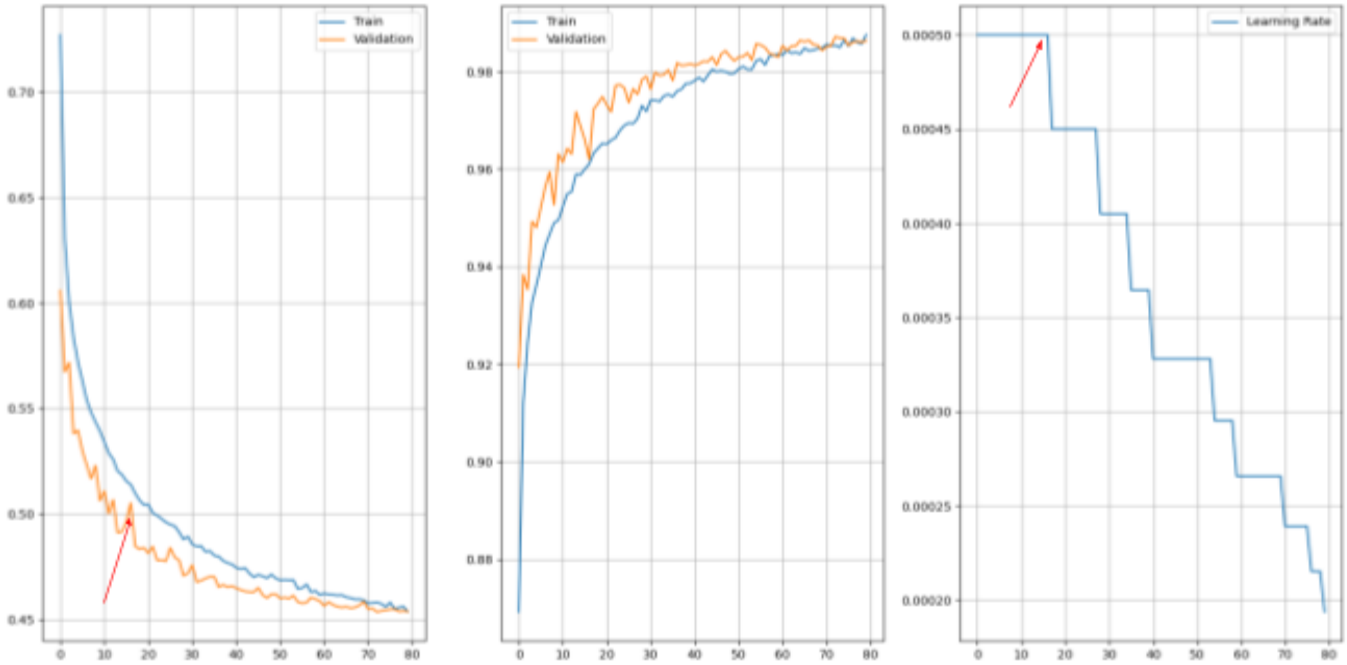
sub model: המודלים הקטנים מאוד דומים למודל הגבוה, אך הם רדודים וצרים יותר. חשוב לציין כי כאשר ניסינו להשתמש באותה ארכיטקטורה למודל הקטן והגדול, ראינו כי המודלים הקטנים נוטים ל-Overfitting קיצוני, כי המודל גדול מדיי וחזק מדיי עבור כמות קטנה יחסית של מידע בשונות נמוכה ממיילא, בייחוד עבור Clusterים עם כמות נמוכה של Classים.

במודלים הקטנים ויתרנו על שכבת ה-Skip Connection.

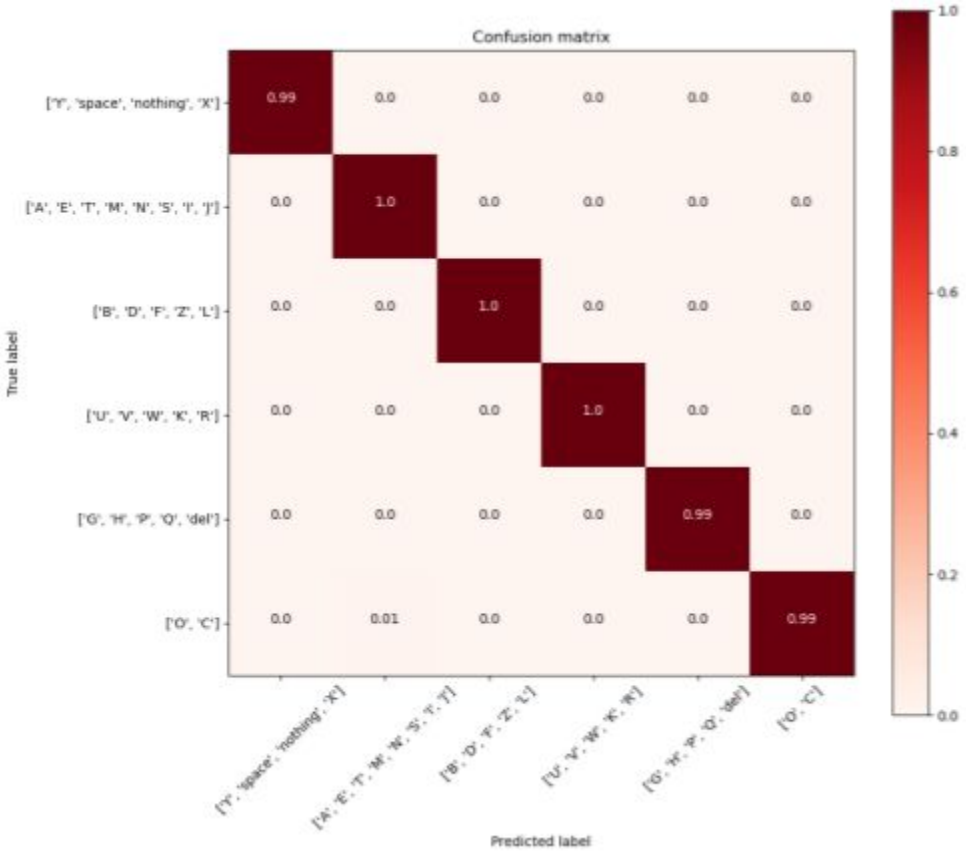


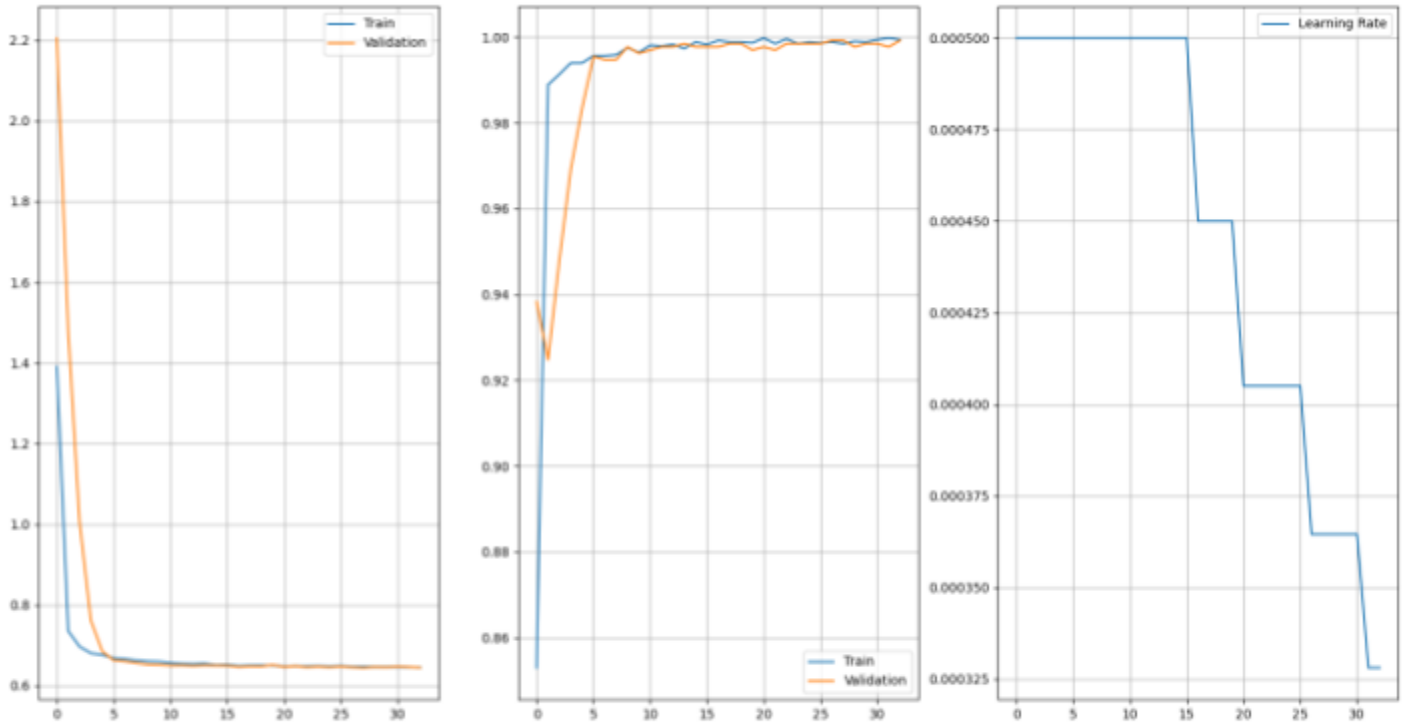
ארכיטקטורת המודלים הקטנים.

תוצאות

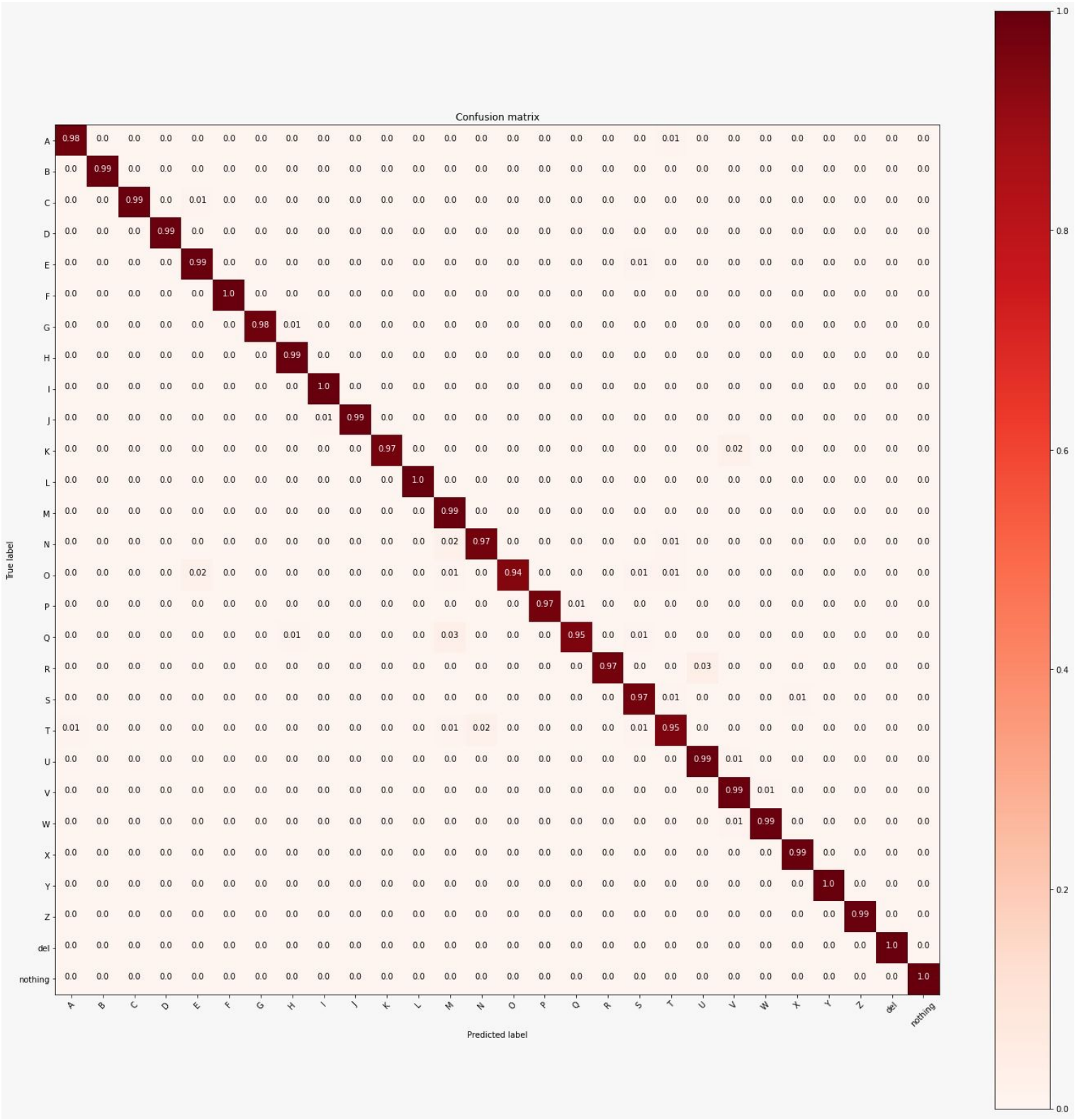


תוצאות המודל הגבוה (High Model).
משמאל לימין: גרף התקדמות ה Loss, גרף התקדמות ה Accuracy וגרף ה-Learning Rate.
ניתן לראות את ההשפעה של הורדת מקדם הלמידה (באדום), ה-Validation בכתום החל לזנק, ולאחר הורדת מקדם הלמידה, חזר למסלול הירידה.





אותם גרפים, על המודל sub1, כלומר המודל עבור ה-Cluster:
["A", "E", "T", "M", "N", "S", "I", "J"]


Confusion matrix
True label
Predicted label
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z del nothing
1.0
0.8
0.6
0.4
0.2
0.0

צד הלקוח

כפי שהוזכר, ישנו שימוש במודלים בצד הלקוח המוציאים לנו את ה- Keypoints של היד. בצד הלקוח, אנו יכולים לראות את ה-Feed המתקבל ב- RealTime מהמצלמה. כאשר המודלים מזהים את היד ואת ה-Keypoints אנו נראה על גבי הFeed את ה-Keypoints. לאחר מכן, צד הלקוח שולח את ה- Keypoints אל עבר מודל ה- Classification המוזכר לעיל, שנמצא בשרת.

בצד הלקוח התמודדנו עם 2 בעיות עיקריות:

1. איך לדעת אם הלקוח התכוון לשתי אותיות רצוף, לדוגמא AA. או שמא תנועתו התארכה במידה מסוימת אך עדיין התכוון למופע אחד של האות.
2. איך להתמודד עם תנועות ידיים מזבלות הנעשות תוך כדי מעבר בין תנועות ידיים מכוונות.

עם שתי הבעיות הנ"ל התמודדנו בצורה הבאה:

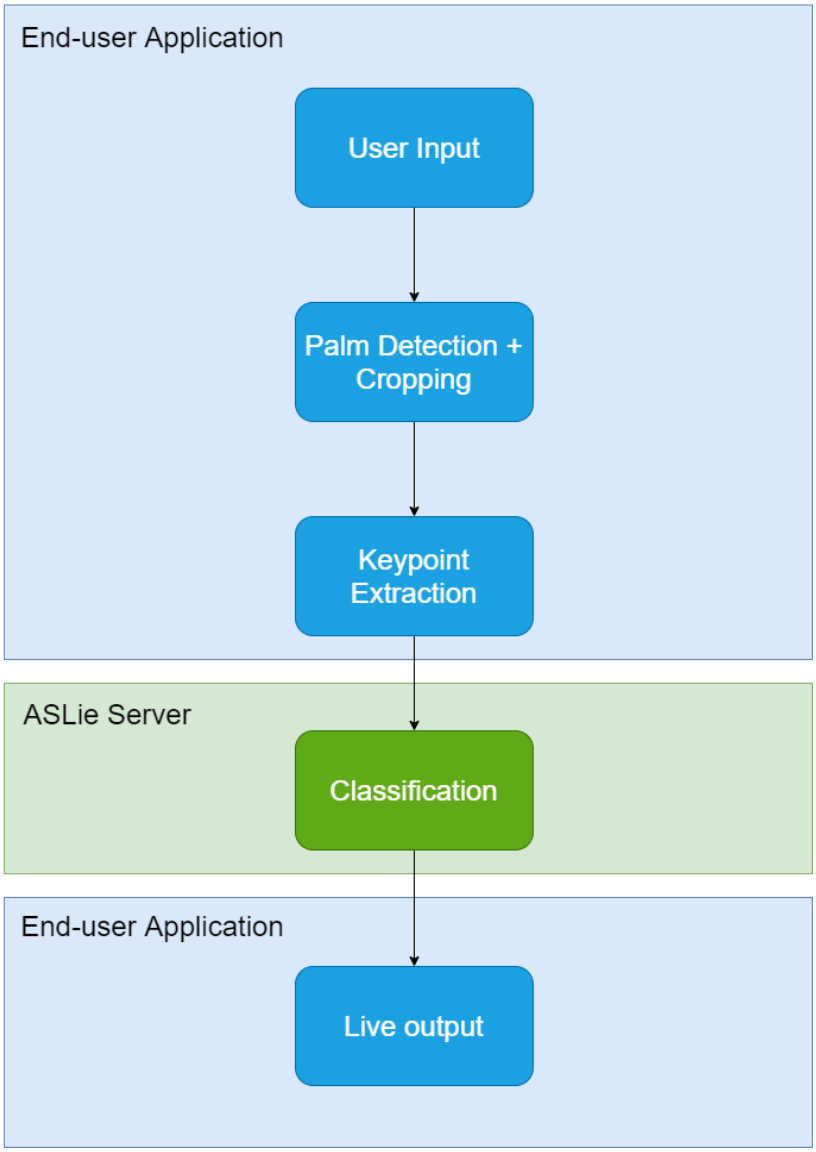
ניסינו בצורה נאיבית להגדיר Threshold המייצג את כמות הפעמים אשר חיזוי של אות אמור להתקבל ברצף. במידה והתקבל חיזוי אחר מקודמו, נאפס את המונה. פתרון זה פתר במידה מסוימת את הבעיות הנ"ל אך הציג בעיה חדשה. כעת נוצר שיהוי יחסית גדול בין עשיית תנועת היד להצגת האות. הבעיה נוצרת כיוון שמידי פעם ובתנאים מסוימים המודל החזיר אחרי רצף ארוך וזהה של אותו חיזוי, חיזוי שונה ובכך הלקוח היה צריך לחכות שוב עד אשר יגיעו החיזויים מחדש.

על מנת לפתור את הבעיות, החלטנו ליצור מנגנון חכם יותר. המנגנון החכם מתבסס על הכרעת רוב, בנוסף המנגנון מתייחס רק לפרדיקציות מעל ערך סף מסוים.

כעת, כאשר חוזרת מהשרת פרדיקציה אנו בודקים בכמה המודל היה בטוח בכך, במידה והוא בטוח מעל ערך סף מסוים נכניס את הפרדיקציה לתחשיב. בצורה זאת אנו מצליחים להתמודד עם בעיה 2 בצורה טובה.

הכרעת הרוב בודקת איזו פרדיקציה חזרה הכי הרבה פעמים בתוך כמות מסויימת של פריימים וכמה פעמים הופיעה, לאו דווקא ברצף. באמצעות הכרעת הרוב והתחשיב אנו מצליחים לפתור את הבעיה הראשונה.

תרשים מודולים כללי של המערכת



כלי הפיתוח, תוכנה וחומרה

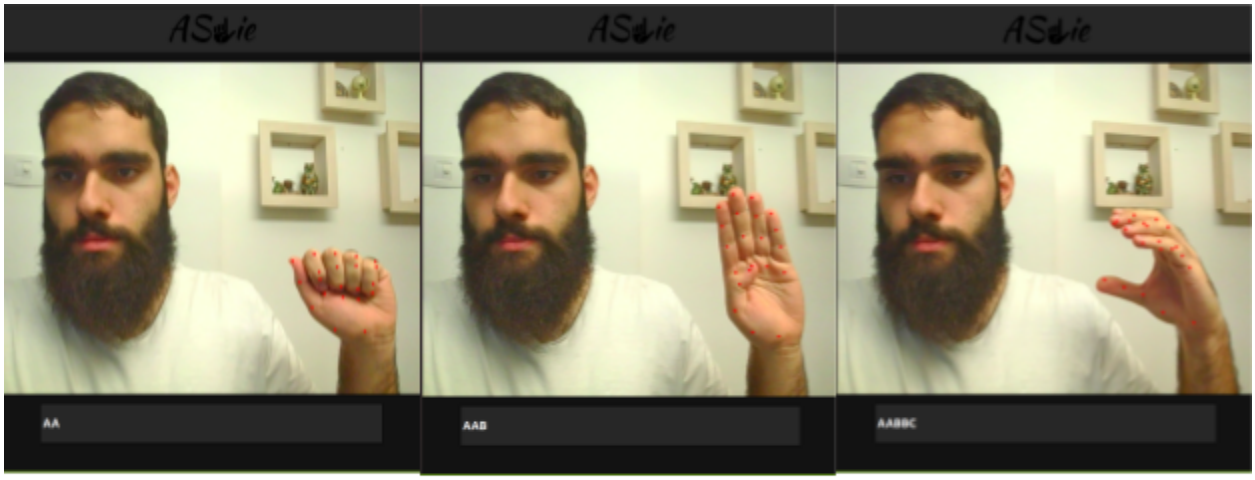
כלי הפיתוח בהם עשינו שימוש: Pycharm, Visual Studio Code, Kaggle.

השרת כתוב ב Python ומשתמש ב Flask כדי להנגיש את ה Endpoint של המודל ב HTTP.

הקליינט נכתב ב Javascript ומשתמש ב React.

מבחינת חומרה, חלק מהאימון הורץ לוקאלית על המחשבים האישיים שלנו (RTX 2070 / GTX 1060) וחלק רץ ע"ג הפלטפורמה של Kaggle בענן המאפשרת אימון רשתות קל נוח ומהיר ישירות מהדפדפן.

תוצר סופי



דוגמא להרצת האותיות ABC על המודל

קישורים לקוד

https://www.kaggle.com/tomeryacov/kernel602456cf2d

https://www.kaggle.com/tomeryacov/asl-class

https://github.com/TomerYacov/ASLie