הפקולטה להנדסה

המעבדה לעיבוד אותות

אבחון פרמטרים של חדר לא ידוע ע"י שימוש בהקלטות רב ערוציות

צופית חדד

בר ינוביץ

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחה : מרדכי מוראדי

מנחה אקדמי: פרופ' שרון גנות

אוקטובר 2024

תוכן עניינים

רקע לפרוייקט..... 3

מטרת הפרויקט..... 4

מושגים הקשורים לייצור הדטא.....5-8

חלק א'- ייצור הדאטא..... 9-15

חלק ב- מושגים חשובים לצורך בניית מודל.....16-29

חלק ב- יצרת המודל שלנו.....30-38

חלק ג- אימון המודל..... 39-45

הצעות לשיפור..... 46

רקע לפרויקט

בפרויקט זה אנחנו מציגים שיטה חדשה אשר מבצעת אומדן באופן "עיוור"- כלומר מבלי לקבל מידע מוקדם כלשהו, ומבצעת שערוך של פרמטרי החדר.

כל פרויקט זה מבוסס **מבוסס על המאמר:**

Blind Room Parameter Estimation Using Multiple Multichannel Speech Recordings- https://arxiv.org/pdf/2107.13832.pdf

את שערוך פרמטרי החדר נבצע על ידי שימוש בהקלטות דיבור רב- ערוציות ממספר מיקומים הנבחרים באופן אקראי של מקורות ומקלטים.

מציאת הפרמטרים אותם ציינו עשויים להיות שימושיים בתחומים שונים כמו:

- לאחרונה יש עניין רב ב'מציאות אודיו מוגברת'- שבה ניתן לשמוע מקורות קול כאילו המאזין נמצא בסביבה

באופן אמיתי.

- זיהוי פלילי.
- בשביל להשיג את התחושה כאילו המאזין נמצא בסביבה באופן אמיתי עלינו להשיג את המידע על החדר שהמאזין נמצא בו ובפרט את הפרמטרים האקוסטיים שציינו.

במאמר אנו מציגים שיטה חדשה לאמוד פרמטרים אלו ללא מידע על החדר.

במאמר עליו אנו מתבססות לפרויקט שלנו אנו רוצים למצוא מהם הנתונים האקוסטיים והגיאומטריים לחדרי "shoebox" –חללים בעלי גיאומטריה מלבנית ולא חדרים אשר מורכבים מבחינה גיאומטרית. אנחנו נייצר ;20,000 חדרים כאשר גודל החדרים שלנו מוגרל באופן הבא:

האורך והרוחב כל אחד מוגרל בין $[3,10]\mathrm{m}$ והגובה בין $[2.5,4]\mathrm{m}$. כך שנקבל כי השטח הוא בין $[48,360]\mathrm{m}^2$ והנפח הוא בין $[22.5,400]\mathrm{m}^3$.

השערוך אותו אנו רוצים לעשות מבוסס על Deep Neural Network(DNN) שמקבלת כקלט ספקטוגרמות חד ערוציות ובין ערוציות ומחזירה שיערוך עבור הפרמטרים ביחד עם השגיאה שלהם.

במאמר מאמנים את המודל על מאגר נתונים גדול שנוצר בצורה מלאכותית בעזרת סימולציות אקוסטיות, כלומר על ידי **RIR** ('room impulse response'- ראו הסבר בהמשך), והוכח במאמר כי הוא משפר באופן משמעותי את השערוך בהשוואה לשיטות קודמות גם על נתונים מלאכותיים וגם על נתונים אמיתיים. השימוש שנעשה באותות בין-ערוציים והעובדה שאנו משלבים מידע ממספר מיקומי מקור- קלט מפחיתים את השגיאה באופן משמעותי.

מטרת הפרויקט

מטרת הפרויקט שלנו היא למצוא את הפרמטרים הגיאומטרים של החדר.
אנו רוצים לקבל סך הכל 14 פרמטרים על כל חדר אשר ייצגו את הגיאומטריה של החדר.
הפרמטרים הינם:

- שטח הפנים הכללי של החדר.
- נפח החדר.
- זמן ההדהוד עבור כל אחד משמשת פסי האוקטבה (*הרחבה בהמשך) (Reverberation time- RT60).
- מקדם הספיגה הממוצע עבור כל אחד מקירות החדר (6 קירות).

במאמר זה נשתמש בשיטות של למידה עמוקה ורשתות נוירונים, שיטות אלו צפויות להעניק לנו ביצועים טובים ושערוך מיטבי עבור הפרמטרים של החדר.

אז מה הוא בעצם התהליך אותו אנחנו הולכים לעבור?

*כמובן שנסביר באופן מעמיק על כל שלב ושלב בהמשך

1. יצירת ה DATA- בשלב הראשון נרצה לייצר מאגר הנתונים- כתבנו סקריפט הנקרא data generation. מאגר הנתונים מכיל הקלטות מהדהדות, ספקטוגרמות והפרמטרים הגיאומטריים והאקוסטיים של כל חדר, נייצא ממנו קובץ CSV שיכיל את כל המידע באופן מסודר ובו נוכל להשתמש לצורך למידת המודל.

2. בניית המודל- המודל נבנה תוך שימוש ברשת ניורונים, המודל מאומן על מאגר הנתונים שנוצר, והמטרה היא ללמד אותו לחלץ את הפרמטרים האקוסטיים והגיאומטריים על סמך הקלטות השמע בלבד.

3. אימון המודל- המודל מאומן באמצעות פונקציית האובדן, כאשר בכל שלב הוא משפר את ההערכות שלו עבור הפרמטרים. האימון נעשה על מאגר נתונים גדול, כאשר כל דגימה מכילה הקלטה מהדהדת והפרמטרים הגיאומטריים והאקוסטיים הנכונים.

4. הערכת המודל- לאחר האימון, המודל נבדק על סט נתונים שלא ראה קודם כדי להעריך את הביצועים שלו ולוודא שהוא מצליח לאמוד את הפרמטרים המבוקשים בצורה מדויקת.

5. שימוש במודל- בסוף התהליך, המודל יוכל לקבל הקלטה של חדר לא ידוע, ולספק הערכות מדויקות של הפרמטרים הגיאומטריים והאקוסטיים של החדר (שטח, נפח, מקדמי ספיגה וזמני הדהוד) בהתבסס על הקלטות אודיו בלבד.

מושגים אשר ישמשו אותנו ליצירת הDATA:

- **פס אוקטבה**- האוזן האנושית יכולה לשמוע בטווחה בין $20[\mathrm{Hz}] - 20[\mathrm{KHz}]$, את הטווח הזה חילקו ל10 פסי אוקטבה כאשר עבור כל אחד מהפסים התדירות העליונה שלו גדולה פי 2 מהתדירות התחתונה שלו. כדי לתאר פס אוקטבה אנו משתמשים בתדירות המרכזית שלו,

תדירות תחתונה	22.5	45	90	180	355	710	1,400	2,800	5,600	11,200
תדירות מרכזית	31.5	63	125	250	500	1,000	2,000	4,000	8,000	16,000
תדירות עליונה	45	90	180	355	710	1,400	2,800	5,600	11,200	22,400

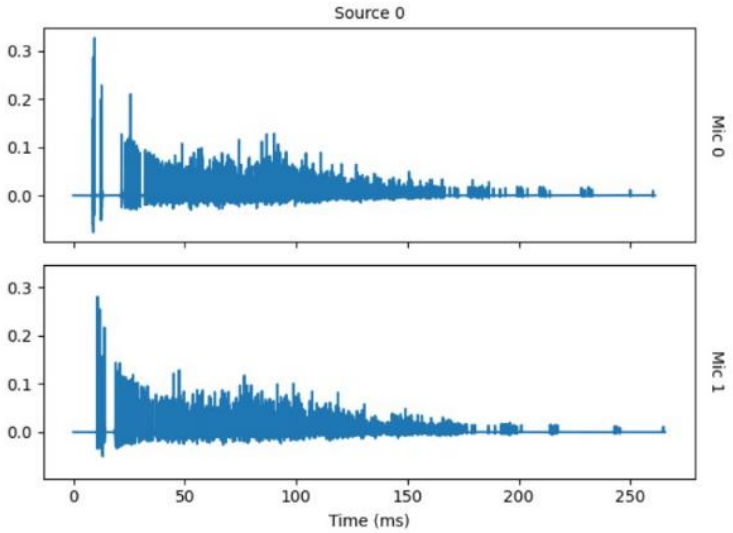
כמוצג בטבלה הבאה:

אצלנו במאמר אנו משתמשים בטווח תדרים של 6 פסי אוקטבה- $125[\mathrm{Hz}] - 4000[\mathrm{Hz}]$.

- **RIR**- 'room impulse response'- מערכת אשר מדמה תגובה להלם של חדר, כלומר אנו לוקחים קטעי קול ומבצעים להם קונבולוציה עם RIR וכך אנו מקבלים הדמיה של איך קטע הקול הזה ישמע בחדר שאותו נבחר.

RIR בעצם פונקצית התמסורת בין מקור הקול לבין מערך המיקרופונים. כך שקונבולוציה בין הקלטה נקיה לבין פונקצית תמסורת זו תתן לנו, באופן אידיאלי, את תוצאת ההקלטה כאילו הוקלטה בסביבה מסוימת שהגדרנו, על ידי מערך מיקרופונים שנבחר.

שימוש במערכת זו חוסך לנו עבודה רבה של ביצוע הקלטות עבור כל סוגי החדרים שבהם אנו מאמנים את המודל שלנו בקוד זה.



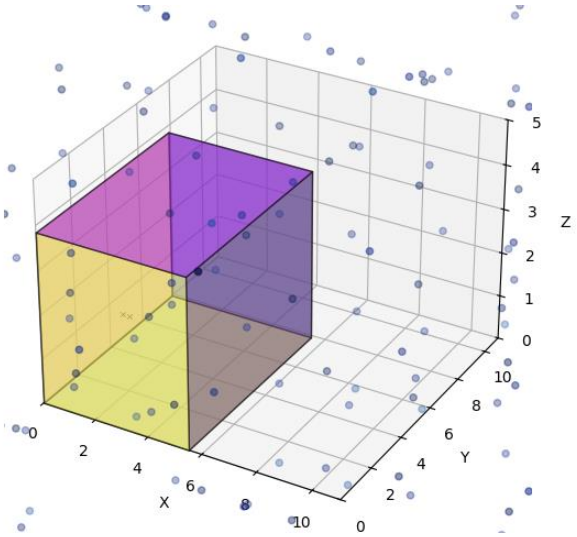
אנו רואים בתמונה את ה RIR עבור כל אחד מהמיקרופונים לsource0 ואנו רואים כי הוא אכן דועך כמו הקול שלנו בדיבור.

- **<u>Image source method</u>-** במאמר שלנו משתמשים בטכניקה הנקראת image source method- טכניקה זו נפוצה בתעשיית האקוסטיקה ועיבוד האותות והיא משמשת למגוון צרכים כגון הפרדת מקורות דיבור, לוקליזציה של מקום אקוסטי, שיפור דיבור ועוד.

בשיטה זו משתמשים בהתפשטות הקרניים כאשר אנו מתייחסים להגעת הקרניים, כלומר אצלנו-הקול, מהמקור למקבל ב2 דרכים:

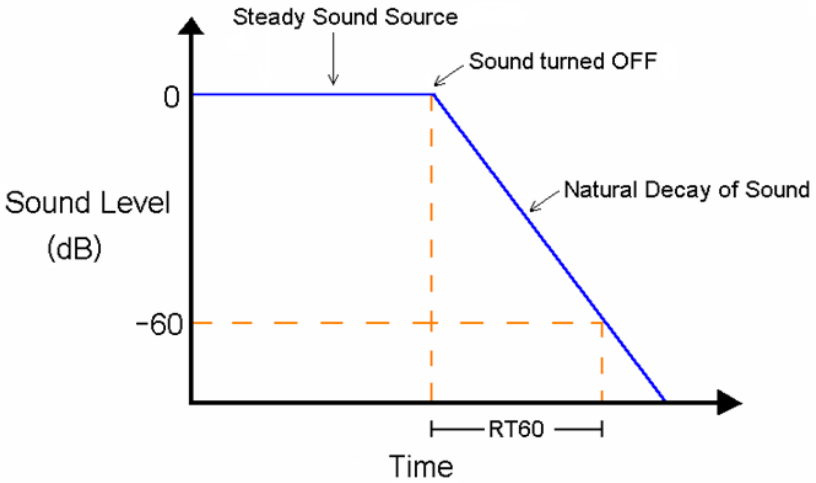
- אלו המתקבלות באופן ישיר- מהמקור האמיתי.
- אלו המתקבלות באופן לא ישיר לאחר התנגשויות- מדמויות הנוצרות מאחורי הreceiver.

אנו מקבלים כי ככל שמספר המקורות גדל המיקום מדויק יותר.



- **<u>RT60-Reverberation Time</u>-** זמן הדהוד של החדר.

פרמטר זה מציג לנו כמה זמן נדרש לאנרגית הקול בחדר לרדת ב 60db – זמני הדהוד תלויים באופן ישיר בגודל החדר ובאופי החומרים בהם משתמשים בבניית החדר.
זהו המדד העיקרי לניתוח אקוסטי של חדר.
במקרה שלנו ביצענו את חישוב זה בחדר באמצעות שימוש בפונקציה מובנת את הספרייה בה אנחנו עובדים וגם באמצעות חישוב אנליטי עליהם נסביר בחלק של בניית המידע לפרויקט.



- **Short Time Fourier transform-STFT**
התמרת פורייה לזמן קצר, זו טכניקה לניתוח ספקטרלי של אותות אודיו.
בטכניקה זו מפצלים את האות לחלונות קצרים ועבור כל חלון מבצעים התמרת פורייה אשר נותנת את תדרי האות באותו רגע. זה מאפשר לנו לראות את הספקטרום של האות כפונקציה של הזמן. במאמר זה משתמשים בשיטה זו על האות האקוסטי על מנת לקבל ספקטוגרמות חד-ערוציות ורב- ערוציות אשר מהוות את הכניסות לרשת של המודל.

- **Spectrogram**- ספקטוגרמה זה בעצם צורת הצגה חזותית של התדרים של אות כתלות בשינוי בזמן- זו בעצם יכולת הצגה של אותות שמע.
נוכל לראות שימוש בספקטוגרמה במגוון רחב של תחומים- אם זה במוזיקה, מכ"ם, עיבוד אות דיבור וכו'. באופן אנליטי, לצורך יצירת הספקטוגרמה נרצה להשתמש בSTFT – short time Fourier transform.

- **Diffuse rain method** - זו שיטה נומרית אשר מדמה הדהוד (reverberation) ב Room Impulse Response. השיטה הזו מבוססת על סימולציית קרניים אקראיות שמוחזרות באופן מפוזר מכל המשטחים שיש בחדר. כל קרן נשלחת לכיוון אקראי חוזרת מהמשטח ומצטרפת לסך ה"צבירה" של ההדהוד. השיטה הזו יעילה לנו ביישום הקוד אשר מתואר במאמר זה כיוון שהיא בעלת יעילות חישובית גבוהה וגם עם יכולת לדמות באופן אמיתי ובצורה סטטיסטית טובה את ההדהוד בחדרים מסובכים. שיטה זו ביחד עם הImage Source Method משלימות אחת את השנייה ונותנות RIR מלא כאשר שיטה הImage Source אחראית על ההחזרות הראשוניות ושיטת הDiffuse rain אחראית על ה"זנב" כלומר על ההחזרות המאוחרות יותר.

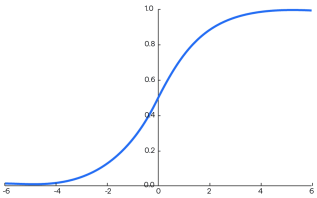
- **SNR- יחס אות רעש-** מייצג את היחס בין עוצמת האות הרצוי לבין עוצמת הרעש הכוללת- בעצם מתאר כמה הרעש הלא רצוי משמעותי.

- **רעש דיפיוזי (Diffuse Noise)**- מתייחס לרעש שמפוזר באופן אחיד בחלל. הוא נוצר כשגלי קול מתפשטים ומפוזרים על פני שטחים שונים, כמו קירות, תקרה ורצפה. רעש דיפיוזי מתאפיין בכך שאין לו מקור סביר וברור, והוא מורכב ממספר גלי קול הפוגעים במקורות שונים בו-זמנית.
רעש דיפיוזי מתפשט באופן אחיד- הרעש מגיע מכל הכיוונים ויוצר סביבה אקוסטית שבה הקולות נשמעים אחידים. בנוסף, קשה לזהות את מקור הרעש, מה שהופך אותו לאידיאלי להדמיה אקוסטית.
בהקשר של המאמר שלך, רעש דיפיוזי נחשב לרעש שמגיע ממקורות בלתי נראים, והוא משפיע על אופן ההערכה של פרמטרים אקוסטיים, כמו זמני תהודה ומקדמי ספיגה. מאמר זה מדגיש את החשיבות של קלטים דיפיוזיים באימון המודל, כיוון שהם מספקים הקשר עשיר יותר לפיתוח אלגוריתמים להערכה אקוסטית.

- **<u>Soft max</u>**- פונקציית Softmax היא פונקציה נפוצה ביישומים של למידת מכונה ושיווק עמוק. הפונקציה משמשת בדרך כלל להמיר סדרה של ערכים לסדרה של ערכים חדשים בין 0 ל־1, שבהם סכום הערכים החדשים שווה ל־1. זה נקרא "פונקציה שלמות", משום שהסכום שלהם שווה ל־1.
הנוסחה עבור פונקציה זו הינה:

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Softmax Function



- **<u>Cross entropy loss</u>**- קרוי גם אובדן log. הוא מודד את הביצועים של מודל סיווג שהפלט שלו הוא הסתברות בין 0 ל1. פעולה זו היא בעצם נותנת מידה לכמה התחזית של המודל קרובה למציאות, ככל שההבדל בין ההסתברות החזויה וההסתברות האמיתית גדול יותר, פוקנצית ההפסד גדולה יותר. כאשר המודל טועה לגבי מחלקה שהיא כן נכונה הוא "מעניש" אותו יותר ובכך מלמד אותו למזער את ההפסד.

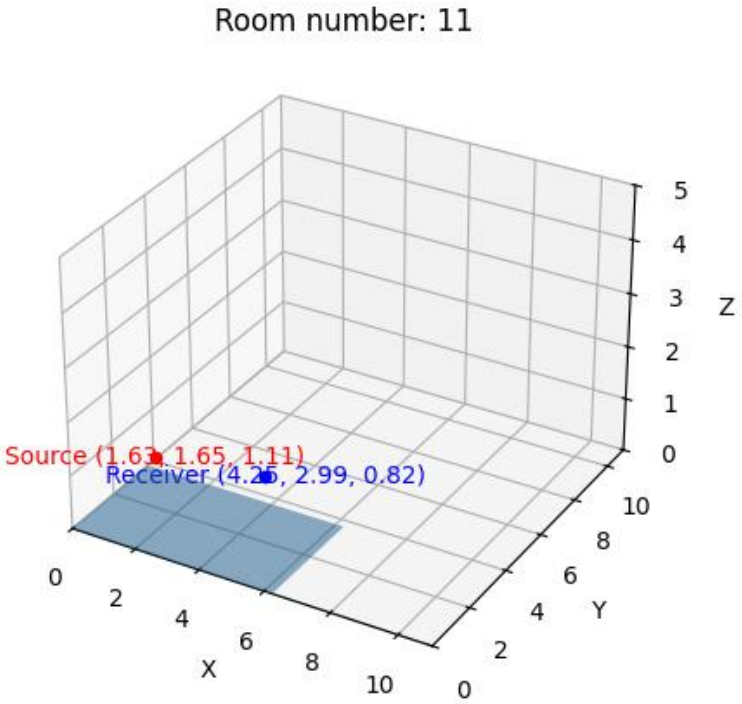
True probability distribution (one-shot)

$$H(p, q) = -\sum_{x \in classes} p(x) log\ q(x)$$

Your model's predicted probability distribution

חלק א'- בניית DATA

בחלק זה אנו ראשית מגרילים את הנתונים הפיזיים של החדר, אורך, רוחב, גובה וממקמים את הsource והreciver לפי התנאים הכתובים במאמר.
כלומר, במרחק 30 ס"מ מהקירות ואחד מהשני, עבור כל חדר שנוצר, נבצע קונבולוציה בין אותות הקול הנקיים שהגדרנו מראש לבין ה RIR-המיוחס לאותו מיקום במרחב.
בעת ייצור החדרים באופן רנדומלי לפי ההגדרות במאמר ייצרנו תמונה תלת מימדית עבור כל חדר- נניח עבור חדר רנדומלי - במקרה הזה 11:



נוכל לראות את התמונה באופן תלת ממדי ככה שבתוכו נמצא המיקרופונים והדובר- הכל לפי ההגדרות של המאמר.

לאחר החישוב הזה, נקבל את האותות המדומים שנשמעים כפי שהיו נשמעים אם הוקלטו באותו חדר אמיתי עם הפרמטרים שנקבעו. עבור כל חדר שנוצר, נבצע **קונבולוציה** בין אותות הקול הנקיים שהגדרנו מראש לבין ה RIR-המיוחס לאותו מיקום במרחב. לאחר החישוב הזה, נקבל את האותות המדומים שנשמעים כפי שהיו נשמעים אם הוקלטו באותו חדר אמיתי עם הפרמטרים שנקבעו..

לאחר מכן, אנו רוצים לבצע חישוב של **מקדמי הבליעה-absorption coefficients** עבור כל אחד מהמשטחים בחדר, כלומר לכל אחד מששת הקירות (4 קירות+ רצפה+ תקרה).

נסביר כי מקדמי בליעה הם בעצם ערכים המתארים את מידת הספיגה של הקול על ידי חומר מסוים, כמה עוצמת הגל קול נחלשת עם המרחק שהקול עובר בתוכו.
מקדמי הקרינה תלויים בתכונות הגל, אורכו או התדירות שלו, בנוסף קיימת תלות בסוג החומר, הצפיפות שלו, מבנה וטמפרטורה- חומרים שונים יבלעו בצורה שונה.

אנו משתמשים בשיטה של "דגימה מוטית רפלקטיבית" עבור חישוב זה- את שיטה זו לקחנו ממאמר המקושר למאמר המקורי שלנו- מאמר מספר 17- המאמר נקרא:

"MEANABSORPTIONCOEFFICIENTESTIMATIONFROMIMPULSE RESPONSES: DEEPLEARNINGVS.SABINE"

הדגימה המוטת רפלקטיבית מייצגת לנו מצב של הטלת מטבע.
כלומר, עבור כל משטח בנפרד קיימת הסתברות של 0.5 להיות תלוי בתדר (בפס אוקטבה שלנו) והסתברות של 0.5 לא להיות תלוי בתדר.
הפרמטר $\alpha_i(b)$ - מתאר לנו את מקדם הספיגה- כך שהאות i מייצגת לנו את הקיר אותו אנחנו מתארים (כלומר סך הכל עבור כל חדר אנחנו רצים עד 6 קירות) ואילו הb מתאם לנו את האוקטבה הרלוונטית- כלומר בסך הכל יש לנו 6 קירות ו6 אוקטבות ונקבל 36 מקדמי ספיגה- עבור כל קיר נקבל 6 מקדמי ספיגה עבור כל פס אוקטבה.

לפי השיטה המתוארת במאמר 17, בהסתברות של 50% המשטח הינו תלוי תדר ואז נבחר את ערכי $\alpha_i(b)$ לכל אחד מפסי האוקטבה בהתאם לטבלה הבאה:

freq. (Hz)		125	250	500	1k	2k	4k
Walls	α_i^{min}	0.01	0.01	0.01	0.01	0.01	0.01
	α_i^{max}	0.50	0.50	0.30	0.12	0.12	0.12
Floor	α_i^{min}	0.01	0.01	0.05	0.15	0.25	0.30
	α_i^{max}	0.20	0.30	0.50	0.60	0.75	0.80
Ceiling	α_i^{min}	0.01	0.15	0.40	0.40	0.40	0.30
	α_i^{max}	0.70	1.00	1.00	1.00	1.00	1.00

Table 1. Absorption coefficients ranges in the RB dataset.

(הטבלה לקוחה מתוך המאמר https://inria.hal.science/hal-03045556/document).

כלומר, במידה ויצא כי הקיר הi תלוי בתדר עבור כל אחד מפסי האוקטבה שלנו נגריל את מקדם הבליעה שלו בהתאם להתפלגות אחידה לפי המיקום בטבלה.
במידה ויצא לנו כי הקיר הספציפי הינו בלתי תלוי בתדר- כלומר בהסתברות של 50% המשטח אינו תלוי תדר ואז נגריל את $\alpha_i(b)$ בטווח בין [0.1,1.2].

עבור כל משטח אנו מחשבים את מקדם הספיגה לכל אחד מפסי האוקטבה.
לאחר מכן אנו מחשבים את מקדם הספיגה הכולל של כל פס אוקטבה עבור כל הקירות על ידי הנוסחה הבאה:

$$\alpha^{-}(b) = \sum_{i=1}^{k} \frac{\alpha_i(b) \cdot S_i}{S}$$

כאשר:

S_i- הינו השטח של כל אחד מהקירות.

S- השטח הכולל של החדר.

$\alpha_i(b)$- לקוח מההסבר המפורט לעיל.

```python
def calculate_absorption_coefficients(self, Sroom_dim, surface_areas, octave_bands, i):
    # Calculate absorption coefficients for each surface and octave band
    absorption_coeffs = {}
    for surface, area in surface_areas.items():
        absorption_coeffs[surface] = {}
        if np.random.rand() < 0.5:
            print(F'for room number{i}, {surface}-acording freq-table')
            for freq in octave_bands:
                if (surface == 'ceiling'):
                    for freq in octave_bands:
                        if freq == 125:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.70)
                        if freq == 250:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.15, 1)
                        if freq == 500:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.40, 1)
                        if freq == 1000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.40, 1)
                        if freq == 2000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.40, 1)
                        if freq == 4000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.3, 1)
                if (surface == 'floor'):
                    for freq in octave_bands:
                        if freq == 125:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.20)
                        if freq == 250:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.3)
                        if freq == 500:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.05, 0.50)
                        if freq == 1000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.15, 0.60)
                        if freq == 2000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.25, 0.75)
                        if freq == 4000:
```

```python
                            absorption_coeffs[surface][freq] = np.random.uniform(0.3, 0.8)
                if surface == 'east' or surface == 'west' or surface == 'north' or surface == 'south':
                    for freq in octave_bands:
                        if freq == 125:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.50)
                        if freq == 250:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.50)
                        if freq == 500:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.30)
                        if freq == 1000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.12)
                        if freq == 2000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.12)
                        if freq == 4000:
                            absorption_coeffs[surface][freq] = np.random.uniform(0.01, 0.12)
        else:
            print(f'for room number{i}, {surface}- not acording freq-just random')
            for freq in octave_bands:
                abs_coeff = np.random.uniform(0.02, 0.12)
                absorption_coeffs[surface][freq] = abs_coeff
    # Calculate average absorption coefficient for each octave band
    avg_absorption_coeffs = {}
    total_surface_area = sum(surface_areas.values())
    for freq in octave_bands:
        weighted_sum = sum(absorption_coeffs[surface][freq] * area for surface, area in surface_areas.items())
        avg_absorption_coeffs[freq] = weighted_sum / total_surface_area
    return absorption_coeffs, avg_absorption_coeffs
```

את חישוב הrt60 ביצענו בשתי צורות- באופן הפשוט השתמשנו בפונקציה המובנת של הספרייה בה אנחנו משתמשים, ואילו בחישוב האנליטי אשר ביצענו בעצמנו יצרנו פעולה אשר מחשבת את הrt60 המתקבל עבור כל פס אוקטבה- כלומר עבור כל תדירות בנפרד.

עבור כל אחד מפסי האוקטבה נעזרנו במשוואה הבאה המוזכרת במאמר:

$$\mathrm{RT}_{60}(b) \approx 0.16 \frac{V}{\bar{\alpha}(b) \cdot S}$$

כך ש:

V נפח החדר

S שטח הפנים של החדר

$\overline{\alpha\,(b)}$ הינו המקדם ספיגה הממוצע עבור פס אוקטבה- חישבנו ערך זה בפונקציה של חישוב מקדמי הספיגה.

```
def rt60_for_each_f(self, octave_bands, volume, surface_ca, avg_absorption_coeffs):
    arr_rt60 = {}
    for freq in octave_bands:
        arr_rt60[freq] = ((0.16 * volume) / (avg_absorption_coeffs[freq] * surface_ca))
    return arr_rt60
```

```
t60_effective = pra.experimental.measure_rt60(room.rir[0][0], fs=room.fs, plot=True)
#print(f"rt60 effective is: {t60_effective:.3f} s")
```

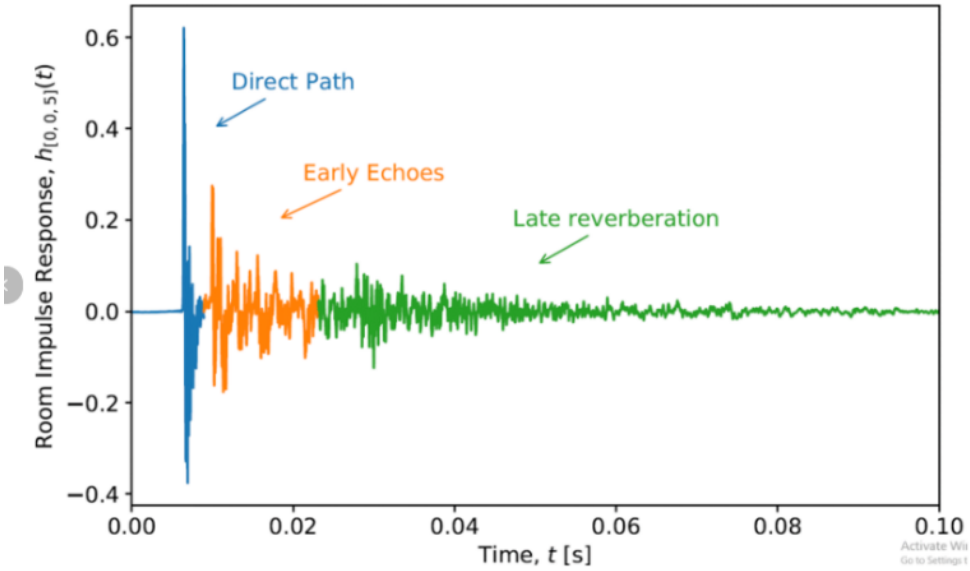
לאחר מכן, אנו רוצים גם להוסיף רעש.

ישנם 2 סוגי רעשים אותם אנו מוסיפים.

1. רעש לבן שהוספנו לכל אחד מהמקירופונים.
2. רעש דיפיוזי- רעש זה מדמה לנו המצאות בחדר אשר אנו שומעים רעש מכל הכיוונים.

את הרעש הזה נוסיף באופן הבא: עבור כל אחד מהמיקרופונים: נגריל מיקום בחדר, כלומר דובר אשר מייצר לי רעש במרחק מסוים אותו קבענו מהמיקרופונים והקיר ונייצר RIR.

מהRIR ניקח רק את החלק השלישי- ראה תמונה מצורפת.
כלומר ההגעות האחרונות (בחלק הירוק- Late reverberation) חלק זה יהיה מ 50ms והלאה. זה למעשה יהיה הRIR החדש אותו ייצרנו, ניקח הקלטה חדשה אותה נעביר בRIR החדש וזה יהיה הרעש הדיפיוזי שהגרלנו ואותו נוסיף לקול המהדהד הראשון שייצרנו.




```python
def add_noise(self, reverberant_signal, room, sample_rate, static_noise_snr_range, diffuse_noise_snr_range):
    #reverberant_signal- the signal we want to add noise to
    #static_noise_snr_range- snr= signal/noise static noise
    #diffuse_noise_snr_range- snr= signal/noise diffuse noise
    static_noise_power = np.mean(reverberant_signal ** 2) / (10 ** (static_noise_snr_range / 10))
    static_noise_power = np.maximum(static_noise_power, 1e-10)
    static_noise = np.random.normal(0, np.sqrt(static_noise_power), reverberant_signal.shape)

    late_rir = room.rir[0][int(0.05 * sample_rate):] if len(room.rir[0]) > int(0.05 * sample_rate) else []
    if len(late_rir) > 0:
        diffuse_noise = np.random.normal(0, 1, len(late_rir) + len(reverberant_signal) - 1)
        diffuse_noise = np.convolve(diffuse_noise, late_rir, mode='full')[:len(reverberant_signal)]
        diffuse_noise_power = np.mean(reverberant_signal ** 2) / (10 ** (diffuse_noise_snr_range / 10))
        diffuse_noise_power = np.maximum(diffuse_noise_power, 1e-10)
        diffuse_noise *= np.sqrt(diffuse_noise_power) / np.std(diffuse_noise)
    else:
        diffuse_noise = np.zeros_like(reverberant_signal)

    noisy_signal = reverberant_signal + static_noise + diffuse_noise
    return noisy_signal
```

נזכור כי המטרה היא לשערך מספר פרמטרים גיאומטריים ואקוסטיים של החדר באופן "עיוור", כלומר מבלי לדעת מראש את המידע על החדר אלא על בסיס הקלטות שמע בלבד.

כלומר, במוצא של הסקיפט הזה בעצם נייצר קובץ CSV שיכיל בתוכו מידע אותו נרצה לשערך וכן מידע שיכול לעזור לנו לאימון המודל.
הפרמטרים אותם נרצה לשערך כוללים:

פרמטרים גיאומטריים:

1. שטח הפנים הכולל של החדר – (S) סך כל שטחי המשטחים של החדר (כלומר, 4 קירות, תקרה ורצפה).
2. נפח החדר – (V) הנפח הכולל של החדר, אשר מחושב על פי ממדי האורך, הרוחב והגובה.

פרמטרים אקוסטיים:

3. זמן ההדהוד – (RT60) הזמן הדרוש לאנרגיית הקול לרדת ב-60 דציבלים לאחר שהקול נפסק. נרצה לשערך את זמן ההדהוד עבור כל אחד מששת פסי האוקטבה (תדרים בין 125 הרץ ל-4000 הרץ).
4. מקדם הספיגה הממוצע – ($\bar{\alpha}$) נמדד עבור כל אחד מששת פסי האוקטבה. מקדם זה מתאר כמה מהאנרגיה של הקול נספגת על ידי המשטחים השונים בחדר, כלומר כמה מהקול נשאר ומהדהד, וכמה נספג.

סה"כ, נרצה לקבל סך הכל **14פרמטרים**:

- 6מקדמי ספיגה ממוצעים עבור כל פס אוקטבה.
- 6זמני הדהוד (RT60) עבור כל פס אוקטבה.
- שטח הפנים הכולל של החדר.
- נפח החדר.

קובץ שמירת המידע שלנו יתקבל באופן הבא תחת סוג קובץ של csv:

```
room number,surface,volume,length,width,height,rt60-octave=125,rt60-octave=250,rt60-octave=500,rt60-octave=1000,rt60-octave=2000,rt60-octave=4000,mean a
3,110.85712397048044,74.42648461088561,3.144641281152374,6.404515920793731,3.69547344217172,0.5019000875370727,0.7010862682196229,0.4543125290134379,0.4
4,238.19494044955678,226.44757690779915,8.203484721979262,7.445640449910038,3.7073812057435442,1.2456710345361819,1.2988128107043702,1.862801429965971,1
5,254.72496222953924,246.05475337112836,6.8870853775458505,9.573580833476765,3.731830276993532,1.7323815369412945,1.8843223088899751,1.3399360493002077,2
6,190.28107438666336,161.64445834587903,7.970465888496748,6.027584682897769,3.364602724361892,0.7063803550814155,0.6039772592343582,0.3622290176279728,0
7,66.52743122128098,36.72296520693541,3.083806129717625,3.1785709423194515,3.7464398558392227,0.7637336489741519,0.399832446184599,0.4442754629454674,0.
8,90.46778992300884,55.99094617198174,5.422577826021456,3.4062640067045704,3.031333417931412,0.9242453361761207,1.1481678450860275,0.8618417989067523,1.
9,84.31836202305848,51.93532067649881,4.416287095866835,3.1493624340450195,3.7340736207982395,0.3369821004153220 6,0.4507830178164306,0.3862574393081510 4
10,123.50726260401571,91.58523821371571,5.674599808638099,4.1376954096503695,3.9006027510125336,0.6097360778631479,0.4067774811549101 4,0.3956297956536 98
11,102.51202077136514,65.58268574011318,6.409107678424548,3.1016057062669518,3.2991726455248918,0.8761163254407514,0.5776824294777184,0.436041899604322,
12,166.7340160564925,141.0088334978016,5.916785214419169,6.247261046502899,3.8147919896322042,0.8417608374769714,0.4235056915337 8015,0.6202524041284564,
13,139.92818636152325,99.7813908488341,8.428579804142636,3.6742349579215867,3.2220199503489635,1.476088763578134,1.2280510557922912,0.6449841774625612,1
14,133.87276912816088,87.14701039644953,9.468344448177454,3.3496232581658396,2.747783298116343,0.3110786230723220 4,0.3452166200481868,0.4880772696698792,
```

ata.csv > data

```
octave=2000,rt60-octave=4000,mean absorption-octave=125,mean absorption-octave=250,mean absorption-octave=500,mean absorption-octave=1000,mean absorption-octave=2000,mean absorpti
62682196229,0.4543125290134379,0.4241644116199685,0.3953952917660365,0.4015464569491315,0.2140260372113348,0.1532189285127052 7,0.2364444736860964 2,0.2532501168622736,0.271676696
88128107043702,1.862801429965971,1.2998951824207021,1.2984405144407658,1.866614858238739,0.1221101493141070 3,0.1171139326389765,0.0816560872117486 2,0.1170164164622946 6,0.11714751
4322308899751,1.3399360493002077,2.5661756328004817,1.378802735566783,1.6438868816316599,0.0892147553564690 6,0.0820209972955873 9,0.1153443069786789 5,0.0602273644199543 06,0.112092
9772592343582,0.3622290176279728,0.3888013991651629 3,0.3872111516335408 6,0.3200454442761470 6,0.1924183983353017,0.2250425400330297 6,0.3752338159718745 5,0.3495887021809521 7,0.3510
832446184599,0.4442754629454674,0.5517958798420048,0.519086918980691,0.5264394753611987,0.1156418534722488,0.2208914648355437,0.1987946265160620 8,0.1600584164415711 5,0.1701440962
678450860275,0.8618417989067523,1.2730258533241696,1.0168852118620544,0.9618249615977674,0.1071412084694278 2,0.0862458939814001,0.1148990016099883 1,0.0777869215943770 7,0.09738047
07831078164306,0.3862574393081510 4,0.2376288180417506,0.3819488203475700 5,0.5399597795782 5,0.2924514681707 77,0.2186216119921 3,0.2551430729470313 7,0.4147262559561191 7,0.2580212
0677748115491014,0.3956297956536982 7,0.3932160300672 24,0.6886533606528626,0.5244604608479617,0.1945857680861963 8,0.2916728888334957,0.2998913740680906,0.3017322641210274,0.172286
776824294777184,0.436041899604322,0.2803121919191253,0.4194789478595377,0.2721692303857445 3,0.1168349058497460 3,0.1771924558772541 8,0.2347503037877349 8,0.3651677356432233,0.24401
50569153378015,0.6202524041284564,0.6537303064885769,0.5693902308998618,0.4886928384015977,0.1607508859073778,0.3195088119276582 3,0.2181592516947914 2,0.2069871918182576 4,0.237646
0510557922912,0.6449841774625612,1.2343866202368436,1.3530731714548903,0.6549261618308 58,0.0772950809694582 1,0.0929068686006626,0.1768948828291 25,0.0924300366095898 3,0.084322417
52166200481868,0.4880772696698792,0.3100657157791196,0.3244727561620612,0.4908467571826544,0.3348189738349548,0.3017091857991285 3,0.2133986395832628 8,0.3359127438431861,0.3209977
9407130003882 2,0.8432695684233021,0.6116777441044562,0.6401032520215914,0.7942924619805691,0.1038561696625310 4,0.1861004571134500 5,0.1531740158109351 2,0.2111683602215441 7,0.20179
66753418225413,0.8355486170514 3,0.5127637938899298,0.577319365265294,0.9869591448886625,0.1267761345851620 4,0.1464885424576164 7,0.1589584933723491 9,0.2590228695717076 7,0.23005905
04021777216636,1.037610658461216,0.8877864800482109,1.3468791570564946,1.1899864409997312,0.1117109336948068,0.0793684499909151 6,0.1315962332241860 4,0.1538046109908638 7,0.1013792
8590788016628,0.5240153967748992,0.5838998654337496,0.6027255033271623,0.4781163531731205,0.1048079673576116 2,0.1421383545312044 4,0.1993293726024891 2,0.1788862550868943 7,0.173298
```

```
00,mean absorption-octave=2000,mean absorption-octave=4000,rt60 effective,picke_path
4447368609642,0.2532501168622736,0.2716766968361984 4,0.2675149660832141,1.381583333333333 2,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_3_noisy.pickle
5608721174862,0.1170164164622946 6,0.1171475122131291 9,0.0814892667076459 6,2.2229166666666664,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_4_noisy.pickle
34430697867895,0.0602273644199543 06,0.1120928984368047 4,0.0940174149019773 2,4.836270833333334,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_5_noisy.pickle
3752338159718745 5,0.3495887021809521 7,0.3510244370981437 4,0.424691489821709,1.3079375,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_6_noisy.pickle
62651606208,0.1600584164415711 5,0.1701440962910877 6,0.1677677660207705,1.2699583333333335,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_7_noisy.pickle
9900160998831,0.0777869215943770 7,0.0973804723335630 2,0.1029550762288846,1.5702083333333332,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_8_noisy.pickle
14307294703137,0.4147262559561191 7,0.2580212446370502 3,0.1825152793615645 6,0.685125,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_9_noisy.pickle
2998913740680906,0.3017322641210274,0.1722869150429844 3,0.2262248003386861 7,1.1575208333333333,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_10_noisy.pickle
23475030378773498,0.3651677356432233,0.2440193218720743 7,0.3760930956494229,1.5778958333333333,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_11_noisy.pickle
815925169479142,0.2069871918182576 4,0.2376468597515452,0.2768892639989156,2.0245208333333333,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_12_noisy.pickle
894882829125,0.0924300366095898 3,0.0843224172246349 9,0.1742095630749066,1.6357083333333335,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_13_noisy.pickle
339863958326288,0.3359127438431861,0.3209977521411513 4,0.2121945878931564,0.8895416666666667,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_14_noisy.pickle
5317401581093512,0.2111683602215441 7,0.2017908607691222 8,0.1626189248787194 8,1.1008333333333333,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_15_noisy.pickle
5849337234919,0.2590228695717076 7,0.2300590579441449 4,0.1345724896452813 5,1.388625,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_16_noisy.pickle
59623332418604,0.1538046109908638 7,0.1013792911497523 7,0.1147455546569494 4,1.9009166666666668,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_17_noisy.pickle
932937260248912,0.1788862550868943 7,0.1732988892897206 2,0.2184649397159697,1.4647708333333334,/dsi/gannot-lab-UG/Bar_Tsofit/audio_after/output_room_18_noisy.pickle
```

וככה ממשיך עבור כל הערכים שאנחנו מייצאים.

בהמשך נשתמש בקובץ הCSV הנ"ל בשביל לגשת למידע שלו.

בנוסף, בעזרת ספריית pickle נשמור את ההקלטה המהדהדת והרועשת שנשמעת במיקרופונים כך ששני הערוצים יהיו מופרדים- כך שנקבל בעצם מן מערך עם מספר שורות כמספרים החדרים ו2 עמודות כמספר הערוצים בהקלטה.

חלק ב'- יצירת המודל

תחילה נרצה להסביר מושגי בסיס בלמידה עמוקה לצורך הבנת המודל שלנו:

בפרויקט שלנו, המטרה היא לשערך פרמטרים אקוסטיים וגיאומטריים של חדר באמצעות מודל למידה עמוקה שמתבסס על רשת נוירונים קונבולוציונית (CNN).

המודל משתמש בספקטוגרמות המיוצרות מההקלטות כדי להפיק את הפרמטרים המבוקשים של החדר.

בהמשך נסביר על המודל עצמו, אך קודם נבנה שני class שבעזרתם נוכל להכניס את המידע למודל ולאמן כמות גדולה מאוד של מידע- נצטרך לבנות DATASET, DATALOADER.

כאשר עובדים על פרויקט למידת למידה עמוקה, חשוב מאוד לארגן את הנתונים בצורה נוחה וקלה לשימוש. Dataset ו Dataloader - שניהם מרכיבים קריטיים לאימון מודלים מבוססי למידה עמוקה כמו רשתות נוירונים קונבולוציוניות (CNN).

Dataset – מאגר הנתונים:

ה-**Dataset** הוא הCLASS שמייצג את מאגר הנתונים עבור המודל, כלומר את כל הדוגמאות מהן המודל ילמד, מכיל את כל הנתונים שאנחנו רוצים לאמן את המודל עליהם. זה יכול לכלול תמונות, טקסטים, קבצי שמע, או כל סוג אחר של נתונים- במקרה שלנו קבצי שמע (שמייצגות את הקלטות השמע שעברו המרה) וגם את הנתונים של הפרמטרים האקוסטיים והגיאומטריים של כל חדר.

מטרת מחלקת ה-Dataset היא לאחסן את הנתונים, לארגן אותם, ולספק להם גישה נוחה במהלך האימון של המודל. ב-PyTorch, על מנת לבנות מחלקת Dataset מותאמת אישית, יוצרים מחלקה שמרחיבה את `torch.utils.data.Dataset`. עליך להגדיר שלוש פונקציות עיקריות:

1. init – אתחול מחלקת ה-Dataset, בה מאחסנים את הנתונים ועמודות התוויות.

2. len – הפונקציה מחזירה את אורך הדאטאסט (כלומר, כמה דוגמאות יש במאגר).

3. get item – הפונקציה מחזירה דגימה ספציפית מהדאטאסט (כלומר, זוג של קלט ותווית).

בdataset שכתבנו קראנו את הנתונים ששמרנו מהקובץ CSV - הפרמטרים הגיאומטריים והאקוסטיים של החדרים : שטח, נפח, RT60 עבור כל אוקטבה, מקדמי ספיגה ממוצעים עבור כל אוקטבה, RT60 אפקטיבי.

בנוסף, בקובץ CSV דאגנו לשמור את הpath עבור הקובץ מסוג PICKLE שבו שמורות 2 ההקלטות עבור הערוצים- הקוד משתמש במסלולי הקבצים שמאוחסנים ב CSV-לטעון את קבצי ה Pickle- שמכילים את נתוני הקלט לאחר הטעינה, הנתונים מאורגנים כמערך דו-ממדי 2 x Num samples מה שתואם את השימוש במיקרופונים כמקור נתונים לתהליך הלמידה.

לאחר מכן, הקוד ממיר את נתוני הקלט (הסיגנל מהשמע) ואת התוויות (הפרמטרים האקוסטיים) למבני torch.Tensor כדי להקל על עבודה עם PyTorch במהלך האימון.

לאחר מכן נבנה את ה **Dataloader** – טוען הנתונים:

ה-Dataloader הוא מחלקה נוספת ב-PyTorch, שאחראית על הטעינה של הדאטאסט (Dataset) בצורה יעילה במהלך האימון. בעוד שה-Dataset מאחסן את הנתונים, ה-Dataloader אחראי לטעון את הדוגמאות לפי batch, לבצע shuffling של הנתונים, ולהתמודד עם טעינת הנתונים בצורה מקבילית על מנת להאיץ את האימון ולשמור על אמינותו.

ברוב המקרים, לא מאמנים את המודל על כל הדאטאסט בפעם אחת, אלא מאמנים אותו על קבוצות קטנות של נתונים שנקראות "מנות" (batches). ה-Dataloader אחראי לחלק את הדאטאסט למנות בגודל מסוים (batch_size) ולהעביר אותן למודל במהלך האימון.

במהלך כל שלב אימון, חשוב שהנתונים יעברו ערבוב (shuffling) כך שהמודל לא ילמד דפוסים ספציפיים בסדר הדוגמאות. ה-Dataloader מבצע ערבוב של הDataset לפני כל Epoch (מחזור שלם של אימון על כל ה dataset).

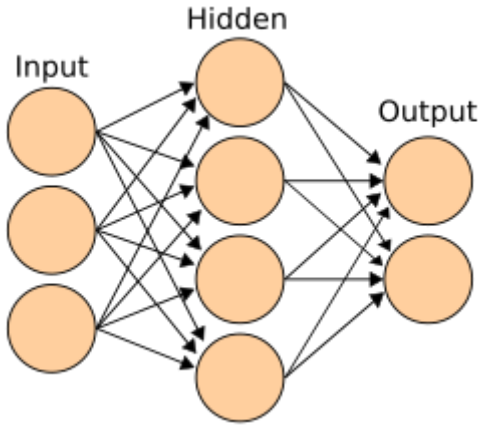
בכדי לייעל את תהליך הטעינה של הDataset, ה-Dataloader משתמש בthreads כדי לטעון את הנתונים בצורה מהירה יותר. כך המודל לא צריך להמתין לטעינת הנתונים מהזיכרון בכל פעם.

כעת, נחזור להתעסק במודל שלנו.

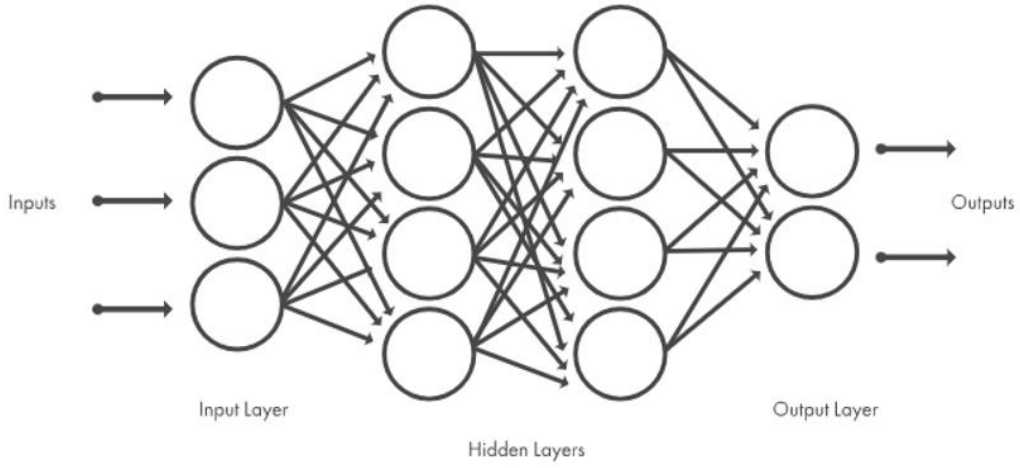
תחילה, ברצוננו להסביר מספר מושגים חשובים להמשך הבנת המודל:

- אז מה הן בעצם רשתות ניורונים?
  Deep neural Networks – ו Artifical Neural Networks

<u>רשתות עצביות (ANN)</u> הן מערכות למידה מפוקחות המורכבות ממספר רב של יחידות פשוטות הנקראות נוירונים. כל נוירון מקבל החלטות בסיסיות ומעביר אותן לנוירונים אחרים, המאורגנים בשכבות המקושרות ביניהן. יחד, רשת עצבית יכולה לחקות כמעט כל פונקציה ולפתור כמעט כל בעיה- בתנאי שיש לה מספיק נתוני אימון וכוח חישוב מתאים.
ברשת עצבית "רדודה" יש שלוש שכבות נוירונים בלבד
Input layer - מקבלת את ה – training data של המודל שאותו נרצה לאמן.
Hidden layer - שכבה אחת נסתרת
Output layer - שכבת פלט שמייצרת את החיזויים.



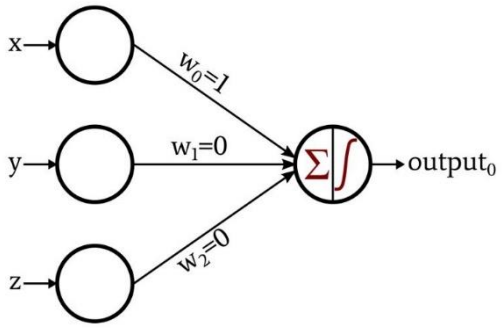
<u>רשתות עצביות עמוקות (DNN)</u> בנויות באופן דומה לרשתות רדודות, אך כוללות שתיים או יותר "שכבות נסתרות" של נוירונים. למרות שרשתות נוירונים רדודות מסוגלות לפתור בעיות מורכבות, רשתות עמוקות מספקות <u>דיוק גבוה יותר</u>, והדיוק משתפר ככל שמוסיפים שכבות נוירונים נוספות, עד שמגיעים לגבול של 9-10 שכבות, מעבר לו יכולת החיזוי מתחילה לרדת. כיום, רוב המודלים והיישומים מבוססי רשתות עצביות משתמשים ברשתות עמוקות עם בין 3 ל-10 שכבות.



רשת נוירונים עמוקה כמו זו שאנחנו משתמשים בה, היא דומה לרשתות ניורנים פשוטות רק שיש להן שכבות נסתרות של נוירונים, נרצה להבין יותר לעומק מה זה אומר רשתות ניורנים ואיך בונים אותם. כמו שנראה בהמשך, המודל שלנו בנוי מ3 מודלים עיקריים שעבור כל אחד מהם יש לנו מספר שכבות עליהם נלמד בהמשך, כל שנראה כי אכן יש לנו רשת עמוקה מאוד.

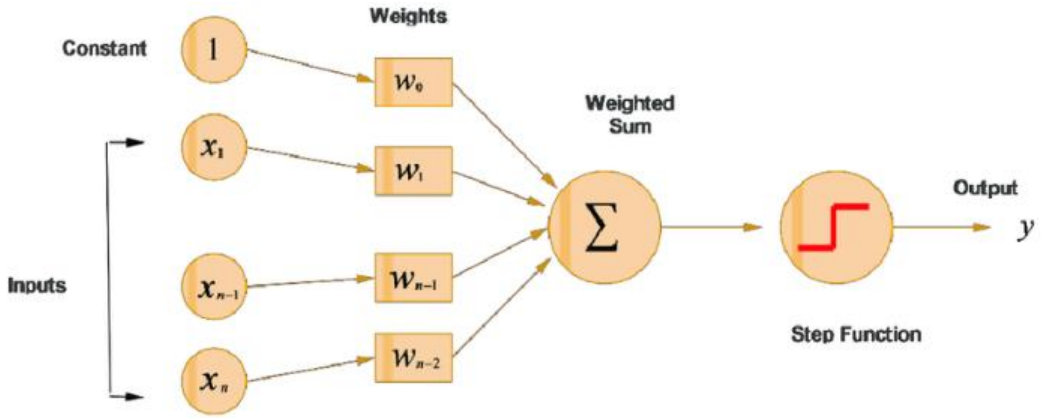
<u>Perceptron</u> הוא אלגוריתם סיווג בינארי, אלגוריתם זה תפקידו הוא לנסות לחקות את המוח האנושי - הוא נועד לחקות את הנוירון. למרות שיש ל - perceptron מבנה פשוט, יש לו יכולת ללמוד ולפתור בעיות מורכבות מאוד.

כאשר- w הינו המשקולות



עבור המוצא נוכל לקבל האם הוא 1/0 בהתאם להיותו אלגוריתם סיווג בינארי

ואז נוכל בעצם לשלב את האלגוריתם ולקבל:



תהליך ההתקדמות מתבצע כך שהכניסות שנכנסות לרשת מוכפלות במשקולות מתאימות ואז מתבצע סכימה, לסכום מתווסף bias כלשהו שמטרתו להזיז את הפלט. הסכום המתקבל עובר בפונקציית אקטיבציה והפלט שלה הוא המוצא של התהליך perceptron.

רשת ניורנים בסוג DNN מורכבת מהתהליך הנ"ל ומעוד כל מיני סוגים של רכיבים, נסביר אותם שלב אחרי שלב.

היפר-פרמטרים (Hyperparameters) הם פרמטרים שלמודל למידת מכונה אין שליטה עליהם במהלך תהליך האימון. הם ערכים שאנו מגדירים מראש ומשפיעים על אופן פעולתו של המודל, על הליך הלמידה שלו, ועל ביצועיו הסופיים. הם נשארים קבועים לאורך כל שלבי האימון, בשונה מפרמטרים כמו המשקלים (weights), אותם המודל לומד ומשנה במהלך האימון.

דוגמאות להיפר-פרמטרים:

א. Learning Rate קובע את גודל הצעד שהמודל מבצע בעדכון המשקלים לאחר כל איטרציה. ערך גבוה מדי עלול לגרום לאי-יציבות באימון, בעוד ערך נמוך מדי עלול לגרום לאימון איטי מאוד.

ב. Batch Size מספר הדוגמאות שהמודל רואה לפני עדכון המשקלים. גודל באץ' גדול יוביל לאימון יציב אך ארוך יותר, בעוד גודל קטן יוביל לאימון מהיר יותר אך פחות יציב.

ג. Number of Epochs מספר הפעמים שהמודל רואה את כל הדאטה סט במהלך האימון. יותר אפוקים יכולים לשפר את ביצועי המודל, אך גם עלולים לגרום ל-overfitting.

ד. Regularization Parameters מטרת רכיב זה הינו להקטין את ההתאמה ולשפר את הכלליות שלו- כלומר למנוע over fitting, כאשר מאמנים מודל עם הרבה נתונים כמו המודל שלנו יש סיכון שהמודל ילמד לא רק את הדפוסים הכללים אלא גם את הרעש והפרטים הלא רלוונטים בנתונים של האימון- לכן אנחנו עלולים

לקבל ביצועיים גרועים מהמודל, הרגולציה מטרתה להעניש את המודל.
בפועל לפונקציית המטרה של המודל (פונקציית הפסד שכבר נפרט עליה), נוסיף איבר רגולציה כך שבעצם:

$$L(\theta) = Loss(\theta) + \lambda \cdot Regularization(\theta)$$

כאשר λ הינו פרמטר הענישה והפונקציה הינה פונקציה המודדת את המורכבות של המודל יש שלל סוגי רגוליזציה- אנחנו נשתמש ברגוליזציה מסוג L2.
משמעותה- שפונקציית הרגוליזציה היא סכום ריבועי הפרמרטרים:

$$Regularization(\theta) = \sum_i \theta_i^2$$

ה. Dropout Rate אחוז הנוירונים ברשת ש"נזרקים" (מושמטים) באופן אקראי בכל שכבה במהלך האימון, מטרת שיטה זו היא גם כן למנוע התאמת יתר. הנוירונים שנזרקו אינם משתתפים בתהליך החישוב הערכים או בעדכון המשקולות באותו מחזור אימון- בעצם הערך במשתנה זה הינו ההסתברות להשמיט ניורנון במהלך האימון. לרוב נפעיל את הdrop out בחלק של ה Hidden Layers , צריך לשים לא לעשים ערכים גבוהים מדיי ואז אנחנו עלולים להגיע למצב של under fitting.

ו. Optimizer סוג האלגוריתם שמשמש לאופטימיזציה של המשקלים במודל, כמו Adam, SGD (Stochastic Gradient Descent) וכו', עליהם נפרט בהמשך.
בגדול האופטימייזר משתמש בגזירה של פונקציית ההפסד ביחס לפרמטרים של המודל כדי לקבוע את יוון וגודל השינוי שיש לבצע בכל פרמטר- כלומר לשפר את המשקולות, מטרתו המרכזית לצמצם את פונקציית ההפסד.
בשלב האופטימיזציה נחשב את פונקציית האובדן, נחשב את הגרדיאנט ונעדכן פרמטרים.

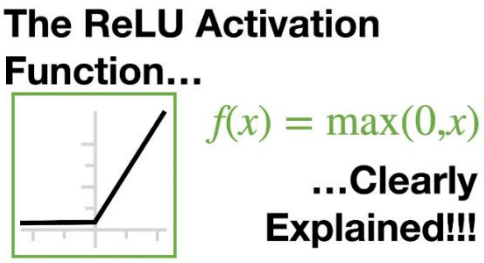
Activation Function פונקציית אקטיבציה (activation function) היא פונקציה שקובעת את הפלט של כל נוירון ברשת נוירונים. תפקידה הוא לקחת את הקלט מהנוירון ולעבד אותו לכדי פלט סופי. בניגוד ל-perceptron הקלאסי שמשתמש בפונקציה בינארית כמו פונקציית מדרגה, ברשתות נוירונים מודרניות נעשה שימוש בפונקציות אקטיבציה שמפיקות ערכים רציפים.
ערכים אלה נעים לרוב בטווח בין 0 ל-1, או בין -1 ל-1, בהתאם לסוג הפונקציה. לדוגמה, פונקציות הפעלה קלאסיות כוללות את sigmoid ו-tanh, שמבצעות מיפוי כזה.
פונקציות הפעלה חדשות יותר, כמו ReLU ו-Swish, נוצרו כדי לשפר את הביצועים החישוביים ולייעל את הלמידה ברשתות נוירונים עמוקות.

דוגמאות לפונקציות activation function נפוצות ותפקידן:

a) <u>ReLU (Rectified Linear Unit)</u>

זוהי הפונקציה הנפוצה ביותר ברשתות נוירונים עמוקות. אם הערך של נוירון חיובי, הוא מועבר כמות שהוא, ואם הוא שלילי, הערך שלו נקבע ל-0.

זוהי פונקציה פשוטה לחישוב, אך מצד שני נוירונים שהערך שלהם תמיד שלילי יכולים לא להתקיים מבחינת האיומן ולא לתרום לאימון.

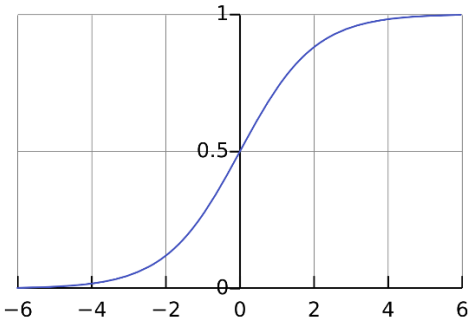


b) <u>Sigmoid</u>

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} = 1 - \sigma(-x).$$

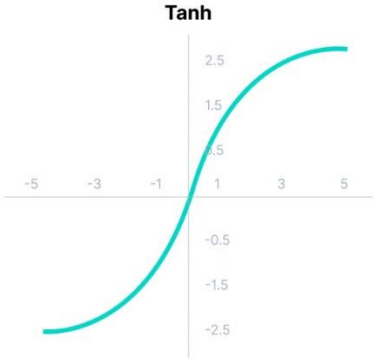
פונקציה זו ממירה כל ערך לערך בין 0 ל-1, ולכן היא טובה למידול של הסתברויות.

זוהי הפונקציה הכי טובה עבור משימות כמו סיווג בינארי (כן או לא).
מצד שני, נוטה לגרום לבעיית ה gradient vanishing, שבה השיפועים של הפונקציה קטנים מאוד וגורמים לאימון להיות איטי.

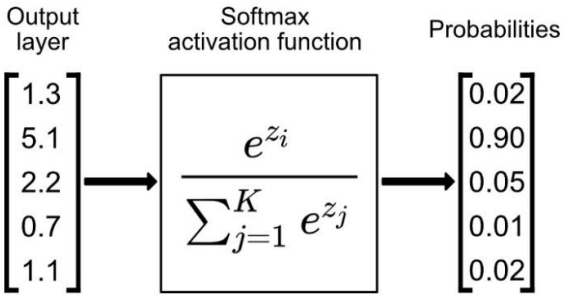


c) <u>Tanh (Hyperbolic Tangent)</u>

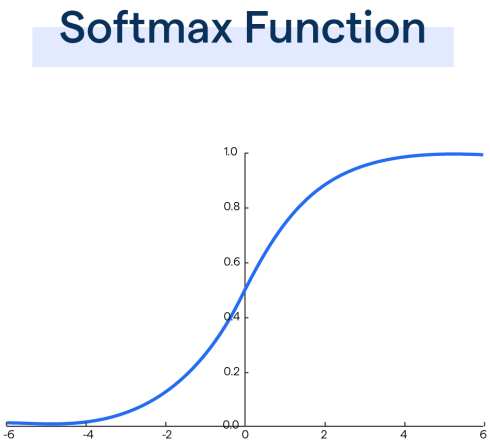
פונקציה זו דומה ל sigmoid, אך הערכים שלה נעים בין -1 ל-1, ולכן מרכז הכובד של הפלטים קרוב יותר ל-0.



d) <u>SoftMax</u>



זוהי פונקציה שמשמשת בעיקר בשכבת הפלט במשימות סיווג מרובות קטגוריות. היא ממירה את הערכים ליחסים שהסכום שלהם הוא 1, כך שכל ערך מייצג הסתברות לכל מחלקה.



-Backpropagation and Gradient Descent

שני מושגים מרכזיים בלמידת מכונה, במיוחד באימון רשתות נוירונים. הם עובדים יחד כדי לאפשר למודל ללמוד ולהתאים את עצמו לדאטה, וכך להפוך למדויק יותר.

Gradient Descent:

כמו שהסברנו, Optimizer משתמש בגזירה של פונקציית ההפסד ביחס לפרמטרים של המודל כדי לקבוע את כיוון וגודל השינוי שיש לבצע בכל פרמטר- כלומר לשפר את המשקולות, מטרתו המרכזית לצמצם את פונקציית ההפסד.
בשלב האופטימיזציה נחשב את פונקציית האובדן, נחשב את הגרדיאנט ונעדכן פרמטרים.

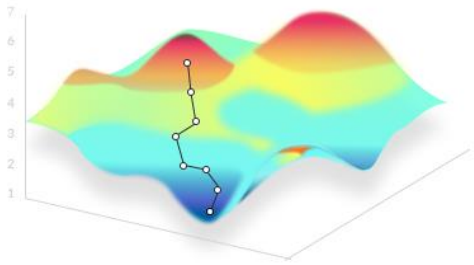
הוא אלגוריתם אופטימיזציה שמשמש לעדכון משקלי המודל כך שפונקציית ההפסד- השגיאה (loss) בין תחזיות המודל לתוויות האמיתיות (labels) תקטן. הרעיון המרכזי הוא למצוא את ערכי המשקלים שמביאים למינימום של פונקציית השגיאה.

המודל מחזיק בסט של משקלים שמכתיבים את הפלט עבור קלט נתון, תחילה המודל מנחש פלט על סמך הקלט, משווה אותו לתוצאה הרצויה, ומחשב את השגיאה.

בשלב הבא, Gradient Descent מחשב את השיפוע (gradient) של פונקציית השגיאה ביחס למשקלים. השיפוע מציין את הכיוון שבו פונקציית השגיאה עולה.

אנו מעדכנים את המשקלים בכיוון ההפוך לשיפוע (כיוון המינימום), תוך שימוש ב-קצב הלמידה (learning rate) שקובע כמה גדולים הצעדים שאנחנו עושים.

שחישבנו את הנגזרות החלקיות, נרצה לעדכן את המשקולות כך שנגיע לנקודת מינימום בפונקציית השגיאה.



יש מספר סוגים של Gradient Descent:

1. Batch Gradient Descent – מחשב את השיפועים על כל הדאטה סט בבת אחת.

2. Stochastic Gradient Descent (SGD) – מחשב את השיפועים על דוגמה אחת בכל פעם, מה שמוביל לשינויים תכופים במשקלים אך גם לאימון מהיר יותר- כלומר מתבצע בצורה פשוטה על סמך ה באץ' הנוכחי שנכנס לרשת מבלי להתחשב בבאצ'ים קודמים.

3. Adam (Adaptive Moment Estimation) - הוא אחד האלגוריתמים הפופולריים והיעילים ביותר לאופטימיזציה באימון רשתות נוירונים. זהו שדרוג של אלגוריתם ה-Gradient Descent המסורתי, שמנסה להתמודד עם האתגרים של אימון רשתות נוירונים גדולות ומורכבות בצורה טובה יותר.

Adam הוא שילוב של שתי שיטות אופטימיזציה נפוצות:

א. Momentum– שיטה המוסיפה "זיכרון" לשיפועי העבר ומאיצה את הלמידה בכיוון הנכון.

ב. RMSProp– שיטה שמסתכלת על היסטוריית הגרדיאנטים ומאזנת את השיפועים כך שמהירות הלמידה תתעדכן בצורה מתאימה לכל פרמטר.

Adam עוקב אחרי הגרדיאנטים ומוסיף זיכרון כדי לייצר צעדים יעילים יותר בכל כיוון, וגם מתאים את קצב הלמידה באופן דינמי לכל פרמטר במודל, בהתאם לעוצמת השיפוע שלו.

בכל איטרציה של האימון מתבצעות הפעולות הבאות:
מחשב את השיפועים (gradients) של פונקציית השגיאה ביחס למשקלים ושומר שני ערכים נוספים שמעדכנים את המשקלים באופן מותאם:

מומנטום ממוצע של הגרדיאנטים- (momentum of gradients) – ערך ממוצע משוקלל של השיפועים שהתקבלו עד כה, שמשמש כדי לאזן בין עדכוני המשקלים בהווה לבין השיפועים מהעבר.

ריבוע ממוצע של הגרדיאנטים (RMS of gradients) – ערך משוקלל של הריבועים של השיפועים, שמשמש כדי להתאים את גודל הצעד בהתאם לשונות בגרדיאנט.

עדכון המשקלים:

א. חישוב ממוצע משוקלל של הגרדיאנטים (first moment)

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

ב. חישוב ממוצע משוקלל של ריבועי הגרדיאנטים (second moment):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

3. תיקון הטיה– כדי למנוע ערכים נמוכים מדי בתחילת האימון, מוסיפים תיקון כדי לאזן את ההשפעה הראשונית של המומנטום:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

4. עדכון המשקלים:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

Adam בעצם מאפשר לנו התאמה אוטומטית של קצב הלמידה, מאחר וכל פרמטר במשקל מקבל קצב למידה מותאם לו, זה מייעל את תהליך הלמידה עבור משקלים שונים במודל.
בנוסף, עושה שימוש בזיכרון של גרדיאנטים קודמים על מנת שנוכל לשמור על היציבות בעדכון המשקלים ומונע קפיצות

לכן נשתמש באלגוריתם זה ברשתות נוירונים עמוקות, שבהן יש הרבה פרמטרים ועדכונים מורכבים.

Backpropagation:

הוא האלגוריתם שבו משתמשים כדי לעדכן את המשקלים ברשת נוירונים בהתאם לשגיאות המודל. זהו החלק המרכזי בלמידת הרשתות הנוירונים.

איך הוא עובד?

1. Forward Pass – בשלב זה המודל מקבל את הקלט ומעביר אותו דרך כל שכבות הרשת עד שמתקבל פלט. אז משווים את הפלט לתוצאה האמיתית ומחשבים את השגיאה.

2. חישוב הגרדיאנטים– האלגוריתם משתמש ב-Gradient Descent כדי לחשב את השיפועים של פונקציית השגיאה ביחס לכל אחד מהמשקלים ברשת.

3. Backpropagation– השיפועים מחושבים אחורה, מהשכבה האחרונה לשכבה הראשונה, כך שאנו יודעים איך כל משקל ברשת השפיע על השגיאה הכוללת. זה מתבצע על ידי כלל השרשרת של נגזרות.

4. עדכון המשקלים – אחרי שחישבנו את הגרדיאנטים עבור כל המשקלים, אנו משתמשים בהם כדי לעדכן את המשקלים בכיוון שמקטין את השגיאה.

בעצם בסך הכל קיבלנו כי- Gradient Descent עוזר לנו למצוא את הסט האופטימלי של משקלים שממזער את פונקציית השגיאה ומביא לתחזיות טובות יותר.

Backpropagation מאפשר לעדכן בצורה מדויקת את המשקלים של כל שכבה ברשת נוירונים, גם אם מדובר ברשת עמוקה עם הרבה שכבות

השילוב של השניים מאפשר לרשתות נוירונים ללמוד דפוסים מורכבים, לשפר את הביצועים שלהן, ולבצע תחזיות מדויקות יותר.

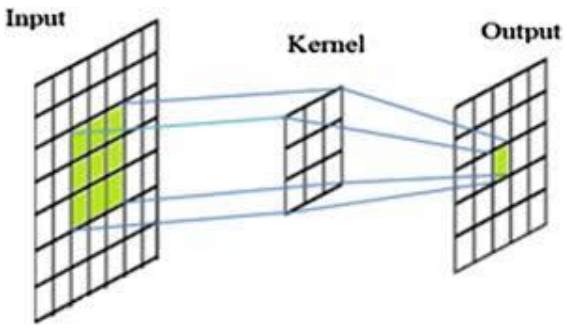
Convolutional layer

הפרמטרים של שכבת הקונבולוציה (Convolutional Layer) בנויים מסט של פילטרים הניתנים ללמידה.
כל פילטר הוא קטן יחסית בממדים המרחביים (רוחב וגובה) אך משתרע לאורך עומק הקלט.
לדוגמה, פילטר בעל מימדים AXBXC -> כאשר, A רוחב, B גובה וC עומק פילטר .
עומק נפח הפילטר- מספר הפילטרים שאנחנו מגדירים בשכבה, עומק הפלט שווה למספר הפילטרים כך שכל אחד מהם מתמקד בזיהוי תכונה אחרת בקלט.

במהלך הforward pass-הפילטר מוחלק (נפוץ גם המונח "מוזז") על פני רוחב וגובה הקלט, ומבצע בכל מיקום הכפלה איבר-איבר של הפילטר עם החלק המתאים בקלט. תוצאה זו יוצרת activation map שהיא ייצוג של תגובות הפילטר באזורים שונים של הקלט. כל פילטר לומד לזהות מאפיין מסוים של הקלט, כגון קצוות, טקסטורות, או תכונות חזותיות מורכבות יותר כמו תבניות.

במהלך האימון, הרשת לומדת להתאים את המשקלים של הפילטרים כך שכל פילטר יתמקד בתכונה מסוימת. בשכבות הראשונות, הפילטרים נוטים לזהות תכונות בסיסיות כמו קצוות (edges)או גווני צבע, בעוד שבשכבות העמוקות יותר הם לומדים לזהות תכונות מורכבות יותר, כמו חלקים של אובייקטים או דפוסים מורכבים יותר.

כאשר יש לנו כמה פילטרים באותה שכבה, כל אחד מהם מייצר מפת הפעלה נפרדת. כל מפת הפעלה מתארת כיצד הפילטר מגיב למידע השונה שהועבר דרכו. בשלב הסופי, ממפות הפעלה אלו מאוחדות לאורך ממד העומק ויוצרות את פלט השכבה.



<u>עומק נפח הפלט</u>- מספר הפילטרים שאנו מגדירים בשכבה.
עומק הפלט שווה למספר הפילטרים, כאשר כל אחד מהם מתמקד בזיהוי תכונה אחרת בקלט.

<u>צעד ההחלקה(Stride)</u> קובע בכמה פיקסלים המסנן זז בכל פעם שהוא מוחלק על פני הקלט. אם הצעד הוא 1, הפילטר עובר פיקסל אחד בכל פעם, ואם הצעד הוא 2, הוא קופץ פיקסל אחד נוסף בכל החלקה, מה שגורם להקטנת הגודל המרחבי של פלט השכבה.

ריפוד(Padding) לעיתים מוסיפים ריפוד של אפסים סביב הקלט כדי לשמור על גודלו המרחבי של הפלט זהה לקלט. למשל, ריפוד של שכבה עם אפסים יכול לעזור בכך שהפילטר יוכל להיות מוחלק גם על קצוות התמונה, וכך נבטיח שכל פיקסל בקלט ישפיע על הפלט הסופי. ריפוד כזה מאפשר לשמר את הגובה והרוחב של הקלט, מבלי להקטין את הממדים שלו.

פילטרים בשכבת הקונבולוציה יכולים להתמקד באזורים קטנים של הקלט ולהפיק מהם תכונות רלוונטיות. זה חשוב במיוחד בתמונות, שבהן מידע חשוב עשוי להופיע באזורים מקומיים שונים.

בניגוד לשכבה Fully Connected, שבה לכל נוירון יש חיבור לכל נוירון בקלט, שכבת קונבולוציה מקושרת רק לחלק קטן מהקלט. זה מקטין את כמות הפרמטרים שצריך ללמוד, ובכך משפר את יעילות המודל ומקטין את הסיכוי לover fitting.

עולה השאלה, למה בעצם אנחנו לא מסתפקים רק בקונבולוציה רגילה?

שכבת קונבולוציה חזקה מאוד בזיהוי מאפיינים מקומיים, אך לא תמיד יש לה גישה למידע גלובלי מהקלט כולו. לכן, בשכבות הסופיות של הרשת, לעיתים נשתמש בשכבות Fully Connected עליהן נפרט מיד- כדי לשלב את כל המידע שהרשת למדה מכל הפילטרים וליצור החלטות סופיות המבוססות על כל התמונה או הקלט Fully Connected Layer. מאפשרת לכל נוירון לקחת בחשבון את כל הקלטים שנאספו, מה שיכול להיות קריטי למשימות מסוימות כמו סיווג תמונות מורכבות.

Fully connected layer

היא **שכבה ברשת נוירונים שבה כל נוירון בשכבה מחובר לכל הנוירונים בשכבה הקודמת**, מה שיוצר מערכת קשרים מרובת קישורים שמאפשרת לשכבה לעבד את כל המידע שנאסף על ידי השכבות הקודמות.
בשכבה Fully Connected, כל נוירון מקבל את כל המידע מהשכבה הקודמת (כלומר כל נוירון מקבל את כל ערכי הפלט של הנוירונים בשכבה הקודמת). פעולה זו מתבצעת כך שכל נוירון בשכבה מחזיק בקבוצת משקלים, שכל אחד מהם מותאם לכל נוירון בשכבה הקודמת.
לאחר חישוב הכפלת המשקלים, תוצאת הסכימה עוברת דרך פונקציית אקטיבציה.

אם יש n נוירונים בשכבה L, ו- m נוירונים בשכבה $L-1$, אז הפלט של הנוירון j בשכבה L יהיה:

$$\text{activation}\left(\sum_{i=1}^{m} w_{ji}x_i + b_j\right) = y_j$$

כאשר:

- w_{ji} הם המשקלים שמחברים את הנוירון j לנוירון i בשכבה הקודמת.
- x_i הוא הפלט של הנוירון i בשכבה הקודמת.
- b_j הוא הביאס (bias), פרמטר נוסף שהמודל לומד.

מטרת השכבות הללו היא:

שילוב מידע :השכבות הקודמות (כמו שכבות קונבולוציה או שכבות אחרות) מחלצות מאפיינים (features) שונים מהמידע הגולמי, כמו תמונה או צליל.

שכבת ה Fully Connected-משלבת את כל המאפיינים הללו כדי לייצר את ההחלטה הסופית של המודל.

סיווג: בשכבות האחרונות, שכבת Fully Connected מעבדת את המאפיינים שחולצו כדי להפיק תחזית סופית.

Dropout layer

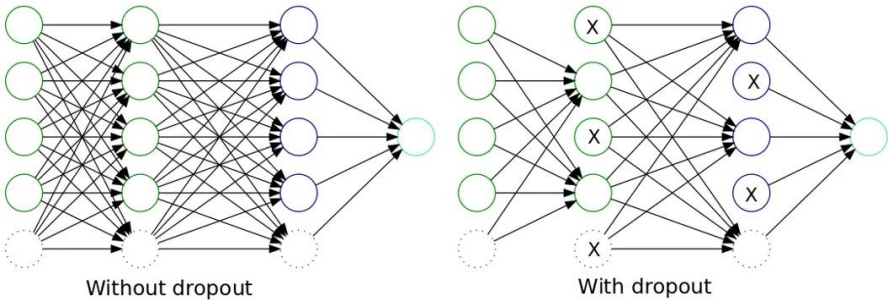
שכבת Dropout היא טכניקה נפוצה ברשתות נוירונים שנועדה להפחית את תופעת ה - overfitting(התאמת יתר) באימון מודלים. שכבת Dropout פועלת כך שהיא משביתה באופן אקראי ניורנים מסוימים במהלך כל איטרציה של אימון, כך שאותם נוירונים לא לוקחים חלק באותה סבב של אימון- דבר זה משפר את האקראיות של הלמידה וגורם למצב שהרשת לא מתאימה את עצמה ניורנים ספציפיים.
בפועל, במהלך כל איטרציה של האימון:

- נבחרת באקראי קבוצת נוירונים מסוימת שלא תשתתף באימון באותו סבב.
- כלומר, כל נוירון שנבחר לא יפעיל את הקשר שלו (לא יעביר את הפלט שלו לשכבה הבאה).
- תהליך זה מתרחש עם הסתברות מוגדרת מראש, שנקראת קצב השמטה dropout rate, אם קצב השמטה הוא 0.5, אז 50% מהנוירונים בשכבה יושמטו בכל איטרציה.
- במהלך שלב הבדיקה כל הנוירונים נשארים פעילים, אך התפוקות של כל הנוירונים מותאמות באופן פרופורציונלי כדי לקחת בחשבון את ההשמטה שהתרחשה באימון.

למה להשתמש בDrop Out ?

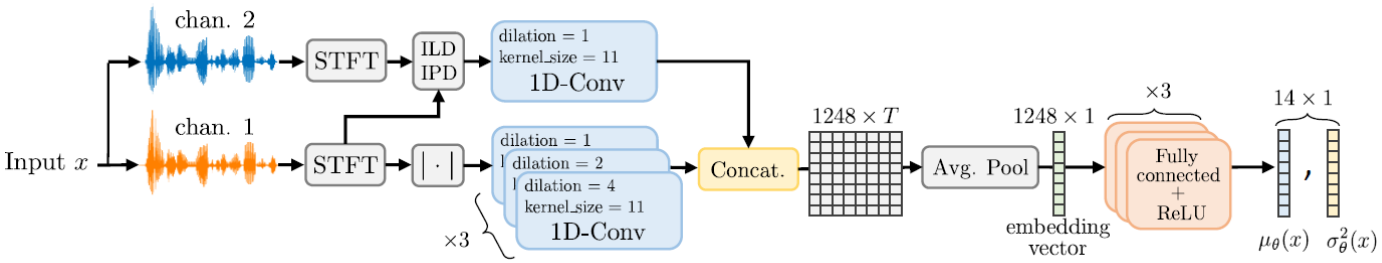
מטרתה העיקרית של שכבת Dropout היא להקטין את התלות של המודל על נוירונים מסוימים ולהפוך את הרשת ליותר עמידה- כאשר נוירונים נבחרים באופן אקראי להישמט, המודל נאלץ למצוא דפוסים שלא תלויים אך ורק בנוירונים ספציפיים. כלומר, שכבת Dropout מאלצת את המודל ללמוד ייצוגים כלליים יותר של הנתונים.

שימוש בשכבה זו גורמת למניעת – overfitting כאשר כל נוירון בשכבה יכול להיות מושמט, המודל אינו תלוי יותר מדי בנוירונים בודדים. זה עוזר לרשת ללמוד תכונות כלליות יותר שמתאימות לדאטה החדש, במקום ללמוד את הדאטה הספציפי באימון באופן מופרז. בנוסף, גוררת שיפור ה- Robustness – הרשת לומדת להיות עמידה יותר בפני שינויים או רעש בנתונים, מכיוון שהיא מתאמנת במצב שבו חלק מהנוירונים לא פעילים.

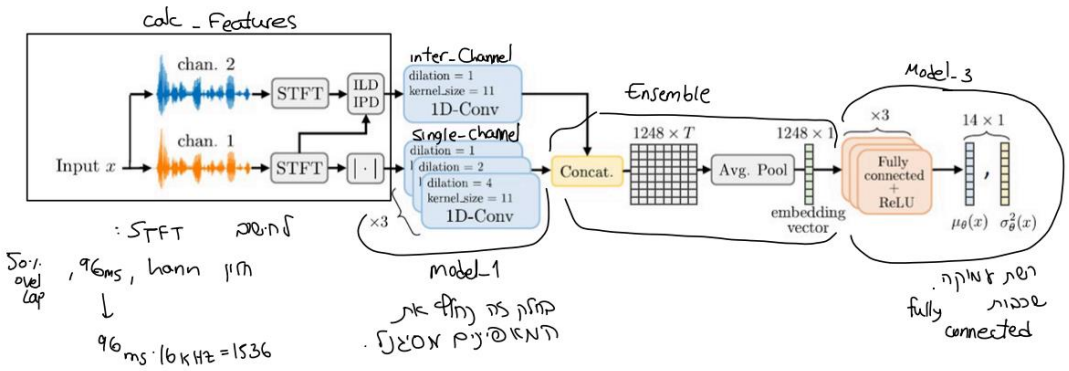


המודל שלנו...

כעת, לאחר שהבנו מושגי יסוד לצורך הבנת החומר, **נראה כי המודל שלנו בנוי בצורה הבאה:**



לצורך בניית המודל נרצה להפריד את המודל כולו לחלקים.
נסביר את השלבים ונתאים כל שלב לפונקציה במודל-



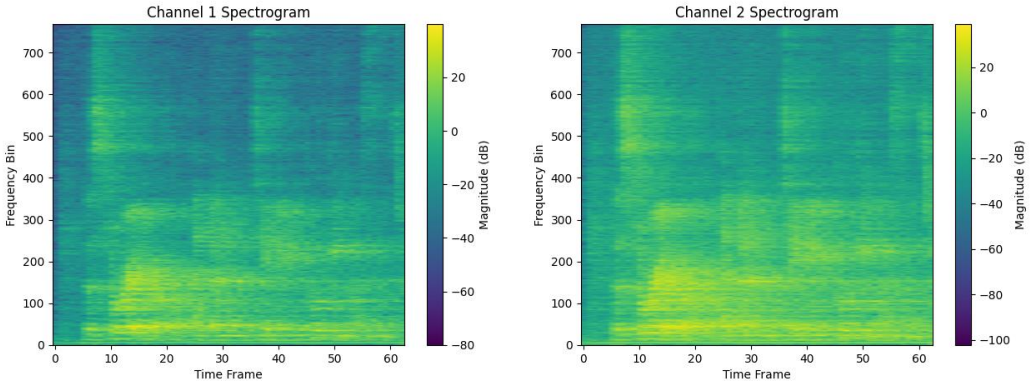
1. **<u>Calc_Fearures</u>**

תחילה, אנחנו מקבלים את שתי הקלטות- אחת עבור כל ערוץ- Ch1,ch2 הם אותות האודיו משני ערוצים שונים, ערוצים של מיקרופונים שונים הממוקמים במרחקים שונים ממקור הקול כמו שייצרנו בחלק א.

נבצע המרה ספקטוגרמות, השלב הראשון הוא המרה של האותות הסאונד לתצורת ספקטוגרמה באמצעות התמרת פורייה לזמן קצר (STFT)- המושג הוסבר החלק של המושגים- אנחנו בעצם ממירים את האותות ממרחב הזמן לתדר.

הספקטוגרמות מייצגות את השינויים בתדירות של האותות לאורך הזמן, וכך מספקות מידע רב עבור המודל.

סתם לשם הדוגמה נראה כי קיבלנו ספקטוגרמה מהצורה הזאת עבור אחד הbatchים:



ספקטוגרמה מספקת מידע על ההרכב התדרי של אות לאורך זמן.
מניתוח התמונה הזו ניתן לראות כי עבור האזורים הבהירים יותר (הצהובים), התדרים חזקים יותר ובעוד האזורים הכהים יותר התדרים חלשים יותר.
ניתן לראות כי בתדרים נמוכים יותר יש אנרגיה יציבה יותר, ניתן לראות הבדל בין הערוצים.
באופן כללי רצינו לראות תמונה בעיקר בשביל לראות שאכן קיבלנו ספקטוגרמה הגיונית.

כעת נבצע עבור שני ספקטוגרמות שקיבלנו חישוב בין-ערוצים- inter channel- חישוב ILD, IPD:

ILD (Inter-Channel Level Differences) : הבדל בעוצמת האות בין שני המיקרופונים.

$$\mathrm{ILD}(f,t) = \log|X_1(f,t)| - \log|X_2(f,t)|$$

IPD (Inter-Channel Phase Differences) : הבדל במופע האות בין שני המיקרופונים.

$$\mathrm{IPD}(f,t) = \left[\mathrm{Re}, \mathrm{Im}\left(\frac{X_1(f,t)X_2^*(f,t)}{|X_1(f,t)X_2^*(f,t)|}\right)\right]$$

וגם לחשב את ה Single channel features- עבור ch1 נחשב אמפליטודה.

את כל פעולות אלו אנחנו מבצעים בקוד תחת הפונקציה 'cal_features'- פונקציה זו מתוכננת לחשב את ה features.

```python
def cal_features(ch1, ch2):
    batch_size = ch1.shape[0]  # Actual batch size
    n_fft = int(0.096 * 16000)  # 96ms * 16kHz = 1536
    hop_length = n_fft // 2      # 50% overlap
    window = torch.hann_window(1536).to(ch1.device)
   # STFT computation
    enc_ch1 = torch.stft(ch1.view(batch_size, -1), n_fft=n_fft, hop_length=hop_length, window=window, return_complex=True)
    enc_ch2 = torch.stft(ch2.view(batch_size, -1), n_fft=n_fft, hop_length=hop_length, window=window, return_complex=True)
    # Plot spectrograms for the first item in batch
    plt.figure(figsize=(15, 5))
    # Channel 1 spectrogram
    plt.subplot(121)
    spec1 = torch.abs(enc_ch1[0]).cpu().numpy()
    plt.imshow(20 * np.log10(spec1 + 1e-8), aspect='auto', origin='lower')
    plt.colorbar(label='Magnitude (dB)')
    plt.title('Channel 1 Spectrogram')
    plt.xlabel('Time Frame')
    plt.ylabel('Frequency Bin')

    # Channel 2 spectrogram
    plt.subplot(122)
    spec2 = torch.abs(enc_ch2[0]).cpu().numpy()
    plt.imshow(20 * np.log10(spec2 + 1e-8), aspect='auto', origin='lower')
    plt.colorbar(label='Magnitude (dB)')
    plt.title('Channel 2 Spectrogram')
    plt.xlabel('Time Frame')
    plt.ylabel('Frequency Bin')

    # Save the plot
    os.makedirs('spectrograms', exist_ok=True)
    plt.savefig(f'spectrograms/batch_spectrograms_{time.time()}.png')
    plt.close()

    sc_features = torch.abs(enc_ch1)  # Magnitude spectrogram

    cc = enc_ch1 * torch.conj(enc_ch2)
    ipd = cc / (torch.abs(cc) + 1e-8)
    ipd_ri = torch.view_as_real(ipd)
    ild = torch.log(torch.abs(enc_ch1) + 1e-8) - torch.log(torch.abs(enc_ch2) + 1e-8)
    #ic_features
    ic_features  = torch.cat((ipd_ri[:, :, :, 0], ipd_ri[:, :, :, 1], ild), axis=1)
    return sc_features, ic_features   # return features for both channels
```

*בשלב זה חשוב לשים לב כי בעצם יש לנו שני מסלולים שונים איתם אנחנו פועלים-
מסלול הsingle-Channel - שזהו בעצם מסלול העובד על ערוץ אחד בלבד, ממנו נלמד בעיקר על הסיגנל עצמו, נוכל ללמוד על עוצמת הסיגנל, תבניות בתדירויות שונות, מאפייני הדהוד בערוץ בודד.
ואילו עבור ה Intra-Channel- נוכל ללמוד על ההשוואה בין הערוצים, נחשב את הild IPD , נלמד מידע מרחבי, על כיוון הגעת הקול, מאפיינים של החדר אשר משפיעים על שני הערוצים.

2. שכבות קונבולוציה (Convolutional Layers)- בקוד נמצא חלק זה תחת Model_1

```python
class Model_1(nn.Module):
    def __init__(self):
        super(Model_1, self).__init__()

        # Single-channel path (3 blocks)
        self.sc_blocks = nn.ModuleList([
            nn.Sequential(
                    SeparableConv1d(769, 769, kernel_size=11, dilation=2**i),
                nn.ReLU(),
                nn.LayerNorm([769, 63])
            ) for i in range(3)])

        # Inter-channel path (1 block)
        self.ic_block = nn.Sequential(
           SeparableConv1d(2307, 479, kernel_size=11, dilation=1),
            nn.ReLU(),
            nn.LayerNorm([479, 63]))

    def forward(self, sc_features, ic_features):
        # Process single-channel features
        x_sc = sc_features
        for block in self.sc_blocks:
            x_sc = block(x_sc)
        # Process inter-channel features
        x_ic = self.ic_block(ic_features)
        return x_sc, x_ic
```

בשלב הראשון, נעשה שימוש ב Separable Convolution -כדי לשפר את היעילות של שכבות הקונבולוציה, תוך שמירה על כמות גדולה של מאפיינים שניתן ללמוד.

```python
class SeparableConv1d(nn.Module):
    """separable 1D convolution"""
    def __init__(self, in_channels, out_channels, kernel_size, dilation=1, padding='same'):
        super(SeparableConv1d, self).__init__()
        self.depth_wise = nn.Conv1d(
            in_channels=in_channels,
            out_channels=in_channels,
            kernel_size=kernel_size,
            dilation=dilation,
            padding=padding,
            groups=in_channels
        )
        self.point_wise = nn.Conv1d(
            in_channels=in_channels,
            out_channels=out_channels,
            kernel_size=1
        )

    def forward(self, x):
        x = self.depth_wise(x)
        x = self.point_wise(x)
        return x
```

במודל 1 נשתמש בקונבולוציה רגילה, כאשר מבצעים קונבולוציה רגילה, אנחנו מבצעים קונבולוציה על כל התכונות בכל הערוצים במקביל- נשתמש במטריצה של פילטרים על כל הערוצים- כל פילטר נועד בכדי שנפיק מפת פילטרים חדשה על סמך השכבה הקודמת- בדרך זו אנחנו מבצעים את הפעולה כל פעם עבור כל הפיצ'רים, לכן אנחנו צריכים לבצע המון קונבולוציות שהן יקרות מאוד מבחינת זמן חישוב ומשאבי מערכת- לכן בחרו להשתמש ב**Separable Convolution** - זוהי בעצם סוג של שכבת קונבולוציה במימד אחד שהיא יעילה יותר.

שכבה זו בעצם מחלקת את הפעולה לשני שלבים נפרדים:
א. קונבולוציה depth wise , הפעולה מתבצעת עבור כל ערוץ בנפרד, כל פילטר מוחל רק על ערוץ אחד ולא על שני הערוצים יחד- כלומר נשתמש בפילטר קונבולציוני קטן וכל פילטר מוחל רק על ערוץ בודד בכל פעם.
ב. קונבולוציה point wise – אחרי שביצענו את הפעולה עבור כל ערוץ בנפרד עלינו לשלב את המידע משני הערוצים. בחלק זה נשתמש בפילטר של 1X1 , נאחד את התכונות למטריצת תכונות משותפות.

נראה דוגמה, עבור קונבולוציה רגילה:

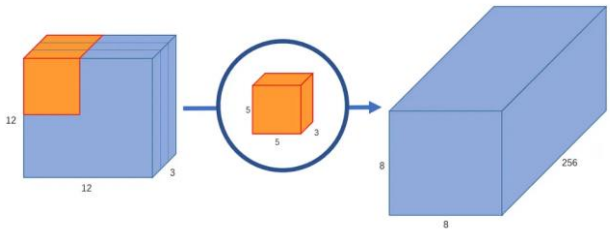
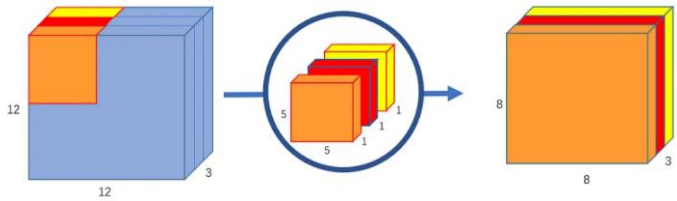
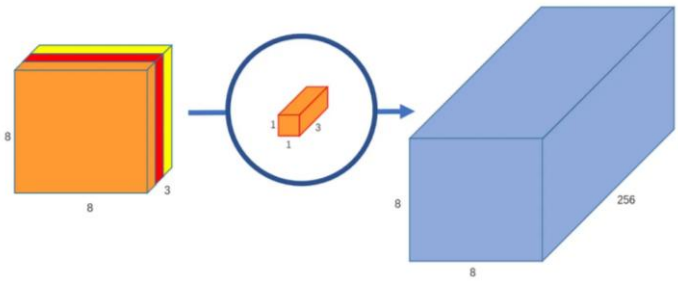


Image 5: A normal convolution with 8×8×256 output

עבור Depth wise:



עבור Point wise:

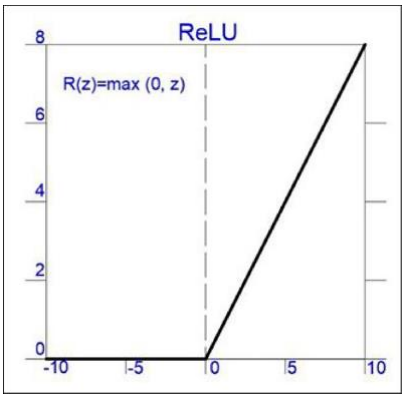


בסך הכל, בעזרת שימוש בשיטה זו אנחנו מצמצמים את כמות הפרמטרים בתהליך האימון, המודל שלנו קל ומהיר יותר, דרישות זיכרון נמוכות יותר, יש לנו פחות פעולות חישוב לכן העומס קטן והרשתות מהירות יותר, למרות כל זאת המודל עדיין מצליח לשמור על הדיוק והיכולת לזהות תבניות משמעותיות.
נשים לב כי המסלול החד ערוצי נכנס ל3 שכבות 1D-conv ואילו המסלול הבין ערוצי נכנס לקובלוציה בודדת.

נסתכל על המסלול החד ערוצי- כל אחד משלושת היחידות קונבולוציה מכילה מאפיינים שונים, כל אחת מהן מכילה שכבה של קונבולוציה נפרדת אותה הסברנו, פונקציית אקטיבציה ReLu שמוסיפה אי ליניאריות וLayerNorm - שזוהי שכבת נרמול שמטרתה להפחית שונות בין הדוגמות, עבור כל אחת מהשכבות בחלק זה יש dilation שונה- זהו בעצם המרווח בין הדגימות של הקרנבל.

עבור המסלול הבין ערוצי- נכנס לשכבת קונבולוציה בודדת הכוללת שכבה של קונבולוציה מופרדת, פונקציית אקטיבציה ReLu ו LayerNorm.

ReLu -היא פונקציית האקטיבציה- מטרתו להוסיף אי-ליניאריות למערכת- מחזירה את הערך אם הוא חיובי, ואם הערך של שלילי, הפונקציה מחזירה 0.



$$\max(0, x) = \mathrm{ReLU}(x)$$

Layer Normalization - אחרי כל שכבת קונבולוציה, שכבת נרמול (LayerNorm) משמשת לייצוב התהליך האימוני- נרמול זה מונע חוסר יציבות במהלך הלמידה ומאפשר למודל להתמקד במידע החשוב באותות.

3. שכבות Model_3-Fully Connected

Model_3 הוא המודל הFully connected – מטרתו להעריך את הפרמטרים של החדר מתוך המאפיינים שחילצנו ע"י Model_1. המודל לוקח וקטור קלט אחד ומבצע עליו עיבוד דרך מספר שכבות מקושרות כדי להפיק שני סוגי פלטים- הערכה של הממוצעים והערכה של השוניות- הקוד המוצג הינו הקוד הסופי לאחר הניסיון לתקן בעיות של over fitting שנתקלנו בה.

```
class Model_3(nn.Module):
    """Parameter estimation model as described in the paper"""
    def __init__(self):
        super(Model_3, self).__init__()
        self.fc1 = nn.Linear(1248, 96)
        self.dropout1 = nn.Dropout(0.3)
        self.fc2 = nn.Linear(96, 48)
        self.dropout2 = nn.Dropout(0.2)  # Define the dropout2 layer here
        self.fc3 = nn.Linear(48, 28)  # 14 means + 14 variances
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.dropout1(x)
        x = self.relu(self.fc2(x))
        x = self.dropout2(x)
        x = self.fc3(x)
        # Split output into means and variances
        means, log_vars = torch.split(x, 14, dim=1)
        vars = torch.exp(log_vars)  # Ensure positive variances
        return means, vars
```

נראה כי בחלק זה, לאחר שהיה לנו בעיה של over fitting ביצענו פעולות של dropout במטרה שישפרו את האקראיות בחלק האימון בלבד.

4. **Ensemble**- מייצרת את המודל הכולל של המאמר.

```
# Ensemble model
class Ensemble(nn.Module):
    """Complete model as described in the paper"""
    def __init__(self, model1, model2):
        super(Ensemble, self).__init__()
        self.feature_extractor = model1
        self.parameter_estimator = model2
        self.avg_pool = nn.AdaptiveAvgPool1d(1)

    def forward(self, sc_features, ic_features):
        # Extract features
        x_sc, x_ic = self.feature_extractor(sc_features, ic_features)

        # Concatenate features
        x = torch.cat([x_sc, x_ic], dim=1)  # Along channel dimension

        # Global average pooling
        x = self.avg_pool(x)
        x = x.view(x.size(0), -1)  # Flatten

        # Estimate parameters
        means, vars = self.parameter_estimator(x)
        return means, vars
```

חלק זה משלב בין שני התת מודלים שראינו- המודל בו מתבצע חילוץ המאפיינים משני סטים של נתונים והתת מודל שאחראי לשערך את הפרמטרים של החדר.

לאחר מכן- nn.AdaptiveAvgPool1d שכבה זו בעצם מבצעת מיצוע גלובלי לאורך מיימד מסוים.

כעת, אחרי שהסברנו את המודל- אנחנו בעצם עוברים להגדרת פונקציית ה Train, Val שלנו.

- **<u>פונקציית Train:</u>** מטרתה לאמן את המודל על ידי חישוב הלוס ואז עדכון המשקלים באמצעות backpropagation.
בפונקציה אנחנו מבצעים מספר שלבים, ראשית נבנה את המודל, נגדיר כי אנחנו במצב אימון, כלומר dropout פעיל ואנחנו מעדכנים את המשקלים.
נקבל מ DataLoader את הקלטים, נבצע הפרדה בין שני הערוצים, ונקבל את ה target שלנו- שזה בעצם מה שאנחנו רוצים לשערך.
נחשב את הפיצ'רים דרך הפונקציה שהסברנו במודל ונחשב את הפלטים במודל.
חישוב הLOSS שבעזרתם נגדיר ונשפר את המשקלים.

עדכון המודל- נאפס את הגרדיאנטים optimizer.zero_grad(), נחשב אותם loss.backward() ונעדכן את המשקלים באמצעות ה optimizer.step()

בנוסף, הוספנו מגנון של <u>Early Stopping</u> בחלק מהניסיון שלנו לפתור את בעיית הover fitting שהייתה לנו. הפסקת האימון כאשר המודל מתחיל להראות ירידה בביצועים של נתונים הוולידציה. המודל ישמור את המשקלים שעבורם השיג את הביצועים הכי טובים ולא ימשיך כאשר יש יציבות בערך הLOSS.

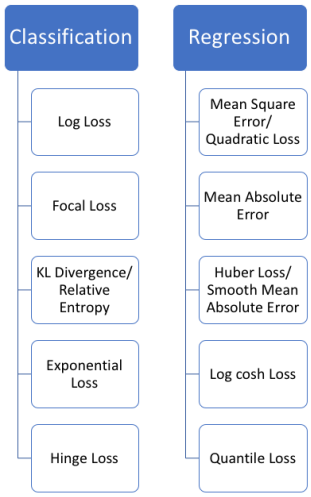
- **<u>פונקציית Val:</u>** מטרתה להעריך את ביצוע על סט ולידציה- ללא עדכון המשקלים.
באופן כללי נעבור את אותה הדרך כמו בפונקציית האימון רק שלא נחשב את הגרדיאנטים, אין קריאה לאופטימייזר או לפונקצית train.
נחשב את הloss אך החישוב יתבצע רק למטרות העערכה ולא משפיע על המשקלים.

<u>פונקציית שגיאה (LOSS) :</u>

פונקציית השגיאה מודדת את ההבדל בין הערך שהמודל חזה לבין הערך האמיתי היא מספקת מדד כמותי לביצועי המודל- משמשת להכוונת תהליך הלמידה ועדכון המשקולות, בחירה לא מתאימה עלולה להוביל לביצועים גרועים, מאוד חשוב להתאים את הפונקציה בה נשתמש לסוג הבעיה.

מבחינת סוגי פונקציות LOSS:

כאשר החלוקה מתחלקת בין



פונקציית ההפסד בה אנחנו משתמשים: **The negative log likelihood loss - NLLloss**
היא מחשבת את הלוס ע"י ההנחה שהפלטים מתפלגים גאוסיאנית (נורמלית):

$$\log(\sigma) + \frac{^2(y-\mu)}{^2 2\sigma} = \text{Loss}$$

כאשר:

- y: הערך בפועל (target).
- μ: הערך החזוי (mean).
- σ^2: שונות חזויה (variance).
- הלוס מעניש סטיות גדולות ו/או שונות לא ריאלית.

יצירת הDATA ואימון המודל

כעת, עברנו לשלב בו אנחנו רוצים לייצר את הDATA ולאמן אותו.

<u>קביעת ערכים התחלתיים במערכת</u>

האופן שבו נקבע את ערכיהם ההתחלתיים ישפיע משמעותית על יעילות תהליך הלמידה ורמת הדיוק שנשיג. אפשרות פשוטה היא להתחיל עם ערך אפס לכל המשתנים.
גישה זו בעייתית – בשביל לא לקבל מצב בו כל השינויים יהיו זהים , נדרוש כי תנאי ההתחלה לא יהיו 0, וכך הנגזרות לא יהיו זהות- נבחר ערכים קטנים ואקראיים.

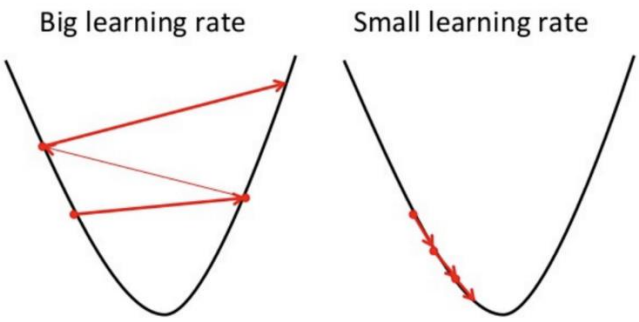
הגדרת hyperparameter ,<u>גודל קבוצת האימון (Batch Size)</u> כשבוחרים קבוצה קטנה מאוד (למשל גודל 1), התהליך מהיר אך רועש - העדכונים מתבססים על מידע מצומצם מאוד. היתרון הוא שחישוב ה-forward/backward pass מהיר. לעומת זאת, קבוצה גדולה תניב תוצאות מדויקות יותר כי השונות קטנה (בזכות מיצוע על יותר דוגמאות), אבל התהליך יהיה איטי יותר.
בתחילת האימון כדאי לנסות מספר ערכים ולבחור את זה שנותן את התוצאות הטובות ביותר.

<u>קצב הלמידה (Learning Rate):</u>

אחד הפרמטרים החשובים ביותר באימון רשתות נוירונים ומודלים של למידת מכונה. תכונה זו חשובה מכמה סיבות, הראשונה היא הינה ההשפעה על התכנסות- קצב למידה גבוה מדי עלול לגרום לאי-התכנסות או "קפיצות" מעבר למינימום הרצוי.
קצב למידה נמוך מדי יוביל להתכנסות איטית מאוד או "תקיעה" במינימום לוקאלי ואילו קצב למידה אופטימלי יאפשר התכנסות מהירה ויציבה למינימום הגלובלי

בחירת קצב למידה טוב נותן לנו איזון בין מהירות לדיוק, נדרש איזון עדין בין התקדמות מהירה לבין דיוק בלמידה, קצב גבוה מאפשר התקדמות מהירה אך עם פחות דיוק ואילו קצב נמוך מספק דיוק גבוה אך דורש זמן רב יותר.

תכונה זו משפיעה באופן ישיר על איכות התוצאות הסופיות, על על זמן האימון הכולל -בחירה נכונה של קצב הלמידה היא קריטית להצלחת תהליך האימון, והיא דורשת הבנה עמוקה של האלגוריתם והנתונים, יחד עם ניסיון מעשי בתחום.



בקצב למידה נמוך, הלוס אמנם יורד אך בקצב איטי מאוד. כשנגיע לקצב הלמידה האופטימלי, נוכל לראות ירידה מהירה(כלומר נגזרת הגרף באיור 31 הכי גדולה עבור קצב הלמידה האופטימלי) בפונקציית הלוס.

חלוקת הDATASET

מחלקים את מאגר הנתונים לשלושה חלקים עיקריים, כשלכל חלק תפקיד ייחודי:

1. נתוני אימון (Training Dataset)- המידע העיקרי המשמש ללמידה פעילה, באמצעותו המודל מעדכן ומכייל את המשקולות שלו, זהו החלק הגדול ביותר מתוך המאגר.

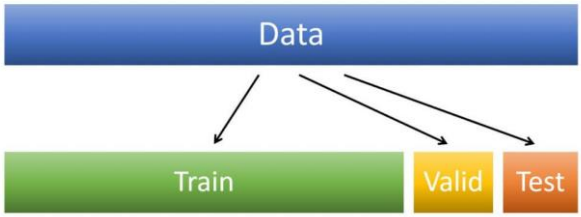
2. נתוני אימות (Validation Dataset)

משמש לבדיקת ביצועי המודל במהלך האימון, נבדק בסוף כל מחזור אימון (epoch) או מקבץ (batch), עוזר בכיוון הפרמטרים הגבוהים (hyperparameters), המודל "רואה" את הנתונים אך לא לומד מהם ומספק משוב חיוני לשיפור תהליך האימון

3. נתוני בחינה (Test Dataset)

נשמר לבדיקה סופית של המודל המאומן, מודד את יכולת ההכללה על נתונים חדשים

ובעצם משקף את הביצועים הצפויים בעולם האמיתי



כעת, לאחר שעשינו את כל זה, הרצנו את המודל בעזרת פונקציית main שבנינו שתפקידה לנהל את התוכנית ולהפעיל את המודל והבדיקה.
בחלק זה אנחנו בעצם מגדירים את התשתית למודל, יוצרים תיקיות- הקוד מכין תיקיות עבור הפלט של התוכנית, כולל תיקיות לשמירת נתוני האימון, מודלים שמורים, וגרפים.

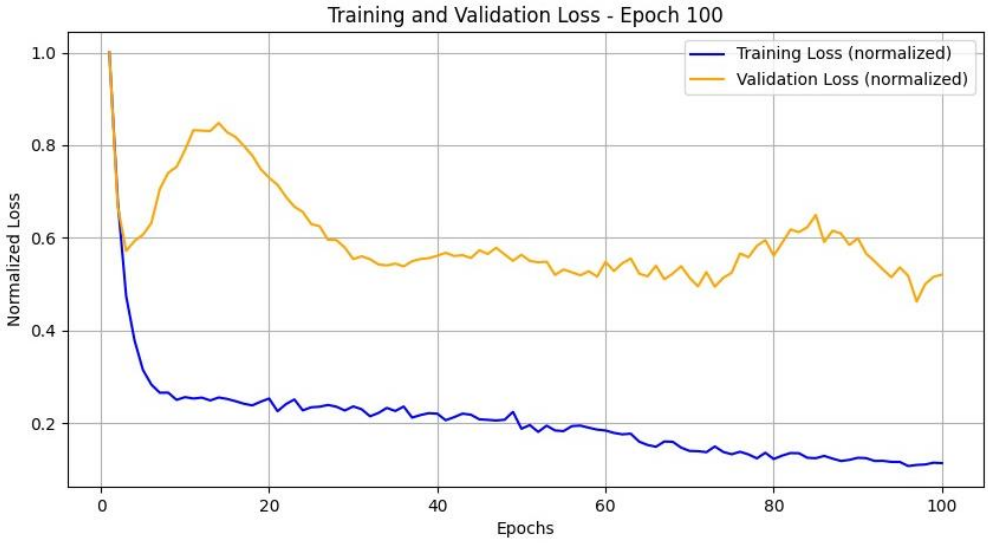
הפונקציה מטעינה ומכינה את ערכות הנתונים לאימון, validation וtest , כולל החלוקה לקבצים, בנוסף בחלק זה אנחנו נגדיר פרמטרים לנורמליזציה- נקבע פרמטרים של סטיית תקן עבור ערכים שונים.

בחלק זה אנחנו נרכיב את לוגיקת האימון, הכנת האימון- יצירת המודלים והגדרתם יצירת והגדרה של מודלים, אובייקטים לאופטימיזציה, ופרמטרים נוספים. לולאת האימון כוללת אימון ואימות של המודל על פי הנתונים שהוכנו, כולל שמירת המודל הטוב ביותר, עדכון סטטוס והצגת התקדמות.

כוללת שמירת המודל לאחר כל עידן מסוים, הצגת תוצאות האימון, והדפסת סיכום ההתקדמות.

התחלנו מלהריץ את המודל עבור כמות דטא קטנה- כך שהיה לנו קל ונוח יותר לעשות דיבאג ולהבין אם והיכן יש שגיאות; תהליך מציאת השגיאות לקח לנו זמן אך לבסוף הצלחנו- וקיבלנו את הגרף

הבא עבור דטא של 50 חדרים-



אכן נראה כמו גרף תקין באופן חלקי- בהתחלה של האימון הרשת אכן לומדת- הלוס בולידציה והאימון שניהם יורדים, אולם לאחר מספר סבבים (לא רב) רק הלוס של האימון ממשיך לרדת ואילו הולידציה מתחיל לעלות ולאחר מכן קצת יורד ומגיע לאיזה מצב יציב כלשהו.

Training loss- יורד מאוד במהירות בתחילת האימונים ומתייצב על ערך נמוך מאוד. זה מראה שהמודל לומד את הדטא האימונית ביעילות.

Validation Loss – נראה ירידה בהתחלה ולאחר מכן עלייה ושוב ירידה, עם יציבות יחסית בסביבות ערך אחד. תנודות בגרף יכולות להצביע על כך שהמודל אולי מתקשה להכליל מהדטא האימונית לדטא שלא ראה קודם.

שוב, זהו רק גרף ראשוני- אי אפשר באמת לדעת את הלך הרוח, כעת- נגדיל את הדטא.

במשך תקופה יצרנו דטא של 20,000 חדרים השמורים בקובץ CSV כאשר עבור כל חדר שמור 20 משתנים חשובים:

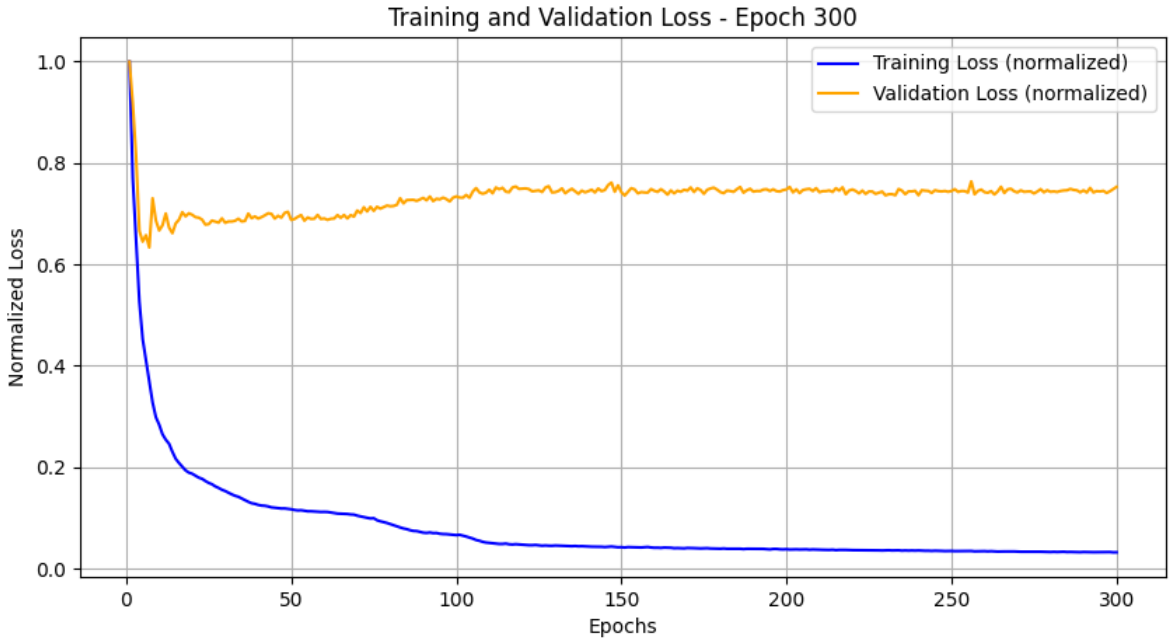
מבחינת מיימדי החדר: מספר חדר, שטח פנים, נפח, אורך, רוחב.

rt60 עבור כל אחד מהאוקטובות (125,250,500,1000,2000,4000)-6 פסי אוקטובה.

mean absorption- מקדם הספיגה הממוצע נמדוד עבור ששת פסי האוקטבה (125,250,500,1000,2000,4000)-. מקדם זה מתאר כמה מהאנרגיה של הקול נספגת על ידי המשטחים השונים בחדר, כלומר כמה מהקול נשאר ומהדהד, וכמה נספג.

rt60 effective – שזה בעצם מתאר את זמן ההדהוד האפקטיבי.

לאחר שסיימנו לייצר את הדטא עבור כל החדרים שלחנו את המודל לאימון עבור 20,000 חדרים ובסופו קיבלנו את הגרף הבא:



מבחינת הקו הכחול אשר מתאר את הדטא השייך ל Train:

הקו הכחול, שמייצג את הLOSS עבור האימון, יורד בחדות באפוקים הראשונים ולאחר מכן ממשיך לרדת בהדרגה. זה אופייני למודלים של למידת מכונה, במיוחד רשתות נוירונים, שבהם המודל לומד מהר בהתחלה ולאחר מכן מאט כאשר הוא מכוונן את הפרמטרים שלו.

ערך הLOSS באימון בסיום- בסוף 300 אפוקים, איבוד האימון נמוך מאוד, כמעט אפסי. הדבר מצביע על כך שהמודל מתאים את עצמו לנתוני האימון בצורה טובה, שזהו סימן חיובי לכך שתהליך הלמידה פועל כראוי. נוכל לראות כי יש ירידה חלקה באיבוד האימון, ללא תנודות משמעותיות, מצביעה על כך שתהליך הלמידה של המודל יציב.
במקרים מסוימים, תנודות יכולות להופיע אם קצב הלמידה גבוה מדי, אך זה לא נראה כבעיה כאן.

מבחינת הקו הכתום אשר מתאר את הדטא השייך ל Val:

נראה כי הירידה הראשונית דומה למה שראינו עבור האימון- המודל לומד דפוסים כלליים בנתונים, המודל מתקדם ומתחיל ללמוד להכליל נתונים חדשים.

לאחר הירידה הראשונית נראה התייצבות בגרף והוא נשאר יחסית קבוע- המודל הגיע לנקודה בה הוא לא מצליח להשתפר יותר, לכן נוצר מצב כי עבור האימון נקבל תוצאות טובות יותר באופן משמעותי מהוולידציה- תופעה זו מצביעה על כך שהמודל מתאים את עצמו לנתוני האימון ולא מצליח לבצע הכללה יעילה על נתונים חדשים- **overfitting**.

Overfitting הוא מצב שבו מודל מתאים את עצמו בצורה מדויקת מדי לנתוני האימון. במקום ללמוד את הדפוסים הכלליים והמשמעותיים שנמצאים בנתונים, המודל לומד גם את הרעש או הפרטים הקטנים, המיוחדים והלא-רלוונטיים בנתונים. זה מוביל לכך שהמודל מצליח מאוד על נתוני

האימון אך מתקשה לבצע הכללה על נתונים חדשים, כלומר ביצועיו על נתונים חדשים עלולים להיות גרועים-וזה לצערנו מה שאנחנו רואים פה.

התאמת יתר מתרחשת כאשר מודל לומד את הפרטים וה"רעש" של נתוני האימון באופן שמזיק לביצועיו על נתונים חדשים. במילים אחרות, המודל "זוכר" את נתוני האימון אך לא מצליח להכליל היטב.

התאמת יתר אומרת שהניבויים של המודל על נתונים חדשים עלולים לא להיות מדויקים או אמינים.

יש מספר דרכים לפתור בעיה של Overfitting:

- **ריגולריזציה** שימוש בטכניקות ריגולריזציה כגון ריגולריזציה מסוג L2 (הנקראת גם דעיכת משקלים) או Dropout יכולה לעזור למנוע מהמודל ללמוד רעשים או פרטים עודפים בנתוני האימון.
- **עצירה מוקדמת** הפסקת האימון כאשר המודל מתחיל להראות ירידה בביצועים של נתונים הוולידציה. המודל ישמור את המשקלים שעבורם השיג את הביצועים הכי טובים ולא ימשיך כאשר יש יציבות בערך הLOSS.
- **Hyperparameter** כוונון פרמטרים כגון קצב הלמידה או גודל ה batch-עשוי לשפר את ההכללה. לדוגמה, קצב למידה נמוך יותר עשוי לעזור למודל ללמוד דפוסים עדינים יותר בנתונים.
- **שימוש ב K-Fold Cross Validation –** שיטה זו אומרת כי בעצם נחלק את הקבוצת נתונים לK חלקים שווים בגודלם, בכל איטרציה נשתמש ב k-1 חלקים כקבוצת אימון ואילו החלק הנותר משמש אותנו בתור קבוצת בדיקה.
  השימוש בכמה קבוצות בדיקה ואימון מפחית את הסיכוי לכך שהמודל ילמד "לזכור" רק את הדוגמאות הספציפיות בקבוצת אימון אחת. התוצאות הכלליות נותנות תמונה מדויקת יותר של היכולת של המודל להכליל על נתונים חדשים.

  דוגמה איך עובד האלגוריתם:
  אם יש לך 1000 דוגמאות נתונים ו:K=5-

  תחלק את הנתונים ל-5 קבוצות בגודל 200 כל אחת, באיטרציה הראשונה, הקבוצה הראשונה (200 דוגמאות) תשמש לבדיקה, ו-800 האחרות לאימון, התהליך יחזור 5 פעמים, ובכל פעם קבוצה אחרת תשמש לבדיקה, לאחר מכן תחשב את ממוצע הביצועים מכל 5 האיטרציות.

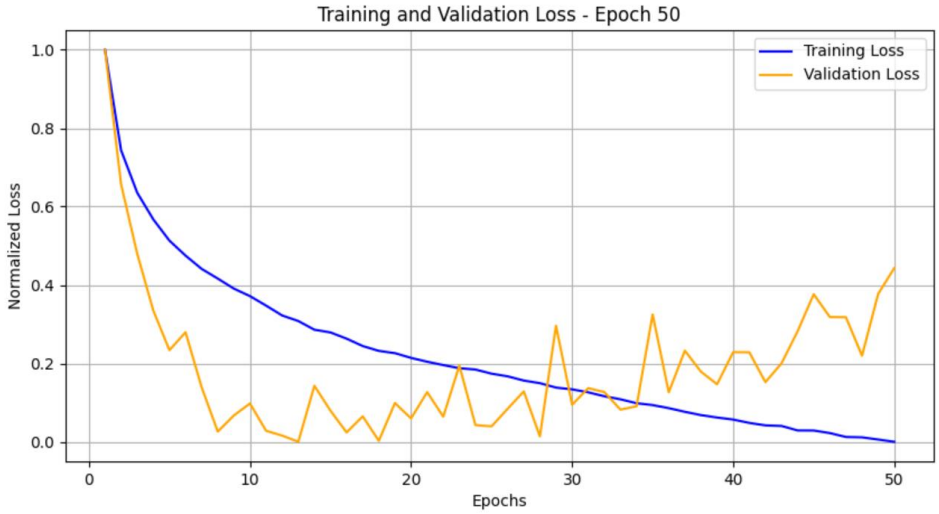
- להיגרר כתוצאה מבעיה **בdata** שלנו.

בפרויקט שלנו בשלב זה בתור התחלה ניסינו להוסיף שכבות Drop out בשלב מודל3– פעולה זו בעצם מבצעת ביטול לחלק מהניורנים שמתבצע בצורה רנדומלית עבור ניורנים במהלך האימון- זה בעצם גורם למודל לא לסמוך על נוירונים מסוימים, דבר הגורם למודל להצטרך להיות יותר מוכלל.

בנוסף שינינו את הקצב למידה שלנו- יכול להיות שהקצב למידה שלנו לא היה תואם למודל. הוספנו יכולת של Early stopping כך שהמודל יעצור במקרה שאנחנו נתקלים בover fitting.

וכן ביצענו k-cross-validation, קבענו את k=4- לאחר מכן מחקתי אותה מאחר והיא לא תרמה.

הרצנו את המודל עם שלל שינויים ועדיין קיבלנו כי הבעיה קיימת, המודל תנודתי ולא מתון וכן קיים פער משמעותי בין תוצאות ההפסד בין הtrain ל הval :



הגרף מציג את אובדן האימון והאובדן של האימות לאורך 50 איפוקים של אימון.
ניתן להבחין במספר דפוסים:

1. אובדן האימון מתחיל ברמה גבוהה מאוד ויורד במהירות בעשרת האיפוקים הראשונים. זה מצביע על כך שהמודל מסוגל ללמוד בצורה יעילה על נתוני האימון- סימן חיובי.
2. עם זאת, אובדן הואלידציה לא עוקב אחר אותו מסלול כמו אובדן האימון. במקום זאת, הוא נשאר יחסית גבוה ומשתנה בצורה בלתי יציבה לאורך תהליך האימון. לאחר הירידה התלולה הראשונית, אובדן האימון ממשיך לרדת בהדרגה, בעוד שאובדן האימות מתנודד למעלה ולמטה ללא מגמה ברורה. זוהי תופעה טיפוסית של יתר התאמה, שבה המודל מתבצע היטב על נתוני האימון אך נכשל בהכללה לנתוני האימות.
3. לקראת סוף 50 האיפוקים, אובדן האימות למעשה מתחיל לעלות שוב, בעוד שאובדן האימון נשאר נמוך. זה מחזק את הרושם שהמודל סובל מיתר התאמה ואינו מסוגל להכליל היטב.

בכללות, הגרף הזה מצביע על כך שהמודל אינו מתפקד כפי שהיה רצוי. הפער הגדול בין אובדן האימון לאובדן האימות, יחד עם העלייה באובדן האימות בסוף, מרמזים שהמודל מתקשה ללמוד ייצוגים משמעותיים שניתן ליישם על נתונים חדשים ובלתי מוכרים. זהו דפוס מדאיג שיביא סביר להניח לביצועים גרועים על נתוני בדיקה בעולם האמיתי.

מה שגרם לנו לחשוב שהבעיה נובעת אכן מהDATA.

הצעות לשיפור

כמו שאנחנו רואים, הרשת שלנו לא מבצעת את תפקידה נאמנה וסובלת מבעיית התאמת יתר- במטרה לשפר את הרשת נוכל לבצע מס' דברים:

1. מציאת הבעיה ביצירת הדטא, שפוגעת ביכולת הלמידה של הרשת, שכן היא לומדת נתונים המיוצרים בצורה שגויה.
2. הגדלת גודל הדגימה של הנתונים - ככל שיש יותר נתונים לאימון, המודל יוכל ללמוד ייצוגים כלליים יותר ולהתמודד טוב יותר עם הנתונים החדשים.
3. שימוש בטכניקות רגולריזציה נוספות, כמו L1/L2 regularization אוDropConnect.
4. מודל חלופי, אולי ארכיטקטורת המודל לא טובה לנתונים שלנו.
5. שימוש בטרנספורמציות על הנתונים כדי להרחיב את מגוון הדוגמאות שהמודל רואה באימון.
6. ביצוע הכוונה מחדש (fine-tuning) של המודל על אוסף נתונים קטן יותר שדומה יותר לנתונים שבהם המודל אמור להשתמש בסופו של דבר.

בסך הכל, קיימות מספר אפשרויות נוספות שניתן לבחון כדי להתמודד עם בעיית ההתאמת יתר ולשפר את ביצועי המודל על הנתונים החדשים. החשוב הוא להמשיך לנסות ולבחון אפשרויות שונות