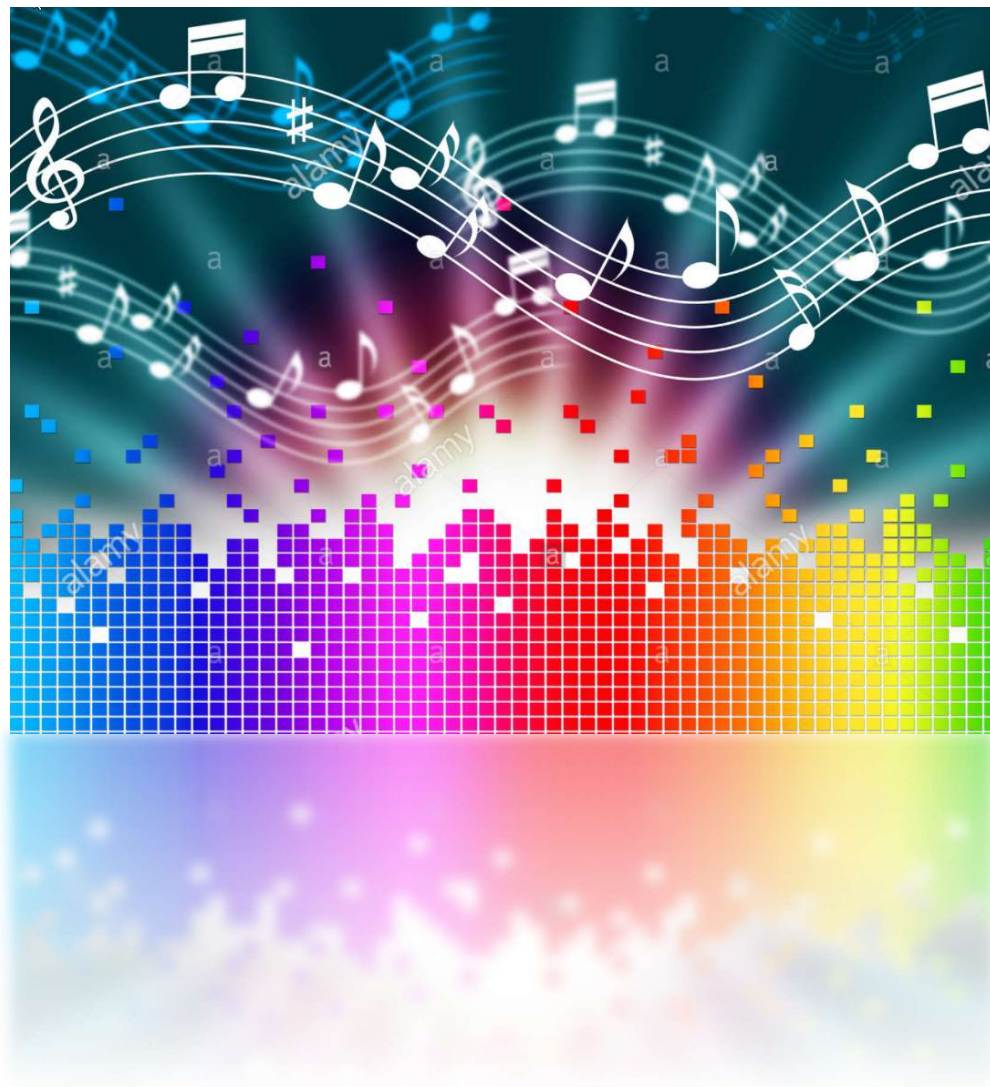


דו"ח פרויקט מסכם בינה מלאכותית – 236502

Generating Melodies

מערכת ליצירת מנגינות



מגישים:

טל רוזנצוויג 307965806

שני אופיר 204512396

ליאור שרמן 307932277

תוכן עניינים

2תוכן עניינים
3הקדמה
3 Midi בקע כלי במוזיקה ובפרוטוקול
6 אתגרים ובעיות ביצירת מלודיות באמצעות בינה מלאכותית
7 הצגת מודלי הלמידה
7 מבוא ומוטיבציה לשימוש במודל <i>RNN – LSTM</i>
7 מבוא ומוטיבציה לשימוש במודל ה- <i>GAN</i>
8 ארכיטקטורת המודל
9 אתגרים במודל ה- <i>GAN</i> הבסיסי
10 דרכי פעולה
10 (1 בחירת <i>Dataset</i>
11 (2 הגדרת בעיית חיזוי על המאגר
11 (3 עיבוד מקדים של ה- <i>Dataset</i>
11 עבור מודל ה- <i>RNN</i>
17 עבור מודל ה- <i>GAN</i>
19 (4 תהליך הלמידה – <i>High and Low level</i>
19 עבור מודל ה- <i>RNN</i>
21 עבור מודל ה- <i>GAN</i>
23 (5 יצירת לחנים – התוצר הסופי
23 תהליך היצירה של המלודיה באמצעות מודל ה- <i>RNN</i>
24 תהליך היצירה של המלודיה באמצעות מודל ה- <i>GAN</i>
25 תיאור הניסויים תוצאות ומסקנות
25 עבור מודל ה- <i>RNN – LSTM</i>
25 ניסוי ראשון
26 ניסוי שני
27 ניסוי שלישי
27 עבור מודל ה- <i>GAN</i>
27 ניסוי למציאת אופטימיזר מתאים
30 סיכום ודיון בתוצאות
30 דיון בתוצאות
33 כיוונים נוספים להמשך המחקר
34 ביבליוגרפיה

הקדמה

כיום, לבינה מלאכותית קיימת השפעה ניכרת על כמעט כל התעשיות כמו בריאות, פיננסים, ניהול וכך גם על תעשיית המוזיקה. הטכנולוגיה העדכנית ביותר מגלה דפוסים ותבניות במערכות נתונים גדולות ומסייעת ביצירת קטעים טובים להאזנה אך ייתכן שאלו לא יהיו שירים נצחיים במקומות גבוהים במצעדים. כלומר, הבינה המלאכותית בתחום המוזיקה יכולה ליצור יצירה שעובדת היטב עבור סרטוני וידיאו, משחקי רשת או אפילו מסע בחנויות. המוזיקה הנוצרת היא לא רק זולה יותר, אלא ברוב המקרים אף עדיפה על גרסאות אנושיות.

אם כן, נבין תחילה מהי מלודיה וכיצד היא מתקשרת לבינה מלאכותית. מלודיה היא צירוף של סידרת צלילים מוזיקליים בסדר קבוע, עולה או יורד הנשמעים ברצף. סידרת צלילים אלו הם אשר מרכיבים את המנגינה/מלודיה המרכזית אשר שומעים בשיר או יצירה. מאחר והמלודיה מורכבת מרצף של צלילים עם "מנוחות", נוכל להפוך כל מלודיה לסדרה של זמנים (*Time series*), כאשר סדרת זמן הינה מבנה נתונים שבו המידע הוא רצף שנלקח בנקודות זמן רצופות שוות מרווח, כלומר רצף של נתונים בזמן בדיד כדוגמת גבהים של גאות ושפל באוקיינוס. בדרך זו, נוכל להשתמש במודלים של ניתוח סדרות זמן של זרימה מוזיקלית על מנת ליצר אלגוריתם המייצר זרמים מוסיקליים נוספים וחדשים. למעשה, נקבל רדוקציה לבעיית יצירת המנגינות שהופכת לבעיית סדרת זמנים.

אם כן, לאחר מחקר רב בתחום החלטנו ליצור מכונה שמייצרת מלודיות תוך ציפייה שנקבל לחנים חדשים ו"נעימים" להאזנה.

רקע כללי במוזיקה ובפרוטוקול Midi

כל צליל מורכב מגובה הצליל וממשך הזמן שבו הוא מתנגן:

$$\text{Note} = \text{Pitch} + \text{Duration}$$

התאוריה של המוזיקה מחלקת את האוקטבה, המרווח בין שני צלילים שתדירותו של האחד מהם היא חצי (או פי 2) מתדירותו של השני, לסדרה של 12 תווים שבהם ניתן להשתמש על מנת להלחין יצירות שונות. סידרה זו נקראת "סולם כרומטי" והיא נראית כך:



סימון גובה מדעי (*SPN*), הידוע גם בשם סימון גובה תקן אמריקאי (*ASP*) וסימון גובה בינלאומי (*IPN*), הוא שיטה לציון גובה המוסיקה על ידי שילוב של שם תו מוזיקלי ומספר זיהוי האוקטבה של התו, כדוגמת (*C4*). סימון זה מספק אמצעי חד משמעי לזיהוי תו במונחים של סימון טקסטואלי ולא תדר ולכן ניעזר בו על מנת לבנות מודל של ה-*dataset* איתו נעבוד.

A#7	106	107	C8
G#7	104	105	A7
F#7	102	103	G7
D#7	99	100	E7
C#7	97	98	D7
A#6	94	95	B6
G#6	92	93	A6
F#6	90	91	G6
D#6	87	88	E6
C#6	85	86	D6
A#5	82	83	B5
G#5	80	81	A5
F#5	78	79	G5
D#5	75	76	E5
C#5	73	74	D5
A#4	70	71	C5
G#4	68	69	B4
F#4	66	67	A4
D#4	63	64	G4
C#4	61	62	F4
A#3	58	59	E4
G#3	56	57	D4
F#3	54	55	C4
D#3	51	52	B3
C#3	49	50	A3
A#2	46	47	G3
G#2	44	45	F3
F#2	42	43	E3
D#2	39	40	D3
C#2	37	38	C3
A#1	34	35	B2
G#1	32	33	A2
F#1	30	31	G2
D#1	27	28	F2
C#1	25	26	E1
A#0	22	23	D1
		21	C1
			B0
			A0

מישכי הצלילים בתווים

שלם - משך של 4 רבעים

חצי - משך 2 פעמות

רבע - משך פעמה אחת

שמינית - חצי פעמה

אחד חלקי שש עשרה רבע פעמה

כל המשכים מתייחסים לפעמה במשך של רבע

הגדרת מפתח:

מפתח הוא קבוצת הצלילים המהווים את מרכז כל האלמנטים המרכיבים חתיכה של היצירה.
מפתח יהיה בנוי כך:

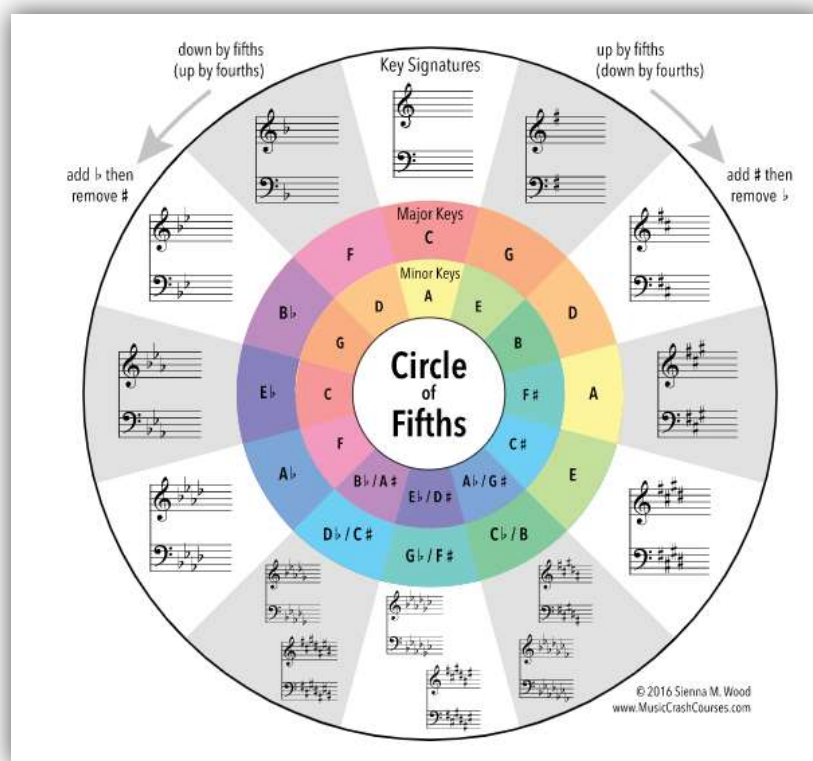
$$\text{Key} = \text{Tonic} + \text{Mode}$$

כך שהמפתח מורכב מ:

❖ *Mode*, כלומר סוג הסולם, הוא דרך לארגן מחדש את הצלילים של סולם כך שמוקד הסולם ישתנה בכל פעם. במפתח אחד, כל *Mode* מכיל בדיוק את אותם הצלילים. כמו כן, באמצעות שינוי המוקד, נוכל לקבל צלילים חדשים ומעניינים. הסולם הוא למעשה דרך לארגון של מקבץ צלילים כך שקיימים לנו שני סולמות: מג'ור (*Major*) ומינור (*Minor*) ואילו יהיו ערכי הפרמטר *Mode* במפתח.

❖ *Tonic* מוגדר להיות כדרגת הסולם הדיאטוני, כלומר הצליל הראשון בסולם והמרכז הטונאלי או צליל הרזולוציה הסופית המשמשים בדרך כלל בקצב הסופי במוזיקה קלאסית טונלית. *Tonic* מכונה גם המפתח המרכזי במוזיקה, ומשמש כמוקד ללחן ולהרמוניה.

כפי שהצגנו, קיימים לו 12 צלילים ו-2 סולמות ולכן נקבל כי קיימים 24 מפתחות וכך הם נראים:



המפתחות ישמשו אותנו לצורך תהליך ה-*preprocess* של ה-*dataset* בדרך נוחה ונכונה.

מבנה מלודיה:

מלודיה נמדדת בזמן כאשר היא בנויה מאבני הבניין הבאות:

- פעמה – יחידת הזמן הבסיסית ביותר (משך ניגון תו) *beat*.
- תיבה – יחידת זמן המורכבת ממספר פעמות *bar*.
- משפט – יחידת זמן המורכבת ממספר תיבות *phrase*.

phrase 1	phrase 2	phrase 3	phrase 4
bar 1	bar 2	bar 3	bar 4
beat 1	beat 2	beat 3	beat 4

אתגרים ובעיות ביצירת מלודיות באמצעות בינה מלאכותית

יצירת מלודיות באמצעות בינה מלאכותית מביאה עמה אתגרים רבים וביניהם:

1. מוזיקה כאמנות התלויה בזמן - מלודיה מורכבת מרצף של תווים שנשמעים באופן סדרתי הניתנים למדידה לאורך זמן. על כן, תהליך היצירה מורכב יותר בעקבות תלות זו, שלא כמו שאר בעיות החיזוי שאינן תלויות במרכיב הזמן כדוגמת חיזוי נתונים, יצירת תמונות או בעיות סיווג.
2. הקשר בין אבני הבניין של המוזיקה – מעבר לקשר הכרונולוגי במוזיקה, קיימים קשרים פנימיים בין אבני הבניין, לדוגמה לקשר אפשרי תהיה אם במידה והופיע בבית הראשון אקורד C , גם בבית האחרון יופיע האקורד C בהרמוניה גבוהה או נמוכה יותר. על כן, ישנו קושי ליצור קשר בין אבני הבניין כך שיתאימו מבחינת מוזיקלית אחד לשני.
3. הקשר בין הכלים השונים המנגנים יחדיו – לרוב בקטעי מוזיקה קיימים מספר רב של כלים והם מנגנים יחד באופן מסונכרן, לכן לכל כלי יש עצמאות משלו ובנוסף הוא צריך להסתנכרן עם שאר הכלים. על כן, הקושי במקרה זה הוא ביצירת מלודיה המורכבת ממספר כלים תוך ביצוע סנכרון ביניהם.
4. קושי בעיבוד גלי קול – על מנת לעבד גלי קול צריך לרדת לרזולוציית יחידות זמן מאוד גבוהה. בנוסף, קיים קושי בהפרדת הכלים בהינתן גל קול המורכב ממספר כלים המנגנים יחד.

הצגת מודלי הלמידה

מבוא ומוטיבציה לשימוש במודל $RNN - LSTM$

מלודיה מורכבת מדפוסים מבניים לטווח ארוך ומחזרה של תבניות אלו, גם אם נעשו בהן שינויים קלים כדוגמת גובה הצליל או קטעים שעברו "מתיחה" או "כיווץ".

זיכרון לטווח קצר - ארוך ($LSTM$) הוא ארכיטקטורת רשת נוירונים חוזרת (RNN) המשמשת בתחום הלמידה העמוקה.

המודל $LSTM$ יודע לעבד רצפים שלמים של נתונים, ולא רק נקודות נתונים בודדות. כמו כן, רשתות ה- $LSTM$ מתאימות לסיווג, עיבוד וחיזוי על סמך נתוני סדרות זמן ($Time series$), כלומר נתונים שמתקדמים בזמן כך שיש קשר של זמן ביניהם. כיום, קיימים שימושים רבים במודל זה לצרכי לימוד דקדוק, זיהוי קול, זיהוי וסיווג שפות וכדומה.

אם כן, מאחר ומודל זה משמש לעיבוד וחיזוי של שפה (טקסט), בחרנו לנסות להסתכל על מוזיקה כעל "שפה", כטקסט המכיל קשר של זמן, וכך לבצע חיזוי בכל פעם של התו הבא בתהליך יצירת המלודיה. אם כן, בחרנו להשתמש במודל זה על מנת ליצור את המכונה ליצירת מלודיות.

מבוא ומוטיבציה לשימוש במודל ה- GAN

$Generative Adversarial Network - GAN$ - הינו מודל גנרטיבי ללמידת מכונה שתוכנן על ידי *Ian Goodfellow* ועמיתיו בשנת 2014. לפי מודל זה, בהינתן סט אימון, ניתן להשתמש במודל על מנת ללמוד את ההתפלגות שממנה מגיעים הנתונים. לאחר לימוד זה, נשתמש בתוצאותיו על מנת ליצור דגימות חדשות שנראות כמו דגימות ששייכות למאגר הנתונים וכך לקבל יצירה חדשה.

מודל ה- GAN בנוי משתי יחידות:

- 1) גנרטור - תפקיד הגנרטור הוא להתחקות אחר התפלגות ההסתברות של הדוגמאות עליהן ביצע למידה ולאחר מכן לייצר דוגמא המתאימה להתפלגות זו.
- 2) המסווג - תפקיד המסווג ללמוד כיצד להבדיל בין דוגמא אמיתית, כלומר אחת כזו שמגיעה מה- $dataset$, לבין דוגמא שאינה אמיתית, כזו שנוצרה על ידי הגנרטור. בתהליך האימון של הרשת, שתי יחידות אלו מקיימות קשרי אימון ביריבות, כלומר הגנרטור מנסה לעבוד על המסווג בזמן שהמסווג מנסה להתמודד עם בעיית הסיווג לצד שיפור התקדמות הלמידה של הגנרטור.

במהלך השנים מודלים שונים של ה- GAN הראו תוצאות טובות בעיקר בחילול (יצירה) של תמונות, טקסט וידאו ובפרט במוסיקה, אך המשימה האחרונה נותרה מאתגרת.

כאמור לעיל, מוזיקה הינה אמנות של זמן. לכן, הקושי בשימוש במודל זה לצורך חילול מלודיה, נובע מכך שארכיטקטורת המודל עצמו היא שהוא מבצע את תהליך הלמידה על דוגמאות הקבועות בזמן. לכן, מודל זה ידוע בטיבו לחילול תמונות אך לבעיות התלויות בהתקדמות הזמן, כדוגמת עולם המוזיקה, הבעיה עדיין נותרה מורכבת.

ארכיטקטורת המודל

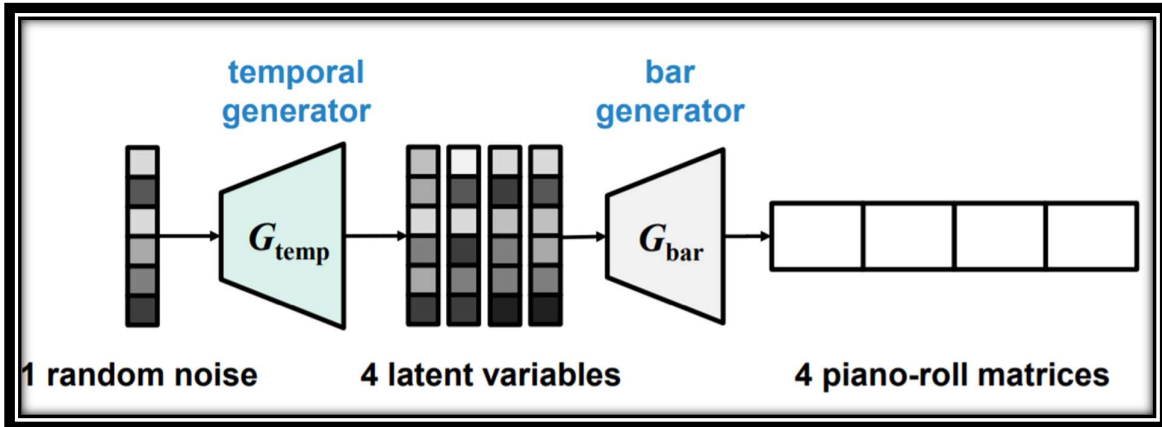
בדומה ל-GAN בסיסי, המודל שלנו מחולק לשני רכיבים עיקריים - גנרטור ומבקר.

גנרטור:

על מנת לתת מענה לבעיה שהצגנו מעלה לפיה למוזיקה יש מורכבות של זמן, השתמשנו ברשת מורכבת יותר - רשת טמפורלית.

רשת טמפורלית בנויה משני גנרטורים באופן הבא:

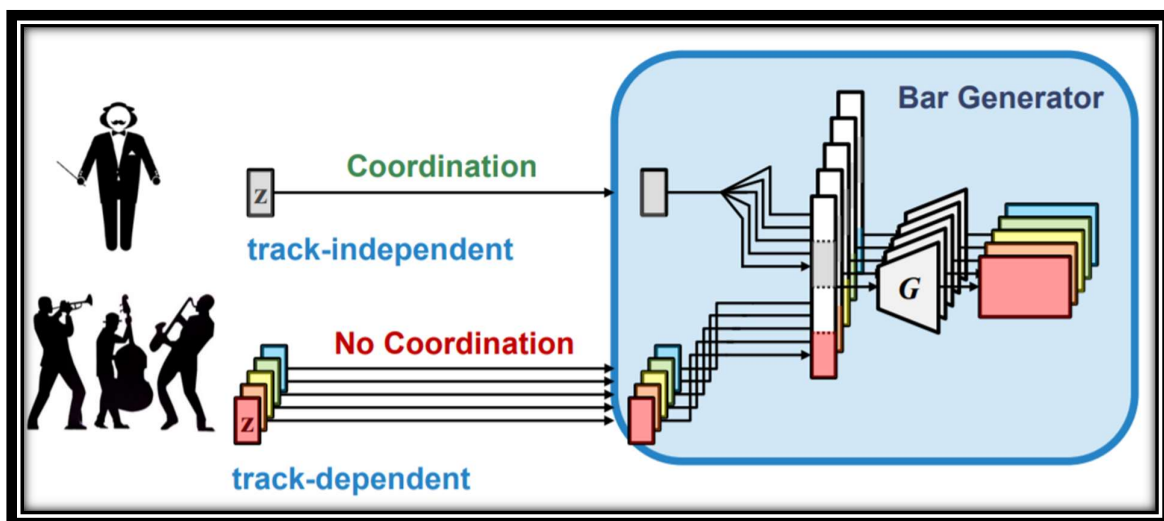
הגנרטור הראשון מקבל וקטור רעש רנדומלי ומייצר ממנו מטריצה המכילה 4 וקטורים כך שכל אחד מהם משמש כקלט לגנרטור השני שיוצר את הבית הבא וכך נוצר קשר בין הבתים - ניתן לראות זאת באיור הבא:



איור זה נלקח מהמאמר המוצר בחלק הביבליוגרפיה בשם *MUSEGAN SLIDES*.

עד כה, תיאור זה משמש עבור כלי אחד ולכן על מנת ליצור מלודיות מרובות כלים, השתמשנו במספר של רשתות טמפורליות - אחת לכל כלי.

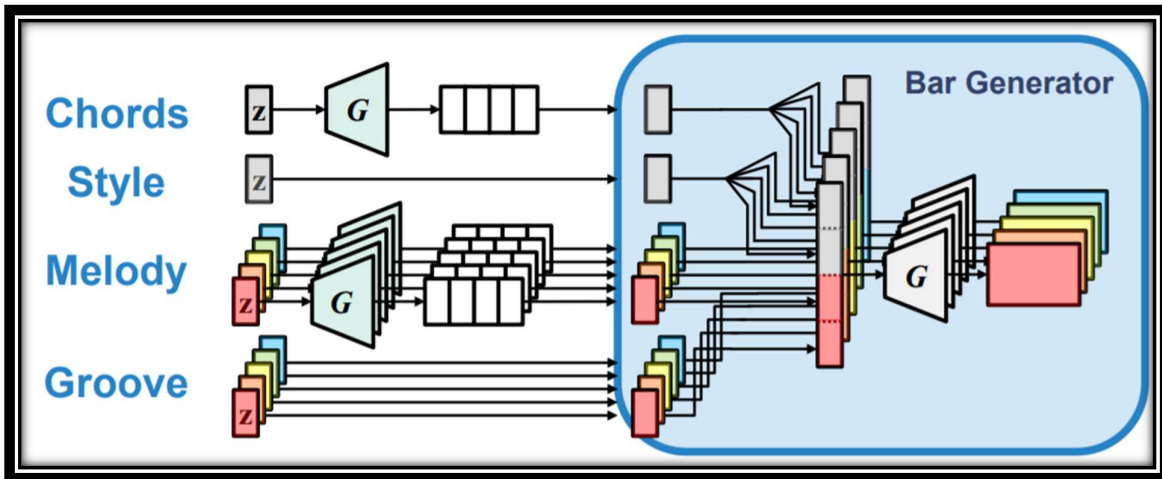
בשלב זה, קיים סנכרון בין כל כלי לעצמו באמצעות הרשת הטמפורלית, אך מתעוררת בעיה של סנכרון בין כל הכלים, כלומר אחד לשני. במציאות קיים מנצח האחראי על סנכרון בין הכלים ולכן על מנת להתמודד עם בעיה זו השתמשנו ברשת רב כלית הבאה לידי ביטוי באמצעות הוספת וקטור רעש אקראי שיהיה משותף לכל הכלים וכך מתקיים קשר עצמאי בין כל כלי לעצמו ובין כל הכלים יחדיו. האיור מתאר כיצד מבצעים סנכרון בין הכלים:



איור זה נלקח מהמאמר המוצר בחלק הביבליוגרפיה בשם *MUSEGAN SLIDES*.

הגנרטור המלא:

אם כן, הגנרטור המלא משתמש בשני הרעיונות, רשת טמפורלית ורשת רב כלית שהוצגו לעיל, ובנוי בצורה הבאה:



איור זה נלקח מהמאמר המוצר בחלק הביבליוגרפיה בשם *MUSEGAN SLIDES*

באיור ניתן לראות את גנרטור התיבות אשר מקבל כקלט וקטור רעש המורכב מארבעה חלקים שלכל אחד מהם תפקיד שונה ביצירת התיבה:

- *Chords* – משותף בין כל הכלים ומשתנה בין תיבה לתיבה, יוצר קשר בין הכלים כתלות בזמן.
- *Style* – משותף בין כל הכלים וזהה לכל התיבות, יוצר קשר בין הכלים ללא תלות בזמן.
- *Melody* – מקנה עצמאות לכל כלי בנפרד כתלות בזמן.
- *Groove* – מקנה עצמאות לכל כלי בנפרד ללא תלות בזמן.

מבקר:

תפקידו של המבקר הוא להתאים ניקוד לכל דוגמא ולמדוד את ה"אמיתיות" של הדוגמא.

אתגרים במודל ה-GAN הבסיסי:

במודל *GAN* תהליך האימון של הגנרטור והמסווג מורכב מתחרות ביניהם במשחק "סכום אפס", וקיים צורך לשמור על יציבות ביחסי הכוחות ביניהם כאשר משימה זו מהווה אתגר.

בעיה נפוצה אחרת היא קריסת המודל לפיה הגנרטור דבק בדוגמאות ספציפיות שמצליחות לעבוד על המסווג, אך אינן מייצגות את ההתפלגות האמיתית של הנתונים. הדבר מתבטא בכך שהגנרטור יוצר מספר מצומצם של דוגמאות שמצליחות לעבוד על המסווג, אך לא מצליח לייצר גיוון מספק.

כדי להימנע מבעיות אלו משתמשים בגרסה משופרת של מודל ה-*GAN*, הנקראת *WGAN*, אשר במקום להשתמש במסווג שמטרתו לסווג דוגמאות לאמיתיות ולא אמיתיות, הוא משתמש במבקר שתפקידו להתאים ניקוד לכל דוגמא ולמדוד את ה"אמיתיות" של הדוגמא.

בנוסף, הגרסה המשופרת משתמשת בפונקציית $loss$ שונה מזו השייכת למודל הבסיסי ושמה $wassertein loss$ אשר מבוססת על $Earth moving distance$.
בגרסה זו מטרת הגנרטור לצמצם את המרחק בין ההתפלגות של הנתונים להתפלגות הדוגמאות שהוא מייצר בעוד שמטרת המבקר היא לחשב את המרחק בין ההתפלגויות.
בפרויקט שלנו, נשתמש בשיפורים של $WGAN$.

דרכי פעולה

לאורך הפרויקט שיטת העבודה כללה את השלבים הבאים:

- (1) בחירת ה- $Dataset$.
- (2) הגדרת בעיית חיזוי על המאגר.
- (3) עיבוד מקדים של ה- $Dataset$.
- (4) תהליך הלמידה – $High and Low level$.
- (5) יצירת לחנים – התוצר הסופי.

1 בחירת Dataset

בפרויקט זה ה- $dataset$ הינו אוסף של קבצי KRN וקבצי $MIDI$.
בתחילת תכנון הפרויקט חשבנו לבחור Dataset ולסווג אותו לפי לחנים "שקטים" ולחנים "קצביים" במטרה להקל על תהליך הלמידה של המודל.
עם התקדמות הפרויקט, החלטנו להשתמש במגוון רחב יותר של סגנונות המשלבים באותו השיר את שני המקצבים וזאת על מנת ליצור מכונה גנרית ככל הניתן, כלומר מכונה שתבצע את תהליך הלמידה שלה על מאגר כללי של לחנים ללא סיווג התחלתי.

לבסוף, גילינו שהצורה הטובה ביותר לבחירת ה- $dataset$ תהיה באמצעות בחירת שירים בעלי מנעד דומה, כלומר מנגינות מאותו הסגנון. כמו כן, המנגינות צריכות להיות מנגינות פשוטות כאשר מנגינה פשוטה מוגדרת כמנגינה בעלת מקצב פשוט כדוגמת 4/4 ובעלת תווים חוזרים לאורך השיר ובמנגינות אחרות ב- $dataset$.

עבור מודל ה- GAN בחרנו ב- $Snes$ בתור ה- $dataset$ כאשר הוא מכיל הרבה כלים והמנגינות יחסית פשוטות.

2) הגדרת בעיית חיזוי על המאגר

בפרויקט זה בעיית החיזוי מוגדרת כבעיה של יצירה, ובמקרה שלנו ג'נרנט(יצירה) של מלודיה. ניתן לפרק את בעיית היצירה לשרשור של בעיות חיזוי קטנות כך שבכל אחת מבעיות החיזוי, בהינתן סידרת תווים, נרצה לחזות מהו התו הבא המתאים מבחינה מוזיקלית לרצף השיר. על מנת ליצור מלודיה חדשה, נתחיל מסידרת תווים כלשהי (*seed*), נבצע חיזוי ונשרשר את התוצאה לסוף סידרת התווים. כך, נמשיך לבצע זאת באופן איטרטיבי עד שלבסוף נקבל מלודיה שלמה.

כפי שהצגנו במבוא, השתמשנו בשני מודלי למידה עמוקה לצורך פתרון בעיה זו, מודל ה- *GAN* ומודל ה- *RNN – LSTM*.

3) עיבוד מקדים של ה-Dataset

שלב העיבוד המקדים של ה-*dataset* מתבצע באופן שונה בהתאם למודל הלמידה, כלומר קיים עיבוד מקדים עבור מודל ה-*RNN* ועיבוד מקדים עבור מודל ה-*GAN*, כאשר אין תלות ביניהם.

עבור מודל ה-RNN

עבור מודל ה-*RNN* ביצענו עיבוד מקדים של המידע המכיל *dataset* של שירים בעלי כלי נגינה בודד. לאחר מכן, ביצענו עיבוד מקדים מורכב יותר המסוגל לעבד *dataset* עם מספר כלי נגינה. נתאר זאת בשלבים:

1. סינון קבצים לא רצויים מה-*dataset* באמצעות פרמטרים שהגדרנו כדוגמת תווים נדירים שמופיעים לעיתים רחוקות במנגינות, מקצבים לא פופולריים.
2. ביצוע *parsing* לקובץ עם *Music21*.
3. הפיכת הקובץ לקידוד טקסטואלי (*Time series representation*) שנעשה בשני שלבים – קידוד עבור כלי נגינה בודד וקידוד משופר שמקנה למודל יכולת ללמוד על מספר כלי נגינה.
4. יצירת מיפוי בין כל הסימבולים מתוך כל הקידודים של כל השירים לסדרה עולה של מספרים שלמים.
5. יצירת קידוד אחד ארוך של כל הקידודים משורשרים אחד לשני עם סימבול מפריד ביניהם שיסמן סוף של שיר.
6. המרת המחרוזות לייצוג של מספרים שלמים ושמירת ה-*dataset* המעובד.
7. קידוד סט הלמידה כ- *One – hot vectors* (*One – hot encoding*).

כעת, נפרט על השלבים המורכבים יותר ב-*High level* ושאר השלבים יהיו ברורים מהפסאודו קוד שיוצג בהמשך.

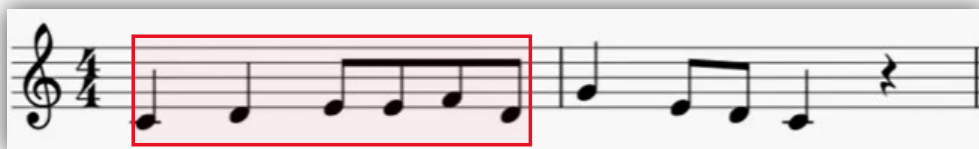
שלב 3:

בשלב ההתחלתי של הקידוד הטקסטואלי, התייחסנו לקידוד עבור כלי נגינה בודד.

על כן, קידוד זה, שיסומן באות P , התבצע באופן הבא:

כל קובץ יהפוך לקידוד הטקסטואלי ע"פ העקרונות הבאים:

- שימוש בסדרות זמנים, מבנה נתונים שבו המידע הוא רצף שנלקח בנקודות זמן רצופות שוות מרווח.
 - דגימת מנגינה בכל התו ה-16.
 - כל תו ייוצג ע"י מספר ה-Midi המיוחס אליו כאשר תו מתנגן – ראה [איור](#).
 - שימוש ב-"_" כסימבול עבור תו ממושך.
 - שימוש ב-"r" כסימבול של "מנוחה".
 - שימוש ב-"/" כסימבול עבור סיום שיר.
- לצורך המחשה, נשתמש בדוגמא הבאה:
עבור המנגינה הבאה:



עבור החלק המסומן באדום נקבל את הייצוג הטקסטואלי:

```
["60", "_", "_", "_",  
"62", "_", "_", "_",  
"64", "_", "64", "_",  
"65", "_", "62", "_"]
```

עד לנקודה זו, הקידוד הטקסטואלי מתייחס אך ורק לתו בודד בכל יחידת זמן כפי שרואים לעיל. אולם, קיימים שירים רבים המכילים אקורדים כאשר אקורד הוא מספר צלילים/תווים המנוגנים יחד באותה יחידת זמן. לכן, הוספנו יכולת לעיבוד אקורדים באופן הבא:
במידה ונרצה לנגן את התווים 60, 62 ו-64 באותו הזמן, כלומר הם יהוו יחדיו אקורד, מבחינת הקידוד הם יופיעו כך: ["60.62.64", ...].

עם התקדמות הפרויקט, שיפרנו את שיטת הקידוד הטקסטואלי וזאת מאחר ורצינו לייצג מספר כלי נגינה בקידוד ולא כלי נגינה בודד כפי שהצגנו עד כה.
אם כן, בעת כל יחידת זמן תייצג n שדות כאשר כל שדה ייצג כלי שונה. נמחיש זאת, באמצעות הדוגמא הבאה:
נסתכל על מערכת התווים הבאה שלקוחה מקלאסיקה של האמן הידוע מוצארט:

1 כלי

2 כלי

3 כלי

4 כלי

$t = 0$

ניתן לראות כי במערכת זו קיימים ארבעה חלקים כאשר כל חלק מייצג כלי נגינה. על כן, הקידוד הטקסטואלי עבור יחידת זמן בודדת, בהינתן הקידוד הקודם שיסומן באות P , הקידוד המשופר יתבצע על פי הנוסחה הבאה: $P'_{t=0} = [P(1)_{t=0}, P(2)_{t=0}, P(3)_{t=0}, P(4)_{t=0}]$. כלומר, נקבל את הקידוד הבא:

$["60.62.63, 64, 60.62.63, 64"]$

תיאור בעיות שנוצרו בעקבות שיטת הקידוד שהוצגה:

לאחר שעבדנו בשיטת קידוד טקסטואלי זו, הגענו למסקנה כי קיימות שתי בעיות עיקריות בעקבות שיטת קידוד זו.

איבוד מידע בשלב הלמידה-

נניח כי המודל מבצע למידה על שירים שיש להם שני כלים ומעלה. כלומר, חלק מהסימבולים יראו כך $[x, y]$ וחלקם יראו ממימד גדול יותר, כאשר האותיות מייצגות תווים. לצורך שלמות ההסבר, נגדיר את המושג $Seed$ שמייצג סידרת תווים בגודל אחד או יותר, ממנה מתחילים לנגן ולפיו המודל מקבל החלטה לגבי חיזוי הסימבול הבא במלודיה החדשה שהוא יוצר.

מאחר ובמלודיות בעלות שני כלים לא נוספים או נמחקים כלים במהלך הלמידה, מתקיים כי מלודיה שמחילה עם שני כלים, ממשיכה גם עם שני כלים ולכן המודל יצמצם את חיזוי הסימבולים שלו לסימבולים מגודל 2 כך ששאר הסימבולים מגדלים השונים מ-2, לא יבואו לידי ביטוי במלודיה זו. נמחיש זאת ע"י הדוגמה הבאה:

במידה והסימבול הראשון, כלומר ה- $seed$, יהיה ממימד 2 כלומר עם 2 כלים, המודל ייצור מנגינה רק עם 2 כלים, בהתאם להסבר הקודם לפיו המודל עובד. במצב זה, כל המידע על השירים המורכבים ממספר כלים השונה מ-2, יאבד מאחר ויצירת המנגינה החדשה תיעשה רק מה- $dataset$ של השירים בעלי 2 כלים ולכן יצירת המלודיות תהיה לא אופטימלית מאחר והיא לא מנצלת את כל ה- $dataset$ אותו היא למדה בשלב הלמידה.

פתרון לבעיה – "ריפוד מאוחר":

לאחר עיבוד וקידוד כל השירים, נבדוק את מספר הכלים המירבי הקיים בשיר מה- $dataset$. נגדיר זאת באופן מתמטי:

$$num\ of\ instruments_i = i \text{ מספר הכלים בשיר } i$$

$$dim_{max} = \max_i \{num\ of\ instruments_i\}$$

כאשר i מוגדר להיות קובץ ב- $dataset$ ו- dim_{max} הוא המימד המקסימלי אותו חיפשנו. כעת, ניקח את כל השירים בעלי מימד קטן יותר מ- dim_{max} ונרחיב אותם ע"י הוספת "r" (מנוחה) בסוף הסימבול וכך נקבל כי כל הסימבולים יהיו במימד של dim_{max} .

לדוגמא, נניח כי ב-*dataset* מספר הכלים המקסימלי הקיים באחד מהשירים המופיעים ב-*dataset* הוא 5 וקיימים שירים נוספים בעלי מימד 3 ו-2. עבור השירים בעלי מימד 3 ו-2, נניח כי סימבול בשיר הוא מהצורה $[x, y, z, r]$ ו- $[a, b]$, כעת הם יורחבו באמצעות הוספת "r" ויהפכו להיות בעלי מימד 5, כלומר: $[x, y, z, r, r]$ ו- $[a, b, r, r, r]$.

הוספת סימבול ה-"r" אינה משנה את השיר וזאת מאחר ואנו לא שומעים סימבול זה ומכאן נובעת הבחירה בהוספתו. בדרך זו התגברנו על הבעיה שהצגנו וכך לא נאבד מידע עליו ביצענו תהליך למידה.

נשים לב כי במצב זה ייתכן ונגיע למקרה בו ייווצר השיר הבא:

נניח כי ה-*seed* הראשון הוא ממימד 5, לכן נקבל כי כל שאר הסימבולים יורחבו במידת הצורך למימד זה ונקבל את השיר הבא:

```
"60, 62, r, 63, 65",
"61, 60, r, r, r",
"r, r, r, r, r"]
```

כלומר נקבל כי קיים כלי נגינה המורכב רק "r" (*rests*) ומצב זה אפשרי בכל מקום בשיר (אמצע/סוף/התחלה), כלומר נקבל כי קיים כלי שלא מנגן כלל אלא רק "נח" ולכן אינו משפיע על המלודיה, כלומר מידע מיותר.

על מנת להימנע מכך, לאחר שלב הלמידה, כלומר בשלב הסופי לאחר קבלת שיר, נבצע מעבר על כל השיר שנוצר ונבדוק האם קיים מצב שבו כלי נגינה שמורכב מ-"r" בכל יחידות הזמן (כמו בדוגמא לעיל), נבצע מחיקה של כלי זה ולמעשה כך נקבל שיר המכיל רק את הכלים שמנגנים.

שלב 4:

בחלק זה נייצר מיפוי בין כל הסימבולים מתוך כל הקידודים של כל השירים ב-*dataset* וניצור מהם סדרה עולה של מספרים שלמים. על מנת לבצע זאת, נייצר תחילה רשימה של כל הסימבולים הקיימים ב-*dataset* ולאחר מכן ניתן אינדקס כל אחד מהם וניצור מפה. כלומר, בהינתן *dataset* המכיל סימבולים שונים, נייצר אינדקס לכל סימבול ונכניס אותו בצירוף האינדקס שלו למבנה של מפה. לדוגמא, בהינתן ה-*dataset* המעובד אשר מכיל את רשימת הסימבולים הבאה:

60 _ _ _ 62 _ _ _ _ 64 _ 65 _ r _ /

נקבל את המפה הבאה:

```
{
"60": 0,
"62": 1,
"64": 2,
"65": 3,
"_": 4,
"/": 5,
"r": 6
}
```

שלב 6:

תחילה, נמפה את הייצוג של השירים במחרוזות לווקטורים של מספרים שלמים באמצעות המפה שיצרנו בשלב 4. לאחר מכן, נשמור את כל ה-*dataset* המעובד בקובץ *numpy* המייצג מערך *numpy*.

שלב 7:

כאשר אנו עוסקים בלמידת מכונה, *hot – One* היא קבוצה של ביטים שביניהם צירופי הערכים החוגגים הם רק אלה בעלי סיבית אחת גבוהה (1) וכל השאר הם נמוכים (0). צורת קידוד זו מתגברת על בעיות שונות המופיעות בתהליך הלמידה כדוגמת יצירת קשרים שגויים על ה-*data* המתבססים על תכונות של המספרים השלמים המייצגים אותם כדוגמת מספרים עוקבים או מספרים קרובים על הציר. לכן, נשנה את הקידוד שנעשה עד כה ונקודד את סט הלמידה כ- *One – hot vectors* (*One – hot encoding*). נמחיש זאת באמצעות הדוגמא הבאה:

נניח כי קיים לנו הקלט הבא:

הרצף $[0, 1, 2]$ כאשר המספרים מייצגים מספר קטגוריות. הן יהפכו לרצפים הבאים – $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$ המהווה *One – hot* ווקטור.

בעיית זיכרון:

לפי שיטת קידוד זו, כל סימבול הופך להיות וקטור של *One – hot* שגודלו כגודל מספר הסימבולים הקיימים. במצב זה במידה וניקח *dataset* המורכב ממגוון רחב של תווים שונים ומורכבים, נקבל כי קיים לנו מספר רב של סימבולים וכתוצאה מכך נקבל דרישה לזיכרון רב לצורך שלב העיבוד המקדים של ה-*dataset*.

בנוסף, כתוצאה מכך לא נוכל לאמן את המודל על שירים ארוכים ומורכבים ונצטרך להסתפק בלמידה על שירים פשוטים יחסית.

פתרון לבעיה זו ע"י שימוש ב- *Data loader* והמרה "עצלה":

במקום להמיר **כל** ה-*dataset* המעובד ל-*One – hot vectors* בשלב העיבוד המקדים, נבצע המרה של מספר קבוע וקטן משמעותית מגודל ה-*dataset* שיקרא *batch* בצורה "עצלה", כלומר רק כאשר נרצה לבצע למידה על ה-*batch* הנוכחי, נבצע המרה שלו ל-*One – hot vectors*. בדרך זו, נימנע משימוש מופרז בזיכרון ונוכל לבצע למידה גם על שירים בעלי מספר רב של סימבולים.

על כן, שלב זה כבר לא קיים יותר בשלב העיבוד המקדים של ה-*dataset* אלא הועבר לשלב הלמידה.

פסאודו קוד של תהליך העיבוד המקדים:

1) Load Midi files
2) Filter bad songs
3) Transpose songs
4) For song in songs: encode(song)
5) Normalize songs
6) Create single file dataset
7) Generate encoded dataset

מספר שלב	הסבר
1	טעינת קבצי <i>Midi</i> על ידי פרסור- לקיחה מקובץ מידי והפיכה לאובייקט <i>M21</i> כאשר פרסור הינו מונח ידוע לפיו מתבצע פענוח או ניתוח של טקסט באמצעות תוכנת מחשוב.
2	סינון שירים שמוגדרים כ"לא טובים" כדוגמת שירים שיש להם משכי תווים לא נפוצים.
3	ביצוע <i>transpose</i> לכל השירים (<i>transpose_songs</i>), כלומר הפיכתם לסולם אחיד כך ששירים בסולם מינור יעברו לסולם <i>A_{minor}</i> ושירים בסולם מג'ור יעברו ל- <i>C_{major}</i> . שלב זה הוא אופציונלי על מנת להקל על הלמידה באמצעות כך שהשירים באותו הסולם. עם זאת, במידה ונעשה שימוש באופציה זו, מגוון השירים שנוצרים הינו קטן יותר ולכן נשתמש בפיצ'ר זה לעיתים רחוקות.
4	לכל שיר ב- <i>dataset</i> בצע קידוד לשיר ושמירתו. הקידוד נעשה באמצעות הפונקציה <i>encode_and_save</i> אשר לוקחת אובייקט מסוג <i>M21</i> והופכת אותו לרשימה של סימבולים כאשר כל סימבול הוא מסוג מחרוזת שמייצג את השיר(כפי שתואר לעיל בשלב 3).
5	קריאה לפונקציה <i>normalize_encoded_songs</i> שמטרתה לממש את הפתרון "ריפוד מאוחר"(תואר לעיל בשלב 3).
6	שרשור שירים – שרשור כל הקידודים של כל השירים לקידוד אחד ארוך ובין קידוד לקידוד נוסף סידרת סימבולי סיום שהגדרנו " / " באורך 64 סימבולים. מספר זה הוא היפר פרמטר. לאחר מכן, ניצור מיפוי מכל סימבול שקיים בקידוד הארוך(השרשור שנוצר) למספר שלם ולאחר מכן נשמור אותו בקובץ <i>.json</i> .
7	ניקח את שרשור הקידודים(רשימה של מחרוזות) ונשתמש במיפוי על מנת להפוך את השרשור לרשימה של מספרים שלמים. את רשימה זו נהפוך ל- <i>NDARRAY</i> כלומר מערך של <i>numpy</i> ונבצע שמירה.

עבור מודל ה-GAN

תחילה, נתאר את הייצוג וההכנה של ה-dataset עבור מודל ה-GAN ולאחר מכן נתאר את העיבוד המקדים שלו.

ייצוג ה-dataset:

ניתן לייצג קטע מוזיקלי של כלי בודד כטנסור (Tensor) ב- $\{0,1\}^{T \times P}$ כך ש:

- T מייצג את מספר יחידות הזמן.
- P מייצג את מספר הטונים (pitches) האפשריים.

כלומר, $x[t, p]$ מקבל ערך 1 במידה וטון p פעיל(מנוגן) בזמן t .

ניתן להרחיב ייצוג זה לייצוג עבור מספר כלים באמצעות הוספת מימד נוסף M כך שיתקבל הייצוג מהצורה - $\{0,1\}^{M \times T \times P}$.

הכנת ה-dataset:

כעת, נבצע הכנה של ה-dataset.

לאחר בחירת ה-dataset, אנו ניצבים בפני אתגר נוסף – לא כל קבצי ה-Midi מורכבים מאותם כלי נגינה. על כן, נפתור בעיה זו באמצעות איחוד כלים דומים לכלי יחיד. כך למשל, קלידים, ופסנתר כנף יאוחדו יחד לפסנתר בודד. לבסוף, לאחר שלב האיחוד, ה-dataset המעודכן יהיה אחיד כך שכל קובץ יכיל 4 כלים בדיוק לפי סדר משפחות הכלים שקבענו.

בנוסף, לאחר בדיקות ומחקר על ה-dataset גילינו שקיימות 4 משפחות כלים נפוצות ביותר והן תופים, פסנתר, באס ומקהלה.

פסאודו קוד של שלב הכנת ה-dataset:

1) Load Midi files

2) Filter bad songs

3) For song in songs:

Merge instruments

מספר שלב	הסבר
1	טעינת קבצי Midi מה-dataset.
2	סינון שירים שאינם במשקל 4/4 ואת כל השירים שאינם מכילים את ארבעת משפחות הכלים.
3	לכל שיר ב-dataset בצע איחוד של כל הכלים הדומים למשפחות כפי שהוגדרו ונתעלם מכילים שאינם שייכים לאחת מהמשפחות שהוגדרו.

עיבוד מקדים של ה-dataset:

לאחר שביצענו הכנה ל-dataset, נוכל לגשת לעיבוד שלו.

העיבוד המקדים מורכב משני שלבים עיקריים:

השלב הראשון כולל בתוכו את טעינת הקבצים לאחר שלב ההכנה והשלב השני מבצע עיבוד שלהם.

פסאודו קוד עבור העיבוד המקדים:

1) Load Midi files

2) Processed data = []

3) For file in Midi files:

3.1) Piano_roll = create_piano_roll(file)

3.2) Filter(piano_roll)

3.3) Samples = split_piano_roll(piano_roll)

3.4) Processed_data.append(samples)

מספר שלב	הסבר
1	טעינת קבצי Midi מה-dataset.
2	אתחול רשימה ריקה
3	לכל שיר ב-dataset בצע: 3.1) מייצרים מקובץ ה-midi מטריצת piano_roll מהצורה $\{0,1\}^{M \times T \times P}$ כפי שהוגדר בייצוג ה-dataset. 3.2) מסננים מהמטריצה טונים נדירים על מנת לחסוך בזיכרון וכדי לא לפגוע בתהליך הלמידה ע"י הכנסת רעש מיותר. 3.3) פיצול המטריצה למשפטים מוזיקליים בגודל 4 תיבות וסינון משפטים "שקטים" שבהם קיימת תיבה המכילה כלי שאינו מנגן כלל 3.4) מוסיפים את הדוגמאות שנוצרו לרשימה המעובדת.

לאחר תהליך זה, נקבל dataset מעובד כמטריצה שכל דוגמא בה היא ממימד 4 ונראית כך:

(n_tracks, n_bars, bar_resulotion, pitches)

הסבר עבור כל איבר:

- n_tracks – מספר הכלים ב-dataset (קיבענו ל-4 כלים).
- n_bars – מספר התיבות ב-dataset (קיבענו ל-4 תיבות).
- bar_resulotion – מספר יחידות הזמן בתיבה – 16.
- pitches – מספר הטונים 84 מתוך 128 לאחר סינון.

4 תהליך הלמידה – High and Low level

עבור מודל ה-RNN

נחלק את תהליך הלמידה לשני חלקים:

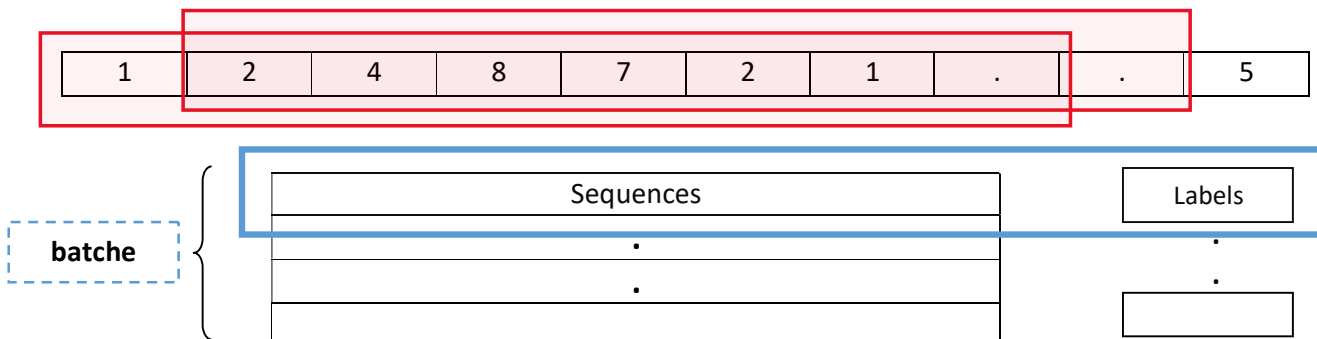
חלק ראשון - הכנת ה-*data* לשלב המידה:

בשלב הראשוני של תהליך הלמידה אנו מבצעים חלוקה אקראית של ה-*dataset* המעובד לשני חלקים כך שהחיתוך בין שני חלקים אלו הוא קבוצה ריקה והאיחוד מהווה את כל ה-*dataset*. החלק הראשון ישמש לצרכי אימון המודל והחלק השני ישמש להערכת המודל, כלומר להעריך עד כמה הוא טוב.

החלוקה של ה-*dataset* נעשית ביחס של 1 ל-9 לטובת האימון וזהו היפר פרמטר, כלומר ניתן לבחור יחס שונה. בחירת ערך זה נעשתה מאחר והוא ערך פופולארי לצרכי אימון. נבצע שימוש ב-*dataloader* שיכין לנו *batches* (אצוות) לתהליך הלמידה. בכל שלילפת *batch* מה-*dataloader*, תבוצע הכנה של רצף למידה והמרתו ל-*One – hot* ווקטור. המרה זו של אצווה בכל פעם ל-*One – hot* ווקטור חוסכת בזיכרון לעומת המרת כל ה-*dataset* בפעם אחת וכך מהווה פתרון לבעיית הזיכרון שהוזכרה בשלב 7 (ראה בעמוד 15). קיימים שני *dataloader* – אחד המשמש להערכה והשני לאימון.

כעת, נסביר מהו רצף למידה:

רצף למידה מורכב מ-64 מספרים שלמים ואחריהם מגיע המספר במקום ה-65 ב-*dataloader* שזהו ה-*label*.



בכל פעם נבחר רצף למידה של 64 ערכים מתוך הקידוד (מסומן ב- ומכיל 64 ערכים), נשים אותו בשורה ב-*Sequences* ואת המספר ה-65 נשים ב-*Batch .labels* היא מטריצה המורכבת מ-64 רצפי למידה בצירוף ווקטור של *labels*. מספר ה-*Batches* תלוי בגודל ה-*dataset*. בסיום בניית ה-*batch*, כל אחד מהמספרים השלמים יהפכו לייצוג *One – hot*.

חלק שני - שלב הלמידה:

תחילה, נבצע אימון במספר איטרציות (*epochs*) כך שבכל איטרציה נעביר את המודל למצב למידה (*pytorch*). נבצע איטרציות על כל ה-*batches* מה-*dataloader* ולכל *batch* נעביר את ה-*batch* ל-*CPU* או ל-*GPU* (בהתאם למשאבים) ולאחר מכן נאפס את הגראדינט של האופטימיזר, נבצע העברה של הניורונים ברשת ה-*RNN – LSTM (forward)*, נחשב את ה-*loss* באמצעות פונקציית *crossentropy*. לאחר חישוב ה-*loss* מחשבים את הנגזרת שלו (*backward*) ונבצע צעד של עדכון פרמטרים באופטימיזר (*optimizer.step*). כעת, בסוף כל אימון של *epoch* בודד, נבצע הערכה של המודל המתבצעת ע"י שליפת *batches* מה-*dataloader* של הערכה, מבקשים מהמודל לחזות את הסימן הבא על כל רצף ובודקים מול ה-*label* של אותו רצף האם המודל צדק או לא. לבסוף, נחשב את דיוק המודל על פי מספר ה-*labels* שצדקנו בהם מתוך כל ה-*labels* שהוא קיבל (הקיימים). הערה: כל תהליך הערכה מתבצע ללא עדכון הפרמטרים של האופטימיזר וללא השפעה על נגזרת של ה-*loss*.

פסאודו קוד של שלב הלמידה:

1) For epoch in range(epochs):

1.1) model.train

1.2) For sequences, labels in train_loader:

```
sequences.to(device)
labels.to(device)
optimizer.zero_grad()
outputs = model(sequences)
loss = loss_function(outputs, labels)
loss.backward()
optimizer.step()
```

1.3) With torch.no_grad()

1.3.1) For sequences, labels in train_loader:

```
sequences.to(device)
labels.to(device)
predicted = model(sequences)
correct = (predicted==labels).sum()
```

1.3.2) accuracy = correct/len(labels)

2) model.save()

מספר שלב	הסבר
1	בתוך הלולאה החיצונית, תחילה נהפוך את המודל למצב של אימון.
אימון 1.1 - 1.2	בתוך הלולאה הפנימית (לולאת אימון), לכל <i>batch</i> המכיל קבוצה של רצפים ו- <i>labels</i> ב- <i>trainloader</i> (ה- <i>dataloader</i> של ה- <i>train</i>) נבצע העברה של ה- <i>batch</i> למעבד הרלוונטי. בנוסף, נבצע איפוס הגראדיאנט של האופטימיזר ונבצע חיזוי של ה- <i>labels</i> , אחד לכל רצף (<i>forward</i>) כך שנקבל ווקטור של <i>labels</i> . כעת, נבצע בדיקה של ה- <i>labels</i> לאחר שלב החיזוי לעומת ה- <i>labels</i> של ה- <i>batch</i> המקוריים שהגיעו מה- <i>dataloader</i> ונחשב את ה- <i>loss</i> . כעת נחשב חישוב נגזרת של ה- <i>loss</i> ונעדכן את הפרמטרים של האופטימיזר לפי הנגזרת.
הערכה 1.3	יציאה מהלולאה הפנימית: נבצע מספר חישובים שאינם משפיעים על הנגזרת ולכן לא משפיעים על האימון ונועדים לצורך הערכה. לולאה נוספת לצורך הערכה: לכל <i>batch</i> נבצע העברה של ה- <i>batch</i> למעבד.

הרלוונטי. כעת, ונבצע חיזוי של ה- <i>labels</i> , אחד לכל רצף (<i>forward</i>) כך שנקבל ווקטור של <i>labels</i> . כעת נספור כמה המערכת "צדקה" כפי שתואר קודם ונחשב את הדיוק.	
יציאה מהלולאה הראשית: ונבצע שמירת המודל ושמירת קובץ <i>json</i> שהוא המיפוי סמלים.	שמירה 2

עבור מודל ה- GAN

תהליך הלמידה של מודל ה- *GAN* מורכב מתחרות בלתי פוסקת בין המבקר לגנרטור המתחרים אחד בשני במשחק "סכום אפס". למידת המבקר מתבססת על:

- מתן ניקוד לדוגמאות אמיתיות מתוך ה- *dataset*.
- מתן ניקוד לדוגמאות לא אמיתיות שנוצרו ע"י הגנרטור לצרכי הערכה בלבד.

הגנרטור מייצר דוגמאות ושולח אותן למבקר למטרת הערכה והפידבק המתקבל ממנו (המבקר) מהווה בסיס לאימון הגנרטור.

בצורה זו, הגנרטור והמבקר מתחרים אחד בשני ובמקביל גם מאמנים אחד את השני.

בעמוד הבא יוצג הפסאודו קוד של שלב הלמידה עבור מודל זה.

1) *for epoch in epochs:*

1.1) *for real_batch in dataset:*

Critic train

1.1.1) *real_score = critic.forward(real_batch)*

1.1.2) *random_noise = vec.random()*

1.1.3) *with torch.no_grad:*

fake_batch= generator.forward(random_noise)

1.1.4) *fake_score = critic.forward(fake_batch)*

1.1.5) *critic_loss = c_loss(real_score, fake_score)*

1.1.6) *critic_loss.backward()*

1.1.7) *critic_optimizer.step()*

Generator train

1.1.8) *random_noise = vec.random()*

1.1.9) *fake_batch = generator.forward(random_noise)*

1.1.10) *fake_score = critic.forward(fake_batch)*

1.1.11) *generator_loss = g_loss(fake_score)*

1.1.12) *generator_loss.backward()*

1.1.13) *generator_optimizer.step()*

מספר שלב	הסבר
1	לכל <i>epoch</i> בצע את 1.1.
1.1	לכל <i>batch</i> מתוך ה- <i>dataloader</i> .
אימון המבקר 1.1.1 - 1.1.7	תחילה נבצע הערכה של המבקר על ה- <i>batch</i> ונשמור את ווקטור ההערכות עבור דגימות אמיתיות. כעת, ניצור דוגמא באמצעות הגנרטור מתוך ווקטור הרעש, את הדוגמא נעביר להערכה של המבקר ולאחר מכן נשמור את ווקטור ההערכות עבור דגימות לא אמיתיות. לבסוף, נחשב את ה- <i>loss</i> בין ווקטור ההערכה האמיתי לבין ווקטור ההערכה הלא אמיתי ונחשב את נגזרת ה- <i>loss</i> . כמו כן, נבצע עדכון לפרמטרים של האופטימיזר השייך למבקר.
אימון הגנרטור 1.1.8 - 1.1.13	ניצור ווקטור רעש וניתן אותו לגנרטור כקלט על מנת שיצור מלודיה ולאחר מכן נבצע הערכה של המבקר עליה. כעת, נחשב את ה- <i>loss</i> של הגנרטור, נחשב את הנגזרת של ה- <i>loss</i> של הגנרטור ולבסוף נעדכן את הפרמטרים של האופטימיזר השייך לגנרטור.

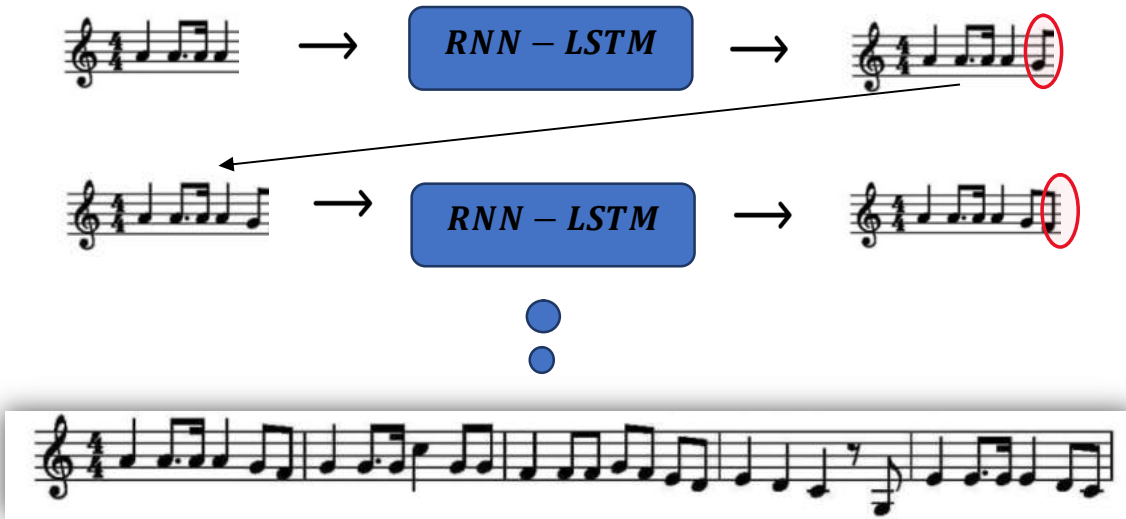
5) יצירת לחנים – התוצר הסופי

תהליך היצירה של המלודיה באמצעות מודל ה- *RNN – LSTM*

לאחר שלב הלמידה, הציפייה היא שה- *RNN – LSTM* ידע לספק לנו כפלט את הסימבול הבא בהינתן סידרת סימבולים (סימבול מוגדר להיות מה שמתנגן ביחידת זמן שלב 4).



למעשה אנו מתחילים ממקור כלשהו "seed" שאנו בוחרים, המורכב ממספר סימבולים, מכניסים אותו למודל והמודל יחזה את הסימבול הבא. בשלב הבא, נשרשר ל- *seed* את הסימבול שהמודל חזה וכך הלאה לאיטרציות הבאות עד לקבלת סימבול הסיום "/".



החיזוי זהה לקידוד המורחב למספר כלי נגינה, במצב זה המודל יחזה סימבול המייצג *timestep* בודד עבור כל כלי הנגינה.

פסאודו קוד של יצירת מלודיה:

```
1) melody = seed
   symbol = none
2) while (symbol != "/")
   symbol = model(melody)
   melody += symbol
3) score = m21.stream.score(melody)
4) parts = decode parts(melody)
5) for part in parts:
   score.insert(part)
6) score.write(format = midi)
```

מספר שלב	הסבר
1	אתחול המלודיה להיות ה- <i>seed</i> .
2	לולאה לפיה כל עוד המודל לא חזה את הסימבול המייצג סיום, הוא מבצע חיזוי לסימבול ומשרשר אותו למלודיה.
3	אתחול אובייקט <i>m21 score</i> .
4	מפרקים את המלודיה למלודיות של כל כלי (אם יש יותר מ-n), כלומר המלודיה מורכבת מסימבולים וכל סימול מופרד ב- ", " כאשר כל חלק בפסיק שייך ל- <i>part</i> מסוים לכן ניקח מלודיה ונפרק אותה לכל ה- <i>parts</i> שלה.
5	כתיבת הכלים ל- <i>score</i> כאשר כל כלי נכתב בנפרד.
6	שמירה בקובץ <i>midi</i> .

תהליך היצירה של המלודיה באמצעות מודל ה- GAN

בשלב הראשון של תהליך זה מייצרים:

- וקטור רעש לשימוש של *chords*.
- וקטור רעש לשימוש של *style*.
- ארבעה וקטורי רעש, אחד לכל כלי לשימוש של *melody*.
- ארבעה וקטורי רעש, אחד לכל כלי לשימוש של *groove*.

לאחר מכן, הגנרטור מקבל את הרעשים כקלט וכפלט מייצר דוגמא מההתפלגות עליה ביצע תהליך של למידה. בשלב האחרון, יש לבצע את התהליך ההפוך (פרסור) לייצוג קובץ *midi* שהוסבר בשלב העיבוד המקדים וזאת על מנת לקבל קובץ *midi*.

פסאודו קוד של יצירת מלודיה:

```
1) random_noises = create_random_noises()
2) batch_sample = generator.forward(random_noise)
3) midi = parse_to_midi(batch_sample)
```

מספר שלב	הסבר
1	מג'נרטים (יוצרים) רעשים רנדומלים לפי ההסבר המופיע בנקודות לעיל.
2	הגנרטור מקבל כקלט את הרעש הרנדומלי שנוצר בשלב 1 ויוצר ממנו דוגמא על בסיס התפלגות הנתונים שהוא למד.
3	שחזור הדוגמא שנוצרה בשלב 2 על פי הייצוג לקובץ <i>midi</i> כאשר הייצוג מתייחס לייצוג שהוסבר בשלב העיבוד המקדים.

תיאור הניסויים תוצאות ומסקנות

עבור מודל ה-RNN – LSTM

המודל משתמש בערכים רבים בתור היפר פרמטרים, כלומר פרמטרים שיש לחקור אותם על מנת לזהות מהו הערך האופטימלי עבור משימת המודל. לצורך כך, ביצענו ניסויים על פי שיטת

KFold cross validation הידוע בתחום ה-*deep learning*.

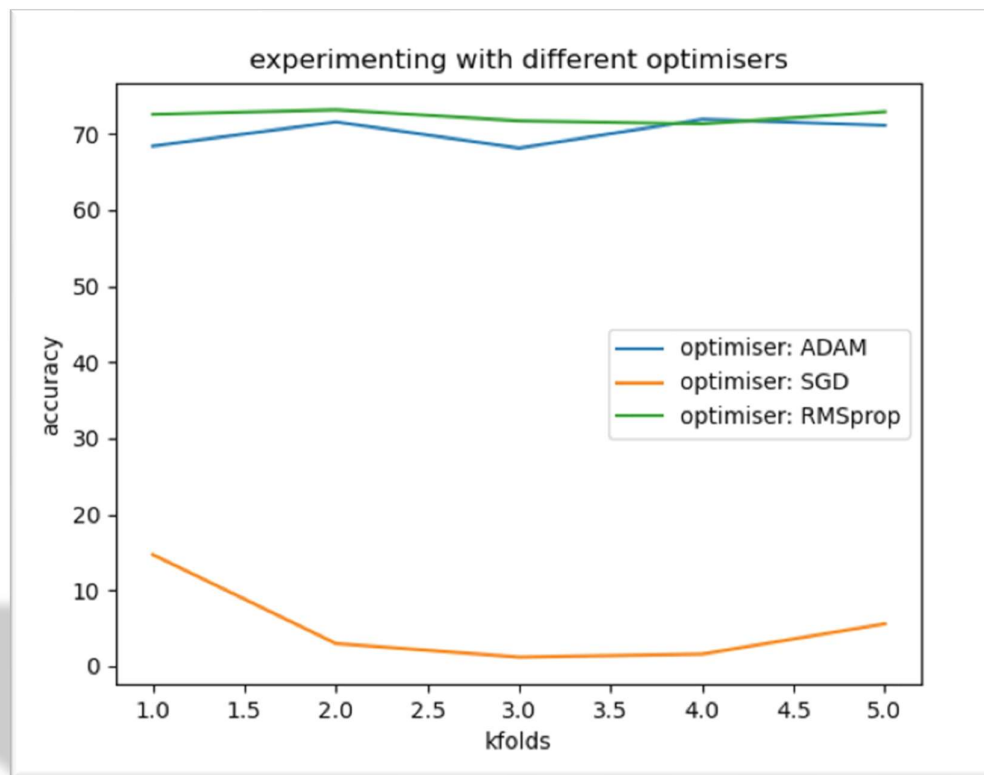
ניתן לראות הסבר על שיטה זו בקישור [*KFold cross validation*](#).

ניסוי ראשון

מטרת ניסוי זה הייתה לצורך בחירת האופטימיזר שבדאי לעבוד איתו. במהלך ניסוי זה, ביצענו לכל אופטימיזר *5Fold cross validation* על *dataset* בגודל ממוצע ובחנו את דיוק המודל בכל *Fold*. האופטימיזרים שנבדקו הם שלושה אופטימיזרים הנפוצים:

- ❖ *SGD*
- ❖ *ADAM*
- ❖ *RMSprop*

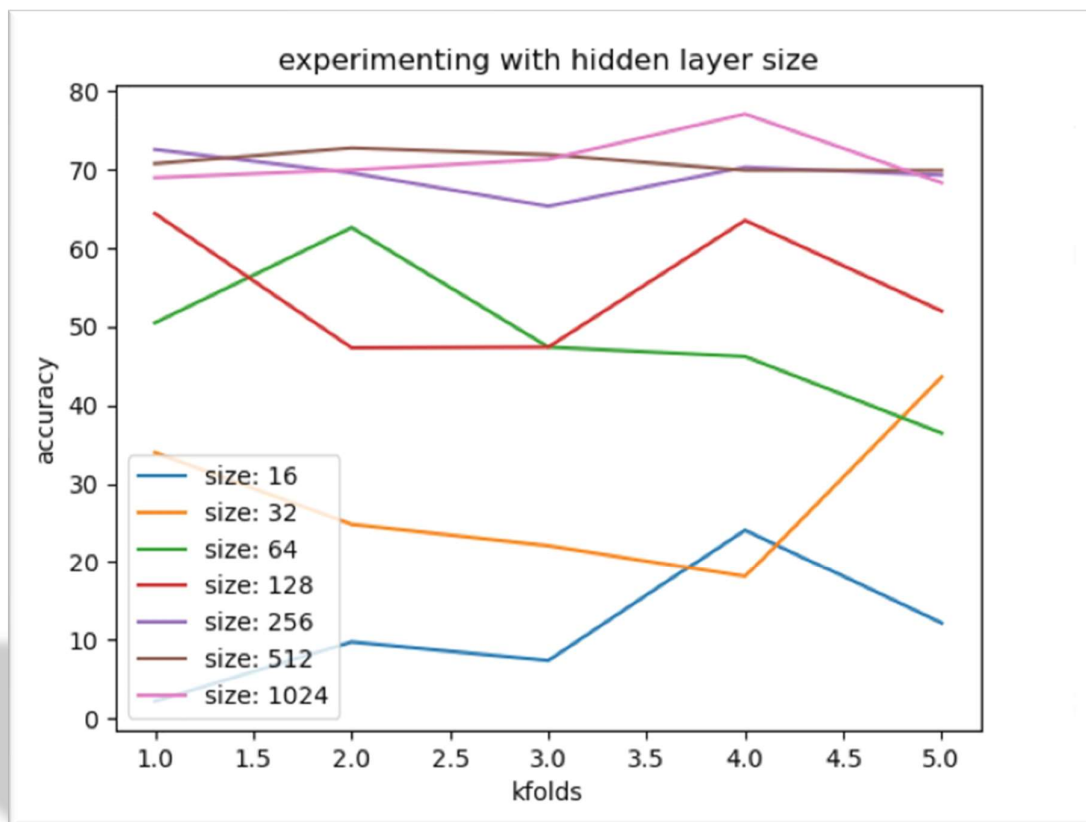
לאחר ניסוי זה, ניתן לראות מהגרף כי האופטימיזר *SGD* אינו מתאים באופן וודאי ולכן בחרנו את *ADAM* באופן שרירותי מאחר והתוצאות שלו ושל *RMSprop* היו כמעט זהות.



ניסוי שני

ניסוי זה הוא למטרת בחירת הגודל של ה-*hidden layer* ברשת הניורונים של ה-*RNN – LSTM*. במהלך ניסוי זה, לכל גודל מבין הגדלים {16, 32, 64, 128, 256, 512, 1024} ביצענו *5Fold cross validation* ובכל *Fold* חישבנו את דיוק המודל.

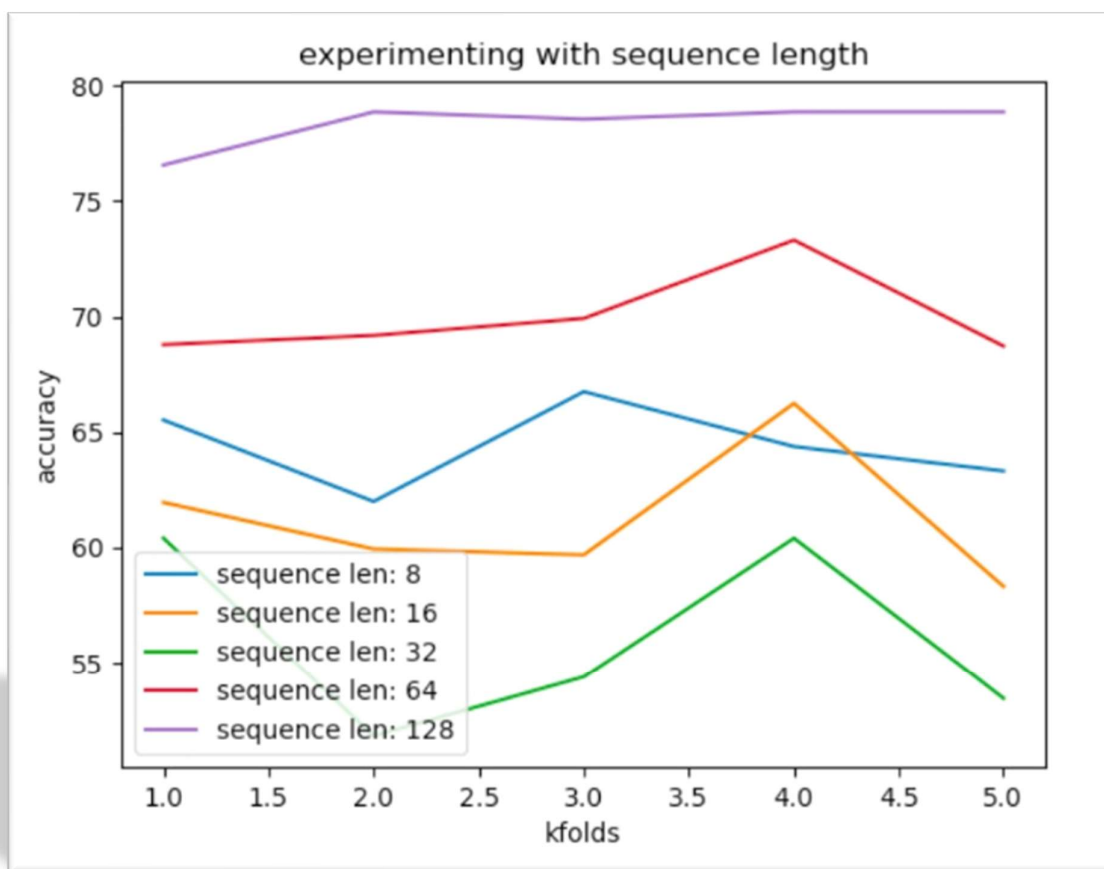
תוצאות הניסוי הראו כי הגדלים 1024, 512, 256 הם הגדלים המתאימים ביותר למודל שלנו לצורך יצירת מלודיות ולכן בחרנו את 256 מאחר ולשלושת הגדלים דיוק כמעט זהה אך עם הגדלת ה-*hidden layer*, זמן האימון גדל באופן משמעותי ולכן בחרנו את הגודל הקטן ביותר שמשגיג תוצאות טובות.



ניסוי שלישי

מטרת ניסוי זה הייתה לבדוק את השפעת אורכי רצפי הלמידה על דיוק המודל. במהלך הניסוי, השתמשנו ב- *5Fold cross validation* לכל אחד מהאורכים ומדדנו את דיוק המודל בממוצע.

תוצאות הניסוי הראי כי הדיוקים הטובים ביותר התקבלו עבור אורכי רצפי למידה גדולים אך עם הגדלת רצף הלמידה, זמן הלמידה גדל בצורה ניכרת. לכן, בחרנו לעבוד עם גודל השווה ל- 64 שמספק דיוק טוב בזמן למידה סביר.



עבור מודל ה-GAN

ניסוי למציאת אופטימיזר מתאים

בניסוי זה רצינו לבדוק באיזה אופטימיזר כדאי להשתמש עבור מודל ה-GAN. לצורך כך, בדקנו את שלושת האופטימיזרים הבאים: *SGD*, *ADAM* ו-*RMSprop*. במהלך הניסוי ביצענו שלושה אימונים, כאשר בכל אחד מהם השתמשנו באופטימיזר אחר עבור הגנרטור והמבקר, ולאחר 100 *epochs* נמדדו תוצאות היצירה הבאות:

עבור SGD:

ניתן לראות את תוצאות הניסוי באיור מטה, אשר מראות כי המודל לא התכנס כלל והמנגינה המוצגת מהווה רעש רנדומלי.

A musical score for a 4/4 piece at 120 BPM. The score includes staves for Drumset, Piano, and Strings. The Piano and Strings parts are filled with dense, chaotic scribbles, indicating random noise. A blue box highlights a section of the Piano staff in the first measure.

עבור ADAM:

ניתן לראות את תוצאות הניסוי באיור מטה, אשר מראות שהמודל אכן ביצע למידה והמנגינה שמוצגת הגיונית וטובה הרבה יותר מתוצאות של ה-SGD.

A musical score for a 4/4 piece at 120 BPM. The score includes staves for Drumset, Piano, Electric Bass, and Violins. The Piano, Electric Bass, and Violins parts show clear, coherent musical notation, including notes, rests, and accidentals, indicating a meaningful melody.

עבור *RMSprop*:

ניתן לראות את תוצאות הניסוי באיור מטה, אשר מראות שהמודל אכן ביצע למידה והמנגינה שמוצגת הגיונית וטובה.

The image displays a musical score for the *RMSprop* model. It consists of four staves, each representing a different instrument or ensemble. The tempo is marked as $\text{♩} = 120$. The time signature is 4/4. The staves are labeled as follows:

- Drumset, Drums:** The top staff, using a drum notation system with vertical lines and flags.
- Piano, Piano:** The second staff, using a standard musical notation with a treble clef.
- Electric Bass, Bass:** The third staff, using a standard musical notation with a bass clef.
- Violas, Ensemble:** The bottom staff, using a standard musical notation with a bass clef.

The score shows two measures of music for each instrument, with various notes, rests, and dynamic markings.

אם כן, לאחר מספר דגימות וניסויים הגענו למסקנה כי האופטימיזרים *ADAM* ו-*RMSprop* מייצרים תוצאות זהות באיכותן ולכן בחרנו באופן שרירותי באופטימיזר *ADAM*.

סיכום ודיון בתוצאות

לאורך הפרויקט בחנו דרכי עיבוד שונות לצרכי ניתוח ה-dataset וקידוד יעיל שלו. כפי שתיארנו בדו"ח, במהלך העבודה נתקלנו במספר בעיות כשהבולטות בהן:

- בעיית איבוד מידע בשלב הלמידה.
 - בעיית הזיכרון.
 - בעיות סנכרון בין מספר כלי נגינה.
 - בעיות סנכרון בין המנגינה לעצמה לאורך התקדמות המלודיה בציר הזמן.
- על כן, עבור בעיות אלו הצלחנו למצוא פתרון בדרכים שונות שהוצגו בדו"ח.

דיון בתוצאות

באופן כללי, לא ניתן למדוד את איכות מלודיה כזו או אחרת בדרכים מתמטיות וזאת מאחר שמוזיקה כשלעצמה היא אומנות כל שכל אדם יכול לאהוב או לא לאהוב יצירה מסוימת.

למרות זאת, קיימת חוקיות במוזיקה הניתנת למדידה, אך אינה מספקת כדי לאמוד את טיב היצירה. חוקיות זו באה לידי ביטוי ב:

- סולמות מוזיקליים מורכבים מתווים שיש ביניהם חוקיות מסוימת כדוגמת המרחקים בין תווים בכל סולמות המינור זהים לכל סולם מינור אחר.
 - אקורדים מייצגים תווים המנוגנים יחדיו באותה יחידת זמן, אולם לא כל קבוצת תווים יכולה להרכיב אקורד. הסיבה לכך היא שלכל תו ניתן להתאים גל קול עם תדירות מסוימת וכאשר קיימת התאמה בין תדירויות אלו (נקודות מינימום ומקסימום של גל הקול), מתקבל אקורד. בפן המתמטי, צריך להיות הפרש קבוע ומסוים בין התדירויות על מנת שיווצר אקורד, דוגמא להפרש שמקיים זאת הוא 15HZ.
 - הרמוניות של תווים משמשים במלודיות וניתן למדוד הפרש תדרים קבוע בין שני תווים על מנת לקבוע האם קיימת ביניהם הרמוניה.
 - מפתח של שיר - לכל שיר קיים מפתח המייצג קבוצת תווים המופיעה באופן דומיננטי לאורך המלודיה, כאשר ניתן לדעת בקלות האם תו מסוים שייך לקבוצה זו.
- למרות הקשרים המתמטיים הקיימים במוזיקה, אין דרך לכמת בצורה מתמטית את טיבה של מלודיה, כלומר לתת ציון גבוה ליצירה טובה וציון נמוך ליצירה גרועה. על כן, נציג דרכים נאיביות ונשלול אותן:
- ניקוד שיר על פי מגוון התווים בו כך שככל שיש מגוון גדול יותר של תווים, כך ניקוד השיר גבוה יותר.
- דוגמא נגדית לכך תהיה יצירת אומנות של בטאובן שהוספנו לה המון תווים באופן רנדומלי עד כדי כך שלא ניתן לזהות את היצירה. על כן, נקבל כי היצירה החדשה תקבל ציון גבוה יותר מיצירת האומנות של בטאובן בניגוד למצופה.
- מדידת כמות האקורדים שאכן מוגדרים בעולם המוזיקה וניתן ציון גבוה לשיר המכיל כמה שיותר אקורדים נכונים. דוגמא נגדית לכך תהיה יצירת אומנות שלא מכילה אף אקורד, תקבל ציון נמוך יותר מרעש רנדומלי של תווים שצדק באקורד אחד.
 - מתן ציון למלודיה באופן הבא - לכל מלודיה יש מפתח המגדיר קבוצת תווים. ניתן נקודות חיוביות שערכן אחד על תווים במלודיה מתוך קבוצת המפתח, ונחסיר נקודות שערכן אחד על תווים שאינם שייכים לקבוצה זו. הבעייתיות בשיטה זו נובעת מכך שקיימות מלודיות וסגנונות רבים שמשנים את מפתח המלודיה באמצע השיר.

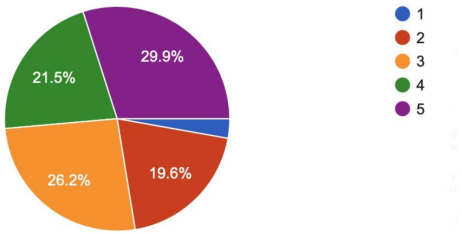
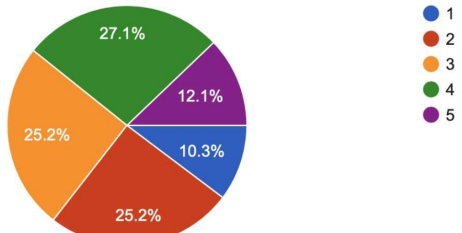
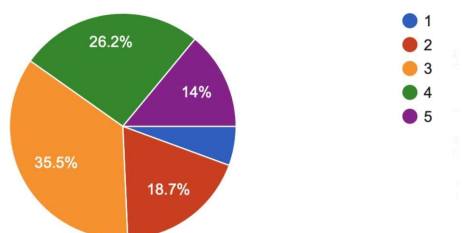
מכאן, נקבל כי כל שיטה בעד עצמה לא טובה מספיק על מנת לכמת את טיבה של מלודיה וכך גם השילוב ביניהן. לכן, השיטה המתבקשת להערכת התוצאות תהיה באמצעות סקר דעת קהל.

על כן, החלטנו לבצע סקר שיכיל 8 מלודיות כך ש:

- 3 מלודיות ממודל ה-*RNN*.
 - 3 מלודיות ממודל ה-*GAN*.
 - 2 מלודיות מתוך ה-*dataset* בקבוצת בקרה.
- בסקר עצמו ציינו כי כל המלודיות נוצרו באמצעות בינה מלאכותית וזאת על מנת לקבל תוצאות אמינות וכנות.

החלטנו שמדד סביר עבור הצלחת הפרויקט, יהיה בהתאם לציון של קבוצת הבקרה. כלומר, ככל שהציון של ששת המלודיות שנוצרו ע"י המודלים יהיה קרוב יותר לציון שקיבלו שתי המלודיות הנמצאות בקבוצת הבקרה, כך נדע להעריך את למידת המודל.

את הסקר ערכנו באמצעות פלטפורמת *google forms* שבסופו התקבלו 105 תגובות. בטבלה מטה, ניתן לראות את סיכום התוצאות:

מספר	מודל	גרף	ממוצע
1	<i>GAN</i>		3.57
2	<i>GAN</i>		3.03
3	<i>GAN</i>		3.25

2.82		<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div>	<i>RNN</i>	4
3.31		<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div>	<i>RNN</i>	5
3.16		<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div>	<i>RNN</i>	6
3.28		<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div>	בקרה	7
3.41		<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div>	בקרה	8

לאחר חישוב פשוט שביצענו, קיבלנו כי ממוצע המלודיות שנוצרו על ידי מודלי הלמידה, ה-*GAN* וה-*RNN*, הוא 3.19 לעומת ממוצע קבוצת הבקרה שהוא 3.34. כלומר, קיבלנו כי האנשים שביצעו את הסקר, כמעט ולא הצליחו להבדיל בין קבוצת הבקרה לבין מלודיה שנוצרה באמצעות אחד מהמודלים.

בנוסף, קיבלנו כי הממוצע הכולל של המלודיות שנוצרו על ידי המודלים היה:

- עבור מודל ה-*GAN* התקבל ממוצע של 3.28.

- עבור מודל ה-*RNN* התקבל ממוצע של 3.1.

כמו כן, נבחין כי מלודיה מספר 1 קיבלה ניקוד גבוה יותר מאלו של קבוצות הבקרה מה שמעיד על איכות היצירה של המודלים.

מכאן, נוכל להסיק כי המלודיות שהמודלים יצרו הינן באיכות טובה.

כיוונים נוספים להמשך המחקר

❖ הוספת מימד של גלים – עבודה עם קבצי *wav*:

במהלך הפרויקט ביצענו עיבוד מקדים של *dataset* המורכב ממלודיות בדידות.

אולם, ניתן לבצע עיבוד מקדים ל-*dataset* המכיל מוזיקה רציפה (לא ניתנת לכתיבה ע"י מקטעי זמן) ובדידה (ניתנת לכתיבה באמצעות מקטעי זמן).

כלומר, אפשר לבדוק כיצד ניתן לבצע יצירה (ג'נרט) של מוזיקה רציפה ולא בדידה.

❖ ניסיון להשתמש במודלים נוספים על מנת לייצר מלודיות והשוואתם מול המודלים שבנינו בפרויקט.

דוגמא למודל שיכול להיות מעניין הוא מודל *VAE* אשר ידוע, בין היתר, כמודל ליצירת מוזיקה.

❖ שילוב רשתות נוירונים *RNN – LSTM* כחלק מארכיטקטורת גנרטור של *GAN*.

ביבליוגרפיה

❖ Dataset:

- עבור מודל ה- $RNN - LSTM$: לצורך יצירת מלודיות המורכבות מכלי נגינה בודד, השתמשנו ב- $EsAC$ (Essen Associative Code) בתור $dataset$ המכיל יותר מ-20 אלף מלודיות מכל העולם - [/http://www.esac-data.org](http://www.esac-data.org)
עבור יצירה המורכבת ממספר כלי נגינה, השתמשנו ב- $dataset$ של ה- GAN שיוצג בנקודה הבאה.
- עבור מודל ה- GAN : ה- $dataset$ עבור מודל זה נלקח מ-
<https://www.vgmusic.com/music/console/nintendo/snes> אשר מכיל מעל 6709 קבצי $midi$ המורכבים ממספר כלי נגינה כאשר המנגינות נלקחו מתוך משחקים של סופר ביטנדו.

❖ מאמרים ושיח בנושא ההשפעה של הבינה המלאכותית על תעשיית המוזיקה:

1. [MUSEGAN](#)
2. [MUSEGAN SLIDES](#)
3. [https://www.niravpatel.online/futureeye/artificial-intelligence-influencing-the-music-world-know-the-secret-behind-it /](https://www.niravpatel.online/futureeye/artificial-intelligence-influencing-the-music-world-know-the-secret-behind-it/)

❖ קישור לסקר שערכנו:

[סקר בקרת איכות המלודיות](#)

הערה: בכל מקום בדו"ח בו כתוב מודל RNN הכוונה היא למודל ה- $RNN - LSTM$.