****

**המחלקה להנדסת חשמל**

שם הפרויקט: הפרדת כלי נגינה וזמר/ת מהקלטות של שירים.

Project Name: separation of musical instruments and singer recordings of songs.

ספר הפרוייקט

|  |  |
| --- | --- |
| שם הסטודנט: | בן ציון צוברי |
| מספר תעודת זהות: |  |
| שם המנחה: | שגיא הרפז |
| חתימת המנחה: |  |
| תאריך ההגשה: | 5.6.2021 |

**תודות:**

תהליך ביצוע פרויקט הגמר היה לא פשוט כלל, הייתי צריך לשלב בין המסגרת הצבאית מרובת האחראיות והשגרה הלא צפויה לבין למידה של חומרים חדשים שלא נחשפתי אליהם לעומק וההתקדמות בפרויקט הגמר, לכן ברצוני להביע את הערכתי ותודתי לאנשים שעזרו לי במהלך הדרך.

* להורים שלי.
* למנחה הפרויקט שגיא הרפז.
* לניר אלנברג וסגל מחלקת פרויקטים ואפקה.
* לחברים הקרובים ולחברים מהצבא.

בלעדיהם לא הייתי מצליח להתקדם הלאה ולסיים את מטלת פרויקט הגמר.

**תוכן עניינים:**

**רשימות** ........................................................................................................................4

**תקציר** ..........................................................................................................................5

...............................................................................................7

**מילון מונחים** .................................................................................................................9

**מבוא** ...........................................................................................................................10

**מטרת הפרויקט, יעדים ומדדים** .......................................................................................12

**מקירת ספרות** ..............................................................................................................13

**ניתוח חלופות**

חלופה מערכתית .....................................................................................................14

חלופה טכנולוגית – ..........................................................................................15

**תכן מפורט**

דרישות בסיסיות ליישום הפרויקט ...............................................................................17

רשת קונבולוציה ......................................................................................................18

אופטימיזציה ...........................................................................................................20

.................................................................................................*21*

...............................................................................................21

..................................................................................................*22*

*המודל של הפרויקט ותהליך האימון ............................................................................23*

*חבילת הקוד של הפרויקט .........................................................................................27*

*הרצת תהליך האימון ושימוש במודל ............................................................................29*

***התוצר*** *.........................................................................................................................32*

***בדיקות והערכה***

*.....................................................................................................................33*

*.....................................................................................................................34*

*.....................................................................................................................35*

*....................................................................................................................36*

***סיכום ומסקנות*** *.............................................................................................................37*

***הצעות עבודה להמשך*** *...................................................................................................38*

***תכנון הפרויקט, ריכוז שינויים וניהול סיכונים*** *.....................................................................39*

***רשימת מקורות*** *.............................................................................................................40*

***נספחים***

*נספח א' – מפרט מערכת ........................................................................41*

*נספח ב' – ..........................................................................................*42

*נספח ג' – פורמט סקר .....................................................................................54*

*נספח ד' – פוסטר הפרויקט .......................................................................................55*

**רשימות:**

**איורים:**

* איור 1: .
* איור 2: .
* איור 3: .
* איור 4: .
* איור 5: ארכיטקטורת .
* איור 6: דיאגרמת בלוקים לאלגוריתם מציאת המודל.
* איור 7: דיאגרמת בלוקים מופשטת ל- .
* איור 8: חבילת הקוד.
* איור 9: הרצת התוכנית.
* איור 10: מחזור שלם.
* איור 11: בחינת אבלואציה סופית למודלים.
* איור 12: ביצוע הפרדה ב- .

**גרפים:**

* גרף 1: חישוב של רכיב ה- .
* גרף 2: חישוב של רכיב ה-
* גרף 3: חישוב של רכיב ה-
* גרף 4: חישוב של רכיב ה-
* גרף 5: חישוב להפרדות שירים מה- .
* גרף 6: תוצאות מדד .

**טבלאות:**

* טבלה 1: ניתוח חלופות מערכתיות.
* טבלה 2: ניתוח חלופות טכנולוגיות.

**נוסחאות:**

* נוסחא 1: חישוב .
* נוסחא 2: חישוב .
* נוסחא 3: .
* נוסחא 4 : .
* נוסחא 5: חישוב .

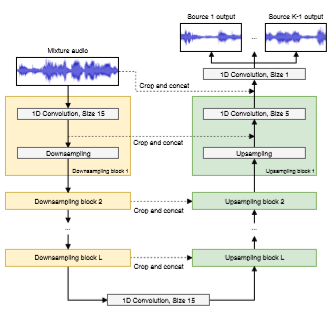
**תקציר:**

כיום, נפוץ שמשתמשים אינטרנטיים נעזרים בתוכנות ייעודיות על מנת להפריד שירים בפורמטים שונים למקורות (זמר/ת, בס תופים, צלילים אחרים), תוכנות אלו עולות כסף, לרוב לא נוחות לשימוש וצורכות זמן למידה של המשתמש, פרויקט זה עוסק בתהליך יישום מערכת קוד מבוסס הניתנת לאימון, שמטרתה לבצע את ההפרדה של השירים למקורות.

מערכות :

* אוטומטיות וקלות לשימוש.
* מבצעות את המשימה במהירות.
* דורשות הרצה חד-פעמית של תהליך אימון למודל.
* קיימת אופציה ללמידה מונחית באמצעות .
* בעלות יכולת להתפתח עם הזמן.

המטרה העיקרית של הפרויקט, מעבר ליישום המערכת, היא להראות שניתן לפתור בעיות מורכבות כמו הפרדות של שירים למקורות באמצעות אלגוריתמי .

במהלך הפרויקט ובדו"ח זה נבחן את ה- , שהוא אלגוריתם ללמידה עמוקה מסוג המיועד לפתירת משימות של הפרדה למקורות במרחב הזמן.

איור 1: Wave-U-Net

ה- *משתמש ב-*  המכיל שירים מקוריים ואת ההפרדות המקוריות שלהם שהוקלטו בערוצים נפרדים ובאמצעותם המודל "לומד" כיצד לבצע את ההפרדות, כל זה קורה בתהליך האימון של המערכת ובסופו מתקבל מודל סופי מבין הרבה שהוא בעל הביצועים הטובים ביותר להפרדות שירים [1].

בתהליך האימון ובניית מודל ה- *השירים מסט האימון עוברים במודל במספר רב של איטרציות כאשר בכל אחת מהן נבחן מודל יחיד ובסופה מוחלט האם הוא טוב יותר או לא לפי ה-* שלוהנותנת את ההפסד הכולל על בחינת שירים מסט הולידציה של ה- .

*כל מודל מכיל שכבות של* שתפקידן לקבל את ה-מהשכבה הקודמת ולבצע *בין המידע בכניסה לשכבה למספר "פילטרים" (נקראים ) ליצירת* שהן המרכיב הכי חשוב של המודל שנבנה ובעזרתן מתבצעים ההפרדות. ו- *בין כל* , סה"כ *שכבות, כדי לחשב* נוספים ברזולוציות זמן שונות.

לאחר בניית המודל המפריד, הוא עובר תהליך של *שבו אנו נרצה לקבל את ערכי המשקלים ב-*  עבורם ה- *של המודל מינימלית וה-* שמחושבים באמצעותם אופטימליים*, נשתמש ב* עם מקדם של כדי להמנע מ- *ולוודא את התכנסות המודל.*

*על מנת לשפר את המודל יותר מתבצע תהליך אימון שני מייד בסיום החלק הראשון הנקרא* שבו מקשיחים את התנאים לאימון המודל ע"י כך שמתבצעים צעדים קטנים יותר בכיוון המינימום של ה- *וממשיכים בביצוע סט איטרציות נוסף של אימון למודל.*

*המודל הסופי הוא התוצר של תהליך האימון המשני ונשמר על המחשב בתיקיית הקוד, מודל זה מכיל את הפרמטרים והמשקלים האופטימליים לביצוע הפרדות שירים למקורות בהתאם ל ובו נשתמש בביצוע הפרדות לשירים.*

*לתוצרי המודל הסופי שהתקבל בסוף התהליך בוצעו מספר בדיקות והערכות לפי מדדי* המתארים את איכות, מובנות וחווית משתמש של תוצרי ההפרדות של המודל, התוצאות נדגמו משירים שהופרדו באופן עצמי מסט האימון של הפרויקט בהשוואה לקבצי המקור. ביצועי המודל שנבחן הראו על תוצאות טובות עם יותר מ- במדד , 70% התאמה במדד , ממוצע של מעל 7.5 במדד ו- מינימלי.

פרויקט זה מציג את יכולותיהם של אלגוריתמי למידה עמוקה ויישומן בפתירת בעיות מורכבות כמו הפרדות של שירים למקורות ביעילות ובהצלחה.

**Executive Summary:**

Nowadays, it is common for personal internet users to use different types of software’s to separate songs to their multi-instrument factors (vocals, bass, drums and other sounds), these software’s require paid subscription for usage and are hard to use and need time to study them. This project suggests a code based platform using trainable Deep Learning methods to accomplish a fully end-to-end song source separation.

Deep learning systems:

* Are automatic and easy to use.
* Accomplishes the task quickly.
* Require a 1 time only training procedure for the model.
* Supervised learning can be applied using labeled dataset.
* Evolves through experience.

The main goal of this project is to show that complex problems such as multi-instrument song separation can be solved using Deep learning algorithms.

During the project we will be using the Wave-U-Net, a multi-scale end-to-end CNN (Convolutional Neural Network) that operates on the time-domain and separates to song to its individual sources.

The Wave-U-Net uses musdb18 dataset that includes original songs with their respective multi-instrument separations for the training procedure of the system model [1].

In the training procedure the files from the dataset are loaded in batches into the Wave-U-Net to create a parametric model of the separator, this procedure keeps going while the system is learning and creating many different models that are being testes and evaluated in accordance to their total loss on song prediction from the validation dataset.

Each model consists on L blocks of convolution layers and down/up sampling operation. CL performs the 1dConvoluion operation between their input and a number of randomly assigned filters (kernels) to compute feature maps for the model, these feature maps are being used by the model to make the song predictions after being optimized. Down/up sampling is performed between each CL, L layers each for 2L total layers, to calculate different feature maps in different resolutions.

The separator model calculated in the current iteration is going through an optimization procedure where we want to minimize the loss function in accordance to the kernel weights, we use ADAM optimizer algorithm for optimization and learning rate factor 0.0001 to avoid over/under fitting of the model.

To further enhance the separator model performance fine tuning is applied after the first set of training, the learning rate is lowered to 0.00001 as we take smaller steps in the minimum direction of the loss function and carry out another set of training.

The final model is the product of the secondary training set and is saved on the PC. We use it to make our song separation prediction because this model holds the optimized parameters and weights for multi-instrument source separation given the musdb18 dataset.

We perform some tests and evaluations on the source estimates final separator model created in the training process, the indices used are SNR and MSE for quality, STOI for intelligibility and MOS for user experience. The results for the calculations of these indices were based on separations from the training set and their references. Our models performance were very good and showed successful results on all indices with over 10 dB on SNR on more than half the songs testes, 70% on most of the songs with STOI, an average score over 7.5 rating on MOS and minimal MSE.

This project showed the capabilities of deep learning algorithms and that they can solve complex problems through training such as multi-instrument song source separation with great results

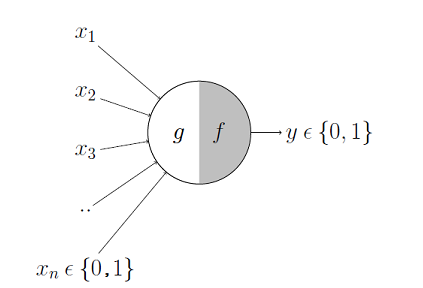
**מילון מונחים:**

1. - הוא תת-תחום בעולם הבינה המלאכותית המתאר אלגוריתמים אשר מסוגלים ללמוד לפתרון בעיות מסוגים שונים בהינתן ידוע מראש.
2. - הוא תת-תחום של המניח שניתן לדמות את פעולת המוח האנושי באמצעות אלגוריתמים לומדים.
3. - משמש המונח לתיאור האלגוריתם העדכני הטוב ביותר בהיבטי ביצועים לפתרון הבעיה הנידונה.
4. – אופרטור וקטורי המתאר את הנגזרת של פונקציה בעלת מספר משתנים.
5. - אלגוריתם לעדכון משקלי הרשת באמצעות ה- של ה- לרשת.
6. - תוצרי שכבת הקונבולוציה שבעזרתן המערכת הלומדת מבצעת פרדיקציות בהתאם לאות המידע.
7. – פרמטר בקונפיגורצית המודל האחראי על ויסות קצת הלמידה של המערכת.
8. – יחס אות לרעש, מדד כמותי שבאמצעותו ניתן לתאר כמה רעש נכנס בתוך קטע האודיו הנדגם.
9. - מדד כמותי המתאר את ממוצע ריבוי השגיאות בין אומד מסויים למה שנאמד.
10. - מדד כמותי לחווית משתמש על בסיס דירוג התוצר.
11. – מדד איכות לרמת המובנות של קטע אודיו.

**מבוא:**

כל שיר מורכב מ- 4 מקורות: זמר/ת, תופים, בס וצלילים אחרים (כמו גיטרות, פסנתר, חצוצרות וכו').  
לרוב הפרדות של שירים למקורות אלו מתבצעות באופן ידני באמצעות תוכנות ייעודיות לעריכת אודיו, חלק מתוכנות אלו דורשות מנוי חודשי לשימוש באפליקציה ויש ללמוד כיצד לעבוד עם הכלים של כל תוכנה כאשר החלק הנותר מפיק תוצאות שאינן איכותיות.

פרויקט זה עוסק בתכנון ויישום מערכת אוטומטית המפרידה שירים למקורות באופן מיידי ואיכותי באמצעות אלגוריתמי , שכיום מיושמים במערכות והתפתחו רבות בשנים האחרונות.

 היא תחום מחקר בעולם המחשבים וה- המתבססת על ההנחה שניתן לדמות באופן ממוחשב את פעולות הנוירונים במוח וכך ליצור מערכות עם היכולת ללמוד ולהשתפר עם הזמן.  
הבסיס המתמטי להנחה זו הוצע ע"י ו- כאשר הם יצרו את ה- באמצעות מודל שפועל באופן דומה (עפ"י מה שהיה ידוע אז) לנוירון במח – נקרא גם .

איור 2:

המודל מקבל בכניסתו סדרה של מידע ומשקלי\0[ם כך שלפי הפונקציה תתקבל תוצאה שהיא '1' או '0', את המשקלים היו מזינים באופן ידני.  
רק יותר מאוחר נבנה ה- ע"י שהיה מעדכן את המשקלים באופן אוטומטי אך לא זכה לתהילה רבה בשל ביצועים שלא עמדו בציפיות.

הנחת יסוד זאת הביאה לפיתוחם של אלגוריתמים נוספים כמו ב- שפותח בשנות ה-60 ע"י אך יושם בפועל החל משנות ה-80 ו- הוצעה לראשונה ע"י .  
אך בתקופה שבין שנות ה-70 לתחילת שנות ה-2000 רווחה אכזבה גדולה מהביצועים של אלגוריתמים אלו שלא עמדו בציפיות גם כן בשל אי עמידה של חומרה ביכולות החישוב וחוסר במשאבים.

החל מאמצע שנות ה-2000, החוקרים ו- הובילו סדרה של ניסויים כאשר תכליתם היה ליצור רשתות 'עמוקות' יותר ויותר.  
סדרת הניסויים שערכו הניבו תוצאות מעולות וזאת בזכות 2 דברים חשובים שקרו במהלך הזמן:

1. תחילת עידן האינטרנט ואיסוף המידע במאגרים וכוננים קשיחים, כמות המידע שהיה ניתן לצבור ולאמן באמצעותו את המערכות גדל אקספוננציאלית.
2. מחשבים ובעיקר חומרה השתפרו מאוד בביצועים ביכולות החישות שלהם, עוצמת החישוב המשתפרת השפיעה באופן ישיר על הצלחות הניסויים.

כך החלו חוקרים רבים ללמוד את התחום וללמד מערכות עמוקות יותר ויותר ומכאן בא המונח .

כיום, עם מעבדי חזקים ונגישים וכמות המידע העצומה שנצברה, הצליחו מערכות לומדות להשתלב במגוון תחומים מרכזיים בחיינו כמו רפואה, זיהוי עצמים ודיבור, הפרדת מקורות, נהיגה אוטונומית, מסחר ועוד.

בפרויקט זה נרצה לפתח מערכת לומדת לפתרון משימת הפרדת מקורות, באמצעות יישום ה- , המערכת תקבל בכניסתה קובץ אודיו המכיל שיר ובמוצאה תפיק 4 קבצים מתאימים למקורות המופרדים של השיר לזמר/ת, לתופים, לבס ולצלילים אחרים כאשר הגישה היא להשתמש באלגוריתמי הפועלים על השיר באופן ישיר.

**מטרת הפרויקט, יעדים ומדדים:**

**מטרת הפרויקט:**

תכנון ויישום מערכת לומדת שתקבל ב- קובץ אודיו, תבצע הפרדה של השיר למקורות ותפיק ב- קובץ מתאים עבור כל מקור מופרד – זמר/ת, תופים, בס, צלילם גבוהים.

**יעדי הפרויקט:**

1. **היעד:** יכולת הפרדה ברזולוציה גבוהה עם מעט רעשים בסיגנל הרצוי, על מנת להבחין באופן ברור שקיימת הפרדה בין השיר למקור הרצוי באופן גורף על פני כל המקורות.

**המדדים:**

* מינימלי של למקורות המשוערכים ב- המערכת.
* מעל 70 אחוז – מעיד על רמת המובנות של הסיגנל ביחס ל- , ככל שהמספר גדל כך הוא יותר מובן.

**השיטה:** חישוב היחס בין ה- של החלק המופרד שנוצר מהפלט של ה- לבין הרעש הוא ה- של התוצר.

עבור שירים עם נבצע חישוב בין השערוכים לבין ה- המתאים לו על פני כל הדגימות: .

לחישוב נשתמש בספריית כדי לקבל את תוצאת החישוב בין כל שיר מופרד למקור.

1. **היעד:** המקורות המשוערכים ב- של המערכת יהיו איכותיים למען המשתמש.

**המדד:** לביצוע מדידה על איכות התוצר נשתמש בשיטת , שיטה זו משתמשת בדירוגים כמותיים שהמשתמש נותן על איכות ההפרדה של התוצר והתוצאה הממוצעת על פני כלל הציונים שהתקבלו עבור כל מקור תייצג את האיכות שלו.

**שיטת ביצוע:** עריכת סקר הכולל השמעת שירים שהופרדו באמצעות המערכת למספר משתתפים שישמעו אותם ויתנו ציון לפי דעתם על כל מקור של שיר בנפרד.

**סקירת ספרות:**

מודלים רבים להפרדת מקורות פועלים על רכיב העוצמה של האות, בדרך כלל זה גורר איבוד מידע של רכיב הפאזה שמקשה על ביצוע הפרדות איכותיות.  
המאמר מציג פתרון לביצוע הפרדת מקורות ( ) תוך שמירה על רכיב הפאזה באמצעות שימוש ב- , שהיא רשת להפרדת מקורות המתייחדת בכך שהיא פועל על האות בציר הזמן ולא דורשת התמרות לציר הספקטרלי.  
על מנת להפיק תוצאות ראשוניות מהמודל אימנו אותו תחת המכיל שירים עם ההפרדות שלהם לביצוע אימון מונחה וערכו השוואה בינם לבין תוצאות של מודל קיים, נמצא מתוצאותייהם שהמודל המוצע בעל ביצועים יותר טובים מה- הנוכחי [1].

בחירת ה- המתאים לאימון מערכות הוא קריטי מאחר ומערכות אלה דורשות כמות חישובים עצומה מאוד כך ש - כבר אינו מספיק.  
 הוא מתמחה לדוקטורט באוניברסיטת וושינגטון שעובד בתחום המחקר של ה- וביצוע אופטימיזציה של חומרה למול תוכנה.  
בסיקור שלו הוא מציג את היתרונות של ה- כרכיב חומרה חשוב ואינטגרלי במערכות , הוא עורך סקירה של מספר דגמים שונים ומשווה ביינהם כעלות למול תועלת.  
החלופות החומרתיות למעבד שנבחר עבור פרויקט זה הסתמך על תוצאות הסיקור וכך נבחרה החלופה המרכזית [2].

מאמר נוסף המיישם את ה- להפרדת הזמר מהשיר לפי המודל מהוצא ב-[1] ומשווה את המודל המאומן על פני למודל באותם תנאים [3].

כחלק מפערי הידע לפרויקט יש לדעת כיצד עובדת ה- שהיא השכבה הבסיסית ביישום רשת ה- להפרדת מקורות.  
המאמר מציג את האג'נדה מאחורי תחום ה- באינטיליגנציה המלאכותית ומספק הסבר מפורט לכיצד עובדות שכבות אלו ברמת האלגוריתם ושימוש ב- לביצוע פרדיקציות בהתאם ל- שעליו התקיים תהליך האימון למשימות כמו זיהוי עצמים, שחזור אודיו, עיבוד תמונה ועוד [4].

תהליך האימון של מערכות הוא בעצם תהליך אופטימיזציה של המודל למול ה- שעליו הוא מבוצע כך שהמודל הסופי יהיה האופטימלי ביותר.  
המאמר מציע אלגוריתם חדש לאופטימיזציה של מסדר ראשון – , אשר יעיל באופן ביצוע החישובים וקל ליישום.  
המאמר מציג את המתמטיקה מאחורי האלגוריתם וכיצד הוא עובד, pseudo קוד ואנליזת התכנסות של האלגוריתם [5].

**ניתוח חלופות**

**חלופה מערכתית:**

פרויקט זה מציע ערכת קוד על גבי עם שלאחר תהליך אימון תבצע הפרדת שירים למקורות באופן אוטומטי ומהיר ה- , כיום בשוק קיימות מערכות דומות וכמו כן גם תוכנות ייעודיות המבצעות את הפרדת השיר למקורות כמו , ו- .

בדומה ל- הן תוכנות לעריכת אודיו שבאמצעותן ניתן לבצע הפרדות של שירים, הן עושות זאת באמצעות שרתים חיצוניים שעובדים עם שמבצע את ההפרדה ולאחר מכן יורד למחשב דרך .  
 הוא כלי להפרדת שירים שזמין להורדה ושימוש באינטרנט.

נסקור את החלופות השונות למימוש הפרויקט וננתח את הבחירה בחלופת הפרויקט.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מדד / חלופה |  |  |  |  | משקל |
| מחיר | דורשת קניה חד פעמית של מחשב עם  4 | דורשת מנוי חודשי/שנתי לשימוש בתוכנה  1 | דורשת מנוי חודשי/שנתי לשימוש בתוכנה  1 | דורשת קניה חד פעמית של מחשב עם  4 | 0.4 |
| מהירות הפרדה | לאחר אימון המודל – ביצוע הפרדה בכמה שניות  5 | שימוש בשרתים חיצוניים להפרדת השיר והורדת התוצרים למחשב לאחר הפרדתם - יכול לקחת זמן רב.  3 | שימוש בשרתים חיצוניים להפרדת השיר והורדת התוצרים למחשב לאחר הפרדתם - יכול לקחת זמן רב.  3 | ביצוע הפרדה בכמה שניות  5 | 0.2 |
| איכות | מצומצם ל- עליו מתבצע תהליך האימון  3 | עם נסיון בתוכנה המשתמש יכול לבצע הפרדות איכותיות  5 | עם נסיון בתוכנה המשתמש יכול לבצע הפרדות איכותיות  5 | מצומצם למודלים המוכנים מראש  1 | 0.3 |
| אופן שימוש | הפעלת תהליך האימון מעט מסורבל אחת להפרדה נדרש רק שורת קוד אחת.  3 | נדרשת למידת התוכנה  4 | נדרשת למידת התוכנה  4 | למידת ה- ואופן שימוש במודלים  4 | 0.1 |
| ציון משוקלל | 3.8 | 2.9 | 2.9 | 3.3 |  |

טבלה 1: ניתוח חלופות מערכתיות

החלופה הנבחרת לפרויקט משקלול סופי היא ה- .

**חלופה טכנולוגית –:**

אלגוריתמי דורשים כמות עצומה של חישובים בתהליך האימון, על מנת לקצר זמן זה באופן משמעותי אינו מספיק וצריך להשתמש גם ב- , שיודע להתמודד עם העברת כמות גבוהה של מידע באופן מהיר.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מדד / חלופה | NVIDIA-RTX2060 | NVIDIA RTX2070 | NVIDIA GTX 1070TI | משקל |
| מחיר [₪] (ציון) | ~1700  (5) | ~2000  (4) | ~2700  (2) | 0.3 |
| RAM [GB] (ציון) | 6  (2) | 8  (4) | 8  (4) | 0.4 |
| base clock [Mhz] (ציון) | 1365  (3) | 1410  (4) | 1607  (5) | 0.1 |
| Num of cores. (ציון) | 1920  (3) | 2304  (4) | 2432  (5) | 0.2 |
| ציון משוקלל | 3.2 | 4 | 3.7 |  |

טבלה 2: ניתוח חלופות טכנולוגיות

חלופה נבחרת: Nvidia RTX2070.

יתרונותיה של החלופה הנבחרת לעומת האחרות בעיקר מתבטאת ביחס עלות מול תועלת מיטבי עבור תכנון והרצת אלגוריתמי אימון למערכת הלמידה בעמוקה, 8GB של זכרון RAM מתאים מאוד למערכות למידה עמוקה מתחילות ובהתאם לגודל הזכרון כך יאפשר להכניס יותר מידע לאימון באיטרציה אחת, כך גם כמות הליבות הקיימות בכל GPU, ובהתאם למחיר של החלופה – Nvidia RTX2070 נבחרה.

**תכן מפורט:**

התכן המפורט יכלול את כל שלבי הפרויקט: בחירת רכיבי החומרה, גרסאות תוכנה לספריות Python מתאימות, תיאוריה בסיסית, תיאור האלגוריתם, יישומו בקוד ואופן השימוש בתוצר הסופי.

**דרישות בסיסיות ליישום הפרוייקט:**

להלן יפורטו הדרישות הבסיסיות עלייהן מושתת הפרויקט כאשר הן מחולקות ל 2 מרכיבים עיקריים: חומרה ותוכנה.

דרישות אלו נועדו על מנת להריץ את ה source code עבור פרויקט זה.

1. דרישות חומרה:
   1. NVIDIA RTX-2070 GPU (מפרט בנספח א').
   2. Intel CPU i5-9400 (מפרט בנספח א').
   3. 16GB RAM.
   4. 1TB HDD.

בחירת רכיבי החומרה לפרויקט הייתה בקפידה רבה מכיוון שצריך לקחת בחשבון כמות עצומה של חישובים מטריציוניים (מכפלות, קונבולוציות, סכימות וכו') כך שבמידה והחומרה אינה מספקת הרצת הקוד יכולה לקחת זמן רב מידיי, מאידך ניתן לקצר זמן זה ע"י בחירה נכונה של חומרה.

1. דרישות תוכנה :
   1. Python – 3.6.8.
   2. Python packages:

יש לוודא התקנה של החבילות קוד הנ"ל על מנת להריץ את הקוד.

* + 1. Numpy - חבילת קוד לתכנות בשפת Python אשר נותן את היכולת לבצע פעולות מתמטיות רב-מימדיות גדולות.
    2. Sacred - חבילת Python המשמשת ככלי לביצוע קונפיגורציה, ארגון והוצאות לוגים מהקוד. שינוי הקונפיגורציה רלוונטי עבור פרוייקט זה.
    3. Tensorflow–gpu - חבילה זו נועדה על מנת לאפשר לקוד לרוץ על ה GPU דרך Python.
    4. Librosa - חבילת Python לאנליזה של אודיו ומוזיקה.
    5. Soundfile - חבילת Python לקריאה/כתיבה של קבצי אודיו.
    6. Lxml - חבילת Python שמאפשרת עבודה יעילה עם קבצי XML, HTML.
    7. Musdb - חבילת Python לתהליך הניתוח של sigsep musdb18 שהוא בעצם ה-dataset של הפרויקט (הסבר על כך בהמשך).
    8. Museval - חבילת Python לביצוע הערכה על שיערוכי מקורות מופרדים.
    9. Google - קישוריות למנוע חיפוש Google.
    10. Protobuf - Protocol buffers בשביל Google.
  1. CUDA 9 for NVIDIA GPU.

סביבת עבודה למעבר הגרפי מאת NVIDIA אשר מאפשרת לפתח ולהריץ תוכנות מחשב על כרטיס ה – GPU, מיועד בעיקר למשימות עיבוד מקבילי מסיבי.

* 1. PyCharm.

סביבת העבודה עליו קוד ה Python בנוי.

ההתקנה של החבילות שונות בגרסאותיהם המתאימות נעשתה באמצעות שהוא כלי עזר המיועד לכך, אפשרות ה- מאפשרת התקנה קלילה של החבילות עפ:י קובץ ה- המוכל בחבילת הקוד לפרויקט זה.

**רשת קונבולוציה:**

רשת קונבולוציה (CNN – Convolutional Neural Network) הינה אלגוריתם של Deep Learning שפותח בשנת 2012 ע"י אלכס קרישבסקי (Alex Krizhevsky) אשר זכה בתחרות ה- ImageNet Large Scale Visual Recognition Challenge, תחרות פתוחה לקהל שבה מפתחים אלגוריתמים שמטרתם לבצע זיהוי עצמים (Image Recognition) בתמונות הלקוחות מ-dataset של ImageNet.

רשת קונבולוציה מוגדרת ככזאת במידה והיא מכילה שכבות קונבולוציה בין השכבות החבויות, ה- , שכבות אלו הן אלו שמגלות את המאפיינים של המידע, הייחודיות שלה היא ביכולת לגלות ("ללמוד") מאפיינים עבור המידע שהיא מקבלת ובאמצעות מאפיינים אלו היא מבצעת פרדיקציות למידע חדש.

רשת זו מכילה 3 סוגים שונים של שכבות:

1. 1d-Convolution Layer.
2. Max Pooling Layer.
3. Fully Connected Layer (FC Layer).

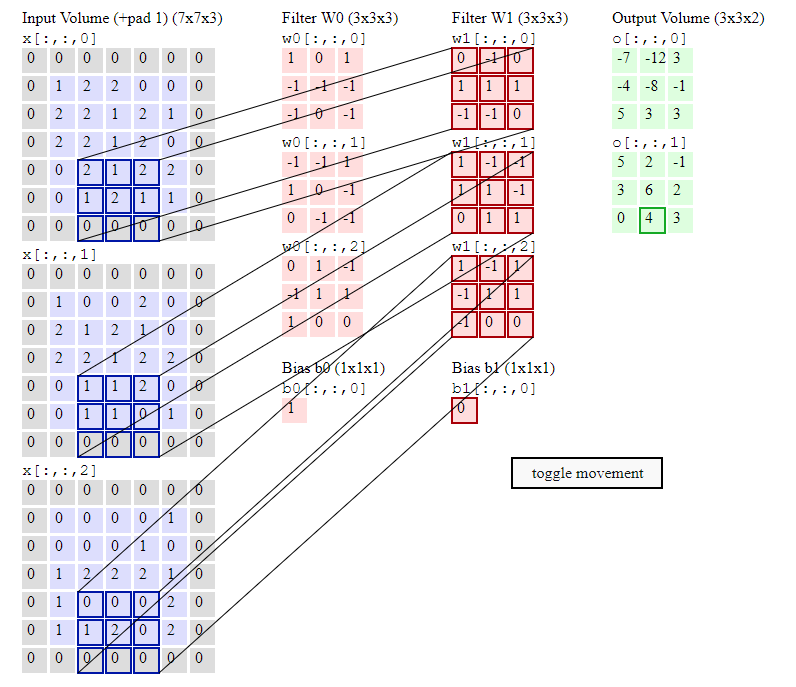
**שכבת הקונבולוציה**:

כשמה היא, מבצעים את פעולות הקונבולוציה בין המידע בכניסה לשכבה לבין מספר מסננים כאשר התוצר מכל פעולת קונבולוציה למסנן אחד נקרא Feature map שזה אחד מן המאפיינים של אות המידע.

פעולת הקונבולוציה עובדת באופן הבא:

נתון שאות המידע בכניסה לשכבה מיוצג כמטריצה שגודלה וגודל המסננים הוא , מבצעים כפל איבר באיבר וסכימה (dot product) בין המסנן לבין חתיכה מאות המידע כגודל המסנן והמספר שיוצא נכנס למטריצת המוצא, מתחילים בראשית מטריצת אות המידע ומתקדמים בצעד אחד ימינה לכל אורך השורה עד שהיא נגמרת ולאחר מכן מתקדמים בצעד אחד למטה וחוזרים על השורה הבאה עד שעוברים על כל מטריצת הכניסה, ניתן גם לעשות יותר מצעד אחד והשינוי נקרא stride.

בסוף, נקבל מטריצה בגודל שהיא ה feature map שנוצרה עבור המסנן, מכיוון שקיימים מספר מסויים של מסננים נוצרים בהתאם מספר זהה של feature maps, כולן מאפיינים שחולצו מאות המידע ומשמשים כ- output לשכבה הבאה ברשת וככל שברשת יש יותר שכבות קונבולוציה כך נקבלת מאפיינים ברזולוציה יותר גבוהה.

דוגמא לפעולת הקונבולוציה בין אות מידע ל-2 מסננים והתוצר - feature map:

איור 3: אופן פעולת ה-

חשוב לציין שבפרויקט שלנו יש רק מטריצת Input Volume אחת (התמונה מתארת אות מידע של תמונה RGB ולכן 3 מטריצות) ולכן הקונבולוציה המתבצעת בשכבת הקונבולוציה הינה חד-מימדית.

**Pooling Layer:**

בדרך כלל שכבה זו נמצאת מיד לאחר שכבת הקונבולוציה ומטרתה להקטין את גודל המימדים של מטריצת ה- feature map במוצא השכבה הקודמת, זאת על מנת להקטין את כוח החישוב הנדרש לעיבוד המידע ובנוסף המאפיינים המחולצים בשכבת הקונבולוציה הבאה יהיו מאפיינים יותר דומיננטיים [4].

אופטימיזציה:

**תהליך האופטימיזציה (תהליך האימון) ברשתות הינו תהליך איטרטיבי שבו בכל איטרציה מתבצע עדכון של משקלי ופרמטרי המודל על מנת לקבל מודל יותר טוב.**

**העדכון של הפרמטרים מתבצע האמצעות מציאת ה- של ה- ולקיחת צעד מאוד קטן בכיוון המינימום של פונקציה זו – נקרא גם .**

**כאשר:**

**- וקטור המשקלים באיטרציה** .

***– קצב הלמידה (***Learning Rate***).***

**– מספר הדגימות.**

**– .**

**באמצעות שיטה זו בכל איטרציה נתקדם יותר ויותר לכיוון המינימום של ה- , משמע, התוצר הסופי לאחר מעבר בכל האיטרציות (ניתן להחלטת המשתמש) יהיה המודל עם הפרמטרים הכי אופטימליים לבעיה אשר נותן את הפרדיקציות הטובות ביותר.**

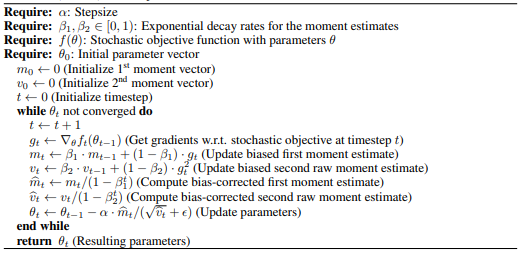
**חשוב לציין שעל קצב הלמידה לא להיות נמוך מספיק שכך תהליך האימון יהיה ארוך מידיי ולא יתכנס למינימום הרצוי (**Under-Fitting**) ושלא יהיה גבוה מידי שכך נפספס נקודות מינימום מקומיות על פונקציית ה-** Loss **(**Over-Fitting**).**

**עבור פרויקט זה נשתמש באופטימייזר** ADAM – Adaptive Moment Estimation**.**

**ADAM Optimizer:**

אופטימייזר ADAM הינו אלגוריתם לאופטימיזציית מסדר ראשון של פונקציות אקראיות [6].

האלגוריתם:

* מיושם בדרך ישירה (פונקציה מובנית של TensorFlow בלולאה).
* יעיל ברמת החישובים וצריכת זיכרון
* מתאים לרשתות בעלי מספר גדול של פרמטרים ומידע.

איור 4:

**MUSDB18 Dataset:**

זהו ה- Dataset שבו נשתמש בפרויקט ועל בסיסו מתרחש אימון המודל, הוא מכיל 150 שירים מלאים מסגנונות שונים עם המקורות המופרדים מהם מראש, והוא חשוב לתהליך האימון כך שלמערכת יהיו "דוגמאות" ללמוד מהן כיצד לבצע את ההפרדה כדי להתאים אותה לצרכים שלנו.

ה- Dataset מכיל 2 תתי-תיקיות, אחת מהן נקראת Train המכילה 100 שירים לאימון המודל, השנייה נקראת Test ומכילה 50 שירים לבחינת המודל.  
בלמידה מונחית יש לבצע את אימון המערכת בשימוש של 2 תתי-התיקיות.

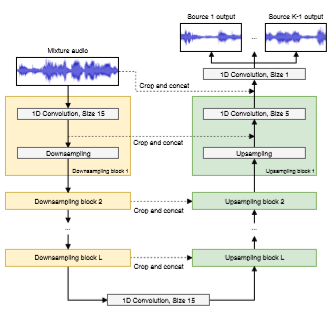
**Wave-U-Net:**

ה- Wave-U-Net הינה מודל לרשת קונבולוציה (CNN – Convolutional Neural Network), המיועד לפתירת משימות של הפרדת מקורות אודיו ופועל ישירות על מקור האודיו במרחב הזמן, מודל זה הוא הבסיס של הפרויקט מכיוון שהוא עונה על מטרתו המרכזית, להפריד שיר למספר מקורות.

המטרה שלנו היא להפריד מיקס ל- מקורות , כאשר לכל , הוא מספר הערוצים ו- הם מספר דגימות האודיו של האות המקורי והמקורות המופרדים בהתאמה [1].

כאשר מעבירים את המידע מה-DataSet ברשת זו (מיקס כלשהו המורכב מארבעת המקורות), היא מחשבת מאפיינים עבור כל אחד מהמקורות ברזולוציות זמן שונות באמצעות בלוקי ה- Downsampling ו- Upsampling במשך שכבות כאשר כל שכבה פועלת בחצי מרזולוציית הזמן מהקודמת לה בהתאמה ובאמצעות מאפיינים אלו היא מבצעת פרדיקציות מתאימות עבור שערוכי המקורות [1].

במוצא המודל אנו נקבל את שערוכי המקורות (Vocals, Bass, Drums, Others) עבור סט הפרמטרים הנוכחי, לאחר מעבר ב ADAM מתבצע עדכון לכל משקלי המודל כדי לשפר את ביצועי ההפרדה.

 ארכיטקטורת המודל בנויה באופן הבא:

איור 5: ארכיטקטורה-

**Mixture Audio:**

משמש כ-input למודל אליו נכנס השיר שאותו אנו רוצים להפריד, בתהליך האימון של המודל עוברים שירים מה- DataSet דרך בלוק זה ומהם מחולצים המאפיינים בשכבות הקונבולוציה שבבלוקים הבאים.

**Down-Sampling Block:**

בלוק זה מורכב משכבת קונבולוציה המלווה בהפחתה של קצב הדגימה במשך שכבות, באמצעות שיטה זו ניתן לחלץ מאפיינים בצירי זמן גסים יותר.  
קצב הדגימה המופחת בכל פעם מזניח כל מאפיין שני ממוצא שכבת הקונבולוציה כדי להקטין פי 2 את רזולוציית הזמן [1].

**Up-Sampling Block:**

בלוק מורכב מהעלאת קצב הדגימה המלווה בשכבת קונבולוציה במשך שכבות נוספות., לכניסה של שכבות הקונבולוציה בבלוקים אלו מצרפים את המאפיינים שחולצו בשכבת הקונבולוציה של בלוק ה- Down-Sampling המתאים לו לפי השכבה (Crop & Concat) כדי לקבל מאפיינים ברזולוציה גבוהה (High Resolution Features) [1].

**Difference Output Layer:**

שכבה זו מחשבת את שערוכי המקור האחרון ע"י הפחתת סך של השערוכים שחושבו מהמידע המקורי - [2].

היתרון של שכבה זו הוא בהגדרה של אשר מונעת פלט לא סביר מהמודל, זאת עלולה להאט את תהליך האימון ולהפחתה בביצועים

**המודל של הפרויקט ותהליך האימון:**

במהלך אימון המודל קבצי אודיו מה- של ה- נדגמות רנדומלית בקבוצות למודל, לאחר ההפרדה המתבצעת ברשת נקבל שיערוכים של 4 מקורות (זמרת, בס, תופים, צלילים גובהים) לכל השירים בקבוצה.  
עלייהם מתבצע חישוב MSE בין המקור המופרד המקורי ב- לבין השיערוך של המודל כחישוב ה-Loss הכולל של המודל.

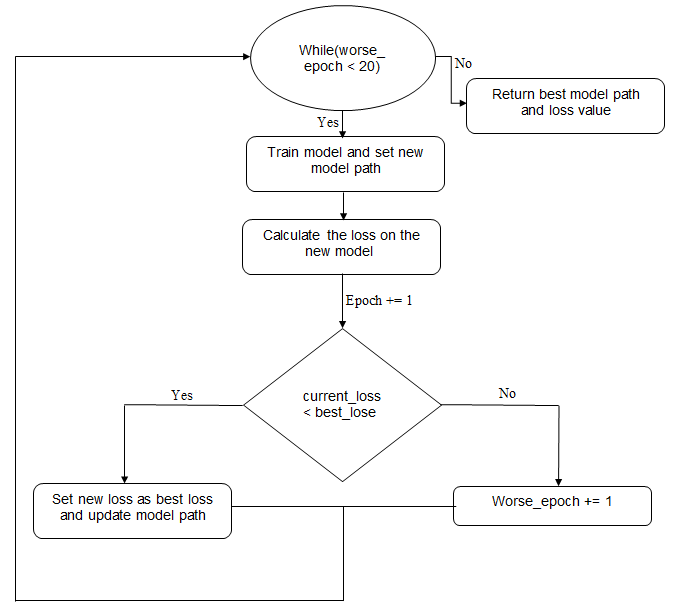
עבור הפרויקט שלנו עפ"י [2] נבחר:

* .
* .
* פילטרים נופים עבור כל שכבת קונבולוציה ().
* גדלי מטריצת הפילטרים עבור בהתאמה.
* .
* פרמטרים ל- ADAM.
* גודלה של קבוצת השירים שנכנסת כל פעם ב- Input למודל, בהתאם ליכולות החומרה ניתן להגדיל מספר זה.
* כל 2000 איטרציות של אימון נגדיר כ- ונדרוש עצירה של האימון לאחר 20 שאין שיפור בולידציה, כלומר, ה- Loss גדל מהמודל הקודם.

בסיום כל האיטרציות נקבל את המודל טוב ביותר ובעל ה-Loss הנמוך ביותר, עליו מתבצע שלב נוסף של אימון הנקרא עבורו:

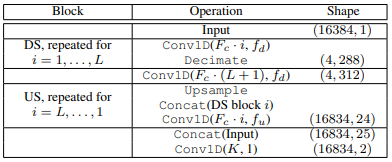
* .
* .

תהליך זה משפר עוד יותר את המודל מאחר והצעדים שעושים בכיוון המינימום אפילו יותר קטנים ומתכנסים כאשר מכניסים יותר מידע (שירים) בקבוצות למודל.

**דיאגרמת בלוקים של האלגוריתם למציאת המודל הטוב ביותר:**

איור 6: דיאגרמת בלוקים לאלגוריתם מציאת המודל

* **While (worse\_epoch < 20)**: נבצע את כל תהליכי האימון ובדיקה בלולאה שבודקת האם חרגנו מהסף המינימלי של איטרציות גרועות אותו הגדרנו ל-20, במידה וחרגנו מסף זה מסתיים תהליך האימון והאלגוריתם יחזיר את הנתיב בו נשמר המודל הטוב ביותר וחישוב ההפסד שלו.
* **Train model and set new model path**: בבלוק זה מתבצע תהליך האימון של המודל, השירים מה- Dataset ב- Training set עוברים בקבוצות דרך ה- Wave-U-Net כדי לייצר את שיערוכי המקורות שלהם באופן הבא [2]:



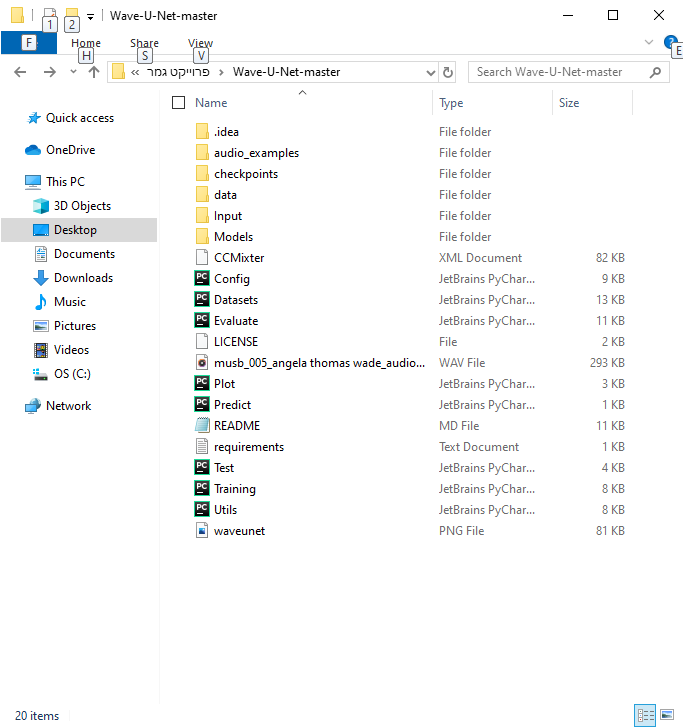
איור 7: דיאגרמת בלוקים מופשטת ל- Wave-U-Net

לאחר מכן מתבצע חישוב של ה- Loss הכולל כפונקציית MSE על פני כל המקורות מופרדים ומתבצע עדכון למשקלי ופרמטרי המודל ע"י אופטימייזר ADAM , במשך 2000 איטרציות.

ניתן לראות את הקוד המיישם תהליך זה תחת נספח ב' – Training.py פונקציית Train().

* **Calculate the loss on the new model**: לאחר 2000 איטרציות של בלוק האימון מתבצע חישוב יותר גס של הפסד המודל מאחר ועורכים בדיקה בנוסף על שירים מה- Test set ב- Dataset שהם שירים שלא עברו בתהליך האימון של המודל, שירים אלו ושירים מה- Training set גם כן עוברים במודל בקבוצות כדי לייצר את שיערוכי המקורות שלהם ועל פנייהם מתבצע חישוב ה- Loss הכולל.
* **Current\_loss < Best\_loss**: מתבצעת בדיקה על ההפסד שחושב בבלוק ה- Test למול ההפסד הטוב ביותר שקיבלנו עד כה בתהליך האימון.  
  אם התנאי לא מתקיים נקדם ב-1 את פרמטר worse\_epoch, המונה איטרציות "גרועות", ונחזור שוב על תהליך האימון, אם התנאי מתקיים נאתחל את worse\_epoch חזרה ל-0 מאחר וקיבלנו שיפור ונבצע עדכון להפסד החדש ושמירת המודל לנתיב המודל הטוב ביותר.

**חבילת הקוד של הפרויקט:**

פרויקט זה מיושם בקוד Python שהיא שפת Object-Oriented ברמה גבוהה, ספציפית בתחום ה deep learning כמות המידע ששפה זו מכילה גדולה מאוד ורוב הפיתוח מתבצע בה.

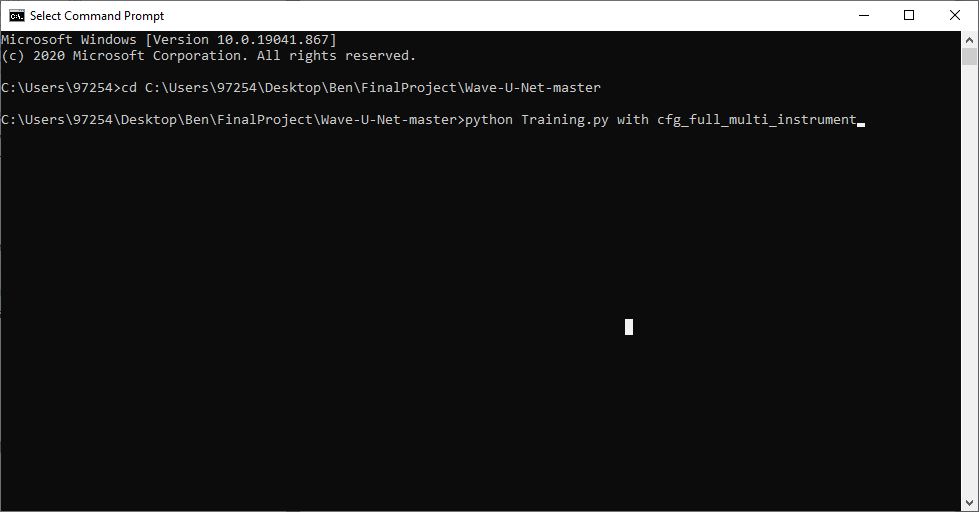
איור 8: חבילת הקוד

חבילת

איור 7: חיבלת הקוד

1. Config.py – קוד זה אחראי על עריכת קונפיגורציית המערכת, בתוכו הוא מגדיר מבנה בעל פרמטרים קבועים של המודל ומעדכן פרמטרים אחרים בהתאם למשימה.  
   עפ"י הקונפיגורציה הנבחרת המבנה מתעכן ועובר לתוכנית המרכזית כפרמטר.
2. Datasets.py – קוד האחראי על תהליך ה- pre-preparing של ה- dataset.  
   קיימות בו פונקציות עזר אשר תומכות בטעינת השירים ובהכנתם לכניסת רשת ה- Wave-U-Net בקוד.
3. Training.py – זהו הקוד המרכזי של הפרויקט בו מתבצעת מציאת המודל הטוב ביתר בעל ה-Loss הנמוך ביותר, הקוד זה מיושמת דיאגרמת הבלוקים שהוזכרה לעיל.
4. Utils.py – קוד המכיל פונקציות עזר בהן נשתמש במהלך הקוד על מנת להקל על סרבול הקוד ויישום של אותה משימה במקומות שונים בתוכנית.
5. Test.py – קוד זה מוכל ב-Traning.py ומיישם את בלוק ב- Calculate loss on the new model, הוא מחשב את ה-Loss של המודל עפ"י שירים ב- של ה- Dataset בנוסף על שירים מה- .
6. UnetAudioSeparator.py – קוד המיישם את ה- Wave-U-Net [1] ואחראי על תהליך בניית המודל ומפיק במוצאו את שיערוכי המקורות.
7. Evaluate.py - קוד זה אחראי על חישוב השיערוכים של שיר עבור מודל מסוים ובנוסף לחישוב השערוכים של שירים מה- Dataset כחלק מתהליך האימון.
8. Predict.py – באמצעות קוד זה נבצע הפרדה של שירים באמצעות מודל נתון, לאחר מציאת המודל הטוב ביותר נשתמש במודל זה על predict.py על מנת להפיק 4 קבצים של המקורות המופרדים.
9. OutputLayer.py – קוד האחראי על יישום ה-Output Layer [1], שהוא חישוב שיערוך המקור ע"י החסרת סך המקורות המשוערכים מהמיקס השלם.
10. Checkpoints – התיקייה אליה נשמרים כל המודלים המיוצרים במהלך תהליך האימון.

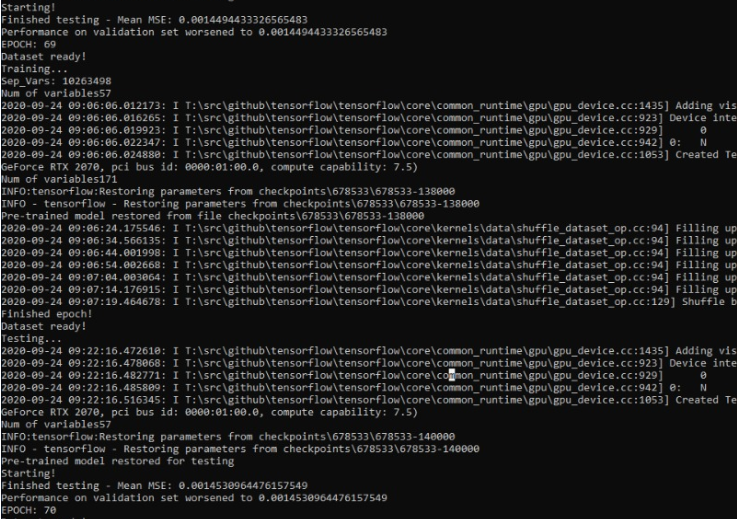
**הרצת תהליך האימון ושימוש במודל:**

ניתן להפעיל תוכניות דרך ה- במחשב, נבחר את הנתיב בו נמצאת חבילת הפרוייקט וכדי להריץ את תהליך האימון למערכת נשתמש בפקודה :

איור 9: הרצת התוכנית

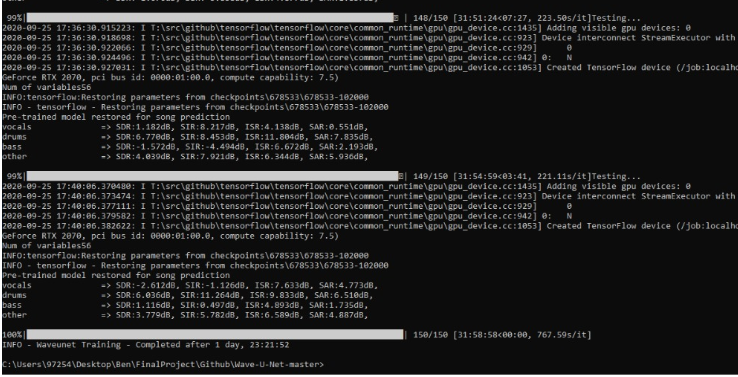
הפקודה מריצה את התוכנית תחת ערכי הקונפיגורציה להפרדת מקורות ב , המתחילה את תהליך האימון של ה- למציאת המודל הטוב ביותר להפרדת מקורות עם ה- הנתון כפי שמתואר בדיאגרמת הבלוקים לעיל.

חשוב לציין, בהרצה הראשונית של התוכנית מתבצע גם תהליך של של ה- כפי שמתואר ב-[1] ולא מתבצע עיבוד של השירים כלל בתהליך.



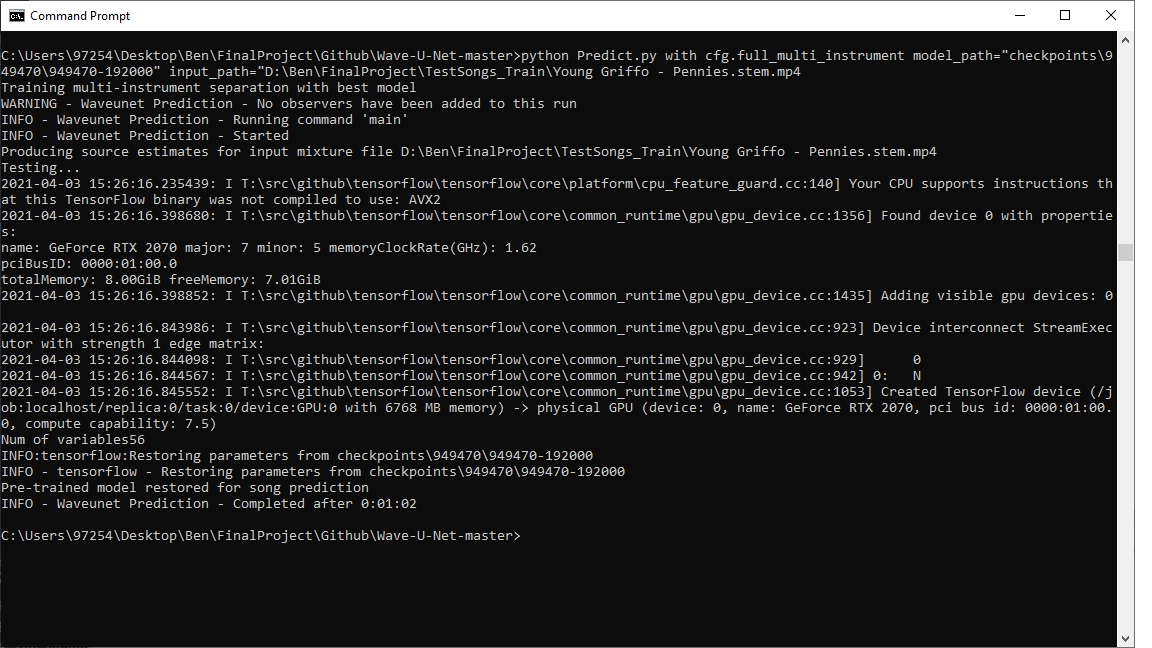
איור 10: מחזור epoch שלם

בכל איטרציה בה נבנה מודל פרמטרי חדש להפרדת מקורות מתבצע תהליך ה ולאחר מכן תהליך ה , כפי שהוסבר בתכן המפורט, שבסופו יוחלט האם האיטרציה הייתה טובה יותר וקיים שיפור בביצועי המודל הנוכחי במידה וקיים שיפור המודל נשמר בפרמטר המכיל את מיקומו במחשב.

בסוף התהליך מתבצעת אבלואציה של המודל הסופי על פני כל השירים מה- ובסיומה מודפסת הודעת סיום האימון ומשך זמן כולל.

איור 11:בחינת אבלואציה סופית למודלים

כדי להשתמש במודל הסופי לאחר תהליך האימון נשתמש בפקודה , הפקודה צריכה בנוסף לקבל את הנתיב למודל שבה נשתמש להפרדה , ואת הנתיב לשיר שאותו רוצים להפריד , במידה ולא נציין את הנתיב לתוצרים הם יווצרו תחת ה- כברירת מחדל.



איור 12: ביצוע הפרדה ב- CMD

**התוצר:**

התוצר הסופי של פרויקט זה הינו מודל מאומן להפרדת שיר למקורות מסוג , המודל שאומן מכיל את הפרמטרים האופטמליים לביצוע הפרדות שירים בהתאם ל הנתון ויפריד כל שיר שיקבל ל-4 מקורות שונים תוך זמן קצר מאוד ובאמצעות פקודה אחת בלבד.

בנוסף, ערכת הקוד של הפרויקט מכילה את כל הקבצים ההכרחיים לביצוע אימון למודלים חדשים במידה ומתקיים שינויי ב- כמו החלפת סוגי השירים, הוספה של שירים וכו', כך שניתן ליצור מודלים נוספים שמותאמים לסוגי שירים שונים.

כל המודלים הנוצרים בתהליך האימון ניתנים לשמירה ולשימוש חוזר, אינם חד-פעמיים ואין צורך לבצע תהליך אימון נוסף.

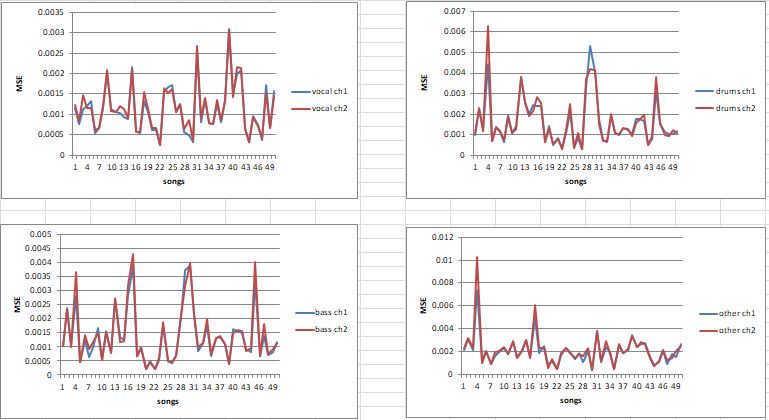
**בדיקות והערכה:**

**:**

לאחר הרצת הקוד וקבלת המודל הטוב ביותר נשתמש בו כדי להפריד שירים מה-, מכיוון שקיימים ב- Dataset קבצים עם המקורות מופרדים מראש ניתן להשתמש בהם כ- לשיערוכי המקורות שנקבל מהמודל ונבצע בינהם חישוב ב אופן הבא:

.

נדגום כל אחד מן השירים בקצב דגימה אחיד ונחסיר בין דגימות תואמות של המקור ושל ה-, מכיוון שזה חישוב נעלה את ההפרש בריבוע ונסכום את כל התוצאות ונחלק בסך כל הדגימות כדי לקבל את ממוצע של השגיאה.

בוצעו חישוב למדגם של 50 שירים מתקיית ה- של ה- , התוצר הוא חישוב ב- 2 ערוצים של mono ו-stereo ל-, ככל שערך זה נמוך יותר כך קיים יותר דמיון בין השערוך למקור.

(2)

(1)

גרף 1-4: חישובי MSE לכל סיגנל בהפרדות שירים מה- DataSet

(4)

(3)

ניתן להסיק מתוצאות אלו שההפרדות שנעשו עבור השירים מה- Dataset היו קרובות מאוד למקוריות עבור כל אחד מן המקורות Vocals, drums, bass, others.  
רוב התוצאות דומות עד כדי פקטור של , אמנם קיימים שירים מסויימים בהם ההפרדה פחות טובה עד כדי פאטקור של .

**:**

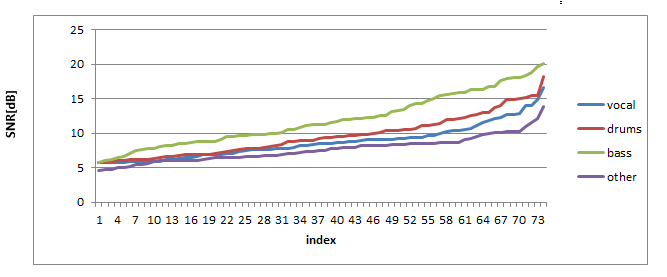
חישוב ה- נועד לתאר את היחס בין הסיגנל הרצוי בשיר המופרד שלנו לעומת כמות הרעש בו. בדומה לחישוב ה- , נבצע הפרדה לשירים מה- בתקיית ה- באמצעות המודל הסופי של הפרוייקט, לשירים אלה גם כן קיימים קבצים מופרדים ללא רעשים שהוקלטו בנפרד המשמשים כ- ויעזרו לנו בחישוב ה- .

כדי לבצע חישוב זה נשתמש בחישוב ה- של הרעש והחלק המופרד.  
בשביל הרעש – נדגום את השיר המופרד ואת ה- ונחסר ביינהם, כך נקבל רק רעש.

לאחר מכן נבצע חישוב עבור הרעש וה- בנפרד באמצעות.

*נחלק בין החישוב של ה- לחישוב של הרעש ונקבל את ה- .*

החישובים בוצעו על כל השירים בהם המודל נעזר בתהליך האימון שלו הנמצאים תחת תקיית של ה-



גרף 5: חישוב להפרדות שירים מה- .

מהגרף ניתן להסיק:

* מרבית השירים הם בעלי של יותר מ- , בשירים אלו ניתן לשמוע בבירור את החלק המופרד אך עדיין ניתן להבחין ברעש בחלקים מסויימים.
* בשירים בעלי ומטה הרעש יותר עוצמתי ויותר קשה להחבין בחלק המופרד.
* להפרדת ה- יש את התוצאות הכי טובות בהיבט ומגיע למקסימום של .
* להפרדת ה- יש את התוצאות הפחות טובות מבין שאר ההפרדות.

**:**

לשם מציאת מדד נערך סקר בין קבוצה בעלת 20 משתתפים פורמט לסקר בנספח ג'), המשתתפים התבקשו לשמוע 5 שירים ואת ההפרדות שלהם למשך כחצי דקה ולדרג את איכות ההפרדה לפי דעתם, השירים שנבחרו לסקר אינם חלק מה- ולכל משתתף הושמעו אותם שירים לאחידות התוצאות.

לאחר שכלל הדירוגים התקבלנו נחשב את ממוצע הדירוגים של כל מקור בשיר עבור כלל המשתתפים, כך נקבל את הדירוג הממוצע עבור כל שיר למקור מסויים ונוכל לבצע חישוב נוסף לממוצע הדירוגים של 5 השירים כדי לקבל את הדירוג הסופי להפרדה של כל המקור בביצועי המודל.

לאחר קבלת הדירוגים מכלל המשתתפים התוצאות נסכמו באופן הבא:

תוצאות המדד מראות על תוצאות גבוהות של שכלול ממוצע הדירוגים, לפי החישוב , ציון משוכלל למודל - .

*לפי חוות דעת של המשתתפים מעבר לרעשי הרקע בחלקים שקטים להפרדה כאשר היא קיימת ומופרדת שומעים אותה באופן ברור לעומת שאר הכלים.*

**:**

**מדד הינו מדד איכות אשר נותן אינדיקציה מספרית באחוזים לרמת המובנות של שיר מסויים ביחס ל- קיים, ניתן לקבל תוצאה למדד זה באמצעות ספריית שבה קיימת הפונקציה לחישוב הפרמטר .**

**הפונקציה מקבלת בכניסתה את החלק המופרד בשיר שהתקבל בפלט מהמודל הסופי, את החלק המופרד בשיר המקורי מה- וקצב הדגימה שבה נדגמו השירים ומחזירה ערך באחוזים כך שערך גבוה יותר מצביע על מובנות גבוה יותר של הההפרדה.**

**התוצאות של מדד זה עבור המודל הסופי חושבו על כל השירים מתקיית של ה-**  ולכל הפרדה בנפרד (), השירים נדגמו בקצב דגימה של בערוץ יחיד בכניסתם לפונקציה, עבור שירים ארוכים החישוב התבצע על פני קטע של 3 דקות.

**נבחן את התוצאות על 20 שירים המובילים באחוזי התאמה למדד :**

גרף 2: תוצאות מדד STOI

**מהגרף ניתן להסיק:**

* **כל ההפרדות של השירים מעל**  **התאמה במדד** .
* **הפרדת ה-**  **היא בעלת התוצאות הטובות ביותר ובה התוצאה הכי טובה למובנות ההפרדה ביחס למקור היא** .
* **כל ההפרדות של**  **גם כן מעל ל-**  **התאמה במדד, המראה על שהמדד אחיד בין כל ההפרדות.**

**סיכום ומסקנות:**

לסיכום, פרויקט זה עסק ביישום מערכת תוכנה מותקנת על *המשתמשת באלגוריתמים של*  לפתרון משימת הפרדת שירים למקורות של *זמר/ת, תופים, בס וצלילים אחרים. לשם כך* השתמשנו ביישום ה- *, שהוא אלגוריתם ללמידה עמוקה מסוג המיועד להפרדת מקורות מקצה לקצה בציר הזמן.*

*תחילה ביצענו תהליך אימון מורכב בעל 2 חלקים (אימון ראשוני ו-* ) *שבו העברנו מידע מקוטלג (שירים וההפרדות שלהם) דרך הרשת שלפיו המערכת למדה להבחין בין הרכיבים השונים של השיר ולבצע הפרדות באמצעות בניית מודל פרמטרי המסתמך על נתונים שנלקחים משירים אלו ובסוף התהליך התקבל מודל סופי מבין רבים שנבחנו שהוא בעל הביצועים הטובים ביותר בהפרדות השירים.*

באמצעות המודל הסופי שנוצר בוצעו הפרדות של שירים מה- ומיוטיוב בשביל מספר בדיקות שמטרתן לבחון את ביצועיו בקוד שצורף לקוד (נספח ב' - ):

* *מדד*  *המתאר את הפרש השגיאה הממוצעת בין ההפרדה למקור נבחן על 50 שירים מסט ה-*  *הראה על תוצאות טובות עם שגיאה ממוצעת של עד .*
* *מדד* לחישוב יחס אות לרעש נבחן על 75 שירים מסט ה- *לא היה מדד אופטימלי לבדיקה אך הראה תוצאות טובות של ומעלה ברוב המקרים.*
* *מדד*  המתאר את רמת המובנות של ההפרדה ביחד למקור נבחן על 20 שירים המובילים מסט ה- *שיקפו ברובן תוצאות מעולות של מעל התאמה.*

*מדד* המתאר חווית משתמש נבחן על 5 שירים חדשים שהופרדו ע"י המודל ודורגו ע"י 20 משתתפים שהעבירו את חוות דעתם על איכות ההפרדה עבור כל מקור מופרד, לפי המשתתפים מעבר לרעשי רקע ההפרדה נשמעת בבירור והציון הכללי למודל שהתקבל הוא .

לפי תוצאות המדדים ניתן לראות שביצועי המודל שנוצר טובים מאוד בהיבטי איכות ומובנות על פני כל ארבעת ההפרדות של השירים (זמר/ת, בס, תופים, צלילים אחרים), סה"כ בוצעו כ- 150 הפרדות של שירים באמצעות המודל שנבחן.

ניתן לקבל מודל טוב יותר בהתאם לחומרת המחשב ושינויי הקונפיגורציה של הקוד, בפרויקט זה נעשה שימוש בחומרה מינימלית לאימון המערכת לפי דרישות המערכת להרצת אלגוריתמי למידה עמוקה ובוצעו בעקבות כך שינויי קונפיגורציה, לכן אפשר להסיק שביצועי המודל הינם בהתאם.

**הצעות לעבודת המשך:**

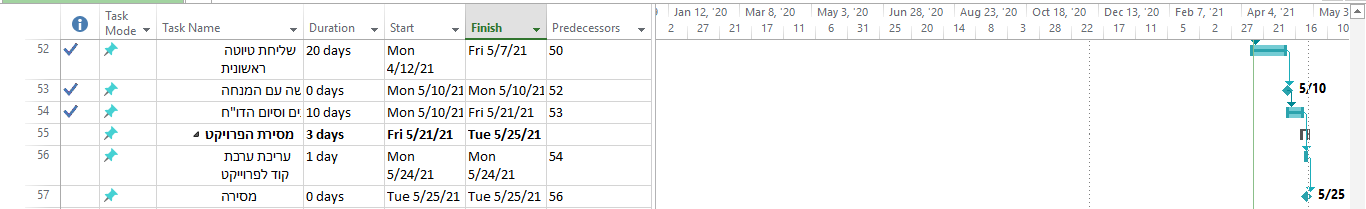
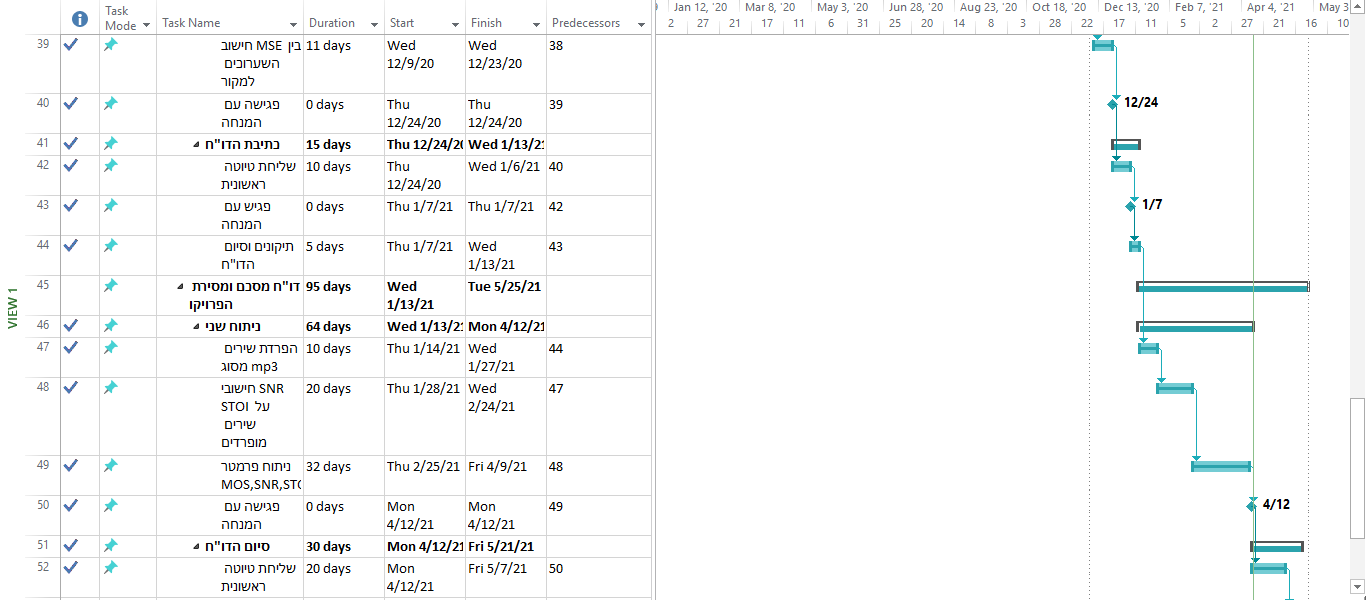
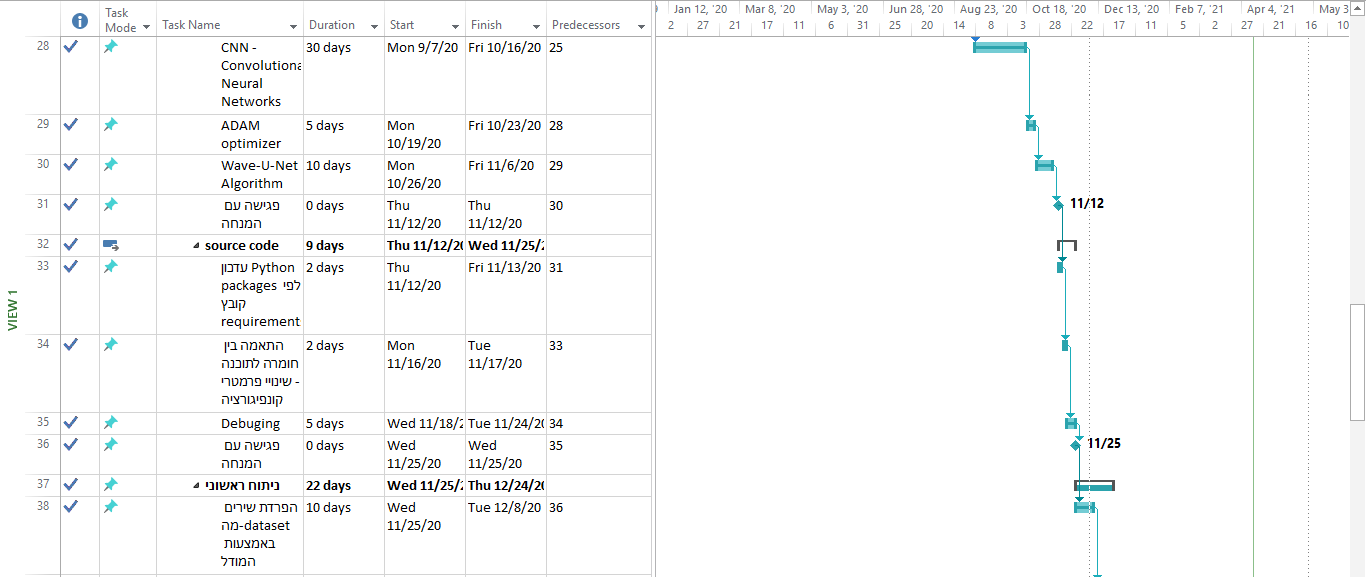
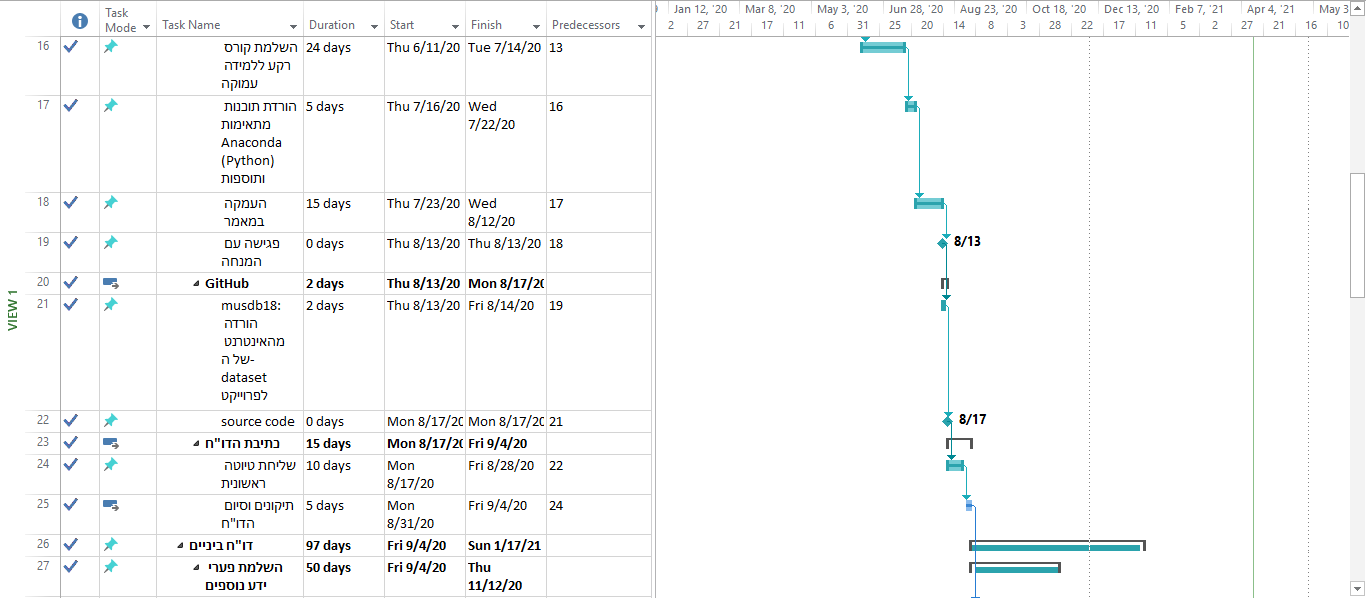
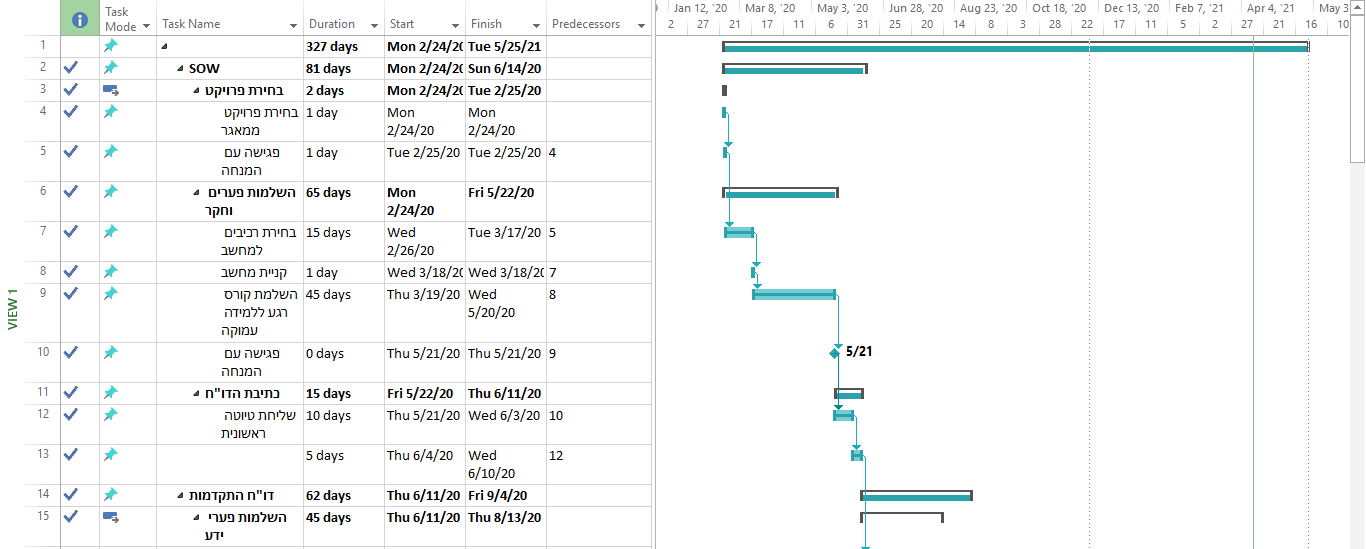
כדי להמשיך ולפתח את תוצר הפרויקט ולנסות לקבל מערכת שמפרידה שירים עם ביצועים גבוהים יותר ניתן לעשות 2 דברים:

1. ניתן לבצע שינויים של חומרה ובהתאם לשנות את הקונפיגורציה.  
   החלפה של רכיבי חומרה ברכיבים חזקים יותר אשר מגדילים את ביצועי המחשב מאפשרת לבצע שנויים של קונפיגורציית המערכת ב- וכך לשפר את זמן חישוב המודל הסופי בתהליך האימון ואת ביצועיו, הגדלת ה- תאפשר ליותר מידע לעבור בו-זמנית במודל, הגדלת כמות הליבות שפועלות ב- במידה וקיימות יותר באמצעות פרמטר .
2. ניתן להרחיב את כמות השירים הנמצאים ה- , ככל שיש למערכת יותר מידע ללמוד ממנו בתהליך האימון שלה כך ביצועי המודל הסופי יגדלו בהתאם.  
   תהליך האימון יהיה ארוך יותר לפי כמות השירים שנוספו ל- אך המודל הסופי כאמור יהיה בעל ביצועים טובים יותר.

עבור כל אחד מן השינויים שהוצעו יתקבל מודל שונה מן המודל הסופי שקיבלנו בפרויקט זה וניתן להשוות בינהם באמצעות חישובי , , ו- .

**תכנון הפרויקט, ריכוז שינויים וניהול סיכונים:**

1. **תוכנית עבודה סופית – תרשים גאנט:**



1. **ריכוז שינויים:**

* נוסף מדד מובנות () לבחינת ביצוע הפרוייקט.
* עדכון קריטריון ליחס אות לרעש .

1. **ניהול סיכונים:**

* כמות המידע המועברת באלגוריתם גדולה מידיי לכדי שה- GPU יוכל להתמודד איתה ולכן ייקח זמן רב לאמן את המערכת, כדי להתמודד עם סיכון זה נבחר רכיב GPU מומלץ להרצת הקוד כך שהסיכון יורד משמעותית.
* השלמת פערים – לימוד של חומר חדש באופן עצמאי והבנתו לעומק, ניתן סיוע מאנשים העוסקים בנושא ובנוסף קורס אינטרטני המלמד את הבסיס לתכנות Deep Learning ב- Python.

במהלך הפרויקט לא היו קשיים בסיכונים העיקריים שפורטו.

**רשימת מקורות:**

* [1] Daniel Stoller, Sebastian Ewert, Simon Dixon, Wave-U-Net: a multi-scale neural network for end-to-end audio source separation, submitted at 2018-06-08.
* [2] Tim Dettmers, Which GPU(s) to get for deep learning: My experience and advice for using GPU’s in deep learning,

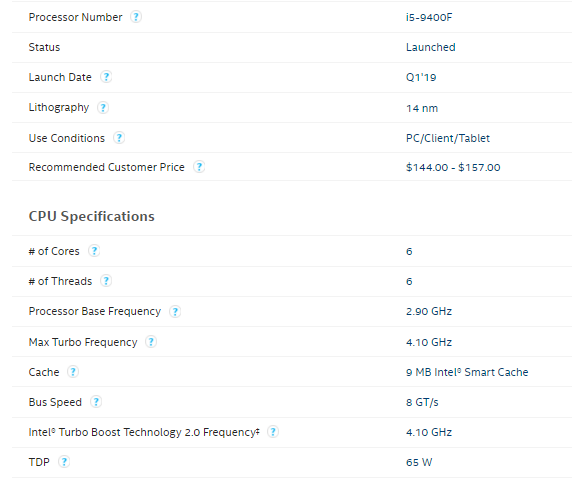
<https://timdettmers.com/2019/04/03/which-gpu-for-deep-learning/>, published at 2019-04-03.

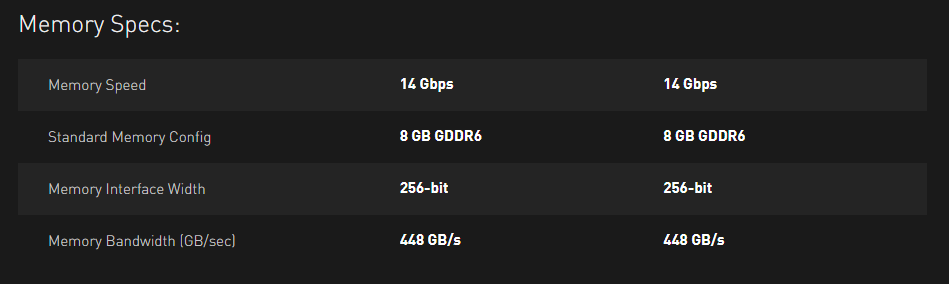
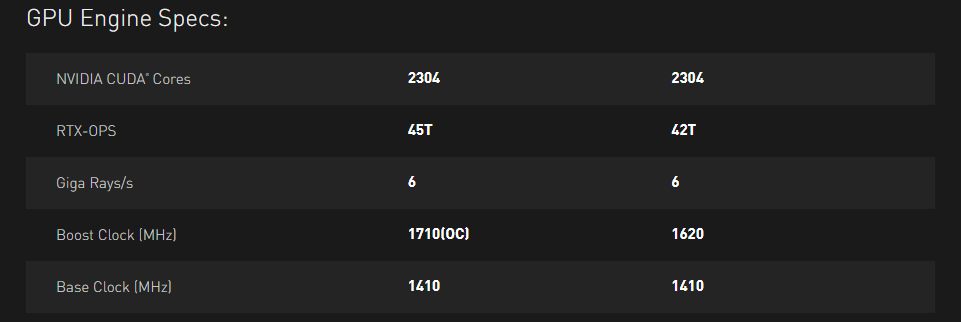
* [3] Craig Macartny, Tillman Weyde, Improved Speech Enhancment with the Wave-U-Net, submitted at 2018-11-27.
* [4] Sumit Saha, A Comprehensive Guide to Concolutional Neural Networks – the ELI5 way,

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, submitted at 2018-12-15.

* [5] Diederik P.Kingma, Jimmy Lei Ba, ADAM: a method for stochastic optimization, Published as a conference paper at ICLR 2015.

**נספחים:**

**נספח א' – מפרט מערכת CPU + GPU:**



**נספח ב' – Python Code:**

**Config.py:**

import numpy as np  
from sacred import Ingredient  
  
config\_ingredient = Ingredient("cfg")  
  
@config\_ingredient.config  
def cfg():   
 model\_config = {"musdb\_path" : r"C:\Users\97254\Desktop\Ben\FinalProject\Github\train-test-dataset",  
 "estimates\_path" : r"C:\Users\97254\Desktop\Ben\FinalProject\Github\estimated\_path",  
 "data\_path" : "data",  
 "model\_base\_dir" : "checkpoints",  
 "log\_dir" : "logs",  
 "batch\_size" : 8,   
 "init\_sup\_sep\_lr" : 1e-4,  
 "epoch\_it" : 2000,  
 'cache\_size': 4000,  
 'num\_workers' : 4,   
 "num\_snippets\_per\_track" : 100,  
 'num\_layers' : 12,  
 'filter\_size' : 15,  
 'merge\_filter\_size' : 5,  
 'input\_filter\_size' : 15,  
 'output\_filter\_size': 1,  
 'num\_initial\_filters' : 24,  
 "num\_frames": 16384,  
 'expected\_sr': 22050,  
 'mono\_downmix': True,  
 'output\_type' : 'direct',  
 'output\_activation' : 'tanh',  
 'context' : False,  
 'network' : 'unet',  
 'upsampling' : 'linear',  
 'task' : 'voice',  
 'augmentation' : True,  
 'raw\_audio\_loss' : True,  
 'worse\_epochs' : 20,  
 }  
 experiment\_id = np.random.randint(0,1000000)  
 # Set output sources  
 if model\_config["task"] == "multi\_instrument":  
 model\_config["source\_names"] = ["bass", "drums", "other", "vocals"]  
 elif model\_config["task"] == "voice":  
 model\_config["source\_names"] = ["accompaniment", "vocals"]  
 else:  
 raise NotImplementedError  
 model\_config["num\_sources"] = len(model\_config["source\_names"])  
 model\_config["num\_channels"] = 1 if model\_config["mono\_downmix"] else 2

def full\_multi\_instrument():  
 print("Training multi-instrument separation with best model")  
 model\_config = {  
 "output\_type": "difference",  
 "context": True,  
 "upsampling": "linear",  
 "mono\_downmix": False,  
 "task" : "multi\_instrument"  
 }

**Training.py:**

from sacred import Experiment  
from Config import config\_ingredient  
import tensorflow as tf  
import numpy as np  
import os  
  
import Datasets  
import Utils  
import Models.UnetSpectrogramSeparator  
import Models.UnetAudioSeparator  
import Test  
import Evaluate  
  
import functools  
from tensorflow.contrib.signal.python.ops import window\_ops  
  
ex = Experiment('Waveunet Training', ingredients=[config\_ingredient])  
  
config = tf.ConfigProto()  
config.gpu\_options.allow\_growth = True  
sess = tf.Session(config=config)  
@ex.config  
# Executed for training, sets the seed value to the Sacred config so that Sacred fixes the Python and Numpy RNG to the same state everytime.  
def set\_seed():  
 seed = 1337  
  
@config\_ingredient.capture  
def train(model\_config, experiment\_id, load\_model=None):  
 # Determine input and output shapes  
 disc\_input\_shape = [model\_config["batch\_size"], model\_config["num\_frames"], 0] # Shape of input  
 if model\_config["network"] == "unet":  
 separator\_class = Models.UnetAudioSeparator.UnetAudioSeparator(model\_config)  
 elif model\_config["network"] == "unet\_spectrogram":  
 separator\_class = Models.UnetSpectrogramSeparator.UnetSpectrogramSeparator(model\_config)  
 else:  
 raise NotImplementedError  
  
 sep\_input\_shape, sep\_output\_shape = separator\_class.get\_padding(np.array(disc\_input\_shape))  
 separator\_func = separator\_class.get\_output  
  
 # Placeholders and input normalisation  
 dataset = Datasets.get\_dataset(model\_config, sep\_input\_shape, sep\_output\_shape, partition="train")  
 iterator = dataset.make\_one\_shot\_iterator()  
 batch = iterator.get\_next()  
  
 print("Training...")  
  
 # BUILD MODELS  
 # Separator  
 separator\_sources = separator\_func(batch["mix"], True, not model\_config["raw\_audio\_loss"], reuse=False) # Sources are output in order [acc, voice] for voice separation, [bass, drums, other, vocals] for multi-instrument separation  
  
 # Supervised objective: MSE for raw audio, MAE for magnitude space (Jansson U-Net)  
 separator\_loss = 0  
 for key in model\_config["source\_names"]:  
 real\_source = batch[key]  
 sep\_source = separator\_sources[key]  
  
 if model\_config["network"] == "unet\_spectrogram" and not model\_config["raw\_audio\_loss"]:  
 window = functools.partial(window\_ops.hann\_window, periodic=True)  
 stfts = tf.contrib.signal.stft(tf.squeeze(real\_source, 2), frame\_length=1024, frame\_step=768,  
 fft\_length=1024, window\_fn=window)  
 real\_mag = tf.abs(stfts)  
 separator\_loss += tf.reduce\_mean(tf.abs(real\_mag - sep\_source))  
 else:  
 separator\_loss += tf.reduce\_mean(tf.square(real\_source - sep\_source))  
 separator\_loss = separator\_loss / float(model\_config["num\_sources"]) # Normalise by number of sources  
  
 # TRAINING CONTROL VARIABLES  
 global\_step = tf.get\_variable('global\_step', [], initializer=tf.constant\_initializer(0), trainable=False, dtype=tf.int64)  
 increment\_global\_step = tf.assign(global\_step, global\_step + 1)  
  
 # Set up optimizers  
 separator\_vars = Utils.getTrainableVariables("separator")  
 print("Sep\_Vars: " + str(Utils.getNumParams(separator\_vars)))  
 print("Num of variables" + str(len(tf.global\_variables())))  
  
 update\_ops = tf.get\_collection(tf.GraphKeys.UPDATE\_OPS)  
 with tf.control\_dependencies(update\_ops):  
 with tf.variable\_scope("separator\_solver"):  
 separator\_solver = tf.train.AdamOptimizer(learning\_rate=model\_config["init\_sup\_sep\_lr"]).minimize(separator\_loss, var\_list=separator\_vars)  
  
 # SUMMARIES  
 tf.summary.scalar("sep\_loss", separator\_loss, collections=["sup"])  
 sup\_summaries = tf.summary.merge\_all(key='sup')  
  
 # Start session and queue input threads  
 sess.run(tf.global\_variables\_initializer())  
 writer = tf.summary.FileWriter(model\_config["log\_dir"] + os.path.sep + str(experiment\_id),graph=sess.graph)  
  
 # CHECKPOINTING  
 # Load pretrained model to continue training, if we are supposed to  
 if load\_model != None:  
 restorer = tf.train.Saver(tf.global\_variables(), write\_version=tf.train.SaverDef.V2)  
 print("Num of variables" + str(len(tf.global\_variables())))  
 restorer.restore(sess, load\_model)  
 print('Pre-trained model restored from file ' + load\_model)  
  
 saver = tf.train.Saver(tf.global\_variables(), write\_version=tf.train.SaverDef.V2)  
  
 # Start training loop  
 \_global\_step = sess.run(global\_step)  
 \_init\_step = \_global\_step  
 for \_ in range(model\_config["epoch\_it"]):  
 # TRAIN SEPARATOR  
 \_, \_sup\_summaries = sess.run([separator\_solver, sup\_summaries])  
 writer.add\_summary(\_sup\_summaries, global\_step=\_global\_step)  
  
 # Increment step counter, check if maximum iterations per epoch is achieved and stop in that case  
 \_global\_step = sess.run(increment\_global\_step)  
  
 # Epoch finished - Save model  
 print("Finished epoch!")  
 save\_path = saver.save(sess, model\_config["model\_base\_dir"] + os.path.sep + str(experiment\_id) + os.path.sep + str(experiment\_id), global\_step=int(\_global\_step))  
  
 # Close session, clear computational graph  
 writer.flush()  
 writer.close()  
 sess.close()  
 tf.reset\_default\_graph()  
  
 return save\_path  
  
@config\_ingredient.capture  
def optimise(model\_config, experiment\_id):  
 epoch = 0  
 best\_loss = 10000  
 model\_path = None  
 best\_model\_path = None  
 for i in range(2):  
 worse\_epochs = 0  
 if i==1:  
 print("Finished first round of training, now entering fine-tuning stage")  
 model\_config["batch\_size"] \*= 2  
 model\_config["init\_sup\_sep\_lr"] = 1e-5  
 while worse\_epochs < model\_config["worse\_epochs"]: # Early stopping on validation set after a few epochs  
 print("EPOCH: " + str(epoch))  
 model\_path = train(load\_model=model\_path)  
 curr\_loss = Test.test(model\_config, model\_folder=str(experiment\_id), partition="valid", load\_model=model\_path)  
 epoch += 1  
 if curr\_loss < best\_loss:  
 worse\_epochs = 0  
 print("Performance on validation set improved from " + str(best\_loss) + " to " + str(curr\_loss))  
 best\_model\_path = model\_path  
 best\_loss = curr\_loss  
 else:  
 worse\_epochs += 1  
 print("Performance on validation set worsened to " + str(curr\_loss))  
 print("TRAINING FINISHED - TESTING WITH BEST MODEL " + best\_model\_path)  
 test\_loss = Test.test(model\_config, model\_folder=str(experiment\_id), partition="test", load\_model=best\_model\_path)  
 return best\_model\_path, test\_loss  
  
@ex.automain  
def run(cfg):  
 model\_config = cfg["model\_config"]  
 print("SCRIPT START")  
 # Create subfolders if they do not exist to save results  
 for dir in [model\_config["model\_base\_dir"], model\_config["log\_dir"]]:  
 if not os.path.exists(dir):  
 os.makedirs(dir)  
  
 # Optimize in a supervised fashion until validation loss worsens  
 sup\_model\_path, sup\_loss = optimise()  
 print("Supervised training finished! Saved model at " + sup\_model\_path + ". Performance: " + str(sup\_loss))  
  
 # Evaluate trained model on MUSDB  
 Evaluate.produce\_musdb\_source\_estimates(model\_config, sup\_model\_path, model\_config["musdb\_path"], model\_config["estimates\_path"])

**Test.py:**

import tensorflow as tf  
from tensorflow.contrib.signal.python.ops import window\_ops  
import numpy as np  
import os  
  
import Datasets  
import Models.UnetSpectrogramSeparator  
import Models.UnetAudioSeparator  
import functools  
  
def test(model\_config, partition, model\_folder, load\_model):  
 # Determine input and output shapes  
 disc\_input\_shape = [model\_config["batch\_size"], model\_config["num\_frames"], 0] # Shape of discriminator input  
 if model\_config["network"] == "unet":  
 separator\_class = Models.UnetAudioSeparator.UnetAudioSeparator(model\_config)  
 elif model\_config["network"] == "unet\_spectrogram":  
 separator\_class = Models.UnetSpectrogramSeparator.UnetSpectrogramSeparator(model\_config)  
 else:  
 raise NotImplementedError  
  
 sep\_input\_shape, sep\_output\_shape = separator\_class.get\_padding(np.array(disc\_input\_shape))  
 separator\_func = separator\_class.get\_output  
  
 # Creating the batch generators  
 assert ((sep\_input\_shape[1] - sep\_output\_shape[1]) % 2 == 0)  
 dataset = Datasets.get\_dataset(model\_config, sep\_input\_shape, sep\_output\_shape, partition=partition)  
 iterator = dataset.make\_one\_shot\_iterator()  
 batch = iterator.get\_next()  
  
 print("Testing...")  
  
 # BUILD MODELS  
 # Separator  
 separator\_sources = separator\_func(batch["mix"], False, not model\_config["raw\_audio\_loss"], reuse=False) # Sources are output in order [acc, voice] for voice separation, [bass, drums, other, vocals] for multi-instrument separation  
  
 global\_step = tf.get\_variable('global\_step', [], initializer=tf.constant\_initializer(0), trainable=False, dtype=tf.int64)  
  
 # Start session and queue input threads  
 sess = tf.Session()  
 sess.run(tf.global\_variables\_initializer())  
 writer = tf.summary.FileWriter(model\_config["log\_dir"] + os.path.sep + model\_folder, graph=sess.graph)  
  
 # CHECKPOINTING  
 # Load pretrained model to test  
 restorer = tf.train.Saver(tf.global\_variables(), write\_version=tf.train.SaverDef.V2)  
 print("Num of variables" + str(len(tf.global\_variables())))  
 restorer.restore(sess, load\_model)  
 print('Pre-trained model restored for testing')  
  
 # Start training loop  
 \_global\_step = sess.run(global\_step)  
 print("Starting!")  
  
 total\_loss = 0.0  
 batch\_num = 1  
  
 # Supervised objective: MSE for raw audio, MAE for magnitude space (Jansson U-Net)  
 separator\_loss = 0  
 for key in model\_config["source\_names"]:  
 real\_source = batch[key]  
 sep\_source = separator\_sources[key]  
  
 if model\_config["network"] == "unet\_spectrogram" and not model\_config["raw\_audio\_loss"]:  
 window = functools.partial(window\_ops.hann\_window, periodic=True)  
 stfts = tf.contrib.signal.stft(tf.squeeze(real\_source, 2), frame\_length=1024, frame\_step=768,  
 fft\_length=1024, window\_fn=window)  
 real\_mag = tf.abs(stfts)  
 separator\_loss += tf.reduce\_mean(tf.abs(real\_mag - sep\_source))  
 else:  
 separator\_loss += tf.reduce\_mean(tf.square(real\_source - sep\_source))  
 separator\_loss = separator\_loss / float(model\_config["num\_sources"]) # Normalise by number of sources  
  
 while True:  
 try:  
 curr\_loss = sess.run(separator\_loss)  
 total\_loss = total\_loss + (1.0 / float(batch\_num)) \* (curr\_loss - total\_loss)  
 batch\_num += 1  
 except tf.errors.OutOfRangeError as e:  
 break  
  
 summary = tf.Summary(value=[tf.Summary.Value(tag="test\_loss", simple\_value=total\_loss)])  
 writer.add\_summary(summary, global\_step=\_global\_step)  
  
 writer.flush()  
 writer.close()  
  
 print("Finished testing - Mean MSE: " + str(total\_loss))  
  
 # Close session, clear computational graph  
 sess.close()  
 tf.reset\_default\_graph()  
  
 return total\_loss

**UnetAudioSeparator.py:**

import tensorflow as tf  
  
import Models.InterpolationLayer  
import Utils  
from Utils import LeakyReLU  
import numpy as np  
import Models.OutputLayer  
  
class UnetAudioSeparator:  
 *'''  
 U-Net separator network for singing voice separation.  
 Uses valid convolutions, so it predicts for the centre part of the input - only certain input and output shapes are therefore possible (see getpadding function)  
 '''* def \_\_init\_\_(self, model\_config):  
 *'''  
 Initialize U-net  
 :param num\_layers: Number of down- and upscaling layers in the network   
 '''* self.num\_layers = model\_config["num\_layers"]  
 self.num\_initial\_filters = model\_config["num\_initial\_filters"]  
 self.filter\_size = model\_config["filter\_size"]  
 self.merge\_filter\_size = model\_config["merge\_filter\_size"]  
 self.input\_filter\_size = model\_config["input\_filter\_size"]  
 self.output\_filter\_size = model\_config["output\_filter\_size"]  
 self.upsampling = model\_config["upsampling"]  
 self.output\_type = model\_config["output\_type"]  
 self.context = model\_config["context"]  
 self.padding = "valid" if model\_config["context"] else "same"  
 self.source\_names = model\_config["source\_names"]  
 self.num\_channels = 1 if model\_config["mono\_downmix"] else 2  
 self.output\_activation = model\_config["output\_activation"]  
  
 def get\_padding(self, shape):  
 *'''  
 Calculates the required amounts of padding along each axis of the input and output, so that the Unet works and has the given shape as output shape  
 :param shape: Desired output shape   
 :return: Input\_shape, output\_shape, where each is a list [batch\_size, time\_steps, channels]  
 '''* if self.context:  
 # Check if desired shape is possible as output shape - go from output shape towards lowest-res feature map  
 rem = float(shape[1]) # Cut off batch size number and channel  
  
 # Output filter size  
 rem = rem - self.output\_filter\_size + 1  
  
 # Upsampling blocks  
 for i in range(self.num\_layers):  
 rem = rem + self.merge\_filter\_size - 1  
 rem = (rem + 1.) / 2.# out = in + in - 1 <=> in = (out+1)/  
  
 # Round resulting feature map dimensions up to nearest integer  
 x = np.asarray(np.ceil(rem),dtype=np.int64)  
 assert(x >= 2)  
  
 # Compute input and output shapes based on lowest-res feature map  
 output\_shape = x  
 input\_shape = x  
  
 # Extra conv  
 input\_shape = input\_shape + self.filter\_size - 1  
  
 # Go from centre feature map through up- and downsampling blocks  
 for i in range(self.num\_layers):  
 output\_shape = 2\*output\_shape - 1 #Upsampling  
 output\_shape = output\_shape - self.merge\_filter\_size + 1 # Conv  
  
 input\_shape = 2\*input\_shape - 1 # Decimation  
 if i < self.num\_layers - 1:  
 input\_shape = input\_shape + self.filter\_size - 1 # Conv  
 else:  
 input\_shape = input\_shape + self.input\_filter\_size - 1  
  
 # Output filters  
 output\_shape = output\_shape - self.output\_filter\_size + 1  
  
 input\_shape = np.concatenate([[shape[0]], [input\_shape], [self.num\_channels]])  
 output\_shape = np.concatenate([[shape[0]], [output\_shape], [self.num\_channels]])  
  
 return input\_shape, output\_shape  
 else:  
 return [shape[0], shape[1], self.num\_channels], [shape[0], shape[1], self.num\_channels]  
  
 def get\_output(self, input, training, return\_spectrogram=False, reuse=True):  
 *'''  
 Creates symbolic computation graph of the U-Net for a given input batch  
 :param input: Input batch of mixtures, 3D tensor [batch\_size, num\_samples, num\_channels]  
 :param reuse: Whether to create new parameter variables or reuse existing ones  
 :return: U-Net output: List of source estimates. Each item is a 3D tensor [batch\_size, num\_out\_samples, num\_channels]  
 '''* with tf.variable\_scope("separator", reuse=reuse):  
 enc\_outputs = list()  
 current\_layer = input  
  
 # Down-convolution: Repeat strided conv  
 for i in range(self.num\_layers):  
 current\_layer = tf.layers.conv1d(current\_layer, self.num\_initial\_filters + (self.num\_initial\_filters \* i), self.filter\_size, strides=1, activation=LeakyReLU, padding=self.padding) # out = in - filter + 1  
 enc\_outputs.append(current\_layer)  
 current\_layer = current\_layer[:,::2,:] # Decimate by factor of 2 # out = (in-1)/2 + 1  
  
 current\_layer = tf.layers.conv1d(current\_layer, self.num\_initial\_filters + (self.num\_initial\_filters \* self.num\_layers),self.filter\_size,activation=LeakyReLU,padding=self.padding) # One more conv here since we need to compute features after last decimation  
  
 # Feature map here shall be X along one dimension  
  
 # Upconvolution  
 for i in range(self.num\_layers):  
 #UPSAMPLING  
 current\_layer = tf.expand\_dims(current\_layer, axis=1)  
 if self.upsampling == 'learned':  
 # Learned interpolation between two neighbouring time positions by using a convolution filter of width 2, and inserting the responses in the middle of the two respective inputs  
 current\_layer = Models.InterpolationLayer.learned\_interpolation\_layer(current\_layer, self.padding, i)  
 else:  
 if self.context:  
 current\_layer = tf.image.resize\_bilinear(current\_layer, [1, current\_layer.get\_shape().as\_list()[2] \* 2 - 1], align\_corners=True)  
 else:  
 current\_layer = tf.image.resize\_bilinear(current\_layer, [1, current\_layer.get\_shape().as\_list()[2]\*2]) # out = in + in - 1  
 current\_layer = tf.squeeze(current\_layer, axis=1)  
 # UPSAMPLING FINISHED  
  
 assert(enc\_outputs[-i-1].get\_shape().as\_list()[1] == current\_layer.get\_shape().as\_list()[1] or self.context) #No cropping should be necessary unless we are using context  
 current\_layer = Utils.crop\_and\_concat(enc\_outputs[-i-1], current\_layer, match\_feature\_dim=False)  
 current\_layer = tf.layers.conv1d(current\_layer, self.num\_initial\_filters + (self.num\_initial\_filters \* (self.num\_layers - i - 1)), self.merge\_filter\_size,  
 activation=LeakyReLU,  
 padding=self.padding) # out = in - filter + 1  
  
 current\_layer = Utils.crop\_and\_concat(input, current\_layer, match\_feature\_dim=False)  
  
 # Output layer  
 # Determine output activation function  
 if self.output\_activation == "tanh":  
 out\_activation = tf.tanh  
 elif self.output\_activation == "linear":  
 out\_activation = lambda x: Utils.AudioClip(x, training)  
 else:  
 raise NotImplementedError  
  
 if self.output\_type == "direct":  
 return Models.OutputLayer.independent\_outputs(current\_layer, self.source\_names, self.num\_channels, self.output\_filter\_size, self.padding, out\_activation)  
 elif self.output\_type == "difference":  
 cropped\_input = Utils.crop(input,current\_layer.get\_shape().as\_list(), match\_feature\_dim=False)  
 return Models.OutputLayer.difference\_output(cropped\_input, current\_layer, self.source\_names, self.num\_channels, self.output\_filter\_size, self.padding, out\_activation, training)  
 else:  
 raise NotImplementedError

**mse:**

import soundfile as sf  
import numpy as np  
from pystoi import stoi  
from pesq import pesq  
from scipy.io import wavfile  
from scipy.signal import resample\_poly  
ref, fs\_ref = sf.read( 'D:\Ben\FinalProject\Github\Train-test-dataset\Train\Actions - South Of The Water.stem\_vocals.wav') # input - reffrence mixture path  
pred, fs\_pred = sf.read('D:\Ben\FinalProject\TestSongs\_Train\Actions - South Of The Water.stem.mp4\_vocals.wav') # input - prediction from U-Net  
if ref.ndim > 1:  
 ref = np.mean(ref, axis=-1)  
  
if pred.ndim > 1:  
 pred= np.mean(pred, axis=-1)  
if fs\_ref > 16000:  
 ref = resample\_poly(ref[:4410000], 16000, fs\_ref)  
if fs\_pred > 16000:  
 pred = resample\_poly(pred[:4410000], 16000, fs\_pred)  
  
print("pred sample:{}".format(fs\_pred))  
d = stoi(ref, pred, fs\_ref, extended=False)  
print("STOI:{}".format(d))  
print(pesq(16000, ref, pred, 'wb'))  
mse = np.sum((ref - pred)\*\*2, axis=0) / len(pred) # calculation of MSE  
  
noise = pred - ref  
std\_noise = np.std(noise)  
std\_ref = np.std(ref)  
snr = std\_ref / (std\_noise + np.finfo(np.float32).eps)  
snr\_db = 20 \* np.log10(snr + np.finfo(np.float32).eps)  
print("SNR:{}".format(snr\_db))  
print("MSE = {}".format(mse))

**נספח ג' – פורמט סקר :**

**סקר איכות:**

מטרת הסקר היא למצוא את מדדMOS (Mean Opinion Score) של כל מקור שמע של תוצר המערכת.

בתקיית 'שירים מופרדים' ניתן לראות שעבור כל שיר קיימים 4 קבצים נוספים שהם מקורותיו המופרדים שהפיקה המערכת של פרויקט זה: זמר/ת, תופים, בס, צלילים אחרים.

בטבלה הבאה עלייך לדרג בטווח של ­1-10 (כאשר 1 – גרוע, 10 – מצוין) את איכות השמע של המקורות המופרדים לפי דעתך.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Others | Bass | Drums | Vocals | Song name / Source |
|  |  |  |  | **3 Doors Down - Here Without You** |
|  |  |  |  | **Kings Of Leon - Use Somebody** |
|  |  |  |  | **Green Day - American Idiot** |
|  |  |  |  | **Green Day -Boulevard of Broken Dreams** |
|  |  |  |  | **Life Is Beautiful (Acoustic)** |

**נספח ד' – פוסטר הפרויקט:**

