הפקולטה להנדסה

המעבדה לעיבוד אותות

הפרדת דוברים רב ערוצית ע"י רשתות נוירונים מבוססת TasNet

אברהם גולדפישר

הילה יחזקאל

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחה: רננה אופוצ'ינסקי

מנחים אקדמיים: ד"ר שרון גנות

אוקטובר 2021

מטרת הפרויקט:

פיתוח ומימוש שיטת הפרדת דוברים רב ערוצית תוך שימוש ברשתות נוירונים על סמך TasNet. יושם דגש על שיפור השיטות הקיימות בכלים מרחביים, מעצבי אלומה (beamformer) המנצלים את החתימה האקוסטית של המערכת (החדר) ושיפור הביצועים במצבים של הדהוד ורעש.

תקציר:

בשנים האחרונות, תחום הלמידה העמוקה התפתח והתקדם מאוד, ובפרט הושגו תוצאות מרשימות בתחום הפרדת הדוברים על ידי רשתות נוירונים.

מחקרים עדכניים מראים שיפור משמעותי בביצועים של משימות שונות בתחום הדיבור, כמו למשל: הפרדה/ זיהוי אוטומטי של דיבור וכו'.. באמצעות שימוש במעצבי אלומה נלמדים על סמך ה-data (בהינתן מערך מיקרופונים).

במקביל, התפתחו גישות הפרדת דוברים על ידי רשתות נוירונים שמבצעות את ההפרדה במישור הזמן (TasNet) באופו ישיר.

המוטיבציה להפרדה במישור הזמן בלבד נבעה מהבעיות הבאות:

- ייצוג של האות המעורבב במישור הזמן-תדר אינו בהכרח אופטימלי לצורך הפרדת דוברים.
- פירוק של האות למישור הזמן-תדר גורר בעיות טבועות, כגון: התייחסות למגניטודה ולפאזה בנפרד/ חלונות זמן ארוכים וכו'..

כתוצאה מהבעיות הנזכרות למעלה, מפתחי TasNet הציעו את הרעיון הבא. במקום לעבור למישור הזמן-תדר, אפשר יהיה למדל את האות באופן ישיר במישור הזמן, תוך שימוש במסגרת של מקודד-מפענח, ולבצע את הפרדת המקור על המוצאים האי שליליים של המקודד.

היתרונות בשיטת מידול זו:

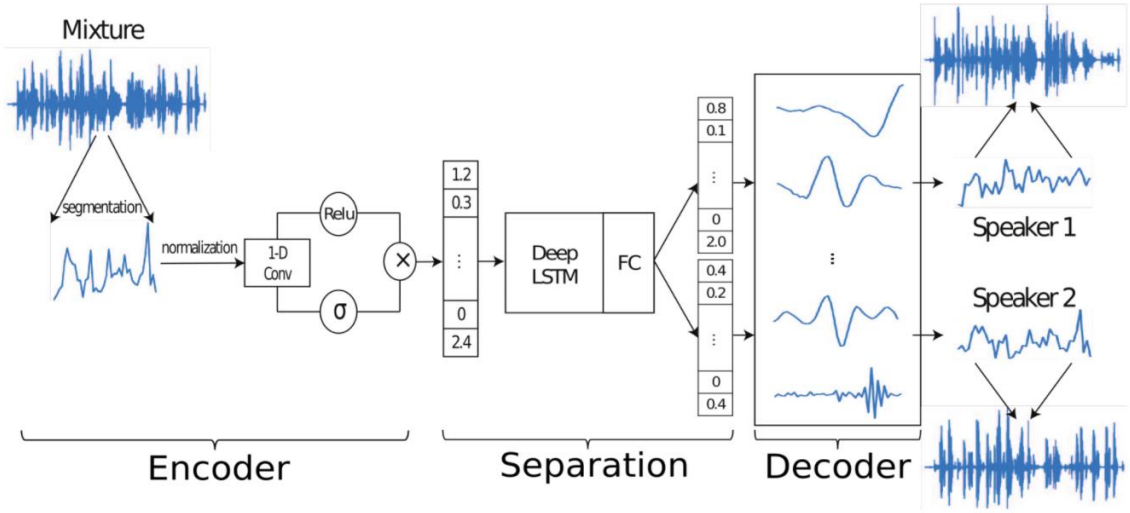
- הפחתה של העלות החישובית עבור ביצוע הפרדת דוברים.
- הפחתה של זמן ההשהייה (latency) עד לקבלת הפרדת האות במוצא.

ללמידה עמוקה הייתה השפעה גדולה על ההתמודדות עם הבעיה של הפרדת דוברים בייחס לשיטות קודמות. תחילה, האלגוריתמים השונים בצעו את ההפרדה תוך חישוב התמרת ה-STFT אשר העבירה את הבעיה למישור הזמן-תדר. העניין הוא שלא ברור האם התמרת פורייה היא אופטימלית לצורך פתרון הבעיה. כמו כן, התמרת STFT מעבירה את האות למישור קומפלקסי, דבר שיוביל אותנו להתעסקות עם המגניטודה והפאזה של האות. לכן, המטרה העיקרית תהיה להתמקד במגניטודה של ה-STFT , על ידי חישוב של ה"המסכה" במישור הזמן-תדר לכל מקור, ולבצע סינטזה תוך שימוש בספקטרום של המגניטודה עם הפאזה המקורית של האות המעורבב. כתוצאה מכך נוכל לקבל חסם העליון עבור ביצועי ההפרדה.

רשת נוירונים יוצרת לנו מודל במישור הזמן, המשתמשת במקודד-מפענח עבור הגל נושא של האות המעורבב, ומבצעת את ההפרדה במוצא המקודד. הגל נושא מתואר על סכום של משקלים אי שליליים (מוצא המקודד) של N אותות בסיס (הפילטרים של המפענח). ההפרדה נעשית על ידי חישוב המשקלים שמתאימים לכל מקור. שחזור המקור נעשה על ידי למידת המפענח.

המשקלים והסיגנלים של הבסיס נלמדים במסגרת המקודד האי שלילי, כאשר אצלנו: במקום להשתמש בייצוגים של התמרת STFT, אנו משתמשים במשקלים של האות המעורבב.

כדי להבין טוב יותר איך המערכת עובדת, נציג תרשים שמראה את השלבים השונים בביצועי ההפרדה:



בתמונה זו ניתן לראות את תהליך ההפרדה עבור 2 דוברים בעזרת TasNet.

התועלת שאנו מקבלים עבור TasNet:

- המערכת יכולה להתמודד בזמן אמת ולבצע השהיות מאוד קטנות.
- ביצועי TasNet עולים על ביצועי STFT.
- הפרדה לא סיבתית יכולה להיות יעילה ולשפר את הביצועים, כאשר אנו משתמשים במידע מהאות השלם.

בחזרה אלינו, בפרויקט זה נרצה לשלב בין שני העולמות ולפתח שיטה להפרדת דוברים רב ערוצית, תוך שימוש בכלים של רשתות נוירונים ותוך שימת דגש על החתימה האקוסטית במערכת עם הדהוד ורעש. נציין, שבניגוד לניסוי שבוצע ב-TasNet, שם השתמשו ב-2 דוברים עבור מיקרופון בודד, אצלנו יהיה יותר ממיקרופון אחד. דובר אחד שמדבר לתוך מספר מיקרופונים, מושפע מההדהוד בחדר/ מרחק מהמיקרופון/ זמן ההדהוד וכדומה, ואנו נשתמש בפרמטרים אלו כדי לפתח הרשת שלנו לצורך הפרדת דוברים.

רקע תיאורטי:

1. הקדמה- למידה עמוקה:

למידה העמוקה הוא תחום מחקר בעולם המחשבים וספציפית בתחום "למידת המכונה" שמניח שהמחשב יכול ללמוד וללמד את עצמו.

למידה עמוקה מבוססת על רשת נוירונים אשר מטרת היא ליצור חיקוי ממוחשב של פעולת המוח האנושי.

למערכות למידה עמוקה יש את היכולת שלהן ללמוד ולהשתפר כל הזמן ודי בעצמן. מפתחי מערכות כאלה בונים בשבילה מעין "רשת סמנטית". זו מערכת שמחקה את המערכת הנוירונים שבמוח האנושי וכאמור פועלת ולומדת כמוהו - ככל שהיא פועלת וככל שמשתמשים בה - היא משתפרת ו"יודעת" יותר.

מכיוון שתחום הלמידה עמוקה הוא תחום רחב שניתן לכתוב רק עליו את כל הפרוייקט ניסינו להתמקד בהקדמה בהסבר על פרמטרים ושיטות שראינו שהיה לנו שימוש בהם ברשת שלנו כדי שההקדמה תיהיה כמה שיותר רלוונטית.

1.1 <u>רגולריזציה:</u>

הנוסחה לרגולריזציה הינה

$$L_\lambda(w) = \sum_t (w^\top x_t - y_t)^2 + \lambda \|w\|^2$$

הרגולריזציה הופכת את המטריצה ליציבה יותר ועוזרת למנוע בעיה של overfitting.
ישנם שני סוגי רגוליזציות 1L ו 2L כאשר הכי נפוץ להשתמש ב 2L מכיוון ש 1L מאפסת הרבה פרמטרים ולכן לרוב גורמת לנו לקבל מודל "דליל".

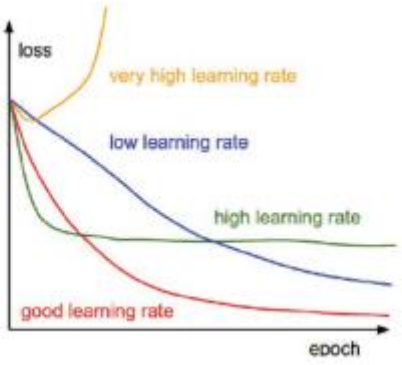
1.2 <u>קצב הלמידה:</u>

Gradient descent:

$$w_{s+1} \leftarrow w_s - \epsilon \frac{dL}{dw}(w_s)$$

ϵ - learning rate

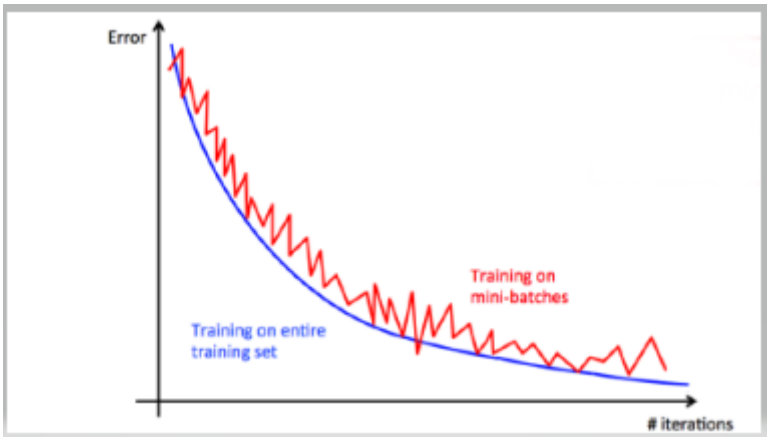
קצב הלמידה הינו פרמטר אשר מוכפל בנגזרת של פונקציית ההפסד בכל איטרציה של GD.



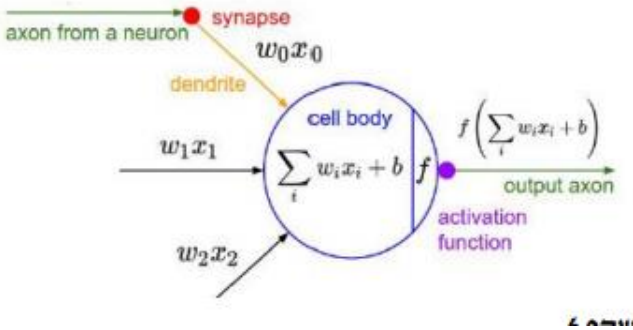
נרצה למצוא קצב למידה "מאוזן" כאשר קצב למידה קטן מדי – יגרום לכך שנרוץ על הרבה מאוד איטרציות עד ההתכנסותואפילו שנוכל להתקע באותו ערך לעומת זאת קצב למידה מהיר יכול לגרום לנו לקפיצה גדולה מדי שתגרום לנו לפספס את המינימום.

1.3 Batch

ניתן לחלק את המידע שלנו ל Batch כלומר לקבוצות וכך בכל איטרציה ללמוד רק חלק מהמידע, גורם לכל איטרציה להיות מהירה יותר אך להתכנסות של המודל להיות רועשת יותר.

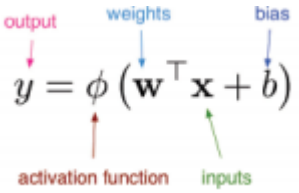


1.4 פונקציות אקטיבציה:



תיאור מתמטי של נוירון- נוירון במוח מקבל קלט מכל מיני תאי עצב סמוכים, מכפיל את המידע במשקולות מכל תא עצב ועליו עושה פונקציה לא לינארית על מנת להוציא פלט. נרצה בצורה רשת שתעבוד על אותו העיקרון. הפונקציה הלא לינארית שאנו מפעילים על המישקול של תאי העצב נקראת פונקציית אקטיבציה.

$$x \rightarrow z = W^T x + b \rightarrow h = \phi(z)$$



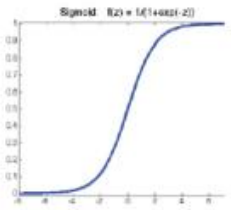
<u>דוגמאות בולטות לפונקי אקטיבציה:</u>

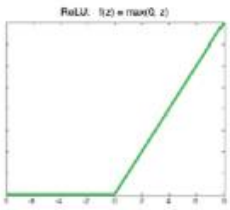
sigmoid: $\varphi(z) = \frac{1}{1+\exp(-z)}$

$\varphi(z)' = \varphi(z)(1-\varphi(z))$

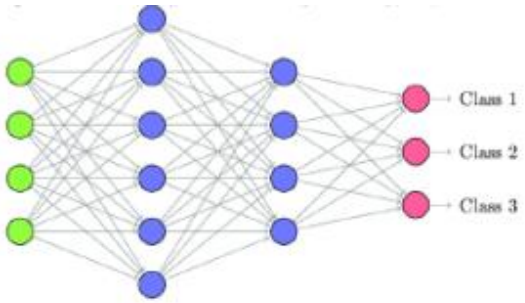
$ReLU(z) = max(0, z)$

$ReLU(z)' = 1_{\{z>0\}}$





1.5 <u>רשת נוירונים מרובת שכבות:</u>



1.6 <u>נירמול:</u>

נרמול של נתוני הכניסה:

$$x_{ti} \leftarrow \frac{x_{ti} - \mu_i}{\sigma_i}$$

ננרמל כך שיהיה תוחלת אפס ושונות 1, יכול לשפר את היציבות וההתכנסות של האלגוריתם.

נרמול batch:

נרצה שהכניסה לכל נוירון וכל שכבה תיהיה עם תוחלת 0 ושונות 1 ולא רק בשכבה הראשונה ולכן יהיה עלינו לבצע נורמליזציה לכל batch ולכל שכבה.

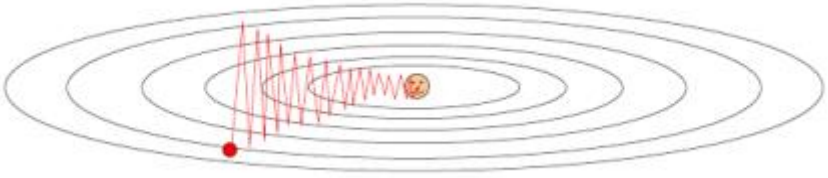
בכל batch אנחנו ננרמל על פי הדאטא שנכנס לאותו ה batch ביחס לנירמול של הדאטא בכניסה ש"התחשב" בכל הדאטא שי לנו.

נירמול ה Batch מאפשר נירמול של השכבות הפנימיות לעומת הנירמול הרגיל שמיועד רק לשכבה הראשונה של הכניסה.

יישנם מקרים שבהם נירמול יוכל לעזור לנו במידה ונתקלנו בגרדיאנט "מתפוצץ".

1.7 מומנט:

עוזר להתכנסות של הגרדיאנט, גורם לכך שלא "נלך" רק לכיוון שבו הגרדיאנט הכי משתנה אלא גם בכיוון שהיו הכמה שינויים האחרונים גורם לגרדיאנט "לנוע" עם יותר רעש ולכן יגרום לכך שלא "נתקע" במקום.



1.8 אלגוריתם ADAM-adaptive moment estimation:

אלגוריתם זה הוא שילוב של מומנט וRMSProp והינו האלגוריתם האדפטיבי המקובל כיום מחושב על ידי:

$$s_j \leftarrow \beta_1 s_j + (1-\beta_1)\left(\frac{\partial L}{\partial w_j}\right)^2$$

$$m_j \leftarrow \beta_2 m_j + (1-\beta_2)\left(\frac{\partial L}{\partial w_j}\right)$$

$$\hat{s}_j = \frac{s_j}{1-\beta_1^t}$$

$$\hat{m}_j = \frac{m_j}{1-\beta_2^t}$$

$$w_j \leftarrow w_j - \frac{\alpha}{\sqrt{\hat{s}_j + \epsilon}}\hat{m}_j$$

כאשר:

Sj - ממוצע נע של הגרדיאנט בריבוע.

Mj – ממוצע נע של הגרדיאנט.

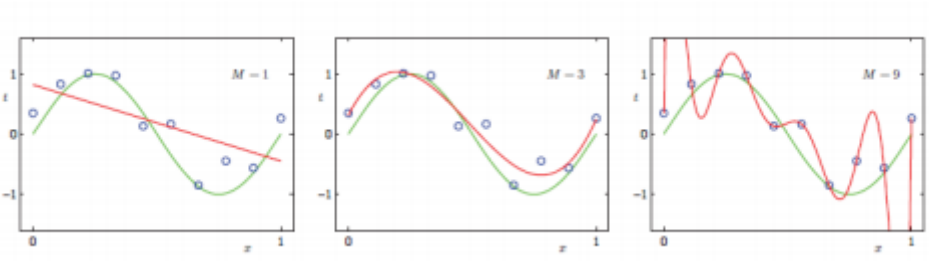
$\hat{S}j$ – עושה נירמול של Sj.

$\widehat{Mj}$ – עושה פעולת מומנטום.

Wj- השינוי במשקולת הרצויה.

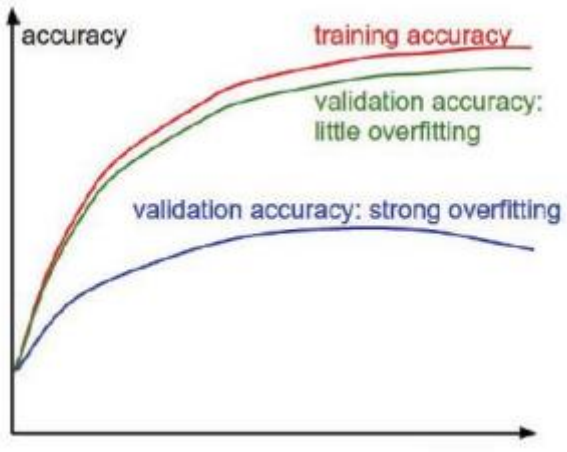
1.9 Underfitting and overfitting:

מצד אחד אם נעשה קירוב לפונקציה ע"י פולינום מסדר נמוך מדי (גרף שמאל) נקבל דיוק ממש גרוע וחוסר התאמה לפונקציה האמיתית, מצב זה נקרא underfitting. מצד שני אם נעשה קירוב לפונקציה ע"י פולינום מסדר גבוה מדי (גרף אמצעי) נקבל כי הקירוב שלנו מאוד דומה למקור ויש חשש שתיווצר התאמה טובה מדי לדאטא, שתגרום לכך שכאשר יגיע דאטא שונה המודל לא יידע איך להתמודד איתו, מצב זה נקרא -overfitting. המצב הרצוי שלנו הוא מצב ביניים (הגרף הימני), שהמודל כן ייתאים את עצמו לדאטא אבל תשמר מידת חוסר התאמה מסויימת כדי שהמודל יוכל להתמודד עם דאטא חדש.



נחלק את הדאטא שיש לנו ל-3 תתי קבוצות:

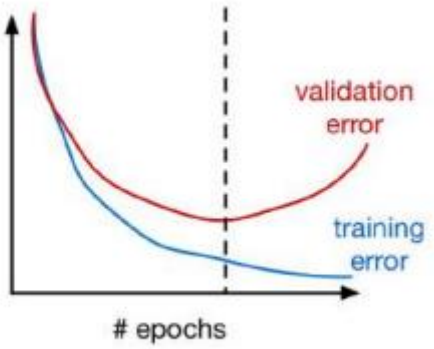
א. Train data- לשם למידת פרמטרי המודל.
ב. Validation data- לא משמש ללמידה, דאטא שמשמש לבקרה, בדיקה האם יש לנו overfitting ובדיקה של הרגולציה ובחירת ההיפרפרמטרים.
ג. Test data- משמש על מנת לבחון את המודל. נשתמש בדאטא חדש שלא שומש בtrain ונבחן מה מידת ההצלחה של פרמטרי הלמידה על הדאטא החדש.



ניתן לזהות בעיית overfitting ע"י כך שדיוק הולידציה יהיה משמעותית נמוך יותר מהדיוק של האימון, זה אומר שהתאמנו את עצמנו מאוד לדאטא של האימון.

1.1 Early stopping :

עצירה מוקדמת באה "טפל" במקרה של overfitting, בשיטה זו נבדוק בזמן הלמידה מתי אנו רואים כי השגיאה של הולידציה מתחילה לעלות ושם נעצור את הלמידה שלנו.



1.2 Vanishing or exploding gradients:

כאשר קיימת לנו הפונקציה:

$$h_t = f(z_t) = f(w_{hh}h_{t-1} + w_{hx}x_t + b_h),$$

נחשב את הנגזרת:

$$\frac{\partial h_n}{\partial x_1} = \frac{\partial h_n}{\partial h_{n-1}} \times \frac{\partial h_{n-1}}{\partial h_{n-2}} \times \cdots \times \frac{\partial h_2}{\partial h_1} \times \frac{\partial h_1}{\partial x_1}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = f'(z_t) w_{hh}$$

במקרה הפשוט whh, זה מספר קבוע עבור כל שלב בנגזרות, ואז בכל נגזרת בשרשרת נכפול בו. לפיכך, אם הוא מספר גדול או קטן מאחד זה מאוד משמעותי במידה והרשת ארוכה מאוד.

אם whh<1 אזי אנו כופלים רבות במספר קטן מאחד ולכן הנגזרת תיהיה מספר ששואף לאפס.

אם whh>1 אזי אנו כופלים פעמים רבות במספר גדול מאחד ולכן הנגזרת תיהיה מספר מתפוצץ.

מדוע גרדיאנטים מתפוצצים או נעלמים?

הדבר קורה בשלב הBP. הוא לא קורה בשלב הForward כיוון שאנו מפעילים פונקציה לא לינארית. במקרה הכללי שבו whx וwhh הן מטריצות:

$$\frac{\partial h_n}{\partial h_1} = \frac{\partial h_n}{\partial h_{n-1}} \times \frac{\partial h_{n-1}}{\partial h_{n-2}} \times \cdots \times \frac{\partial h_2}{\partial h_1}$$

כיצד נטפל במקרה של גרדיאנט מתפוצץ?

אם הנורמה של הגרדיאנט גדולה ננרמל אותו, השיטה נקראת clipping .

תבכל פעם שאנחנו מחשבים נגזרת של x לפי y אז נדאג שהיא תיהיה מנורמלת, ולא תגדל לגדלים גדולים. בכך שכאשר נחשב את בגרדיאנט הכולל – המכפלה של שרשרת הנגזרות – לא נקבל מספר עצום שייגרום לoverflow:

$$if \quad \|g\| > c \quad then \quad g \leftarrow \frac{cg}{\|g\|}$$

בכל רגע שאנו מחשבים גרדיאנט, אנו לא "מרשים" שהוא ייגדל מעל רף מסויים, ובכך בסך הכל בשרשרת הוא לא מתפוצץ.

1.3 LSTM- long short term memory:

רשת עם ארכיטקטורה מסוג RNN, בהשוואה לרשתות רגילות שמשתמשות בfeeaforwoard ל LSTM יש גם חיבורי feedback. כלומר ניתן להכניס אליהם יותר מנקודת אינפוט אחת (כמו תמונה) ולהכניס רצף מסויים (הקלטה או שמע). מורכבת מתאי זיכרון שבהם היא שולטת מתי לאגור או לשכוח מידע.

נבצע ברשת 3 שלבים:

1. נחליט איזה מידע חשוב ואיזה לא חשוב.
2. נחליט באיזו עוצמה אנו רוצים לעדכן את רכיב הזיכרון על ידי הקלט הנוכחי.
3. נחליט ברכיב הזיכרון מה חשוב כרגע לקבלת החלטה.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

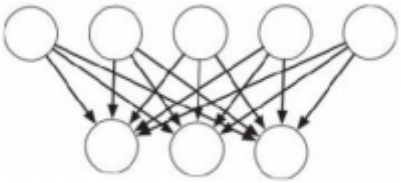
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

-

1.4 Fully-connected layer:

שכבה שבה כל נוירון מחובר לכל הנוירונים בשכבה הבאה.



החלק המעשי:

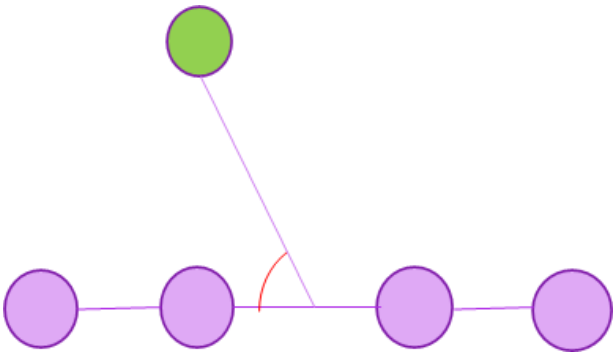
בחלק זה עשינו בנינו דאטא סט חדש רב ערוצי שהיה גם יותר "מתוחכם" מהדאטאסט החד ערוצי מכיוון שהגרלנו בו עוד פרמטרים כמו זווית הדוברים וגודל החדר (מפורט בהמשך), הרצנו רשת Tasnet על דאטאסט חד ערוצי (wsj0) ואז הטעמנו את הדאטאסט שלנו למבנה הרשת וניסינו לשנות פרמטרים ולראות כיצד זה משפיע על התוצאות.

1. בניית דאטאסט חדש רב ערוצי בעזרת מאגר timit

על מנת לייצר את הדאטאסט השתמשנו בקבצי שמע מתוך מאגר timit, אשר מכיל אנשים עם מבטאים שונים אומרים משפטים שונים. נשים לב כי קבצי השמע נקיים וללא רעש. בחרנו מספר קבצים על מנת ליצור מאגר חדש שיתאים לנו, כאשר גיוונו מבחינת המשפטים הנאמרים ומבחינת מין הדובר (נקבה/זכר), מכיוון שמדובר בטווח תדרים שונים עבור זכר או נקבה (9 נשים ו9 גברים). פרט חשוב! לאחר כמה הרצות ראשונות שמנו לב כי הדאטאסט שלנו אינו מספיק מגוון, וישנה חזרתיות בדוברים בשל הדרישה למספר הקלטות רב, ולכן עשינו שינוי ולקחנו מתוך מאשר של כ200 דוברים לכל אחד משלושת השלבים של הרשת- train, cross-val וtest (יפורט בחלק של ההרצה).

בחרנו באופן אקראי שני דוברים מתוך המאגר החדש שיצרנו והוספנו לכל אחד מהם רעש בעל SNR קבוע. לאחר מכן בחרנו את גודל החדר משלושה גדלי חדרים אפשריים בצורה אקראית. בהתאם לגודל החדר שנבחר ניתן יהיה לבחור בשני סידורים לינארים של 4 מיקרופונים, אחד אנכי ואחד אלכסוני. לאחר מכן נבחר את מיקום הדוברים בצורה שיעמדו בהגבלות הבאות:

1. רדיוס זהה ממרכז מערך המיקרופונים
2. מרחק של 0.5 מטר מהקיר (על מנת למנוע הידהוד משמעותי)
3. זווית אקראית בין הדובר למרכז מערך המיקרופונים שנעה בין 0-180 מעלות.
4. הפרש הזוויות בין שני הדוברים לעומת מערך המיקרופונים צריכה להיות מעל 20 מעלות.

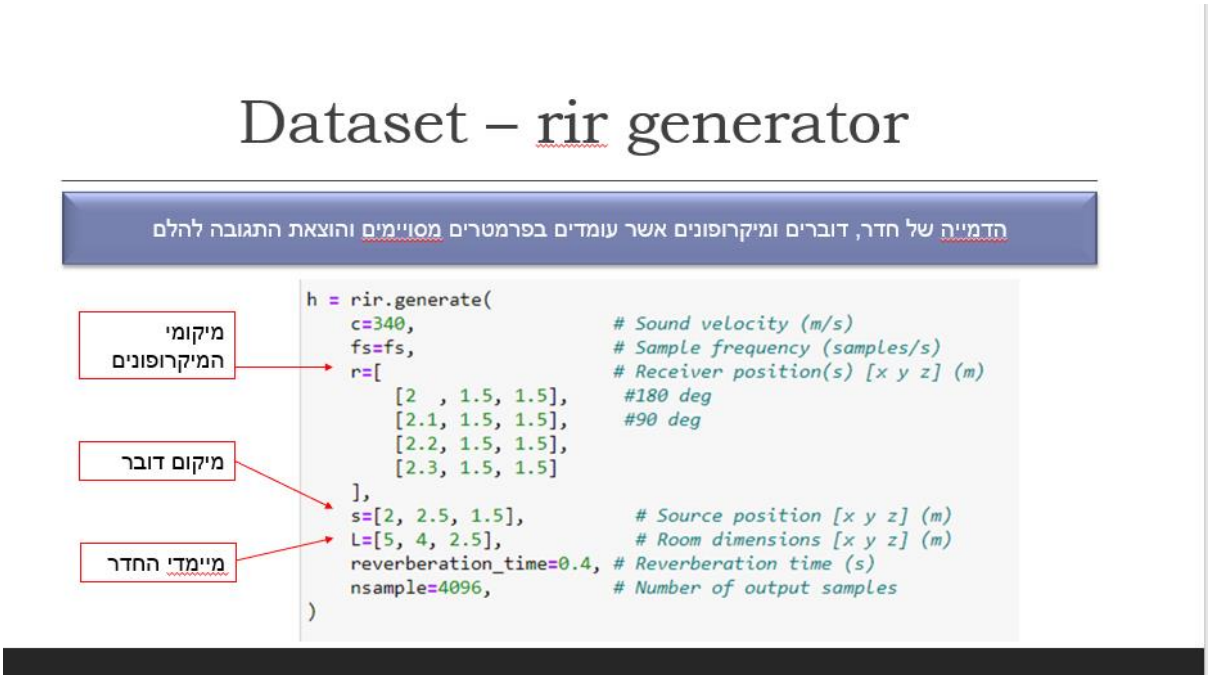


תרשים המתאר את הזווית בין מערך המיקופונים לדובר, סגול- מיקרופון, ירוק - דובר.

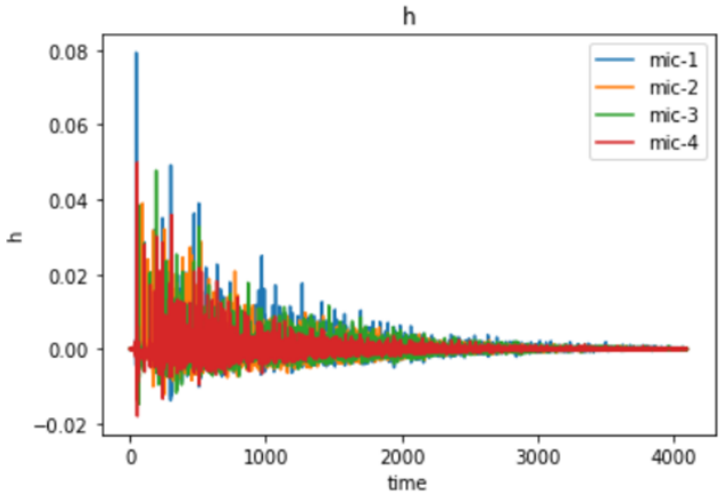
לאחר שקבענו את כל הפרמטרים נכניס אותם אל הrir . הrir הינו קוד פייטון שמייצר הדמייה של חדר (כאשר יש לנו אופציה לקבוע את הפרמטרים שלו- גודל, הידהוד וכו'). בנוסף ניתן למקם בחדר מיקרופונים ודוברים ולהוציא את התגובה

להלם של המיקרופונים עם הדובר בחדר. בעזרת קונבולוציה של התגובה להלם ואות הכניסה ניתן לייצר אות מוצא חדש אשר ייצג את מה שהמיקרופון דלנו קלט במידה וזה אות הכניסה.

בעזרת הrir נוכל לייצר דאטאסט "מלאכותי" של קבצי שמע של שני דוברים בחדר עם פרמטרים כרצוננו אשר נוכל להשתמש בו על מנת ללמד את הרשת שלנו.



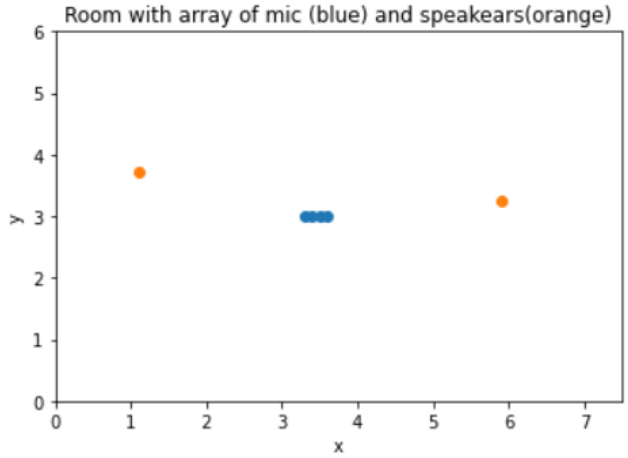
נשים לב כי עבור גודל חדר שונה, מיקרופון שונה, ופרמטר הדהוד שונה נקבל תגובה להלם שונה של החדר והמיקרופון ושל זנב ההתכנסות.



התגובה להלם שמקבלים עבור 4 מיקרופונים שונים. ניתן לראות כי התגובה מושפעת מהמיקום של המיקרופון ביחס לדובר, בנוסף ראינו כי קצב ההתכנסות ואורך ה"זנב" תלויים בפרמטר ההדהוד של החדר.

עבור כל בחירה של פרמטרים שונים (גודל חדר, מערך מיקרופונים אנכי/ עם זווית, מיקום דוברים) עשינו הדפסה של החדר המדמה את גודל החדר ובו מערך המיקרופונים (כחול) ומיקום שני הדוברים (כתום), בנסוף הדפסנו את כל הפרמטרים החשובים:

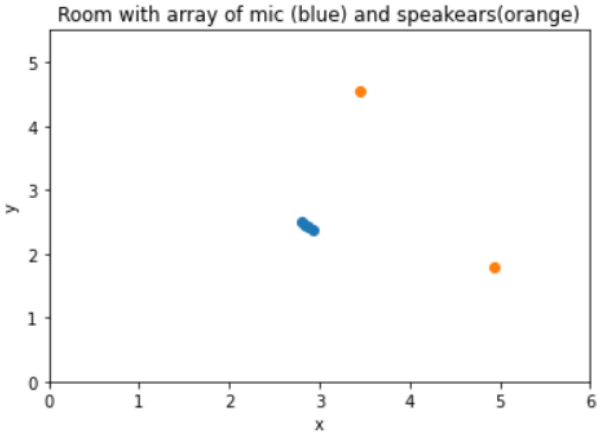
```
0
theta =  [163.17396432   6.1628283 ]
r =  2.45518265702277
speaker1 =  1.0999284750090714 3.710693820693696
speaker2 =  5.890993717793771 3.2635745606782507
mic_center [3.4499999999999997, 3.0, 1.5]
```



הדמייה של חדר עם מערך מיקרופונים אנכי ושני דוברים.

x

```
20
theta =  [342.67815148  74.42451537]
r =  2.1816031713762207
speaker1 =  4.943873957456596 1.78923949656543
speaker2 =  3.446989523789515 4.540277032197725
mic_center [2.8612123092720085, 2.4387876907279913, 1.5]
```



הדמייה של מערך מיקרופונים זוויתי ושני דוברים.

הערה: נציין, שהסיבה לכך שקיבלנו זווית של 342 מעלות, היא משום שמדובר בזווית בייחס לראשית הצירים ולא בייחס לדוברים. בייחס הדוברים אנו עדיין נמצאים בתוך הטווח של 0-180 מעלות, כפי שציינו לעייל בהגבלות שנתונות לנו.

לאחר מכן, מתוך התוצאה שקיבלנו מהrir נייצא 3 קבצי שמע, שני קבצים של כל אחד מהדוברים בחדר בנפרד וקובץ של איחוד של שני הדוברים. לכל אחד מהקבצים נייצא גם את הפרמטרים הבאים:

- מספר ההרצה ("מספר סידורי" על מנת לקשר את הקובץ של של שני הדוברים לשני הקבצים של הדוברים הבודדים)
- הזווית של כל דובר לבין מרכז המערך
- המיקום של כל דובר
- איזה דוברים נבחרו (שם ההקלטה - גבר.איש מספר)
- גודל החדר הנבחר

הערה: במקום להאריך את השם של הקבצים עבור פרמטרים אלו, העדפנו לייצא את הפרמטרים הללו לתוך קובץ אקסל, אשר נבנה מיד לאחר הרצת הקוד.

נשים לב כי הדאטא סט שקיבלנו הינו דו ערוצי, כאשר אנו משנים את הזווית בין דובר למיקרופון ניתן לשמוע (כאשר מרכיבים אוזניות) כי מזווית מסויימת נשמע את הדובר באוזניה הימנית ומזווית אחרת נשמע את הדובר באוזנייה השמאלית.

הקוד:

imports

In [60]:
```python
import numpy as np
import scipy.signal as ss
import soundfile as sf
import rir_generator as rir
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import math
import os
```

In [2]:
```python
# Adding noise using target SNR
def adding_noise_with_target_snr(audio_signal):
    # Set a target SNR
    target_snr_db = 20
    # Calculate signal power and convert to dB
    sig_avg_watts = np.mean(audio_signal)
    sig_avg_db = 10 * np.log10(abs(sig_avg_watts))
    # Calculate noise according to [2] then convert to watts
    noise_avg_db = sig_avg_db - target_snr_db
    noise_avg_watts = 10 ** (noise_avg_db / 10)
    # Generate an sample of white noise
    mean_noise = 0
    noise_volts = np.random.normal(mean_noise, np.sqrt(noise_avg_watts), 1)
    noise_watts = noise_volts ** 2
    out_signal = audio_signal + noise_volts

    return out_signal
```

In [3]:
```python
def rir_genrator(s, fs, r, L, reverberation_time):
    h = rir.generate(
        c=340,                  # Sound velocity (m/s)
        fs=fs,                  # Sample frequency (samples/s)
        r=r,              # Receiver position(s) [x y z] (m)
        s=s,
        L=L,            # Room dimensions [x y z] (m)
        reverberation_time=reverberation_time, # Reverberation time (s)
        nsample=4096,           # Number of output samples
    )

    return h
```

In [58]:
```
def random_speakers_locations(mic_center,room_dimentions,mic_angel):
    valid_locations = 0
    while not valid_locations:

        valid_teta=0
        r=np.random.uniform(1.5,2.5)
        if mic_angel ==0:
            while not valid_teta:
                print('0')
                teta=np.random.uniform(0,math.pi,2)
                valid_teta=check_teta(teta)

        else:
             while not valid_teta:
                print('20')
                teta=np.random.uniform(0.2*180/math.pi,math.pi+0.2*180/math.pi,2)
                valid_teta=check_teta(teta)


        print("theta = " ,teta*180/math.pi%360)
        #print("theta0 = " ,teta[0]*180/math.pi%360)
        print("r = " ,r)
        print("speaker1 = " ,r*math.cos(teta[0])+mic_center[0],r*math.sin(teta[0])+mic_center[1])
        print("speaker2 = " ,r*math.cos(teta[1])+mic_center[0],r*math.sin(teta[1])+mic_center[1])

        speaker_1=[r*math.cos(teta[0])+mic_center[0],r*math.sin(teta[0])+mic_center[1]]
        speaker_2=[r*math.cos(teta[1])+mic_center[0],r*math.sin(teta[1])+mic_center[1]]

        valid_locations = check_location(speaker_1 ,speaker_2, room_dimentions)

    return speaker_1,speaker_2,teta[0],teta[1]
```

In [5]:
```
def check_location(speaker_1 ,speaker_2, room_dimentions):
    if speaker_1[0]>0.5 and speaker_1[0]<room_dimentions[0]-0.5 and speaker_1[1]>0.5 and speaker_1[1]<room_dimentions[1]-0.5:
        if speaker_2[0]>0.5 and speaker_2[0]<room_dimentions[0]-0.5 and speaker_2[1]>0.5 and speaker_2[1]<room_dimentions[1]-0.5:
            return True
```

In [6]:
```
def check_teta(teta):
    if abs(teta[0]-teta[1])>20*math.pi/180:
        return True
```

```
def find_center(mic_array):
    x = np.mean([mic_array[0][0],mic_array[1][0],mic_array[2][0],mic_array[3][0]])
    y = np.mean([mic_array[0][1],mic_array[1][1],mic_array[2][1],mic_array[3][1]])
    z = np.mean([mic_array[0][2],mic_array[1][2],mic_array[2][2],mic_array[3][2]])
    center = [x,y,z]
    return center
```

```
def random_room(room_dimentions):
    index=np.random.randint(0,3)

    return [room_dimentions[index],index]
```

```
def mic_location(num_of_mic,index):
    if index == 0:
        x1 = 2.8
        y1 = 2.5
    if index == 1:
        x1 = 3.8
        y1 = 3.5
    if index == 2:
        x1 = 3.3
        y1 = 3
    mic_array=[[x1,y1,1.5]]

    index_of_location = np.random.randint(0,2)
    if index_of_location == 0:
        for i in range (num_of_mic):
            mic_array.append([x1+0.1,y1,1.5])
            x1 = x1+0.1
        mic_angel=0
    else:
        mic_angel =20
        for i in range(num_of_mic):
            mic_array.append([x1+0.2*math.cos(mic_angel),y1-0.2*math.cos(mic_angel),1.5])
            x1 = x1+0.1*math.cos(mic_angel)
            y1 = y1-0.1*math.cos(mic_angel)

    return mic_array, mic_angel
```

```
#enter audio

path = 'C:/Users/hilay/Desktop/project/ourtrain'

list_of_speakers = os.listdir(path)

# list_of_speakers= ["man1.wav","man2.wav","man3.wav","man4.wav","man5.wav","man6.wav","man7.wav","man8.wav","man9.wav","woman1.wav","woman2.wav","woman3.wav","woman
num_of_data_files=len(list_of_speakers)
num_of_output_files=10

# for file_name in files:
#     list_of_speakers = os.path.join("wav", file_name)

# print("files",files)
# print("files2",list_of_speakers)

print("num_of_files", num_of_data_files)


for num in range(num_of_output_files):
    choose_two_speakers_idx=np.random.randint(num_of_data_files-1,size=(2))
    choose_two_speakers= [list_of_speakers[choose_two_speakers_idx[0]],list_of_speakers[choose_two_speakers_idx[1]]]

    #read the files
    speaker1, fs = sf.read(choose_two_speakers[0], always_2d=True)
    speaker2, fs = sf.read(choose_two_speakers[1], always_2d=True)

    #add noise
    speaker1_noisy= adding_noise_with_target_snr(speaker1)
    speaker2_noisy= adding_noise_with_target_snr(speaker2)

    #room dim options
    options_of_room_dimentions = [[6, 5.5, 2.8],[8, 8, 2.8],[7.5,6, 2.8]]
    [room_dimentions,index] = random_room(options_of_room_dimentions)

    #mic_locations

    #mic_array=[[2.2, 2.5, 1.5], [2.3, 2.5, 1.5],  [2.5, 2.5, 1.5], [2.6,2.5, 1.5 ] ]
    num_of_mic = 4;
    mic_array, mic_angel = mic_location(num_of_mic,index)
    mic_center = find_center(mic_array)

    #get the speakers locations
    speaker_1_loc,speaker_2_loc,teta1,teta2=random_speakers_locations(mic_center,room_dimentions, mic_angel)
    print("mic_center", mic_center)
```

```python
#rir
s = [[speaker_1_loc[0],speaker_1_loc[1],1.5],[speaker_2_loc[0],speaker_2_loc[1],1.5]]

h1=rir_genrator(s[0], fs, mic_array, room_dimentions,0.4 )
h2=rir_genrator(s[1], fs, mic_array, room_dimentions,0.4 )

#plot room and mic
plt.scatter([mic_array[0][0],mic_array[1][0],mic_array[2][0],mic_array[3][0]],
            [mic_array[0][1],mic_array[1][1],mic_array[2][1],mic_array[3][1]])

plt.scatter([s[0][0],s[1][0]],[s[0][1],s[1][1]])
#plt.scatter(s[0][0],s[0][1])
#plt.scatter(s[1][0],s[1][1])

plt.plot(0,1, linestyle='solid');
plt.plot(mic_center[0],0, linestyle='solid')
plt.plot(mic_center[1],0, linestyle='solid')
plt.title("Room with array of mic (blue) and speakears(orange)")
plt.xlabel("x")
plt.ylabel("y")

plt.xlim(0,room_dimentions[0])
plt.ylim(0,room_dimentions[1])

plt.show()

# Convolve 2-channel signal with 3 impulse responses
signal1_room=(ss.convolve(h1[:, None, :],speaker1_noisy[:, :, None]))
signal2_room=(ss.convolve(h2[:, None, :],speaker2_noisy[:, :, None]))

#add two speakers and create a new file
#resize the length of to signalsto the same size - and add them
if signal1_room.shape[0]<signal2_room.shape[0]:
    two_speakers=signal2_room.copy()
    two_speakers[:signal1_room.shape[0]]+=signal1_room

else:
    two_speakers=signal1_room.copy()
    two_speakers[:signal2_room.shape[0]]+=signal2_room

sf.write(f'C:/Users/hilay/Desktop/project/dataset/train/two speakers/SAMP{num}__speaker_1_angel_{teta1}_audio_{choose_two_speakers[0]}__speaker_2_angel_{teta2}_audio_{choose_two_speakers[1]}

sf.write(f'C:/Users/hilay/Desktop/project/dataset/train/speaker1/SAMP{num}__angel_{teta1}_x_{speaker_1_loc[0]}_y_{speaker_1_loc[1]}_audio_{choose_two_speakers[0]}_room_{room_dimentions}.wav'

sf.write(f'C:/Users/hilay/Desktop/project/dataset/train/speaker2/SAMP{num}__angel_{teta2}_x_{speaker_2_loc[0]}_y_{speaker_2_loc[1]}_audio_{choose_two_speakers[1]}_room_{room_dimentions}.wav'
```

2. הרצת רשת Tasnet על דאטאסא חד ערוצי – wsj0

בשלב זה, חיפשנו בgitub רשת TasNet אשר מבצעת את הפרדת הדוברים, עבור המקרה שבו יש לנו 2 דוברים ומיקרופון בודד- כך שהיא תהיה דומה כמה שיותר לרשת עליה קראנו במאמר.

יש לציין כי זהו היה אמור להיות חלק טכני ולא ארוך במיוחד בפרוייקט שלנו, אך מאחר ונתקלנו בהרבה מאוד בעיות טכניות (שיפורטו בהמשך- כולל הפיתרונות שלנו) בהרצה (במספר רשתות שניסינו) חלק זה גזל מאיתנו זמן רב משציפינו.

2.1 מציאת רשת וביצוע כל ההתקנות:

- הרשת שאיתה עבדנו נמצאת בקישור הבא:

https://github.com/kaituoxu/TasNet

ניתן לראות ב-github המצורף, את כל השלבים שעלינו לבצע על מנת שנוכל להריץ את הרשת (כולל התקנות – שבהן נקלנו במספר תקלות טכניות שמפורטות בהמשך).

הרשת רצה על הדאטאסט של wsj0:

הדאטא סט של wsj0 בנוי מהקלטות של דוברים שונים אשר מקריאים משפטים בני 5000 עד 20000 מילים מהעיתון. מדובר בדוברים שונים אשר מקריאים בין 5000 ל20000 מילים. משך ההקלטות נע סביב ה-4 ועד 8 שניות.

תוך כדי הסתכלות בפונקציות השונות ובפרמטרים אשר מופיעים ב-git, הצלחנו למצוא דמיון וקשר בין רשת זו לבין הרשת שעליה קראנו במאמר באופן תיאורטי.

נציג מספר גורמים מקשרים:

- שימוש ב-dataset של wsj0. כלומר, הקלטות רנדומליות של אנשים שונים ב-Wall Street Journal.
- בחירת הפרמטרים: L נבחר להיות 40, N נבחר להיות 500, ישנן 4 שכבות של LSTM שמכילות 1000 hidden layers. ה-batch size נבחר להיות 128.
- בסוף האימון, יש הצגה של הביצועים של הרשת, על ידי חישוב השיפור של ה-SI-SNR ושל ה-SDRi.
- הבנייה של הפונקציות דומה מאוד להסבר התיאורטי של החלקים השונים של הרשת, כפי שמופיע במאמר עצמו.

<u>שלבי התקנה:</u>

1. פתיחת סביבת conda -

```
conda create --name TasNet_project

conda activate TasNet_project
```

2. התקנת פייטון 3.9 ואנקונדה.
3. התקנת החבילות- בעזרת הקובץ requirements.txt:

- librosa
- mir_eval
- visdom

לאחר שסיימנו את כל ההתקנות, נוכל להתחיל בתכנון הרצת הרשת.

הרצת הרשת:

רשת זו מכילה את חמשת השלבים הבאים:

- Stage 0: Convert sphere format to wav format and generate mixture
- Stage 1: Generating json files including wav path and duration
- Stage 2: Training
- Stage 3: Evaluate separation performance
- Stage 4: Separate speech using TasNet

כאשר, בשלב הראשון אני רוצים לבצע המרה של הקבצים שנמצאים ב-dataset של wsj0, כך שהם יהיו בפורמט של קבצי wav. נציין שאת החלק הזה בצענו באופן עצמאי על המחשבים האישיים שלנו מחוץ לשרתים, כפי שנפרט בהמשך. לאחר מכן, הרשת מייצרת קבצי json שמכילים את הנתיב של קבצי ה-wav. בשלב הבא הרשת מבצעת את האימון, אחרי זה מחשבת את הביצועים של ההפרדה. לבסוף, הרשת מבצעת הפרדת דוברים ל-test תוך שימוש ברשת TasNet.

לפני שנוכל להריץ את הרשת, היה עלינו לעדכן את הקובץ run.sh, שהוא הקובץ אשר מבצע את ההרצה של השלבים השונים של הרשת. קודם כל, היינו צריכים לשנות את ה-path:

```
# -- IMPORTANT
# Modify these path to your path
wsj0_origin=/home/ktxu/workspace/data/CSR-I-WSJ0-LDC93S6A
wsj0_wav=/home/ktxu/workspace/data/wsj0-wav/wsj0
# Directory of wsj0 including tr, cv and tt
data=/home/ktxu/workspace/data/wsj-mix/2speakers/wav8k/min/

stage=1  # Modify this to control to start from which stage
```

עדכנו את ה-path באופן הבא:

wsj0_origin=/home/dsi/goldfia1/TasNet/egs/wsj0/data

wsj0_wav=/home/dsi/goldfia1/TasNet/egs/wsj0/data/wsj0-wav/wsj0

data=/home/dsi/goldfia1/TasNet/egs/wsj0/data/wsj0-wav/wsj0/2speakers/wav8k/min/

בנוסף, ניתן לראות שיש אפשרות לשלוט על השלב ההתחלתי שממנו נוכל להריץ את הרשת. כאמור לעיל, כיוון שאת שלב 0 בצענו בעצמנו, אז יכולנו להתחיל בשלב 1.

פעולה נוספת שנדרשה מאיתנו הייתה להשיג את ה-dataset של wsj0 עליה קראנו במאמר. לצורך כך היינו צריכים לקבל גישה ל-drobox של המנחה של הפרוייקט, ומשם הורדנו למחשב האישי שלנו את כל ההקלטות. נשים לב שהקבצים ירדו בפורמט של wv1 / wv2, ולכן לא יכולנו להאזין לקבצי השמע שהורדנו.

נתחיל בשלב 0. כפי שמופיע באתר, ניתן לבצע את ההמרה של הקבצים לפורמט של wav בעצמנו. נציין את הסיבה מדוע לא בצענו את השלב הזה באמצעות הרשת. כאשר ניסינו להגדיר בקובץ run.sh שהרשת תתחיל החל מ-stage 0, אז קבלנו את השגיאה הבאה:

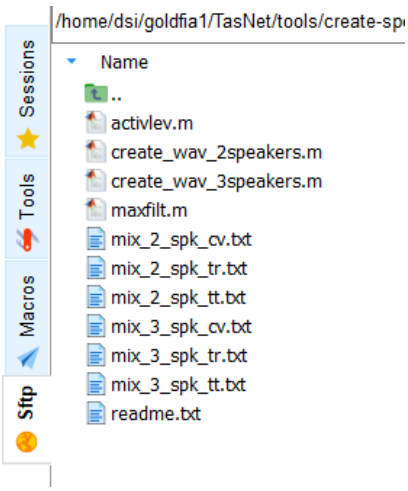
run.sh: line 66: local/data_prepare.sh: Permission denied

כלומר, הגישה לקבצים אלו הייתה חסומה לנו. לכן החלטנו, כפי שגם מופיע בקובץ github, לבצע את ההמרה באופן עצמאי.

לצורך ההמרה, השתמשנו בפקודה הבאה:

cd tools; make

לאחר פקודה זו, נבנתה התיקיה, שמכילה את הקבצים הבאים:



בקבצי txt ניתן לראות את הדוברים השונים עליהם נבצע את ה-mix, ובקבצי מטלב ניתן לבצע את הערבוב של הדוברים. נציין כי קבצי ה-txt יוצרים קומבינציה ספציפית של ערבוב הדוברים, אך מכיוון שמספר ההקלטות בהם הרשת מבצעת את האימון מספיק גדולה, אז זה פותר את הבעיה של גיוון ההקלטות השונות.

כמו כן, ניתן לראות שיש אפשרות לבצע ערבוב ל-2 דוברים וכן גם ל-3 דוברים. אנו נשתמש באופציה הראשונה של 2 דוברים, כפי שמופיע במאמר.

נכנסנו לקובץ הבא במטלב:

create_wav_2speakers.m

שם היה עלינו לעדכן את ה-path הרלוונטי לנו:

wsj0root = '/home/dsi/goldfia1/TasNet/egs/wsj0/data/wsj0-wav/'; % YOUR_PATH/, the folder containing wsj0/

output_dir16k='/home/dsi/goldfia1/TasNet/egs/wsj0/data/2speakers/wav16k';

output_dir8k='/home/dsi/goldfia1/TasNet/egs/wsj0/data/2speakers/wav8k';

לשם נייצא את קבצי ה-wav שנשתמש בהם בביצוע ההפרדת דוברים. הרצנו את הקובץ מטלב, ואז לאחר תהליך ממושך קבלנו את קבצי השמע בפורמט wav ויכולנו להאזין להקלטות השונות. בדומה ל-dataset שיצרנו, הוא מחלק את ההקלטות ל-3 חלקים:

- דובר א'- s1.
- דובר ב'- s2.
- הערבוב ביניהם- mix.

בנוסף, ניתן לראות שההקלטות נכנסות לתוך התיקיות עם השמות הבאים:

- אימון - tr, מכיל 20000 הקלטות.
- ולידציה - cv =cross validation, מכיל 5000 הקלטות.
- הקלטות מבחן - tt, מכיל 3000 הקלטות.

נסביר כעת את פעולת המטלב. כאמור לעיל, המטלב מקבל את ה-path עבור ההקלטות השונות של wsj0, מהם הוא ייצר את הדטאסט. כמו כן, הוא גם מקבל את קובץ txt אשר מכיל בתוכו את הקומבינציה של ערבוב הדוברים. לכן, הוא עובר שורה שורה בקובץ txt, לוקח את ההקלטות שמופיעות שם, ויוצר את המיקס. בנוסף לכל זה, המטלב מיצר את הערבוב על ידי חלוקה ל-4 תתי תיקיות:

- 8kwav- עבור תדר דגימה של 8000 הרץ.
- 16kwav- עבור תדר דגימה של 16000 הרץ.
- Min- מבין 2 ההקלטות שמערבב, לוקח את ההקלטה הקצרה יותר מבין השתיים.
- Max- מבין 2 ההקלטות שמערבב, לוקח את ההקלטה הארוכה יותר מבין השתיים.

נציין, כי על מנת שנוכל לבצע בדיקות מהירות לרשת ולא להריץ אותה כל פעם מחדש, מתוך אותם תיקיות יצרנו תיקיות שמכילות מספר מצומצם יותר של הקלטות. קראנו להם באותם שמות, על מנת שהרשת תוכל להשתמש בתיקיות אלו, באותו האופן בו תריץ את ההקלטות מהתיקיות המקוריות.

נשים לב, שייצאנו את קבצי השמע למחשב האישי שלנו, ולכן היה עלינו להעתיק את כל התיקיות הרלוונטיות לתוך linux, כדי שהרשת תוכל למצוא אותם ולהשתמש בהם בשלבים הבאים.

נחזור להרצת הרשת. בשלב הבא, אנו נרצה להריץ את החלקים השונים של הרשת מתוך הסביבה שיצרנו. לצורך כך נריץ את הפקודות הבאות:

```
(TasNet_project) [goldfia1@dsigpu01 ~]$ cd TasNet/egs/wsj0/
(TasNet_project) [goldfia1@dsigpu01 wsj0]$ sh run.sh
```

כאמור לעייל, אנחנו מתחילים להריץ את הרשת משלב מספר 1. לאחר פקודה run.sh הרשת מתחילה מ-stage 1:

```
Stage 1: Generating json files including wav path and duration
Namespace(in_dir='/home/dsi/goldfia1/TasNet/egs/wsj0/data/wsj0-wav/wsj0/2speakers/wav8k/min/', out_dir='data', sample_rate=8000)
```

בסוף שלב זה, קבצי השמע מומרים לקבצים מסוג json. לאחר מכן מגיע השלב של אימון הרשת:

```
Stage 2: Training
```

כאשר ניסינו להריץ את השלב הזה, קבלנו מספר שגיאות. נציין את השגיאות השונות שבהם נתקלנו, ונציין כיצד הצלחנו להתמודד איתם:

1. <u>שגיאה ראשונה:</u>

ImportError: /lib64/libstdc++.so.6: version ‘CXXABI_1.3.9’ not found (required by /home/dsi/goldfia1/anaconda3/lib/python3.8/site-packages/scipy/sparse/_sparsetools.cpython-38-x86_64-linux-gnu.so)

חפשנו בגוגל פתרון לשגיאה זו, וניסינו לבצע את הפקודות המופיעות בקישור הבא:

https://www.programmersought.com/article/72065145753/

אחת הפקודות שהיה עלינו לבצע:

```
(py27) [root@master lib64]# rm -rf libstdc++.so.6
(py27) [root@master lib64]# ln -s libstdc++.so.6.0.28 libstdc++.so.6
(py27) [root@master lib64]# ll libstdc++.so.*
```

כאשר ניסינו לבצע את השלב הזה, גם פה נתקלנו בשגיאה של Permission denied. הבנו שהדרך לבצע את ההעתקה זה רק דרך מי שאחראי על המחשוב שקשור ל-dsi. לאחר שיחה איתו הבנו שהדרך לעשות זאת היא באמצעות שימוש ב- docker. נציין את הדברים שהוסברו לנו במהלך 2 מפגשים שעשינו מולו.

הסבר על docker:

דוקרים הם מיקרו מערכות הפעלה (מכונות וירטואליות). אנו בנינו דוקר שיכיל את מה שאנחנו צריכים בשביל לעבוד, וכתוצאה מכך יכולנו להתקין בתוכו את כל מה שאנחנו צריכים, וכך יהיו לנו הרשאות. אנו לוקחים דוקר קיים וממנו אנו יוצרים את כל התוספות:

```
FROM nvidia/cuda:11.3.1-runtime-ubuntu20.04

ENV CONDA_PATH=/opt/anaconda3
ENV ENVIRONMENT_NAME=tasnet
SHELL ["/bin/bash", "-c"]

#COPY . /app

# curl is required to download Anaconda.
RUN apt-get update && apt-get install curl wget git -y

# Download and install Anaconda.
RUN cd /tmp && curl -O https://repo.anaconda.com/archive/Anaconda3-2019.07-Linux-x86_64.sh
RUN chmod +x /tmp/Anaconda3-2019.07-Linux-x86_64.sh
RUN mkdir /root/.conda
RUN bash -c "/tmp/Anaconda3-2019.07-Linux-x86_64.sh -b -p ${CONDA_PATH}"

# Initializes Conda for bash shell interaction.
RUN ${CONDA_PATH}/bin/conda init bash

# Upgrade Conda to the latest version
RUN ${CONDA_PATH}/bin/conda update -n base -c defaults conda -y

# Create the work environment and setup its activation on start.
RUN ${CONDA_PATH}/bin/conda create --name ${ENVIRONMENT_NAME} python=3.9  -y
RUN echo conda activate ${ENVIRONMENT_NAME} >> /root/.bashrc


#RUN . ${CONDA_PATH}/bin/activate ${ENVIRONMENT_NAME} \
 # && conda env update --file /app/keras37.yml --prune

#RUN apt-get update
#RUN conda install pip
#RUN pip install tensorboard_logger
#WORKDIR /eli
```

התקנו בדוקר גם תמיכה ב-GPU וגם סביבת conda ושעליה נוכל להריץ. הדוקר מבוסס על תוכנה של Nvidia, מוריד anaconda, מתקין אותו, מייצר סביבה של python 3.9 ומפעיל אותה בסיום ההתקנה. כשנריץ את הדוקר, הוא תמיד יופעל בסביבה הזאת.

```
(base) [yos@dsigpu01 docker4tasnet]$ nano dockerfile
(base) [yos@dsigpu01 docker4tasnet]$ nano dockerfile
(base) [yos@dsigpu01 docker4tasnet]$ docker pull nvidia/cuda:11.3.1-runtime-ubuntu20.04^C
(base) [yos@dsigpu01 docker4tasnet]$ ^C
(base) [yos@dsigpu01 docker4tasnet]$ nano dockerfile
(base) [yos@dsigpu01 docker4tasnet]$ docker build -t tasnet:01 .
Sending build context to Docker daemon  3.072kB
Step 1/13 : FROM nvidia/cuda:11.3.1-runtime-ubuntu20.04
```

במקביל לבנייה של הדוקר, היינו צריכים לבצע מחדש את כל ההתקנות שפירטנו לעיל על מנת להריץ את הרשת על הדוקר.

בשביל לראות שהדוקר נבנה, נרשום את הפקודה הבאה:

Docker images

החדש ביותר אמור להופיע הכי למעלה:

```
(base) [yos@dsigpu01 docker4tasnet]$ docker images
REPOSITORY                          TAG                     IMAGE ID       CREATED          SIZE
tasnet                              01                      0e291286953f   2 minutes ago    6.65GB
20210610_success_prediction         latest                  bc31fe8f802e   10 days ago      13.2GB
20210620_success_prediction         latest                  bc31fe8f802e   10 days ago      13.2GB
20210619_success_prediction         latest                  1832b9cbc268   12 days ago      13.2GB
nvidia/cuda                         11.3.1-runtime-ubuntu20.04   7693942c98ed   3 weeks ago      1.86GB
```

ניתן לראות לפי הכותרות למעלה (משמאל לימין):

- השם של ה-image.
- התג, מאפשר ניהול של ה-image. בדרך כלל רושמים שם את הגרסא.
- ID- לכל image יש תעודת זהות משלו.
- מתי הוא נוצר.
- מה הגודל שלו.

אנו נרצה לקחת את ה-tasnet ולהוסיף לו כל מיני דברים שאנו רוצים.

בשביל הרצה של הדוקר, נשתמש בפקודה הבאה:

```
Docker run -it tasnet:01 bash
```

עכשיו נכנסנו עם משתמש לתוך קונטיינר. אנו בעצם מעתיקים את ה-image, מייצרים עותק ממנו ומריצים אותו. כלומר, אנו רצים על רפליקה שאנו בנינו.

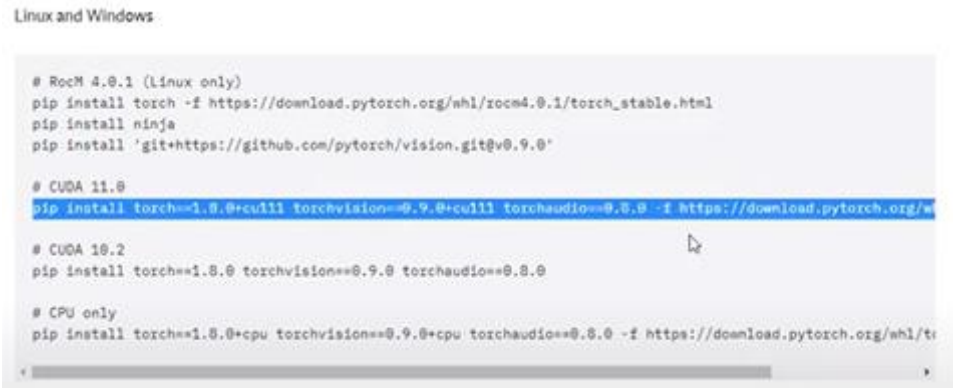
עכשיו כשאנחנו בתוך הקונטרנר, נחזור ל-pytorch, נעתיק את הפקודה שהוא נותן לנו ונתקין את החבילה שמופיעה שם.

לאחר מכאן הרצנו את הפקודה:

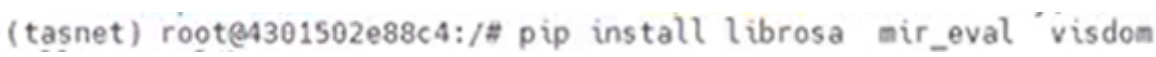
```
(base) [yos@dsigpu01 ~]$ nvidia-smi
Thu Jul  1 09:39:09 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.36.06    Driver Version: 450.36.06    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:02:00.0 Off |                  N/A |
| 25%   33C    P8    11W / 250W |   2763MiB / 11178MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  Off  | 00000000:03:00.0 Off |                  N/A |
| 25%   23C    P8    10W / 250W |      2MiB / 11178MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 108...  Off  | 00000000:81:00.0 Off |                  N/A |
| 25%   23C    P8    11W / 250W |      2MiB / 11178MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  GeForce GTX 108...  Off  | 00000000:82:00.0 Off |                  N/A |
| 25%   23C    P8    11W / 250W |      2MiB / 11178MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

פקודה זו מראה את ה-GPU שיש על השרת. ניתן לראות איזה driver מותקן ואיזה כרטיסים נמצאים ואיזו גירסא של cuda.

צריך לשים לב שיש לנו כאן 11.0 וניסינו להתקין 11.1. לכן העדפנו להתקין גירסא שתהיה לפחות נמוכה מאשר מה שיש על השרת. לכן התקנו את הגירסא הבאה מהאתר:



לאחר מכן התקנו את החבילות שמופיעות בgithub:



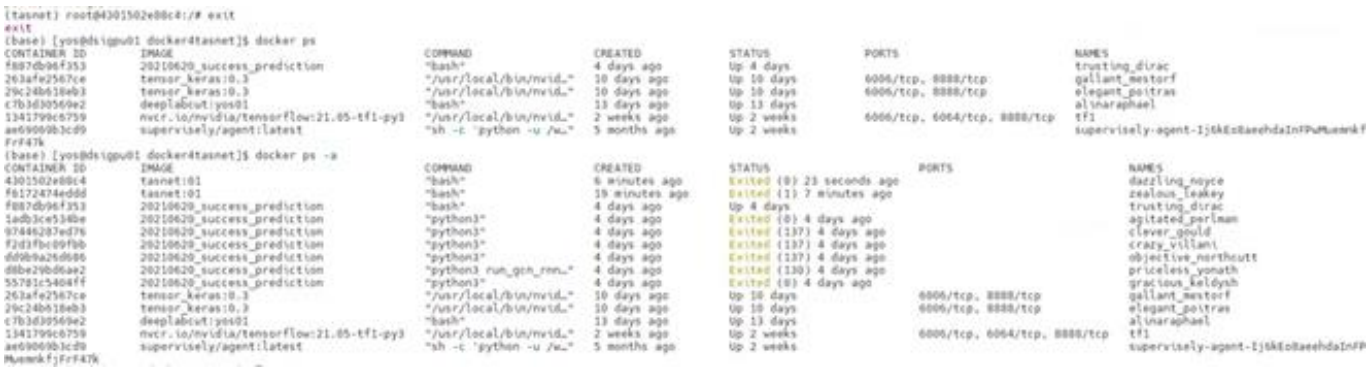
עכשיו כשבנינו את מה שצריך בתוך הדוקרים, אז נרצה לשמור שינויים בקונטרנר. הייעוד שלו, לרוץ ולבצע פקודה, ואז להמחק. נבצע exit לקונטרנר.

נשים לב שנוכל לראות איזה קונטרנרים רצים על ידי הפקודה:

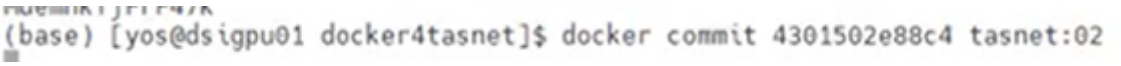
Docker ps

ואם נרצה לראות קונטרנרים שנעצרו:

Docker ps -a



מתוך הקונטרנר הזה נרצה ליצור image. נעשה זאת על ידי הפקודה:



כאשר העתקנו את ה-ID של הtasnet. בעזרת פקודה זו יצרנו בנייה מחדש וכך נוצר image.

נשים לב, שה-image זה סביבה מוכנה מראש שניתן להפעיל אותה בכל רגע נתון. המנוע של הדוקר ניגש ל-image מעתיק אותו הצידה ומריץ אותו.

```
(base) [yos@dsigpu01 docker4tasnet]$ docker commit 4301502e88c4 tasnet:02
sha256:d83b9513d82e527fd13629671f24dc8e6032ff03dfd46135cc8b91b324edc7be
(base) [yos@dsigpu01 docker4tasnet]$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
tasnet                      02                  d83b9513d82e        25 seconds ago      13.5GB
tasnet                      01                  0e291286953f        27 minutes ago      6.65GB
```

בשלב הבא, לאחר שבנינו את הדוקרים, רצינו להריץ את הקונטיינר מהיוזר שלנו.

כדי להריץ את הקונטיינר שלנו נבצע את הפקודה:

```
(TasNet_project) [goldfia1@dsigpu01 ~]$ docker run --rm -it --gpus '"device=1"' -v /home/dsi/goldfia1:/home/dsi/goldfia1 tasnet:02  bash
(tasnet) root@840f7eb9526a:/#
```

בעזרת פקודה זו, אחרי שהקונטרנר ירוץ הוא ימחק אותו כדי שלא יתפוס לנו מקום על השרת. נשים לב שה-gpu במקום ה-0 תפוס, ולכן כדאי לבחור 1. בנוסף, נרצה לבצע מיפוי של הנתיב ממקור אל יעד.

כעת נוכל להגיע למסלול שלנו:

```
(tasnet) root@840f7eb9526a:/# cd /home/dsi/goldfia1/
(tasnet) root@840f7eb9526a:/home/dsi/goldfia1# ls
Downloads  PycharmProjects  TasNet  anaconda3
(tasnet) root@840f7eb9526a:/home/dsi/goldfia1#
```

בנוסף, נצטרך להוסיף לדוקר את הפקודה עבור המסלול:

```
(tasnet) root@840f7eb9526a:/home/dsi/goldfia1# cd TasNet/egs/wsj0/
(tasnet) root@840f7eb9526a:/home/dsi/goldfia1/TasNet/egs/wsj0# sh run.sh
run.sh: 34: utils/parse_options.sh: Syntax error: Bad for loop variable
(tasnet) root@840f7eb9526a:/home/dsi/goldfia1/TasNet/egs/wsj0# exit
exit
(TasNet_project) [goldfia1@dsigpu01 ~]$ echo $PATH
/home/dsi/goldfia1/anaconda3/envs/TasNet_project/bin:/home/dsi/goldfia1/anaconda3/condabin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/dsi/goldfia
1/.local/bin:/home/dsi/goldfia1/bin
(TasNet_project) [goldfia1@dsigpu01 ~]$ docker run --rm -it --gpus '"device=1"' -v /home/dsi/goldfia1:/home/dsi/goldfia1 tasnet:02  bash
(tasnet) root@b7511a6ca909:/# excho $PATH
bash: excho: command not found
(tasnet) root@b7511a6ca909:/# echo $PATH
/opt/anaconda3/envs/tasnet/bin:/opt/anaconda3/condabin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
(tasnet) root@b7511a6ca909:/# PATH=/home/dsi/goldfia1/anaconda3/envs/TasNet_project/bin:/home/dsi/goldfia1/anaconda3/condabin:/usr/local/bin:/usr/bin:/usr/loc
al/sbin:/usr/sbin:/home/dsi/goldfia1/.local/bin:/home/dsi/goldfia1/bin${PATH:+:${PATH}}
```

לאחר ביצוע פעולות אלו, הצלחנו להריץ את הרשת והתגברנו בכך על השגיאה הראשונה שציינו לעיל.

2. <u>שגיאה שנייה:</u>

TypeError: lstm() received an invalid combination of arguments - got (Tensor, Tensor, tuple, list, bool, int, float, bool, int), but expected one of:

* (Tensor data, Tensor batch_sizes, tuple of Tensors hx, tuple of Tensors params, bool has_biases, int num_layers, float dropout, bool train, bool bidirectional) didn't match because some of the arguments have invalid types: (Tensor, Tensor, !tuple!, !list!, bool, int, float, bool, !int!)

* (Tensor input, tuple of Tensors hx, tuple of Tensors params, bool has_biases, int num_layers, float dropout, bool train, bool bidirectional, bool batch_first) didn't match because some of the arguments have invalid types: (Tensor, !Tensor!, !tuple!, !list!, !bool!, int, !float!, bool, !int!)

לאחר חיפוש בגוגל, מצאנו פתרון לבעיה רלוונטי עבור הרשת שלנו ספציפית, שמי שענה עליה היה לא אחר מאשר מי שבנה את הרשת, וזה היה הפתרון לכך לבצע שינוי בפונקציה של train.py שמבצעת את האימון:

add bool() to self.bidirectional

לאחר שעשינו את השינויים הדרושים, הרשת הצליחה לבצע את האימון. נציין שזמן האימון היה ארוך מאוד ונמשך יומיים (הרשת הריצה 100 אפוקים, וכל אפוק לקח בסביבות ה-1500 שניות). בזמן האימון, הרשת מייצרת קובץ log שמראה בכל רגע נתון את התוצאות המתקבלות:

```
992 Epoch 4 | Iter 1951 | Average Loss -9.468 | Current Loss -11.132581 | 668.6 ms/batch
993 Epoch 4 | Iter 1961 | Average Loss -9.474 | Current Loss -11.045516 | 666.8 ms/batch
994 Epoch 4 | Iter 1971 | Average Loss -9.474 | Current Loss -8.294337 | 665.0 ms/batch
995 Epoch 4 | Iter 1981 | Average Loss -9.481 | Current Loss -12.726813 | 663.2 ms/batch
996 Epoch 4 | Iter 1991 | Average Loss -9.486 | Current Loss -11.221352 | 661.2 ms/batch
997 ---------------------------------------------------------------------------------------
998 Train Summary | End of Epoch 4 | Time 1318.35s | Train Loss -9.493
999 ---------------------------------------------------------------------------------------
1000 Cross validation...
1001 Epoch 4 | Iter 1 | Average Loss -7.960 | Current Loss -7.960156 | 1004.3 ms/batch
1002 Epoch 4 | Iter 11 | Average Loss -8.449 | Current Loss -7.811009 | 656.0 ms/batch
1003 Epoch 4 | Iter 21 | Average Loss -8.450 | Current Loss -10.655063 | 596.1 ms/batch
```

על מנת שנוכל לראות בצורה גרפית את ה-loss כפונקציה של ה-epoch, היה עלינו לבצע את הפעולות הבאות:

1. תחילה, היינו צריכים לפתוח טרמינל חדש, כאשר העדיפות היא להשתמש ב-tmux (דרך שבעזרתה הרשת ממשיכה לרוץ, גם כאשר אנחנו לא מחוברים לשרתים), ולרשום את הפקודה:

   visdom

   כזכור לעיל, אנו התקנו את החבילה הזו קודם לכן.

2. על מנת שנוכל להריץ את הרשת כך שנוכל לחשב את ה-loss, השתמשנו בפקודה הבאה:

   bash run.sh --visdom 1 --visdom_id "TasNet - Graph"

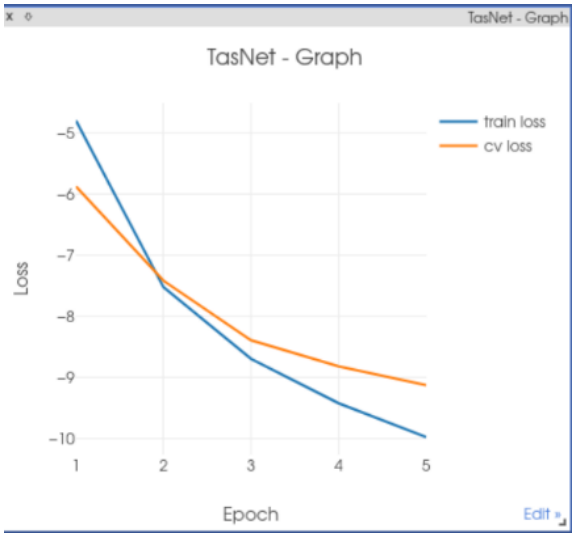
   כאשר, הכותרת של הגרף תהיה TasNet - Graph.

3. פתחנו דפדפן חדש, בעזרת firefox (מתוך הסביבה שאיתה אנו עובדים), ונכנסנו לכתובת הבאה:

   127.0.0.1:8097

בכתובת זו נכנסו לאתר של visdom, שבסוף ההרצה של האימון נוכל לראות את הגרף הרצוי.

במקרה שלנו, התקבל הגרף הבא:

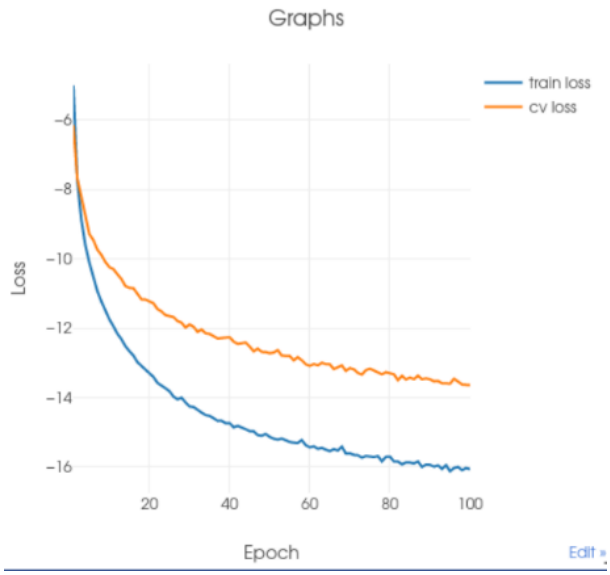


בגרף זה, הקו הכחול מסמן את ה-loss עבור סט האימון, והקו הכתום מסמן את ה-loss עבור ה-cross validation.

ניתן לראות, כפי שאנו מצפים לקבל, שה-loss הולך וקטן כפונקציה של ה-epoch. כמו כן, אפשר לראות שקיבלנו שה-loss מקבל ערכים שליליים. במהלך הקורס שלמדנו בלמידה עמוקה, ראינו שההפסד מגיע לרוב לתוצאה סביב הערך 0, כאשר הקריטריון שלנו הוא cross entropy או MSE. במקרה שלנו, מדובר ברשת קונבולוציה שבו קריטריון ההפסד הוא SI-SNR- (במינוס). במהלך האימון, הערך של SI-SNR הולך וגדל, כך שנהיה גדולים מאפס, ולכן הערך של ההופכי שלו, SI-SNR- נעשה שלילי.

כיוון שהערך של SI-SNR לרוב יהיה קטן מ-20, נוכל להשתמש ב- 20 - SI-SNR עבור קריטריון ה-loss וכך נקבל ערך חיובי להפסד שלנו.

על מנת לקבל תמונה מלאה יותר, ביצענו הרצה עם יותר אפוקים וקיבלנו:



השלב הבא שמתבצע לאחר האימון, הוא חישוב הביצועים של הרשת:

```
Stage 3: Evaluate separation performance
```

בחלק זה לא נתקלנו בשום שגיאה. לאחר הרצת stage 3, התוצאות נשמרו בקובץ log, אשר הכיל את הביצועים עבור SDRi וכן SI-SNRi בצורה הבאה:

```
Utt 98
  SDRi=10.57
  SI-SNRi=10.56
Utt 99
  SDRi=9.21
  SI-SNRi=8.88
Utt 100
  SDRi=13.51
  SI-SNRi=13.14
Average SDR improvement: 10.70
Average SISNR improvement: 10.43
```

כאשר: Utt מתייחס להקלטה של ה-test ועליה הוא מחשב את הביצועים של ההפרדה. כלומר, עבור תוצאה גבוהה נקבל שההפרדה נעשתה בצורה אופטימלית, ועבור תוצאה נמוכה נסיק שההפרדה לא נעשתה בצורה טובה.

בנוסף, בסוף ההרצה ניתן לראות גם את השיפור הממוצע עבור כל אחד מהמדדים.

כדי לקבל תמונה יותר רחבה של הביצועים, הוספנו בפונקציה evaluate.py, אשר מבצעת את stage 3 ברשת, את הוקטורים SDRi, SISNRi, cnt אשר מכילים את כל הערכים עבור כל אחת מההקלטות. בנוסף, הוספנו לקוד את החלק של ה-plot כדי שנוכל להציג את התוצאות על גביי גרף:

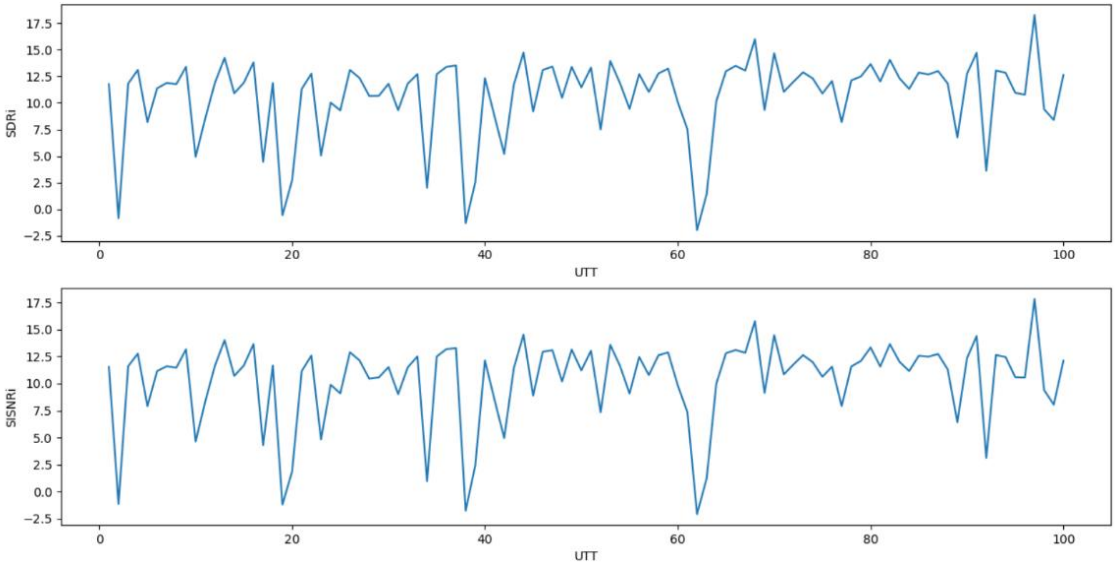
```
            for mix, src_ref, src_est in zip(mixture, source, estimate_source):
                print("Utt", total_cnt + 1)
                cnt.append(total_cnt + 1)
                # Compute SDRi
                if args.cal_sdr:
                    avg_SDRi = cal_SDRi(src_ref, src_est, mix)
                    SDRi.append(avg_SDRi)
                    total_SDRi += avg_SDRi
                    print("\tSDRi={0:.2f}".format(avg_SDRi))
                # Compute SI-SNRi
                avg_SISNRi = cal_SISNRi(src_ref, src_est, mix)
                SISNRi.append(avg_SISNRi)
                print("\tSI-SNRi={0:.2f}".format(avg_SISNRi))
                total_SISNRi += avg_SISNRi
                total_cnt += 1
    if args.cal_sdr:
        print("Average SDR improvement: {0:.2f}".format(total_SDRi / total_cnt))
    print("Average SISNR improvement: {0:.2f}".format(total_SISNRi / total_cnt))

    plt.figure()
    plt.subplot(211)
    plt.plot(cnt, SDRi)
    plt.xlabel('UTT')
    plt.ylabel('SDRi')

    plt.subplot(212)
    plt.plot(cnt, SISNRi)
    plt.xlabel('UTT')
    plt.ylabel('SISNRi')

    plt.show()
```

הגרף הבא מראה עבור כל הקלטה של ה-test מה הייתה רמת הביצוע בהפרדת הדוברים:



ניתן לראות מתוך גרף זה, שברוב המקרים ההפרדה כן הצליחה, אך בחלק מההפרדות- יש להם ביצוע פחות טוב. בסוף ההרצה של הרשת נוכל לבחון מה קורה בכל אחד מהמקרים.

בשלב האחרון של הרשת, אנו מבצעים את הפרדת הדוברים שמופיעים בתיקיית tt (test) באמצעות הרשת:

```
Stage 4: Separate speech using TasNet
```

כאשר הגענו לחלק זה, קיבלנו את השגיאה הבאה:

AttributeError: module 'librosa' has no attribute 'output'

על מנת לפתור את השגיאה הזאת, הוספנו את החלק הבא בקוד (במקום החלק שמופיע כהערה בצבע ירוק):

```
def write(inputs, filename, sr=args.sample_rate):
    import soundfile as sf
    sf.write(filename, inputs, sr)# norm=True)
    #librosa.output.write_wav(filename, inputs, sr)# norm=True)
```

בחלק זה הרשת מייצאת את ההקלטות הבאות:

- האות המעורבב, אותו הרשת מנסה להפריד.
- דובר ראשון.
- דובר שני.

כפי שראינו בחלק הקודם, ישנם מקרים בהם הביצועים לא היו טובים. למשל, עבור הקלטה מספר 2:

```
22 Utt 2
23   SDRi=0.85
24   SI-SNRi=0.54
```

כאשר האזנו להקלטות של הדובר הראשון ושני, באמת שמנו לב שההפרדה לא הצליחה, ושמענו עדיין רעשים מהדובר הנוסף שנמצא בחדר.

לעומת זאת, במקרים בהם הביצוע היה טוב, כמו במקרה הבא:

```
307 Utt 97
308   SDRi=16.69
309   SI-SNRi=16.42
```

אז היה אפשר לשים לב בהקלטה שההפרדה נעשתה בצורה מושלמת, ולא שמענו את הדובר השני ברקע.

בסוף כל התיקונים והשינויים שביצענו, הרשת הצליחה לרוץ גם כאשר שינינו את הפרמטרים ואת מספר ההקלטות בכל אחד מהשלבים (אימון/ test /cross validation).

3. הרצת רשת Tasnet על הדאטאסט שלנו כשהוא חד ערוצי

3.1 התאמת ה-dataset שיצרנו לרשת

בשלב הראשוני, רצינו לראות מה קורה כאשר אנו מעבירים לרשת את ה-dataset שיצרנו בעצמנו לפני הרצת הרשת. כמובן, כפי שציפינו, הרשת לא עבדה, משום שהיא צפתה לקבל את ההקלטות כך שיהיו בפורמט מסוים. כלומר, היא חיפשה הקלטות מסוימות, ולכשלא מצאה אותם אז ההרצה הסתיימה. לכן, היינו צריכים לחזור שוב לשלב 0 של הרשת, רק עבור המקרה שלנו.

לצורך כך, השתמשנו בקוד מטלאב שמשנה את הפורמט של הקבצים ומסדר אותם בתיקיות באותה הצורה של הדאטאסט של wsj0. כזכור, המטלאב מקבל בנוסף להקלטות עליהם הוא יוצר את ה-dataset, את הקובץ txt אשר לוקח את כל ההקלטות ומסדר את ההקלטות בצורה שבה המטלב יבצע את המיקס. כלומר, הוא לא עושה את זה בצורה רנדומלית אלא לפי המסמך שהוא מקבל מראש.

לצורך כך, היינו צריכים להוסיף לחלק של הקוד שלנו, עוד תוספת שתיקח את ה-dataset שאנחנו כבר יצרנו, ותבנה קובץ txt שירשום אילו הקלטות צריך לערבב.

לכן, עבור כל אחת מסוגי ההקלטות השונות (train/ cross validation/ test) הוספנו את השורות קוד הבאות:

```
In [2]: path = 'C:/Users/TAMAR/Desktop/Avi_Study/final_project/dataset/wsj0-hin/tr'

list_of_speakers = os.listdir(path)

#Making list for tr as used to make mixture like wsj
row_list = []

num_of_data_files=len(list_of_speakers)

for x in range(0,num_of_data_files,2):
    #wham_noise = random.choice(os.listdir("/media/reverie-pc/speech/asr/kaldi/egs/Neural-mask-estimation/wham_noise/tr/"))
    #spkr1 = random.choice(os.listdir("C:/Users/TAMAR/Desktop/Avi_Study/final_project/dataset/wsj0-hin/tr/"))
    #spkr2 = random.choice(os.listdir("C:/Users/TAMAR/Desktop/Avi_Study/final_project/dataset/wsj0-hin/tr/"))
    spkr1 = list_of_speakers[x]
    spkr2 = list_of_speakers[x+1]
    #data = [wham_noise,'wsj0/tr/'+spkr1,'wsj0/tr/'+spkr2]
    snr_1 = float(decimal.Decimal(random.randrange(1, 250))/100)
    snr_2 = -snr_1
    data = ['wsj0-hin/tr/'+spkr1, snr_1 ,'wsj0-hin/tr/'+spkr2, snr_2]
    row_list.append(data)
df = pd.DataFrame(row_list)
#df = pd.read_csv(pd.compat.StringIO(row_list), header=None)
#df = pd.read_csv(row_list, header=None)
#df.columns = ['output_filename', 's1_path', 's2_path']
df.to_csv('C:/Users/TAMAR/Desktop/Avi_Study/final_project/dataset/wsj0-hin/mix_2_spk_tr.txt',index=False,sep=' ',header=False)
#df_f = df.apppend()
```

תחילה, הקוד לוקח את ה-path ששם נמצאות כל ההקלטות של אותו חלק (s1,s2,mix) ואז הוא מייצר מהם קובץ txt כך שהמטלב יוכל לקרוא ממנו ולבנות את ה-dataset שייכנס לרשת Tasnet שכבר הרצנו.

נציין, שהקוד לוקח את מה שיצא לנו מתוך הrir ולא מייצר הגרלה נוספת של פרמטרים.

לאחר הרצה אחת של הדאטאסט שבנינו (שמפורט לעיל) הבנו כי צריך לעשות מספר התאמות:

- הוספת דוברים- מכיוון שהכנסנו רק 18 דוברים הייתה חזרתיות- לכן לקחנו כ-200 דוברים לכל אחד משלושת השלבים (train, cross-val, test), כאשר אין חפיפה בין הדוברים בכל אחד מהשלבים.
- בנוסף בשלב זה יצרנו דאטאסט ש"מתחשב" רק במיקרופון אחד על מנת ליצור דאטאסט כמה שיותר זהה לדאטאסט המקורי של wsj0, כדי שנוכל להריץ ולהשוות את התוצאות, בחלק 4 של הפרוייקט אנו נרצה להריץ את הרשת עם הדאטא הרב ערוצי שיצרנו. עשינו זה על ידי שימוש במסנן בקוד מטלאב שמדמה בחירה של מיקרופון אחד.

3.2 <u>הרצות</u>

תחילה הרצנו על הדאטאסט שבנוי על האופצייה של min במטלאב:

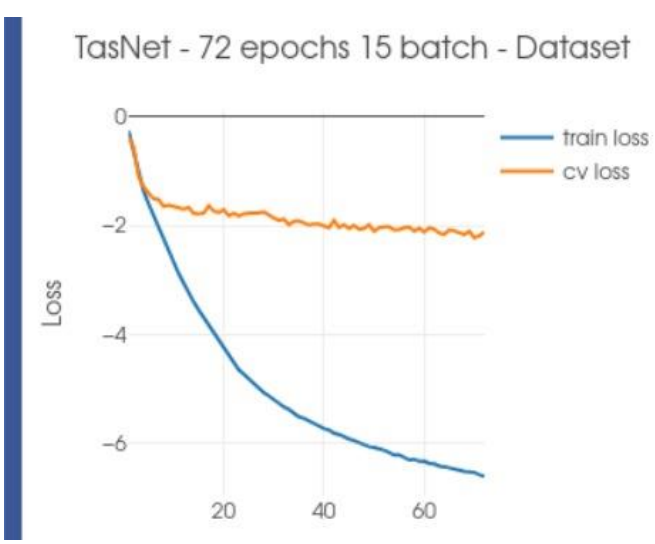
(הגרף נמחק ולכן נציג את הקובץ log):

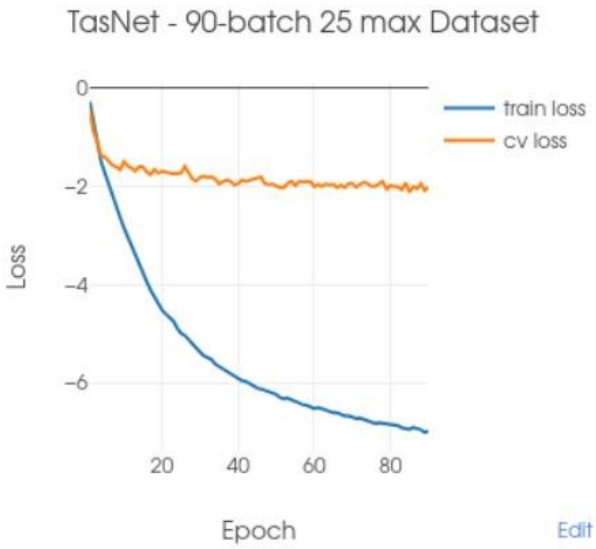
```
Epoch 1 | Iter 1901 | Average Loss 16.490 | Current Loss 22.861437 | 419.8 ms/batch
Epoch 1 | Iter 1911 | Average Loss 16.511 | Current Loss 14.646792 | 418.9 ms/batch
Epoch 1 | Iter 1921 | Average Loss 16.528 | Current Loss 30.280857 | 418.0 ms/batch
Epoch 1 | Iter 1931 | Average Loss 16.502 | Current Loss 18.105444 | 417.1 ms/batch
Epoch 1 | Iter 1941 | Average Loss 16.531 | Current Loss 12.138776 | 416.2 ms/batch
Epoch 1 | Iter 1951 | Average Loss 16.498 | Current Loss 0.830928 | 415.2 ms/batch
Epoch 1 | Iter 1961 | Average Loss 16.475 | Current Loss 4.598598 | 414.1 ms/batch
Epoch 1 | Iter 1971 | Average Loss 16.454 | Current Loss 14.627283 | 413.0 ms/batch
Epoch 1 | Iter 1981 | Average Loss 16.453 | Current Loss 27.572142 | 411.9 ms/batch
Epoch 1 | Iter 1991 | Average Loss 16.467 | Current Loss 12.434778 | 410.7 ms/batch
-----------------------------------------------------------------------------------
Train Summary | End of Epoch 1 | Time 819.31s | Train Loss 16.422
-----------------------------------------------------------------------------------
Cross validation...
```

ניתן לראות כי ההפסד מאוד גבוה ביחס להרצה על הדאטאסט המקורי. כפי שהסברנו למעלה, הבחירה של min גורמת לכך שזמן ההקלטה של ה-mix יהיה על סמך ההקלטה הקצרה יותר מבין השתיים. כתוצאה מכך, חלק מההקלטות היו בזמן של שנייה אחת או שתיים, דבר שחשבנו שגרם לכך שהרשת עברה בצורה כל כך רעה על הדטאסט שלנו.

לכן ניסינו להריץ גם על האופציה של הmax במטלאב, על מנת שנוכל לקבל זמן הקלטה גדול יותר במיקס, דבר שיקל על האימון של הרשת. בנוסף, זמן ההקלטות של הדטאסט שלנו יתקרב כמה שניתן לזמן ההקלטות של הדטאסט עבור wsj0, שעליו ההרצה הקודמת של הרשת עבדה, כפי שהראינו לעייל.

לאחר ההרצה של הרשת, קיבלנו תוצאות סבירות יותר, כפי שניתן לראות בגרפים הבאים:





בנוסף, נצרף את הקובץ log שמראה על הביצועים כעת של הרשת:

```
16451 Epoch 155 | Iter 731 | Average Loss -9.458 | Current Loss -8.783422 | 856.4 ms/batch
16452 Epoch 155 | Iter 741 | Average Loss -9.459 | Current Loss -8.598701 | 852.2 ms/batch
16453 Epoch 155 | Iter 751 | Average Loss -9.445 | Current Loss -8.064981 | 847.8 ms/batch
16454 Epoch 155 | Iter 761 | Average Loss -9.431 | Current Loss -8.359612 | 843.0 ms/batch
16455 ---------------------------------------------------------------------------------------
16456 Train Summary | End of Epoch 155 | Time 645.24s | Train Loss -9.426
16457 ---------------------------------------------------------------------------------------
16458 Saving checkpoint model to exp/train_r8000_L40_N500_h500_l4_bi1_C2_epoch155_half0_norm
16459 Cross validation...
16460 Epoch 155 | Iter 1 | Average Loss -3.059 | Current Loss -3.058898 | 676.4 ms/batch
16461 Epoch 155 | Iter 11 | Average Loss -2.535 | Current Loss -2.027600 | 646.5 ms/batch
16462 Epoch 155 | Iter 21 | Average Loss -2.895 | Current Loss -2.942387 | 618.7 ms/batch
16463 Epoch 155 | Iter 31 | Average Loss -2.749 | Current Loss -2.440747 | 595.4 ms/batch
16464 Epoch 155 | Iter 41 | Average Loss -2.788 | Current Loss -1.526500 | 582.3 ms/batch
16465 Epoch 155 | Iter 51 | Average Loss -2.824 | Current Loss -4.820712 | 567.4 ms/batch
16466 Epoch 155 | Iter 61 | Average Loss -2.684 | Current Loss -2.953311 | 557.4 ms/batch
16467 Epoch 155 | Iter 71 | Average Loss -2.760 | Current Loss -1.767817 | 549.9 ms/batch
16468 Epoch 155 | Iter 81 | Average Loss -2.696 | Current Loss -2.408313 | 542.6 ms/batch
16469 Epoch 155 | Iter 91 | Average Loss -2.679 | Current Loss -5.348856 | 538.4 ms/batch
16470 Epoch 155 | Iter 101 | Average Loss -2.559 | Current Loss -2.340805 | 531.5 ms/batch
16471 Epoch 155 | Iter 111 | Average Loss -2.575 | Current Loss -2.191837 | 525.9 ms/batch
16472 Epoch 155 | Iter 121 | Average Loss -2.515 | Current Loss -2.878057 | 520.9 ms/batch
16473 Epoch 155 | Iter 131 | Average Loss -2.510 | Current Loss -2.871092 | 514.7 ms/batch
16474 Epoch 155 | Iter 141 | Average Loss -2.436 | Current Loss -1.011664 | 510.0 ms/batch
16475 Epoch 155 | Iter 151 | Average Loss -2.447 | Current Loss -1.340779 | 505.0 ms/batch
16476 Epoch 155 | Iter 161 | Average Loss -2.485 | Current Loss -3.034184 | 500.5 ms/batch
16477 Epoch 155 | Iter 171 | Average Loss -2.491 | Current Loss -3.664438 | 493.5 ms/batch
16478 Epoch 155 | Iter 181 | Average Loss -2.524 | Current Loss -4.691522 | 488.2 ms/batch
16479 Epoch 155 | Iter 191 | Average Loss -2.528 | Current Loss -1.217839 | 482.1 ms/batch
16480 ---------------------------------------------------------------------------------------
16481 Valid Summary | End of Epoch 155 | Time 738.64s | Valid Loss -2.524
16482 ---------------------------------------------------------------------------------------
```

```
9021 Average SDR improvement: 2.96
9022 Average SISNR improvement: 2.20
```

הרצנו על batch של 15 ו25 , ניתן לראות כי קיבלנו תוצאות טובות יותר מאשר עם האופציה של המin אך עדיין פחות טובות מהדאטאסט המקורי. משמיעה של הקבצים ניתן לשמוע שאין הפרדה מוחלטת בין הדוברים כמו שהיה בדאטאסט המקורי אך כן קיימת החלשה של דובר אחד לעומת השני.

מהרצות אילו הסקנו שככל הנראה התוצאות פחות טובות של הדאטאסט שלנו לעומת המקורי, ונביא לכך מספר סיבות:

- ככל הנראה התוצאות נובעות מאורך ההקלטות שיותר קצר מהדאטאסט המקורי, דבר זה מסתדר עם ההגיון מכיוון שככל שהקלטה ארוכה יותר, כך לרשת יש יותר זמן ללמוד כל הקלטה ולהצליח להפריד בין דוברים.
- בדטאסט שעליה הרשת עבדה, wsj0, כפי שציינו לעייל, מדובר על הקלטות נקיות של דוברים. כלומר, אין שם את המימד של דוברים בתוך חדר. לכן, במקרה שלנו מדובר בדטאסט שמכיל יותר גורמים שעלולים להקשות על האימון, כגון: הדהוד/ גודל החדר/ מיקום הדוברים/ מיקום המיקרופונים.

לצערנו לא יכלנו לשחק יותר מדי עם גודל הbatch כי בגודל גדול יותר מ26 קיבלנו את הerror הבא:

```
return forward_call(*input, **kwargs)
File "/home/dsi/goldfia1/TasNet/src/tasnet.py", line 36, in forward
  est_mask = self.separator(mixture_w, mixture_lengths)
File "/home/dsi/goldfia1/anaconda3/envs/TasNet_project/lib/python3.9/site-packages/torch/nn/modules/module.py", li
  return forward_call(*input, **kwargs)
File "/home/dsi/goldfia1/TasNet/src/tasnet.py", line 155, in forward
  packed_output, hidden = self.rnn(packed_input)
File "/home/dsi/goldfia1/anaconda3/envs/TasNet_project/lib/python3.9/site-packages/torch/nn/modules/module.py", line
  return forward_call(*input, **kwargs)
File "/home/dsi/goldfia1/anaconda3/envs/TasNet_project/lib/python3.9/site-packages/torch/nn/modules/rnn.py", line 682
  result = _VF.lstm(input, batch_sizes, hx, self._flat_weights, self.bias,
RuntimeError: CUDA out of memory. Tried to allocate 9.40 GiB (GPU 0; 10.92 GiB total capacity; 3.69 GiB already allocate
GiB reserved in total by PyTorch)
# Accounting: time=25 threads=1
# Ended (code 1) at Sun Sep  5 20:15:02 IDT 2021, elapsed time 25 seconds
```

ניסינו לשחק עם הlr על מנת לראות אם ניתן לגרום ללמידה להיות מהירה יותר, ניתן לראות שב lr =10^-2 קיבלנו את הerror הבא:

```
Epoch 24 | Iter 361 | Average Loss 15.483 | Current Loss 32.962112 | 812.3 ms/batch
Epoch 24 | Iter 371 | Average Loss 15.336 | Current Loss 12.345653 | 807.9 ms/batch
Epoch 24 | Iter 381 | Average Loss 15.527 | Current Loss 36.259361 | 803.7 ms/batch
Epoch 24 | Iter 391 | Average Loss 15.594 | Current Loss 12.588418 | 799.7 ms/batch
Epoch 24 | Iter 401 | Average Loss 15.583 | Current Loss 12.557402 | 795.8 ms/batch
Epoch 24 | Iter 411 | Average Loss 15.550 | Current Loss 7.082075 | 792.0 ms/batch
Epoch 24 | Iter 421 | Average Loss 15.722 | Current Loss 33.637650 | 788.2 ms/batch
Epoch 24 | Iter 431 | Average Loss 16.003 | Current Loss 15.068383 | 784.5 ms/batch
/home/dsi/goldfia1/TasNet/src/solver.py:180: FutureWarning: Non-finite norm encountered in torch.nn
.utils.clip_grad_norm_; continuing anyway. Note that the default behavior will change in a future
 release to error out if a non-finite total norm is encountered. At that point, setting
 error_if_nonfinite=false will be required to retain the old behavior.
  torch.nn.utils.clip_grad_norm_(self.model.parameters(),
Epoch 24 | Iter 441 | Average Loss nan | Current Loss nan | 780.6 ms/batch
Epoch 24 | Iter 451 | Average Loss nan | Current Loss nan | 776.3 ms/batch
Epoch 24 | Iter 461 | Average Loss nan | Current Loss nan | 772.1 ms/batch
Epoch 24 | Iter 471 | Average Loss nan | Current Loss nan | 768.0 ms/batch
Epoch 24 | Iter 481 | Average Loss nan | Current Loss nan | 764.0 ms/batch
Epoch 24 | Iter 491 | Average Loss nan | Current Loss nan | 760.1 ms/batch
Epoch 24 | Iter 501 | Average Loss nan | Current Loss nan | 756.2 ms/batch
Epoch 24 | Iter 511 | Average Loss nan | Current Loss nan | 752.3 ms/batch
Epoch 24 | Iter 521 | Average Loss nan | Current Loss nan | 748.6 ms/batch
Epoch 24 | Iter 531 | Average Loss nan | Current Loss nan | 744.9 ms/batch
Epoch 24 | Iter 541 | Average Loss nan | Current Loss nan | 741.1 ms/batch
Epoch 24 | Iter 551 | Average Loss nan | Current Loss nan | 737.4 ms/batch
```

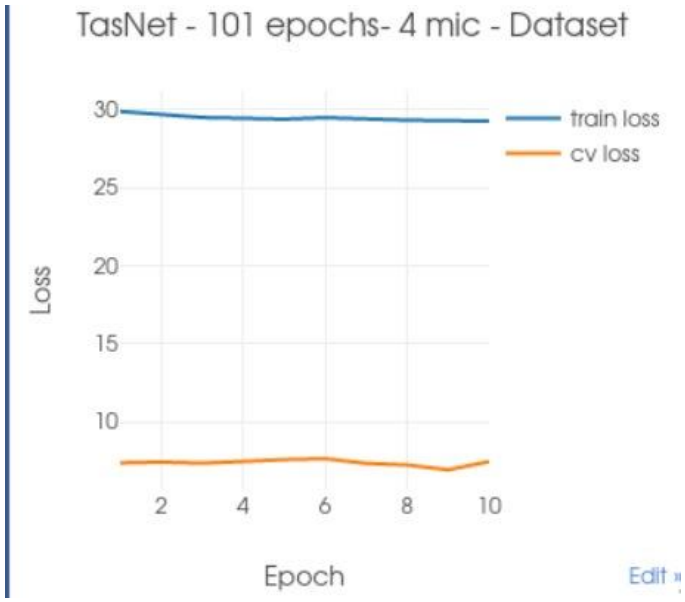
מחיפוש באינטרנט הבנו שמדובר במקרה של exploadind gradient ולכן נשארנו עם lr 10^-3 שמכיוון שהוא קטן יותר לא גורם לגרדיאנט להשתנות בצורה "קיצונית".

בכל מקרה, ניתן לראות שהצלחנו לקבל תוצאה סבירה שכן מוכיחה לנו שהרשת מבצעת תהליך של הפרדה. כמובן שלא קיבלנו תוצאות כמו הרשת המקורית, בהתאם לנימוקים שציינו, אך בכל מקרה תוצאות אלו היו מספיק טובות על מנת שנוכל להתקדם לשלב הבא והאחרון.

4. הרצת רשת Tasnet על הדאטאסט שלנו כשהוא רב ערוצי

בשלב זה הפכנו את הדטאסט שלנו לרב ערוצי על ידי שינוי במטלב, שלא ישתמש במסנן, ושישמור על 4 המיקרופונים שהכנסנו בקוד הפייטון שיוצר את הדטאסט.

הרצות:



הרצנו על הדטאסט של 4 הערוצים. ניתן לראות שקיבלנו תוצאות מאוד גרועות. בנוסף לכך, מכיוון שהדטאסט עצמו כבד מאוד לא הצלחנו להריץ יותר מ-10 אפוקים, מכיוון שנגמר הזיכרון בשרת.

מכיוון שראינו שהרשת עצמה לא מספקת את התוצאה הרצויה, אז הבנו שעלינו לקרוא מאמרי המשך של הרשת. לאחר חיפוש מצאנו 2 מאמרים שרלוונטיים למקרה שלנו, עבור ההפרדה הרב ערוצית:

- End-to-End Multi-Channel Speech Separation.
- Beam- TasNet: Time-domain Audio Separation Network Meets Frequendy-domain Beamformer.

ראינו ששני מאמרים אלו מתבססים על הרשת TasNet, ולכן חשבנו שהם יכולים לעזור לנו.

מכיוון שהזמן לא היה לטובתנו, רצינו לנסות לקחת את הרשתות הנ"ל, לחפש עבורם github (בדומה למה שעשינו עבור הרשת TasNet), ואז לקחת את ה-dataset שלנו במקרה הרב ערוצי ול"השתיל" אותו לתוך הרשת. לצערנו, למרות שניסינו הרבה דברים ונעזרנו בגורמים הרלוונטיים, כפי שנסביר בהמשך, לא הצלחנו להגיע למצב שהם הצליחו לרוץ לנו.

בכל מקרה, נציין את השלבים שכן ניסינו לעשות, ונסביר מה הייתה הבעיה שגרמה לכך שלא הצלחנו להריץ.

- **FaSNet-TAC-PyTorch**

הרשת שאיתה עבדנו נמצאת בקישור הבא:

https://github.com/yoonsanghyu/FaSNet-TAC-PyTorch

בהסתמך על השלבים שרשומים ב-git היה עלינו לבצע את הדברים הבאים:

- להוריד את ה-dataset באותו אופן של הרשת TAC, כפי שמופיע בקישור הבא:

https://github.com/yluo42/TAC/tree/master/data

- לבצע אימון לרשת על ידי הפקודה:

```
python train.py
```

- לבצע חישוב ביצועים עבור ה-test על ידי הפקודה:

```
python evaluate.py
```

ניתן לראות, שבניגוד לרשת הקודמת שאיתה עבדנו, אין קובץ run.sh מסודר, אך ניסינו להתאים את עצמנו לפי מה שמופיע בהוראות שב-git.

תחילה, ניסינו "להתעלם" מהשלב הראשון, ולגרום לכך שהרשת תריץ את ה-dataset שלנו, בכך שניסינו לשנות את ה-path אל המקום ששם נמצאות ההקלטות שלנו. לצערנו, דבר זה לא צלח, משום שהוא ציפה לקבל הקלטות עם פורמט מסוים שלא היה קיים ב-dataset שלנו:

```
FileNotFoundError: [Errno 2] No such file or directory: '/home/dsi/goldfia1/TasNet/egs/wsj0/data/wsj0-wav/wsj0/2speakers
/wav8k/min/tr/sample13555/mixture_mic1.wav'
```

לכן, ניסינו לעבוד לפי "הספר" ולייצר את ה-dataset כפי שמופיע בקישור שצרפנו לעייל.

תחילה, היה עלינו להוריד את ההקלטות הבאות לצורך השלבים השונים של האימון והטסט:

- train-clean-100.
- dev-clean .
- test-clean .

לאחר מכן היה עלינו להכניס אותם לתוך תת תיקיה. בנוסף, היה עלינו להוריד הקלטות של רעש שאמור להתווסף לאותם הקלטות:

- 100 Nonspeech Sounds data.

לאחר מכן, היה עלינו להוריד תוספות לפייטון:

- Soundfile==0.10.0

נעשה בקלות על ידי הפקודה:

Pip install SoundFile

- gpuRIR

לצורך כך, נכנסנו לקישור הבא:

https://github.com/DavidDiazGuerra/gpuRIR

לאחר שהורדנו לרשת את ה-git על ידי פקודת git clone, ניסינו לבצע התקנה של ה-gpuRIR על ידי הפקודה:

```
python setup.py install
```

לאחר הרצת פקודה זו קיבלנו את השגיאה הבאה:

```
running build_ext
-- The CUDA compiler identification is unknown
CMake Error at CMakeLists.txt:5 (project):
  No CMAKE_CUDA_COMPILER could be found.

  Tell CMake where to find the compiler by setting either the environment
  variable "CUDACXX" or the CMake cache entry CMAKE_CUDA_COMPILER to the full
  path to the compiler, or to the compiler name if it is in the PATH.


-- Configuring incomplete, errors occurred!
See also "/home/dsi/goldfia1/TAC/data/gpuRIR/build/temp.linux-x86_64-3.9/CMakeFiles/CMakeOutput.log".
See also "/home/dsi/goldfia1/TAC/data/gpuRIR/build/temp.linux-x86_64-3.9/CMakeFiles/CMakeError.log".
```

ניסינו לקרוא על זה בגוגל, והבנו שזה בעיה ברמת ההתקנת חבילות. לכן פנינו שוב לאחראי של ה-dsi כדי לקבל מענה, והבנו שבשביל לפתור את הבעיה הזאת עלינו שוב להשתמש בדוקרים.

ישבנו עם אחד מהאחראים על מנת לפתור את השגיאה. כן הצלחנו להתקין את ה-gpuRIR בעזרת שימוש בדוקרים, אך למרות ניסיונות חוזרות ונשנות להריץ את הבנייה של ה-dataset שעליו הרשת TAC אמורה לעבוד, לא הצלחנו להתמודד עם העניין, וקיבלנו את השגיאות הבאות:

```
root@e3ca64da56a0:/avidata# python3 create_dataset.py --output-path=/home/dsi/goldfia1/TAC/data/output --avoid-clipping=0 --dataset='adhoc' --libri-pa
noise-path=/home/dsi/goldfia1/TAC/data/noise
GPUassert: no CUDA-capable device is detected /tmp/pip-whv8svag-build/src/gpuRIR_cuda.cu 1002
```

```
root@46c7a2e1e7fc:/avidata# python3 create_dataset.py --output-path=/avidata/output --avoid-clipping=0 --dataset='adhoc'
noise-path=/avidata/noise
GPUassert: system has unsupported display driver / cuda driver combination /tmp/pip-whv8svag-build/src/gpuRIR_cuda.cu 10
```

נראה שלמרות החבילות השונות שהתקנו, עדיין יש משהו שלא תומך ב-cuda שהתקנו לתוך הדוקר.

לצערנו לא הצלחנו למצוא פתרון לבעיה, במסגרת הזמן שהוקצב לנו.

- **Beam-Guided-TasNet**

הרשת שאיתה עבדנו נמצאת בקישור הבא:

https://github.com/hangtingchen/Beam-Guided-TasNet

מדובר ברשת שעושה הפרדה לדאטאסט רב ערוצי כאשר ניתן לבחור כאחד הפרמטרים של הרשת את מספר הערוצים ברשת, חשבנו שהיא תיהיה יותר רלוונטית מהקודמת מכיוון שברשת זו הריצו על הדאטאסט של wsj0 שאיתו עבדנו גם ברשת הtasnet החד ערוצית ולכן יהיה לנו יותר קל לעבוד איתה.

התקנו את כל הדרישות שכתוב שהרשת צריכה והרצנו את קובץ הrequrment.txt על מנת לעשות חלק מההתקנות.

כאשר הרצנו נדרשו התקנות נוספות שאת רובן הצלחנו להתקין ונתקלנו בשגיאה הבאה שלא הצלחנו לפתור:

```
(TasNet_project) [goldfia1@dsigpu01 Beam-Guided-TasNet]$ bash run.sh --stage 3 --tag my_tag --task sep_noisy --id 0,1
Results from the following experiment will be stored in exp/train_convtasnet_my_tag
Stage 3: Training
Traceback (most recent call last):
  File "/home/dsi/goldfia1/Beam-Guided-TasNet/train.py", line 12, in <module>
    from asteroid.data.spatial_wsj0_mix import make_dataloaders
ModuleNotFoundError: No module named 'asteroid.data.spatial_wsj0_mix'
Stage 4: Training
Traceback (most recent call last):
  File "/home/dsi/goldfia1/Beam-Guided-TasNet/train.py", line 12, in <module>
    from asteroid.data.spatial_wsj0_mix import make_dataloaders
ModuleNotFoundError: No module named 'asteroid.data.spatial_wsj0_mix'
Stage 5 : Evaluation
Traceback (most recent call last):
  File "/home/dsi/goldfia1/Beam-Guided-TasNet/eval.py", line 16, in <module>
    from asteroid.data.spatial_wsj0_mix import SPWSJ0MIXDataset
ModuleNotFoundError: No module named 'asteroid.data.spatial_wsj0_mix'
(TasNet_project) [goldfia1@dsigpu01 Beam-Guided-TasNet]$
```

ראינו לאחר מכן כי הדאטא של הרשת עובר "עיבוד" נוסף ע"י הרצת קוד מטלאב בשם:

spatialize_wsj0-mix.zip

שנראה לנו שהשגיאה הזאת נובעת בעקבות זאת שאנו לא הרצנו את המטלאב הזה על הדאטא שלנו לאחר ההרצות האחרות, לצערינו לא נותר לנו מספיק זמן על מנת לבחון את העניין ולנסות להריץ את המטלאב על הדאטאסט שלנו ולהכניס אותו חזרה על השרת.

סיכום ומסקנות:

במהלך העבודה הרבה שהקדשנו על הפרויקט, יצא לנו ללמוד ולהכיר את הנושאים הבאים:

- במהלך השנה לקחנו את הקורסים של "למידה עמוקה" ו"מבוא ללמידת מכונה", אשר הם היו נושאים חדשים שלא יצא לנו ללמוד ולהתעסק בתחומים אלו עד לשנה זו.
- שימוש בכלים של mobaxterm והשרתים, שבעזרתם יכולנו להריץ רשתות מורכבות.
- למדנו על הנושא של דוקרים, בכל מה שקשור להתקנה של חבילות על השרתים, דבר אשר עזר לנו בסופו של דבר בהרצת הרשת המקורית. לצערנו, זה לא הצליח לעזור לנו להריץ במקרה הרב ערוצי, אך הוא כן קירב אותנו אל הפתרון.
- שימוש ברשת שכבר הייתה בנויה, והתאמה של ה-dataset שיצרנו אליה, תוך התייחסות לרשת כאל "קופסא שחורה" אשר יש להפעיל עליה מניפולציות כדי שתוכל להריץ את מה שבנינו.

כמו כן, לאור התוצאות שקיבלנו והדברים שעברנו בדרך, הגענו למסקנות הבאות:

- נראה כי הדאטאסט שיצרנו לא כל כך תאם את הרשת מבחינת אורך ההקלטות, לאחר שניסינו להריץ די הרבה קונסטלציות על הרשת החד ערוצית עם הדאטאסט שלנו הגענו למסקנה שסיכוי גבוה שההקלטות איתן עבדנו קצרות יותר מאשר בדאטאסט המקורי מה שגורם להחטה משמעותית בביצועים. הדבר מנה ממנו לבחון את ההשפעה של השינויים האחרים שעשינו בדאטאסט, כגון שינוי הזווית של הדוברים והמרחק ביחס למיקרופון, גודל החדר וכל מה שמפורט בסעיף של בניית הדאטאסט.
- השרתים של האוניברסיטה מאוד הגבילו אותנו מבחינת ההרצות, היו הרבה מגבלות בהתקנות ומבחינת זיכרון של השרתים שגבו מאיתנו הרבה זמן והתעסקות בפן הטכני של הפרוייקט, ושמנעו מאיתנו לעשות הרבה הרצות ולשנות חלק מהפרמטרים. במבט לאחור אולרי היינו מבקשים לעבוד על שרת של גוגל או קונים יוזר על מנת להריץ את הפרוייקט מכיוון שהבנו ששם יש הרבה פחות מגבלות מהבחינה הזאת.
- יכול להיות שאם היה לנו יותר זמן, היינו כן מצליחים להתמודד עם הבעיות הטכניות שאיתם נתקלנו בכל מה שקשור להתקנת החבילות. זה המקום לציין, שגם ברשת המקורית לקח לנו מספר חודשים עד שהצלחנו להגיע למצב בו הרשת מצליחה לרוץ עם ה-dataset המקורי שהיא מכירה, כך שבמקרה הרב ערוצי כבר לא נשאר לנו מספיק זמן לכך.
- בעקבות התפרצות מגפת הקורונה, כל הפרויקט נעשה בפורמט של עבודה מרחוק/ הבית. אנו חושבים שדבר זה הקשה על העבודה השותפת, על הסנכרון בינינו, וכן על העזרה הטכנית שהייתה חסרה לנו. אנו חושבים שבמצב רגיל, היינו מצליחים להתמודד עם הבעיות השונות בצורה מהירה יותר, דבר שהיה מפנה לנו זמן לחלק האחרון של הפרוייקט.

<u>הצעות להמשך – מה היינו ממשיכים בפרוייקט לו היה זמן:</u>

- להצליח להריץ אחת מהרשתות הרב ערוציות.
- להקליט דאטאסט במעבדה ולהצליח להריץ אותו קודם כל על הרשת החד ערוצית על מנת להשוות את התוצאות לדאטאסטים שהרצנו ולאחר מכן על הרב ערוצית (לאחר שמצליחים להריץ אותה)
- רצינו לבחון את ההשפעה של הפרמטרים השונים (זווית, מרחק וכו') על ההפרדה, מכיוון שהדאטא סט שלנו היה בנוי מהקלטות קצרות יותר מה שראינו שהוריד משמעותיתי את הביצועים את הרשת היינו אולי בונים דאטא סט מהקלטות ארוכות יותר על מנת לבחון את התיאוריה ולבודד השפעה של זה על הביצועים כך שנוכל גם לבחון השפעה של הפרמטרים האחרים.

<u>תודות:</u>

לסיום, כאן המקום למנחה האקדמי פרופ' שרון גנות, אשר תמיד היה אוזן קשבת במהלך השנה, ונתן לנו מידע תיאורטי על הנושאים השונים.

כמו כן, נרצה להודות למנחה רננה אופוצ'ינסקי, שליוותה אותנו במהלך השנה הן במיילים הן בזום והן בפלאפון. אנו מניחים שגם הפורמט הזה והסגרים שנפלו על כולנו, הפכו את המשימה למסובכת יותר גם עבורה.

בנוסף, נרצה להודות ליוסי קליינר ויוסי שיש, אשר סייעו לנו בכל מה שקשור לשרתים GPU-dsi, הסבירו לנו על העבודה עם דוקרים, וכן סייעו לנו בהתקנת החבילות והרצת הרשתות השונות.

רפרנסים:

- Y. Luo and N. Mesgarani, "TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, 2018, pp. 696-700, doi: 10.1109/ICASSP.2018.8462116.
- https://github.com/kaituoxu/TasNet : A PyTorch implementation of "TasNet: Time-domain Audio Separation Network for Real-time, single-channel speech separation", published in ICASSP2018, by Yi Luo and Nima Mesgarani.
- T. Ochiai, M. Delcroix, R. Ikeshita, K. Kinoshita, T. Nakatani and S. Araki, "Beam-TasNet: Time-domain Audio Separation Network Meets Frequency-domain Beamformer," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 6384-6388, doi: 10.1109/ICASSP40776.2020.9053575.
- End-to-End Multi-Channel Speech Separation Rongzhi Gu∗1,3 , Jian Wu∗2,3 , Shi-Xiong Zhang4 , Lianwu Chen3 , Yong Xu4 , Meng Yu4 , Dan Su3 , Yuexian Zou1 , Dong Yu4 1Peking University Shenzhen Graduate School, Shenzhen, China 2Northwestern Polytechnical University, Xi'an, China 3Tencent AI Lab, Shenzhen, China 4Tencent AI Lab, Bellevue WA, USA
- https://github.com/yoonsanghyu/FaSNet-TAC-PyTorch : Full implementation of "End-to-end microphone permutation and number invariant multi-channel speech separation" (Interspeech 2020)
- https://github.com/yluo42/TAC/tree/master/data
- https://github.com/hangtingchen/Beam-Guided-TasNet : Beam-Guided TasNet: An Iterative Speech Separation Framework with Multi-Channel Output