שם בית הספר: מקיף י"א ראשונים

שם העבודה: זיהוי שיר טראנס או שיר ג'אז. (classification)

שם התלמיד: ליאב עובדיה

תעודת זהות: 325697571

שם המנחה: דינה קראוס

שם החלופה: פרויקט בלמידת מכונה

תאריך הבחינה: 5.7

תוכן עניינים:

מבוא:

השנה, במסגרת התמחות Deep Learning Computer Vision שנלמדת זו השנה השנייה בבית הספר שלי, נדרשנו לבצע את פרויקט הגמר שלנו בנושא זה. ספר זה סוקר את הפרויקט שלי בנושא, וכולל בין היתר את הרקע לפרויקט, הוראות הרצה, הסברים ותרשימים על תכולתו עבור משתמשים ומפתחים, מילון מושגים, מסקנות מההרצה וסיכום אישי שלי על התהליך כולו.

הפרויקט שלי הוא מתחום הסיווג בלמידת במכונה(classification) והוא מסווג בין שירי טכנו לג'אז. לפני שאתחיל להסביר על הפרויקט הספציפי שלי אסביר קודם כל על תחום הלמידה העמוקה. למידת מכונה (Machine Learning) הוא תת-תחום במדעי המחשב (Computer science) ובבינה מלאכותית (Artificial intelligence) העוסק בפיתוח אלגוריתמים המיועדים לאפשר למחשב ללמוד מתוך דוגמאות, ופועל במגוון משימות חישוביות בהן התכנות הקלאסי אינו אפשרי. הלמידה העמוקה (Deep Learning), היא תת תחום בMachine Learning אשר מתבסס על רשתות נוירונים בעלות מספר רב של שכבות. בתחום זה אנו מתבססים על ההנחה שהמחשב יכול ללמוד וללמד את עצמו בדומה למוח האנושי, ולמעשה מנסים לחקות אותו. מטרתו של תחום הDL היא ליצור חיקוי ממוחשב של פעולת המוח האנושי.

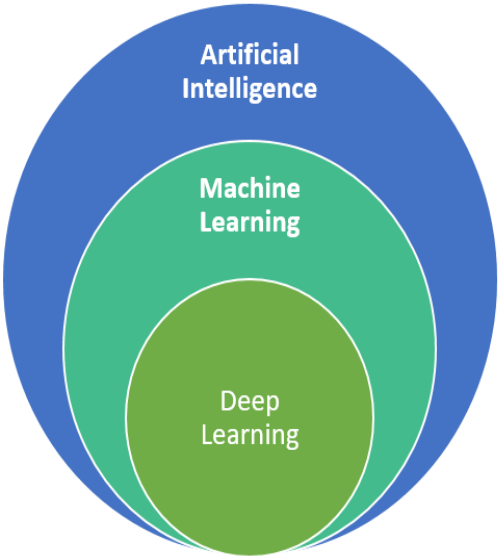


Figure 1: artificial intelligence, machine leaning and deep learning Source: Nadia BERCHANE (M2 IESCI, 2018)

קהל היעד של הפרויקט הוא אנשים שמתחילים את דרכם במוזיקה ורוצים ללמוד ולדעת איזה שיר משתייך לאיזה ז'אנר, נוסף על כך ניתן להשתמש בפרויקט זה בתחנות רדיו מסוימות שמשדרות את שני הז'אנרים הללו ובכך לדעת מה הם משדרים כל פעם, ומזה לדעת כמה פעמים שידרו כל דבר, אחוזים ועוד המון נתונים כאלה ואחרים. הפרויקט עובד בעצם על קטעי מוזיקה מהם הוא לומד ואותם הוא מזהה, אם כי, המחשב אינו יכול לשמוע כמו בן אדם ולזהות רק לפי זה, לכן יש דרך להפוך את קטעי הקול עליהם הוא לומד למטריצה מספרית כאשר לשירי הטראנס יש קשר ישיר בין המטריצות ולשירי הג'אז יש קשר ישיר בין המטריצות ובכך הוא מזהה את ההבדלים בין שני הסוגים. (כי הרי בסופו של דבר קול הוא תנודות באוויר שאפשר למדוד אותם במספרים, ולכל צליל יש מספר אחר, ולכל ז'אנר יש צלילים אחרים).

באופן כללי יש המון תוכנות וקטעי קוד על הנושא הזה, התוכנה המוכרת ביותר היא shazam שמזהה יותר מאת ז'אנר השיר, אלא ממש את השיר עצמו. במחקר שעשיתי מצאתי את השיטה בה אפתור את הפרויקט(הפיכת השיר למספר וחלוקת השירים למקטעי קול שווים בזמנם).

את הקוד מריצים דרך anaconda prompt ואופן ההרצה בשביל המשתמש הוא פשוט מאוד.

את מאגר השירים- the dataset לקחתי מיוטיוב, כאשר מצאתי קישור ל2 פלייליסטים מאוד גדולים של טראנס ושל ג'אז, ולקחתי סקריפט שמוריד את הפלייליסטים האלה לזיכרון ויוצר פולדר שלהם, פולדר של טראנס ופולדר של ג'אז.

האתגר הכי גדול שלי היה להבין בכלל מאיפה אני צריך להתחיל לכתוב את הקוד, קראתי המון על למידת מכונה ועל בינה מלאכותית לפני אך עדיין לקח לי זמן להבין מאיפה להתחיל כדי שיהיה לי בסיס, האתגר הזה גם נבע המון מכך שרוב החברים שלי עשו פרויקטים על תמונות ולכן לא יכלתי לשאול ולהתייעץ איתם לגבי הפרויקט שלי. תוך כדי כמובן צצו עוד אתגרים ועוד Errors אחרים ומעצבנים שהייתי צריך להתמודד איתם וללמוד איך לעקוף אותם.

מבנה/ארכיטקטורה של הפרויקט:

קצת מושגים שבתחום, על מנת שלקורא יהיה בסיס כאשר אני כותב מושגים כאלו ואחרים:

מושגים חשובים בנושא למידת מכונה – deep learning:

Dataset- מאגר השירים: המודל צריך ללמוד תמונות וכדי לעשות את זה אנו נותנים לו מאגר של שירים המסווג לקטגוריות.(טראנס או ג'אז). מאגר זה מתחלק לשלושה מאגרי מידע- train, test, validation.

Accuracy – אחוז ההצלחה של המודל בחיזוי הקטגוריות של השירים (נכון או לא נכון).

Loss - מגדיר כמה קרובות היו תוצאות חיזוי השירים לקטגוריה האמיתית שאליה השיר שייך.

Train data- השירים אותן המודל לומד בעת תהליך האימון, מאגר זה הוא הגדול מבין שלושת תתי המאגרים.

Tast data – התמונות אותן המודל אינו לומד, אלא מנסה לזהות בתום תהליך הלמידה.

Validation data – validation הוא דמוי test שנערך במהלך תהליך הלמידה. מטרתו להציג בפני המשתמש את אחוזי ההצלחה של המודל כבר בעת הלמידה שלו. על כן validation data אלו שירים שהמודל אינו לומד, אלא מנסה לזהות במהלך האימון שלו.

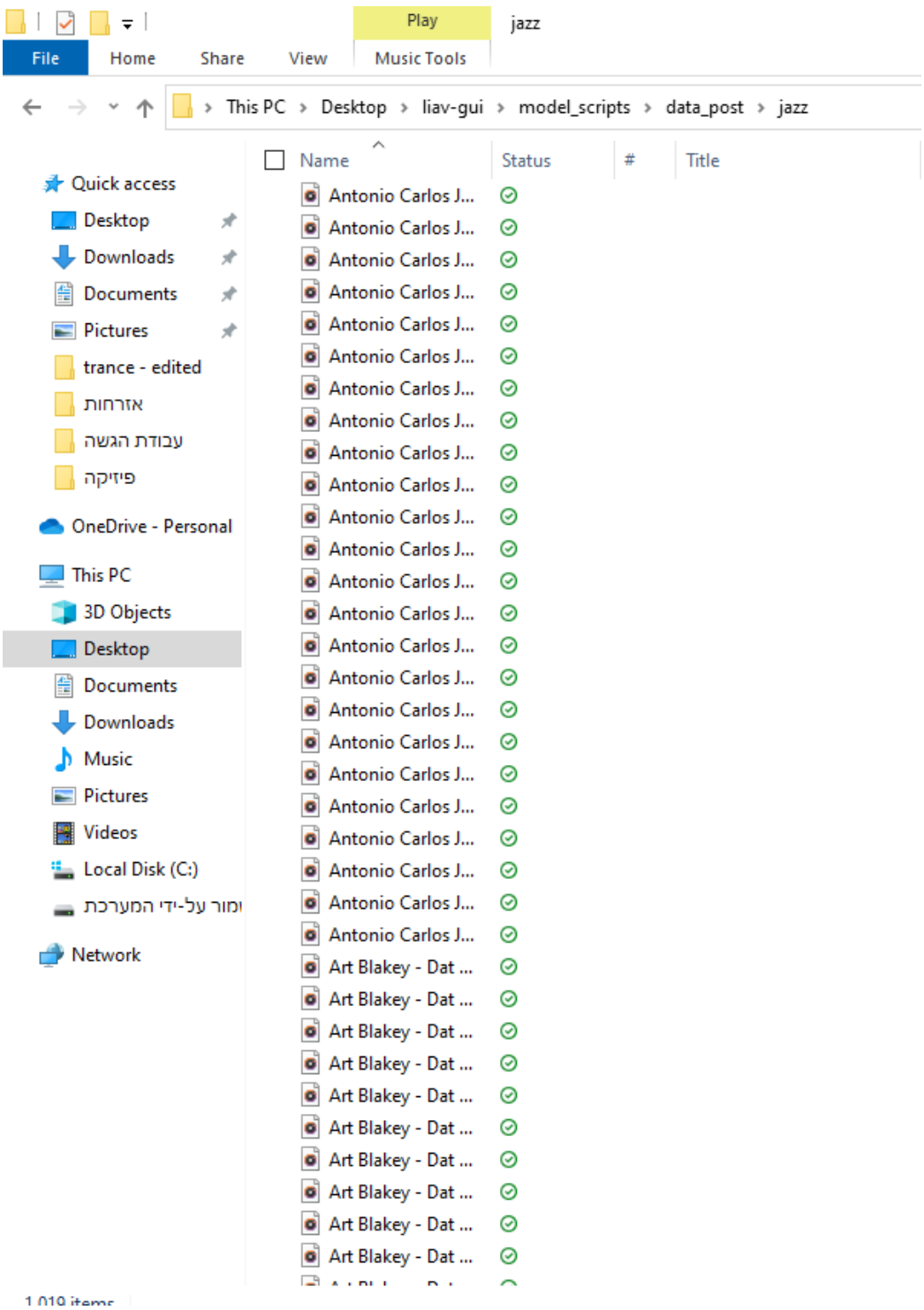
לכל אחד משלושת חלקי המאגר יש ערכי accuracy ו loss. כך ניתן לבחון האם המודל אכן מבצע למידה ואם כן, כמה הוא מצליח.

שתי בעיות שעלולות להיווצר הן: overfitting ו-under fitting:

Overfitting- "התאמת יתר" היא מצב בו המודל מותאם יתר על המידה למאגר אותו הוא לומד, ולכן פחות מצליח בביצוע תחזיות. ניתן לזהות מצב כזה כאשר ה-validation accuracy גבוה משמעותית מן ה-training accuracy או כאשר ה-validation loss קטן משמעותית מן ה-training loss.

Under fitting – ההפך מ-overfitting, מצב בו מאגר הלמידה פשוט מידי ולא כולל מספיק מגוון שלתמונות שונות, או כאשר יש מיעוט בפרמטרים המגדירים את המודל. במצב זה המודל אינו מצליח ללמוד את התמונות. ניתן לזהות מצב כזה כאשר ה-validation accuracy נמוך משמעותית מן ה- training accuracy או כאשר ה-validation loss גבוה משמעותית מן ה-training loss.

epoch- חלוקה ל-epochs הינה הגדרה של מספר פעמים שתהליך הלמידה יתבצע מחדש.

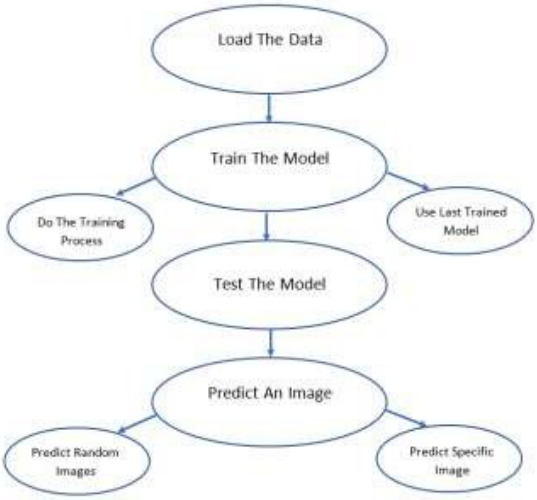


דוגמא: איך הdataset שלי של מוזיקת הג'אז נראה. (רק חלק קטן ממנו כמובן)

ספר פרויקט למידת מכונה

בפרויקט יש לי רק את המחלקה app שאחראית על התקשורת עם המשתמש ועל הgui.

UML Actors diagram



רק עם song ולא עם image, פשוט לא מצאתי תמונה שממחישה את זה טוב יותר

תיאור גרפי של הפרויקט:

ספר פרויקט למידת מכונה

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 214, 32)      320

 max_pooling2d (MaxPooling2D  (None, 63, 107, 32)      0
 )

 batch_normalization (BatchN  (None, 63, 107, 32)      128
 ormalization)

 conv2d_1 (Conv2D)           (None, 61, 105, 64)       18496

 max_pooling2d_1 (MaxPooling  (None, 30, 52, 64)       0
 2D)

 batch_normalization_1 (Batc  (None, 30, 52, 64)       256
 hNormalization)

 conv2d_2 (Conv2D)           (None, 28, 50, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 14, 25, 64)       0
 2D)

 batch_normalization_2 (Batc  (None, 14, 25, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 12, 23, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 6, 11, 64)        0
 2D)

 batch_normalization_3 (Batc  (None, 6, 11, 64)        256
 hNormalization)

 flatten (Flatten)           (None, 4224)              0

 dense (Dense)               (None, 64)                270400

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 364,033
Trainable params: 363,585
Non-trainable params: 448
_________________________________________________________________
```

ספר פרויקט למידת מכונה

שלב איסוף הכנה וניתוח של הנתונים(collect, prepare and analyze data):

שלב איסוף הנתונים מתבצע בקובץ command.txt, בקובץ זה יש סקריפט, אשר בתוכו יש קישור ל2 פלייליסטים מאוד גדולים של ג'אז ומוזיקת טראנס ביוטיוב, תפקידו של הסקריפט הוא בעצם להוריד את הפלייליסטים הללו בפורמט m4a, שזה הכי טוב שאפשר להוציא מן היוטיוב לפי מה שקראתי.

תיאור וניתוח הנתונים הגולמיים: כל השירים שהורדתי מהיוטיוב בפורמט m4a כל השירים האלה, לא מוכנים לשימוש כי הם שירים מלאים (לפחות 150 שניות) וזה גדול מדי וסתם יעמיס על המחשב, מספיק חיתוכים של 5 שניות מהשירים.

תיאור תהליך הכנת הdataset לאימון כולל הסבר אודות שיטת נרמול הנתונים: תהליך הכנת ה dataset שלי מתחלק לכמה קטעים. קודם כל צריך לחתוך את כל השירים למקטעים של חמש שניות, מהאמצע שלהם והפיכתם מפורמט m4a לפורמט wav.
כל התהליך הזה מתרחש בקובץ preprocessing.py האחראי לסידור הdata באופן שיהיה נוח ויעיל לעבוד איתו.

קודם כל אני הופך את הקובץ לפורמט של wav שהוא פורמט שיותר נוח לנרמל ויותר קל לעבוד איתו בתחום הmachine learning כשזה נוגע למוזיקה.

```
# source: path to mp3 file.
# target: path to target wav file.
def convert_to_wav(source, target):
    try:
        sound = AudioSegment.from_file(source)
        sound.export(target, format='wav')
    except:
        print(f'error encountered when trying to convert from m4a to wav source:{source} is probably not formatted correctly...')

# from_dir: origin dir
# to_dir: target dir
def convert_folder_to_wav(from_dir, to_dir):
    if not path.isdir(from_dir):
        raise Exception
    if not path.isdir(to_dir):
        mkdir(to_dir)
    for filename in listdir(from_dir):
        f = path.join(from_dir, filename)
        # checking if it is a file
        if path.isfile(f):
            raw_filename = path.splitext(filename)[0]
            target_path = path.join(to_dir, raw_filename+'.wav')
            convert_to_wav(source=f, target=target_path)
```

לאחר מכן, אני מוצא את האמצע של השירים הללו, כי הרי בסופו של דבר מאמצע השיר המחשב ילמד הכי טוב מה שייך לאיזה לייבל וזה ימנע התחלות של רקע או סופי שירים שלא באמת משקפים את השיר שאני רוצה שהמחשב ילמד.

```python
# gets source path, and returns the middle 150 second long audio segment
def get_150sec_mid_audiosegment(source):
    duration = librosa.get_duration(filename=source)
    if duration < 150:
        print(duration)
        raise Exception('files duration is smaller than 150sec, ignoring')
    left_ms = (duration-150)*1000/2
    right_ms = 150000+left_ms
    audio = AudioSegment.from_file(source)
    return audio[left_ms:right_ms]
```

בשלב השלישי של ארגון הנתונים, אני חותך את השיר מהאמצע שלו לחיתוכים של 5 שניות. בחרתי דווקא בחיתוכים של 5 שניות מפני שקראתי באינטרנט ש5 שניות זה מספיק כדי שהמחשב ילמד כמו שצריך בלי להעמיס יותר מדי על המחשב.(המחשב שלי לא חזק במיוחד, וגם המחשבים בבית הספר שלי).

```python
# source: path to wav file (thats 150 sec long)
# targets: list of length 30 that contains all the paths of the targets
def cut_song_to_5sec(source, targets):
    if not path.isfile(source):
        raise Exception('source file doesnt exist')
    mid_audio = get_150sec_mid_audiosegment(source)

    for p,t in zip(range(150//5), targets):
        left = 5*p*1000
        right = 5*(p + 1)*1000
        part = mid_audio[left:right]
        part.export(t, format='wav')
```

השלב בארגון הדאטה, הוא שמירת הנתונים ויצירת פולדר של טראנס ופולדר של ג'אז.

השלב הסופי והאחרון הוא חלוקת הדאטה לtrain לtest וגם לvalidation לאחר שנרמלתי את השירים והפכתי אותם לmelspectograms בשביל זה השתמשתי בספרייה של sklearn. נוסף על כך, הוא מחלק את הכל רנדומלית, זאת על מנת שהמודל לא ילמד פעם בהתחלה רק על טראנס ואז רק על ג'אז וכך שלב האימון בכל epoch יהיה יותר טוב ומגוון.

```python
# cuts entire genre directory containing raw music files (in our case in m
def cut_genre_to_5sec(from_dir, to_dir):
    if not path.isdir(from_dir):
        raise Exception('from_dir doesnt exist')
    if not path.isdir(to_dir):
        mkdir(to_dir)

    for filename in listdir(from_dir):
        f = path.join(from_dir, filename)
        if path.isfile(f):
            raw_filename = path.splitext(filename)[0]
            target_paths = []
            for i in range(150//5):
                target_paths.append(f'{to_dir}/{raw_filename}_{i}.wav')
            try:
                cut_song_to_5sec(source=f, targets=target_paths)
            except Exception as e:
                print(e)
```

כך בעצם אני מארגן את הdataset שלי בצורה נוחה ויעילה על מנת שיהיה אפשרי לעבוד איתו. את נרמול הנתונים אני עושה על פי..

ספר פרויקט למידת מכונה

שלב בנייה ואימון המודל(build and train deep learning model):

לאחר איסוף וארגון הנתונים הגולמיים, השלב הבא בלמידת מכונה הוא בנייה ואימון המודל.

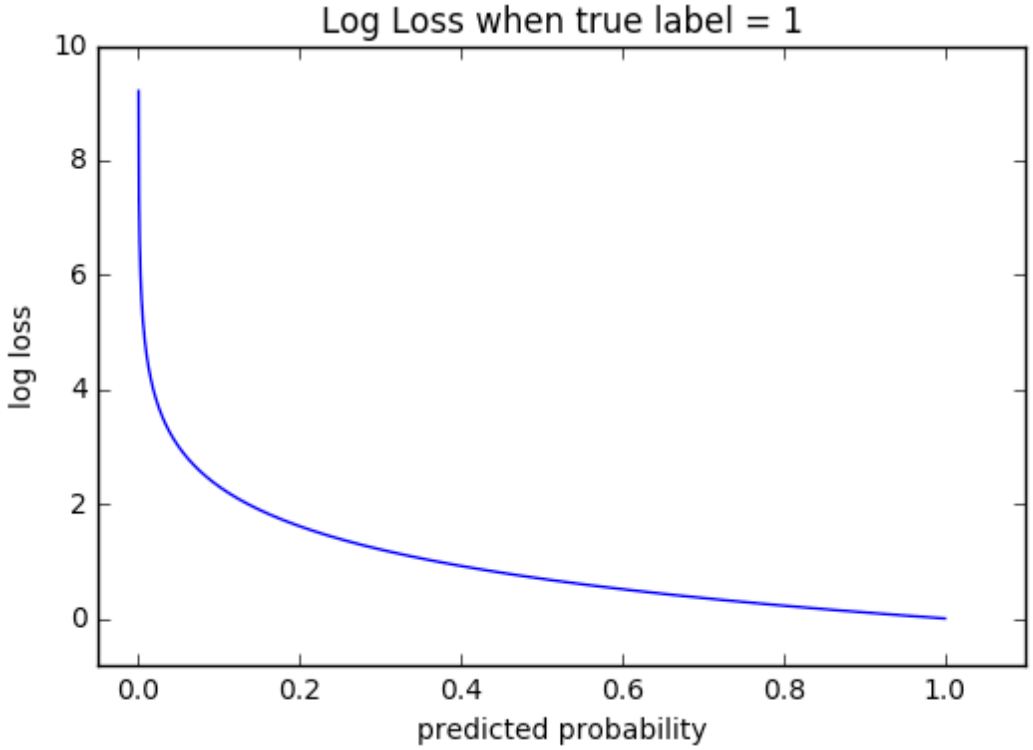
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 214, 32)      320

 max_pooling2d (MaxPooling2D  (None, 63, 107, 32)      0
 )

 batch_normalization (BatchN  (None, 63, 107, 32)      128
 ormalization)

 conv2d_1 (Conv2D)           (None, 61, 105, 64)       18496

 max_pooling2d_1 (MaxPooling  (None, 30, 52, 64)       0
 2D)

 batch_normalization_1 (Batc  (None, 30, 52, 64)       256
 hNormalization)

 conv2d_2 (Conv2D)           (None, 28, 50, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 14, 25, 64)       0
 2D)

 batch_normalization_2 (Batc  (None, 14, 25, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 12, 23, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 6, 11, 64)        0
 2D)

 batch_normalization_3 (Batc  (None, 6, 11, 64)        256
 hNormalization)

 flatten (Flatten)           (None, 4224)              0

 dense (Dense)               (None, 64)                270400

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 364,033
Trainable params: 363,585
Non-trainable params: 448
_________________________________________________________________
```

זה תיאור גרפי של המודל.

ספר פרויקט למידת מכונה

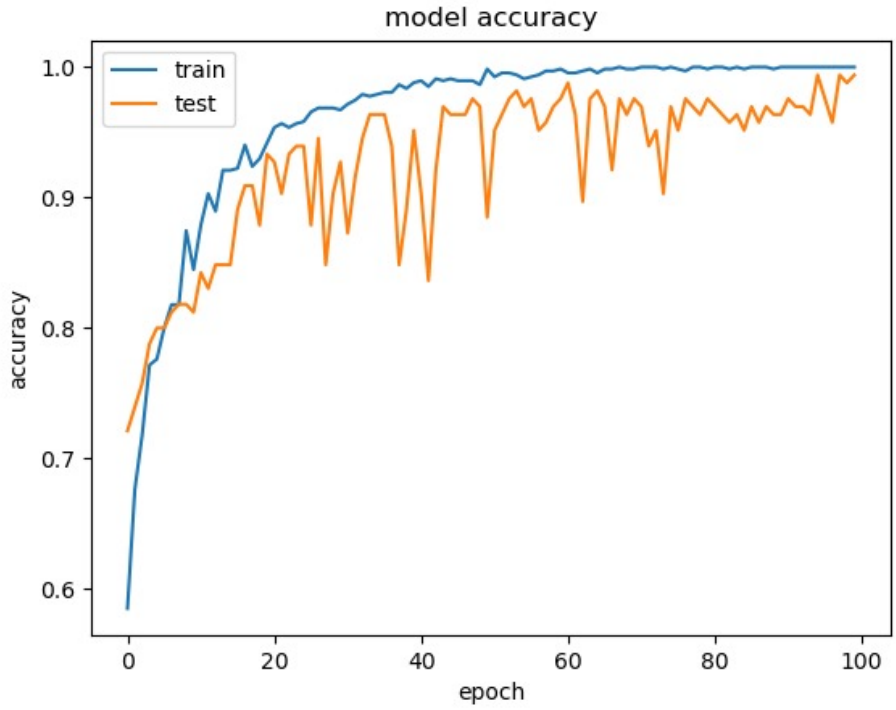
תיעוד והסבר של פונקציית השגיאה: קודם כל, פונקציית השגיאה במקרה של המודל שלי נקראת: Binary crossentropy loss function, היא בינארית מפני שלמודל שלי יש 2 אופציות, 0 או 1. ימין או שמאל, טראנס או ג'אז.

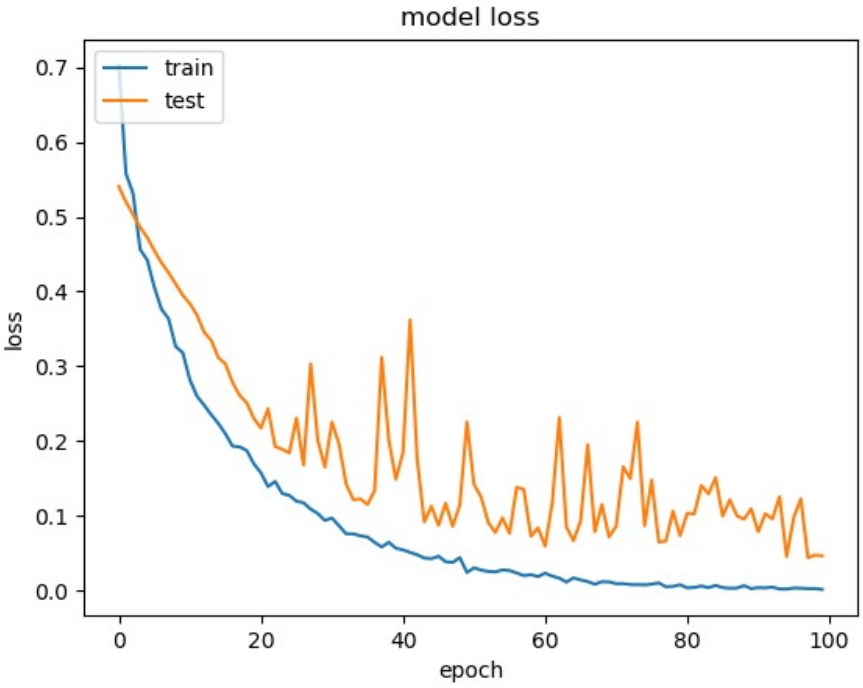
משמעות פונקציית השגיאה הזו היא להראות את השגיאה- את הloss של המודל במהלך ההרצה, על שני האופציות(גם על טראנס וגם על ג'אז) ולא על כל אופציה לחוד.



ברוב הפרויקטים בלמידת מכונה, הפונקציה אמורה להיראות ככה, בהתחלה יש לה loss יחסית גבוה, מפני שהמודל לא מאומן, אך ככל שמאמנים את המודל יותר ויותר כך הloss יורד, כפי שניתן לראות בפונקציה מעלינו.

ספר פרויקט למידת מכונה





ספר פרויקט למידת מכונה

תיעוד והסבר של ייעול ההתכנסות(optimization): הoptamization שלי נעשה באמצעות RMSprop שזה בעצם, אלגוריתם אופטמזיציה שמותאם לרשתות ניורונים.

רשתות ניורונים פירושם בעצם: רשת עצבית
מלאכותית ,(ANN – Artificial Neural Network)רשת נוירונים או רשת קשרית הוא מודל מתמטי חישובי ,שפותח בהשראת תהליכים מוחיים או קוגניטיביים המתרחשים ברשת עצבית טבעית ומשמש במסגרת למידת מכונה .רשת מסוג זה מכילה בדרך כלל מספר רב של יחידות מידע) קלט ופלט (המקושרות זו לזו, קשרים שלעיתים קרובות עוברים דרך יחידות מידע "חבויות .(Hidden Layer) "צורת הקישור בין היחידות, המכילה מידע על חוזק הקשר, מדמה את אופן חיבור הנוירונים במוח.

לכל ניורון יש משקל – weight ו –bias . רשתות ניורונים במחשב אינן כמו של גוף האדם, המחשב אינו נוצר איתם והמשקל שלהם הוא בר שינוי, לכל רשת ניורונים יש את המשקל האולטימטבי בשבילה, על מנת שהמודל ירוץ בצורה החלקה ביותר. כדי למצוא את המשקל הטוב ביותר בשביל רשת הניורונים, יש לבצע תהליך של אופטימזציה. בעצם המודל נותן לי איזשהו גרף בלי פונקציה מוחלטת(כמו f(x)=5x**2) אלא פונקציה מאוד מורכבת כאשר לכל ניורון יש את המשקל שלו ועל איזה משקל הוא פועל בצורה הכי יעילה. RMSprop זה בעצם אלגוריתם שפותח כדי להבין מהגרפים הללו מה המשקל האולטימטבי עבור כל ניורון על ידי השיפוע. כשהגרף נמצא בנקודת הקיצון המקסימום שלו, זאת אומרת שבמשקל הזה הניורון הכי יעיל, ותפקידו של האלגוריתם הזה הוא בעצם למצוא את הנקודת קיצון. פעם הוא הולך על הגרף ימינה, פעם שמאלה, מגדיל את קצבו, מקטין את קצבו, הכל בהתאם לגרף.

באמצעות כך, אני מבצע תהליך אופטמיזציה ומייעל את רשת הניורונים על מנת לבנות את הmelspectograms הטובים ביותר.

הmelspectograms זה בעצם המטריצות, נרמול השירים למספרים מהם המודל מבין לאיזה ז'אנר כל שיר משתייך.

הסבר על שיטת הנירמול:

מפני שאני מנרמל שיר, ולא תמונה, אני מנרמל על ידי ספקטגורמה, את תהליך הנירמול עצמו אני לא באמת יודע(זה נוסחא מתמטית יחד עם קוד שלקחתי מהאינטרנט כמובן), אבל כדי להבין איך זה עובד הכי קל זה פשוט להסביר מה זה ספקטוגרמה.

ספקטרוגרמה הוא ייצוג חזותי של הספקטרום של התדרים של האות כפי שהוא משתנה עם הזמן. כאשר מיושמים על אות אודיו , ספקטרוגרמות נקראות לפעמים סונוגרפיות ,טביעות קוליות או קולגרמות .

כאשר הנתונים מיוצגים בתרשים תלת מימד, הם עשויים להיקרא תצוגות מפל מים.

ספקטרוגרמות נמצאות בשימוש נרחב
בתחומי המוזיקה , בלשנות , סונאר , מכ"ם , עיבוד דיבור , סייסמולוגיה ועוד .ניתן להשתמש בספקטרוגרמות של שמע כדי לזהות מילים מדוברות באופן פונטי , ולנתח קריאות שונות של בעלי חיים.

ספר פרויקט למידת מכונה

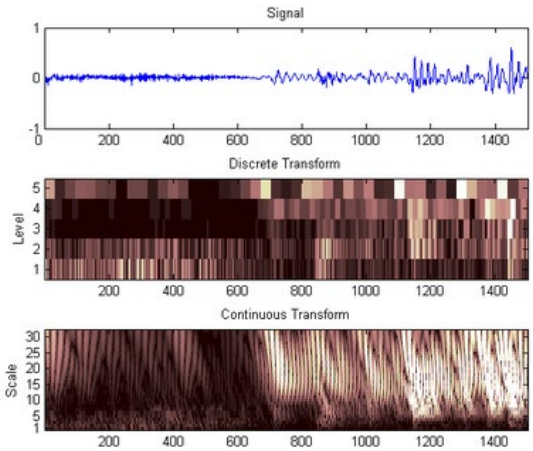
ספקטרוגרמה יכולה להיווצר על ידי ספקטרומטר אופטי , בנק של מסנני פס-פס , על ידי טרנספורמציה של פורייה או על ידי טרנספורמציה של גל) במקרה זה היא ידועה גם כסקאלוגרמה או סקאלוגרמה .

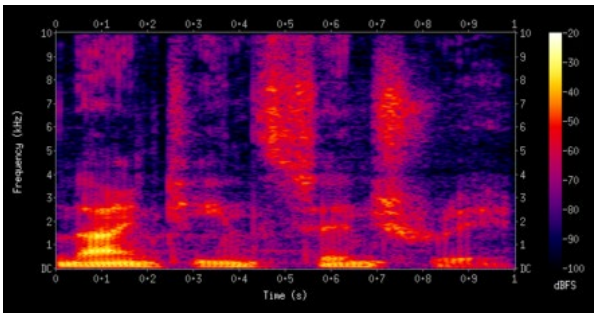
ספקטרוגרמה מתוארת בדרך כלל כמפת חום , כלומר, כתמונה עם העוצמה המוצגת על ידי שינוי הצבע או הבהירות.

פורמט נפוץ הוא גרף בעל שני ממדים גיאומטריים: ציר אחד מייצג זמן , והציר השני מייצג תדר ; מימד שלישי המציין את המשרעת של תדר מסוים בזמן מסוים מיוצג על ידי העוצמה או הצבע של כל נקודה בתמונה.

ישנן וריאציות רבות של פורמט: לפעמים הציר האנכי והאופקי מוחלפים, כך שהזמן רץ למעלה ולמטה; לפעמים כחלקת מפל שבו המשרעת מיוצגת על ידי גובה של משטח תלת מימדי במקום צבע או עוצמה. צירי תדירות והמשרעת יכולים להיות ליניאריים או לוגריתמיים , בהתאם למה שהגרף משמש. אודיו יוצג בדרך כלל עם ציר משרעת לוגריתמי (כנראה בדציבלים , או ,dB והתדר יהיה ליניארי כדי להדגיש יחסים הרמוניים, או לוגריתמיים כדי להדגיש קשרים מוזיקליים, טונאליים.

תמונות להמחשה:





ספר פרויקט למידת מכונה

שלב היישום(software deployment):

תיאור והסבר כיצד היישום משתמש במודל: היישום בעצם משתמש במודל בכך שהוא גם עושה לו train גם עושה לו evaluate וגם עושה לו prediction, כמובן שכל דבר קורה בfile שונה וארחיב יותר על כל file במהשך עם הסבר על הפעולות גם.

תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש: ממשק המשתמש מומש על ידי הטכנולוגיה tkinter.

Tkinterהוא ממשק של שפת התכנות פייתון לערכת התצוגה Tk(ספרית קוד חוצה פלטפורמות של כלים גרפיים). כלול בספריה הסטנדרטית של פייתון בהתקנות של פייתון במערכות ההפעלה Microsoft Windows, Linux, ו-Mac OS X.

ממשק כזה יאפשר גם לאנשים שלא מרגישים בנוח עם שורת הפקודה להריץ את הסקריפט. פייתון מגיעה עם ספריה גרפית מאוד פשוטה שנקראת Tkinter. הספריה מאפשרת לבנות ממשק משתמש גרפי בזמן עבודה מאוד קצר ובאופן שכל מכונה שמותקן עליה פייתון יכולה להציג (בלי קשר למערכת ההפעלה) .
הממשק עצמו לא הכי יפה בעולם, אך זמן הפיתוח הקצר הופך את Tk לאופציה אטרקטיבית כשבאים לפתח ממשק GUI לסקריפטים קטנים.

```
class app:
    def __init__(self, master):
        self.master = master
        self.master.geometry("325x150")
        self.main()

        self.trained_model = False

    def destroy_prev(self):
        for i in self.master.winfo_children():
            i.destroy()

    def main(self):
        self.destroy_prev()
        frame = ttk.Frame(self.master, width=300, height=300)
        frame.pack(pady=20)

        ttk.Label(frame, text='Liav\'s Project').pack(pady=5)

        ttk.Label(frame, text=SKIP_PREDICT).pack(side='top',pady=5)

        self.goto_btn = ttk.Button(frame, text="Go to Preprocessing", command=self.preprocess_window).pack(side='left',pady=5, padx=5)

        ttk.Button(frame, text='Skip To Predict', command=self.predict_window).pack(side='left',pady=5, padx=5)

    def preprocess_window(self):
        self.destroy_prev()
        frame = ttk.Frame(self.master, width=300, height=300)
        frame.pack(pady=20)

        self.master.title('Preprocessing Data')
        ttk.Label(frame, text=SKIP_PREDICT).pack(pady=5)

        ttk.Button(frame, text='Start', command=lambda:preprocess_data(self)).pack(side='left', pady=5, padx=5)

        self.goto_btn = ttk.Button(frame, text="Go to Train", command=self.train_window, state='disabled')
```

ניתן לראות שבאמת באפליקציה יש הרבה מאוד ttk שזה קיצור לtkinter, שזו בקצרה ספרייה שמכילה המון פקודות שנוחות למשתמש וגם למפתח ליצור.

על מנת לייפות קצת את האפליקציה שתהיה יותר יפה לעין של המשתמש, השתמשתי בחבילה נוספת הנקראת tkinter themes המכילה עוד קצת פקודות כדי לייפות את האפליקציה בזמן השימוש.

ספר פרויקט למידת מכונה

תיאור קוד הקולט את ה DATA -שעליו יבוצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי:

הקוד בעצם מקבל איזשהו קובץ של שיר, בכל פורמט שהוא(..m4a, mp3), והופך אותו קודם כל לפורמט wav שאיתו הכי נוח לעבוד, לאחר מכן הוא מחפש את אמצע השיר וממנו הוא חותך ל6 קיטועים של 5 שניות כל אחד. על כל קטע חיתוך כזה של 5 שניות הפרויקט עושה prediction ואז לפי מה שהרוב אומר כך המחשב קובע אם זה טראנס או ג'אז. כלומר, אם 5 מתוך ה6 אמרו שזה ג'אז אז זה ג'אז, אך אם שתיים מתוך השש אמרו שזה ג'אז, כלומר שארבע אחרים אמרו שזה טראנס(כי זה classification, 0 או 1) לכן, הוא יחזה שהשיר הזה הוא שיר טראנס ולא ג'אז.

ספר פרויקט למידת מכונה

מדריך למפתח:

הפרויקט מכיל מספר קבצים ותיקיות:

תכולת הפרויקט:

- תיקיית python המכיל את כל קבצי הpython של הפרויקט:

Main – קובץ המכיל את הקוד האחראי על ניהול הפרויקט ותקשורת עם המשתמש.

מיקומו: liav-gui→main.py

Command – קובץ האחראי על הורדת הדאטה מקישור ביוטיוב.

מיקומו: liav-gui→model_scripts→command.txt

Preprocessing- קובץ האחראי על סידור הדאטה ושמירתו במקום חדש.

מיקומו: liav-gui→model_scripts→preprocessing.py

data- קובץ האחראי חיתוך הדאטה, הפיכתו למטריצה שיהיה אפשרי לעבוד עליה, וחילוקה ל train וtest.
מיקומו: liav-gui→model_scripts→data.py

Training- קובץ האחראי על אימון המודל על הדאטה שנתנו לו, שיוך כל שיר ללייבל המשוייך לו.
מיקומו: liav-gui→model_scripts→training.py

Get_prediction – קובץ האחראי על שיוך שיר חיצוני לז'אנר אליו הוא שייך (טראנס או ג'אז).

מיקומו: liav-gui→get_prediction

Eval- אחראי על הevulate של המודל ועל תהליך האופטמיזציה בו.
מיקומו: liav-gui→model_scripts→eval.py

Model – אחראי על הרצת המודל עצמו, על כל רשתות הניורונים והליירס.
מיקומו: liav-gui→model_scripts→midel.py

Environment.yml- קובץ שמורידים לפני ההרצה על מנת ליצור סביבת עבודה מותאמת להרצת הפרויקט בכל המחשבים.
מיקומו: liav-gui→ environment.yml

ספר פרויקט למידת מכונה

הספריות בפרויקט שלי אותן צריך להוריד כדי שיהיה ניתן להריץ את הפרויקט (סביבת העבודה):

Installation command	Library name
Pip install tensorflow	Tensorflow
Pip install keras	Keras
Pip install numpy	Numpy
Pip install os	os
pip install librosa	librosa
Pip install tk	tkinter
Pip install ttkthemes	ttkthemes
Pip install pydub	pydub
Pip install tk	Tkinter (8.6.10)
Pip install audio segment	pydub
Pip install pickle-mixin	Pickle

בפועל, את כל הספריות למיניהם מורידים דרך anaconda prompt באמצעות הקובץ שיצרתי – environment.yml שזה קובץ המכיל את כל סביבת העבודה הנדרשת. ההורדה מתבצעת כך:

1. חיפוש בanaconda prompt – explorer
2. Cd full path (למיקום אליו שמור הפרויקט)
3. Conda env create –f environment.yml
4. Conda activate liav

ספר פרויקט למידת מכונה

הסבר עבור הפונקציות בפרויקט:

בפועל, על כל פונקציה רשמתי מה היא מקבלת ומה היא מחזירה מעליה, לכן לדעתי הכי טוב יהיה רצף תמונות של כל פונקציה עם הכיתוב מעליה של מה היא מטרתה.

בנוסף, רצף התמונות יהיה בסדר של סדר הפעולות של הפרויקט ומחולק לתיקיות הפרויקט:

Preprocessing:

```
# source: path to mp3 file.
# target: path to target wav file.
def convert_to_wav(source, target):
```

```
# from_dir: origin dir
# to_dir: target dir
def convert_folder_to_wav(from_dir, to_dir):
```

```
# gets source path, and returns the middle 150 second long audio segment
def get_150sec_mid_audiosegment(source):
```

```
# source: path to wav file (thats 150 sec long)
# targets: list of length 30 that contains all the paths of the targets
def cut_song_to_5sec(source, targets):
```

```
# cuts entire genre directory containing raw music files (in our case in m4a format)
#  to 5second long cuts and saves them into a new directory
def cut_genre_to_5sec(from_dir, to_dir):
```

Data:

```
# creates melspectrogram from a song path
def get_melspect(file_path):
```

```
# counts the total tracks, genres labeld, used purely for logging
count = 1
# get a genre directory path containing 5 second long clips of that genre, gets the label of the genre
# (for example, jazz's label is in our case 0), limit which is the limit of entries from that directory.
# returns tracks, genres arrays. while tracks contain all the spectrograms and genres contain all the labels for them.
def get_tracks_genres_for_dir(dir, label, limit=100):
```

```
# gets label dictionary and limit (which limits the amount of enteries).
# returns all_tracks and genres arrays, while all_tracks contain all the spectrograms and genres contain all the labels for them.
def get_tracks_genres(limit=200, lable_dict={'jazz' : 0, 'trance' : 1}):
```

```
# gets num_instances which is the number of instances from the dataset that we want to train/test/validata on,
# gets pickle_save_path which is the save path of all_tracks and genres so we wont need to calculate all the spectrograms
# everytime we want to make changes to the model.
def get_train_val_test_arrays(num_instances=500, pickle_save_path = 'model_scripts/saves/pickles/all_tracks_genres.pickle',
```

ספר פרויקט למידת מכונה

Training:

```
# model_path is the model save path when we finish the training process
MODEL_PATH = 'model_scripts/saves/model/json/model1.json'
WEIGHTS_PATH = 'model_scripts/saves/model/h5/model1.h5'

def train_model_and_save_to_path(data_path, model_save_location='model_scripts/saves/model/json/model2.json', weights_save_location
```

```
# model is the model created in create_model function in model.py
# data is a tuple containing all the dataset generators, created in the get_train_test_generators function in data_generators.py
# epochs is the number of epochs in the training process, defaulted to 75.
# returns history, which contains the history of the training process (i.e. loss and acc for each epoch etc.)
def train_model(model : Sequential, data = None, epochs=75):
```

```
# gets the model which has been trained.
# saves model and weights after the training process to the paths specified in the beginning of the file.
def save_model_and_weights(model, model_save_location, weights_save_location):
```

Get_prediction:

```
# gets model_path and weights_path in order to load the model from them.
# returns the loaded model.
def _load_model(model_path='models/model.json', weights_path='models/weights.h5'):
```

```
# gets the prediction of the genre from an audio file.
def get_prediction(audio_file_path, model_save_location=MODEL_PATH,
```

```
# jazz = 0, trance = 1
labels = ['jazz', 'trance']
# gets model and wav_paths (which are the paths of the 5 second cuts (usually 6 of them are created although it is changeable))
# returns the most likely genre for all the cuts paths it gets (bt calculating the average of the predictions)
def _predict(model, wav_paths):
```

```
# gets the path of 5 sec long wav file
# returns the melspectrogram of a wav file
def _get_melspect(song_path):
```

```
# gets model and path of a 5 sec wav file and uses the model to generate a prediction for the file
# returns prediction of a single wav file's genre
def _predict_single(model, wav_path):
```

```
# source: path to wav file (thats 30 sec long)
# targets: list of length 5 that contains all the paths of the targets
def _cut_song_to_5sec(source, targets):
```

```
# gets source path, and returns the middle 150 second long audio segment
def _get_30sec_mid_audiosegment(source):
```

Model:

```
# creates the model and returns it
def create_model():
```

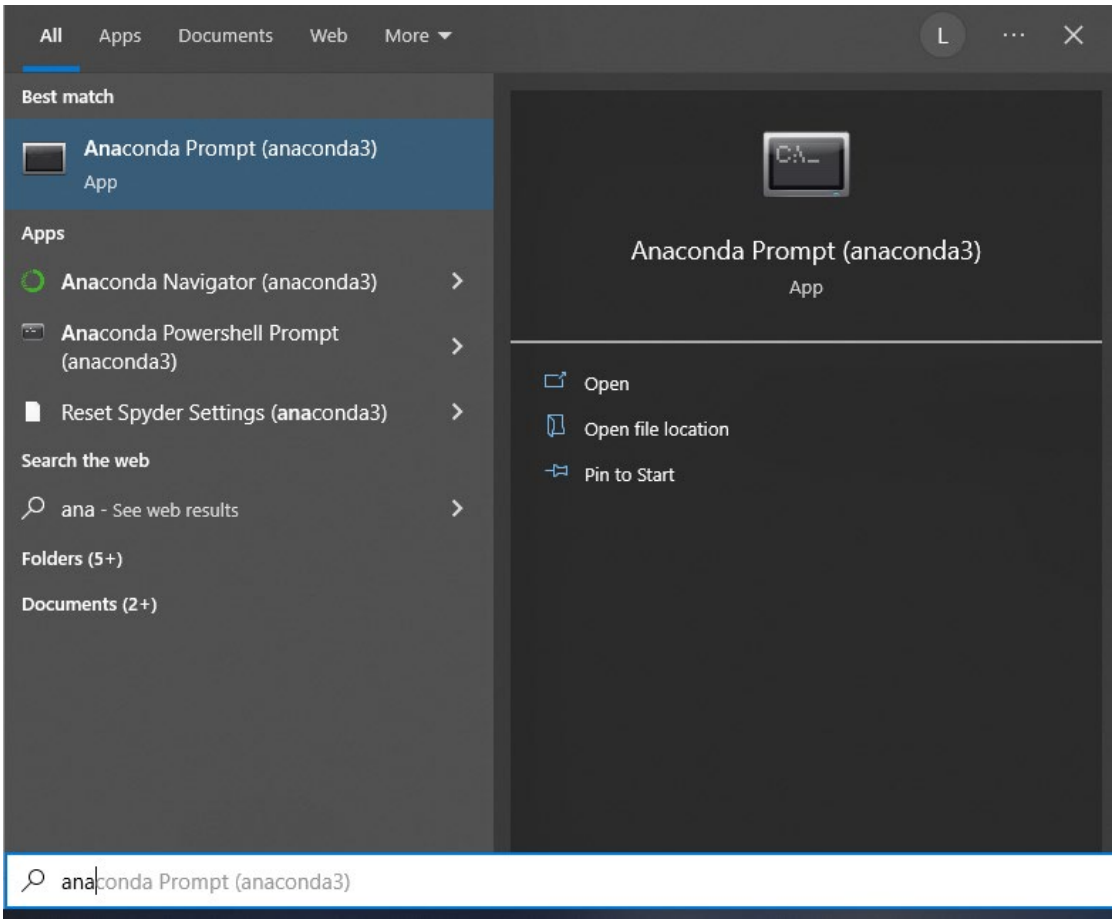
ספר פרויקט למידת מכונה

מדריך למשתמש:

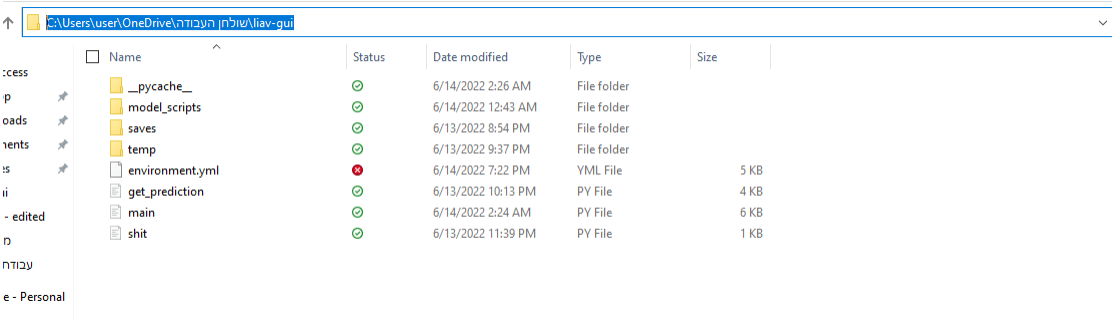
על מנת להשתמש במודל זה יש להשתמש בanaconda prompt לאחר העלאת סביבת העבודה המתאימה.

אצרף תמונה המתאימה לכל שלב בנושא.

קודם כל מחפשים בanaconda prompt explorer.

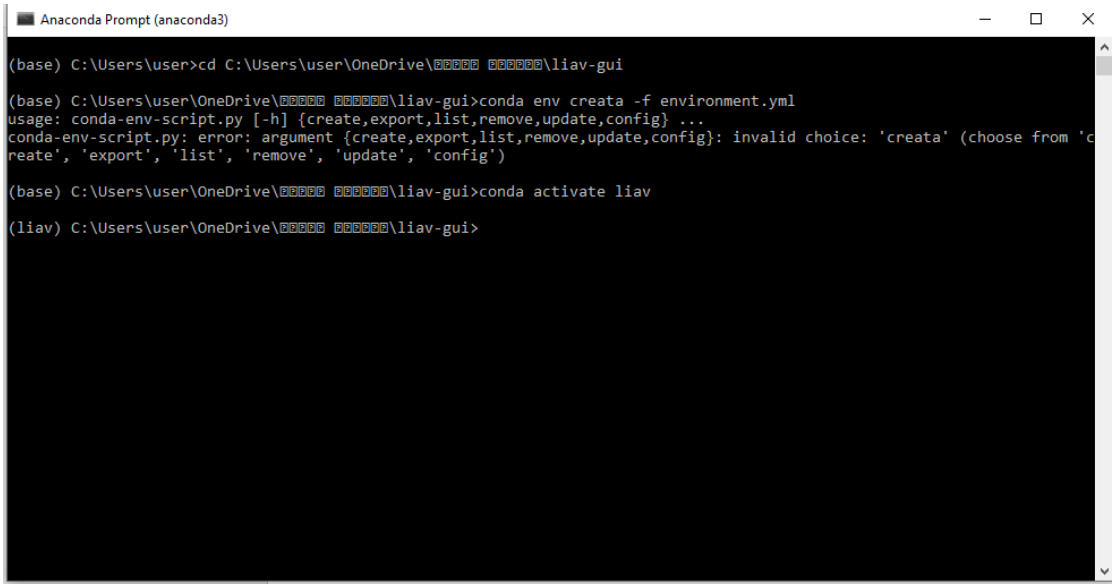


לאחר מכן מחפשים בתיקיית קבצים את התיקייה של הפרויקט כדי להגיע לfull path של הפרויקט:



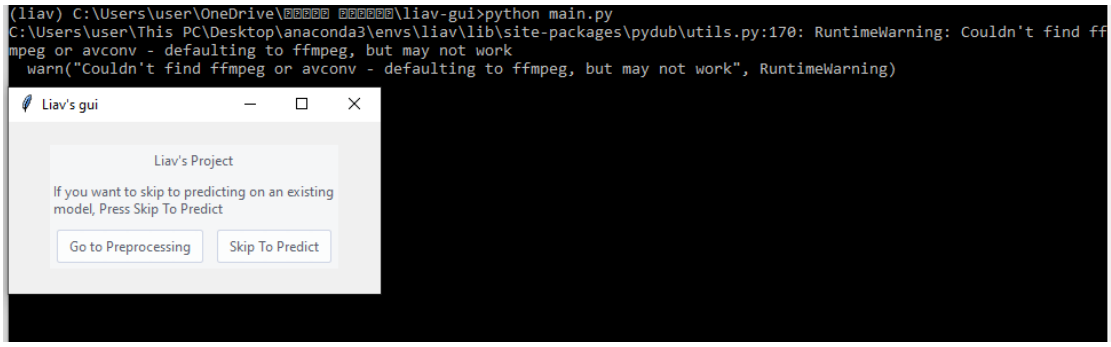
השלב השלישי הוא מספר שורות הרצה על מנת להתאים את סביבת העבודה במחשב ושהכל ירוץ חלק(בanaconda prompt):

ספר פרויקט למידת מכונה



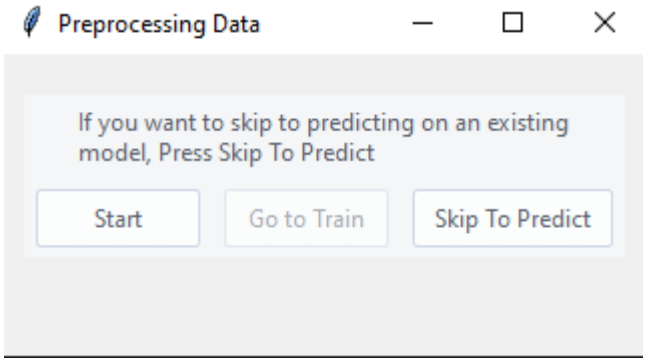
כאשר ההעברה מ(base) ל(liav) מראה על כך שעכשיו סביבת העבודה שלנו מותאמת למודל והכל ירוץ חלק.

לאחר מכן כותבים python main.py (מפני שקובץ הפייתון הזה נמצא באותה תיקייה יחד עם תיקיית סביבת העבודה):



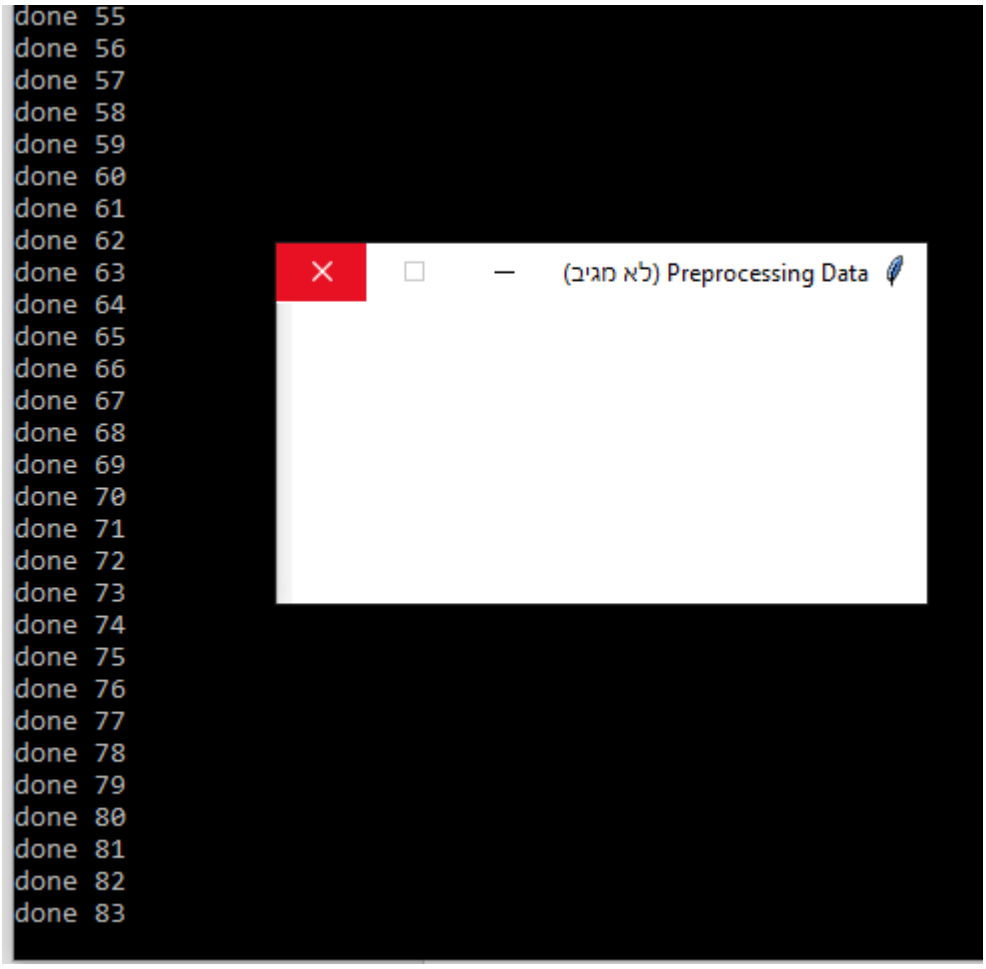
מיד לאחר פקודה זו נפתחת לנו האפליקציה.

נלחץ על go to processing על מנת לראות את כל המודל עובד ולא ישר לקפוץ לסוף.



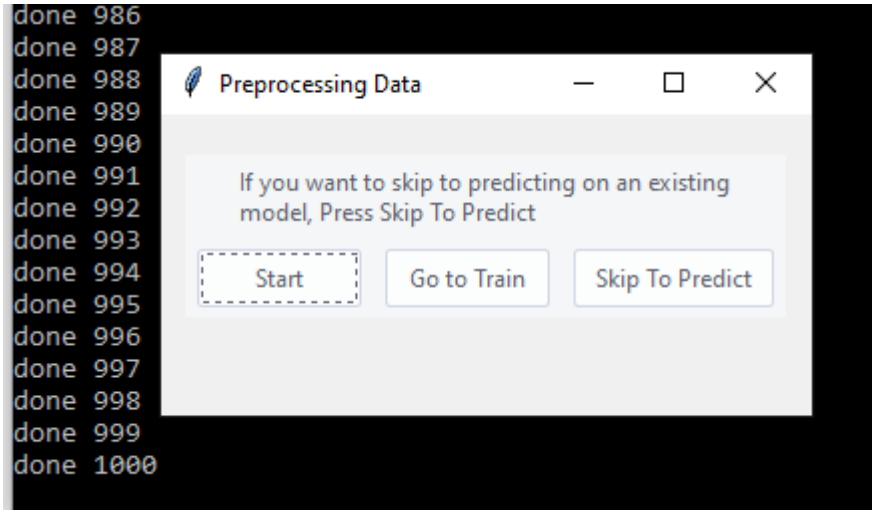
ספר פרויקט למידת מכונה

נלחץ על start:



תחילה הוא לא יגיב עד שהוא יסיים להוריד את כל הdata הנדרשת, מפני שאין לו איך ועל מה לעבוד בזמן שהקוד רץ ומוריד שירים וחותך אותם כפי שהוסבר.

כעת לאחר שירדו 1000 שירים נלחץ go to training כדי לאמן את המודל:

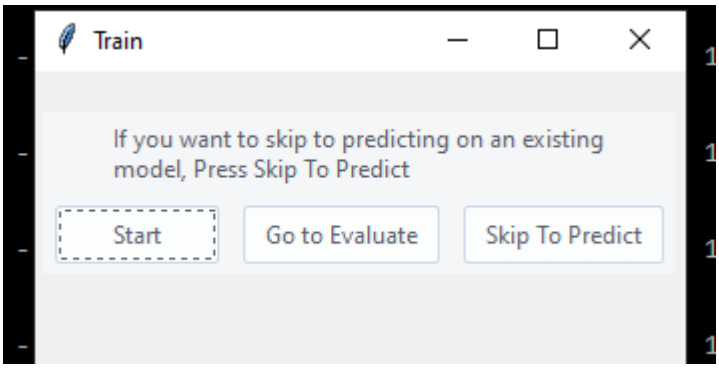


ספר פרויקט למידת מכונה

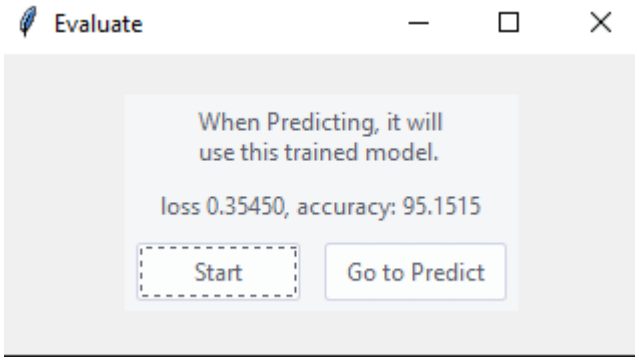
לאחר מכן יתחיל תהליך האימון, והעברה על 100 epcohes תוך כדי העברת מידע על הloss ועל הaccuracy.

```
Epoch 9/100
21/21 [==============================] - 8s 377ms/step - loss: 0.3530 - accuracy: 0.8328 - val_loss: 0.4496 - val_accura
cy: 0.8182
Epoch 10/100
21/21 [==============================] - 8s 396ms/step - loss: 0.3217 - accuracy: 0.8701 - val_loss: 0.4292 - val_accura
cy: 0.8303
Epoch 11/100
21/21 [==============================] - 8s 401ms/step - loss: 0.3068 - accuracy: 0.8761 - val_loss: 0.4099 - val_accura
cy: 0.8606
Epoch 12/100
21/21 [==============================] - 8s 362ms/step - loss: 0.2926 - accuracy: 0.8672 - val_loss: 0.3843 - val_accura
cy: 0.8667
Epoch 13/100
21/21 [==============================] - 7s 358ms/step - loss: 0.2815 - accuracy: 0.8761 - val_loss: 0.3692 - val_accura
cy: 0.8606
Epoch 14/100
21/21 [==============================] - 8s 373ms/step - loss: 0.2870 - accuracy: 0.8642 - val_loss: 0.3527 - val_accura
cy: 0.8788
Epoch 15/100
21/21 [==============================] - 8s 385ms/step - loss: 0.2517 - accuracy: 0.9090 - val_loss: 0.3342 - val_accura
cy: 0.8848
Epoch 16/100
21/21 [==============================] - 9s 418ms/step - loss: 0.2377 - accuracy: 0.9119 - val_loss: 0.3213 - val_accura
cy: 0.8909
```

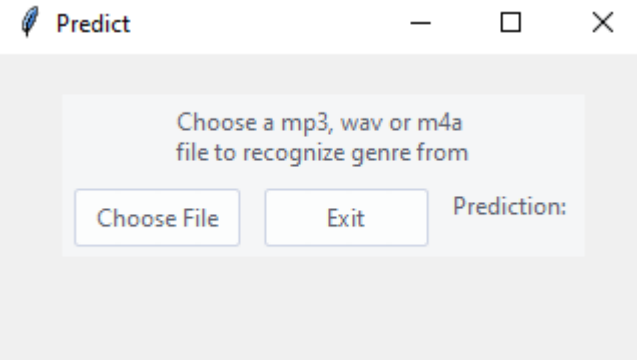
לאחר מכן יפתח לנו חלון נוסף, נלחץ על go to evaluate.



עכשיו זה יראה לנו את המודל:



נלחץ על go to predict לאחר שהמודל מאומן וראינו איך הוא עבד.



ובשלב הסופי והאחרון, נלחץ על choose file, נבחר קובץ של שיר של מעל 150 שניות ששמרנו מלפני(שלא נמצא בדאטה עליו המודל התאמן) והאפליקציה תציג לנו האם זה שיר טראנס או האם זה שיר ג'אז.

ספר פרויקט למידת מכונה

רפלקציה:

במהלך עשיית פרויקט הגמר שלי בהתמחות למידת מכונה, למדתי דברים חדשים והרחבתי את הידע שלי רבות בשפת python בפרט במחשבים ובלמידת מכונה בכלל.

לאורך כל תהליך העשייה שלי היה בי דחף ללמוד ולהגיע לתוצאה הטובה ביותר, לכן המשכתי לשפר את המודל ולנסות להגיע לתוצאות הטובות ביותר בשבילי.

עם זאת, תהליך העבודה על הפרויקט לא היה פשוט. כפי שהסברתי במבוא, התחום הזה היה חדש לי ולא הכרתי אותו לפני תחילת העבודה. מאחורי התחום עומד רקע תיאורטי עצום וכן ידע מעשי שצריך לרכוש כדי ליצור פרויקט. כמו כן, כל חבריי להתמחות נחשפו לתחום החדש, לכן כל הנושא טרי אצל כולנו והיה לנו הרבה ללמוד לקראת הכנת הפרויקט. אמנם הצלחתי להתגבר על אתגרים אלו, בכך שחקרתי ולמדתי בעיקר באופן עצמאי על כל תחום למידת המכונה ו-deep learning, בעיקר ממקורות באינטרנט, ממקורות שהמורה שלחה ומסרטונים ביוטיוב. דבקתי במטרה ולמדתי רבות גם בכוחות עצמי, ואני מרוצה מכך.

מעבר לכך, אני מרגיש כי העבודה על הפרויקט תרמה לי גם לשיפור אוצר המילים שלי באנגלית, שכן הרוב המוחלט של מקורות המידע בנושא זה הינם באנגלית בשפה גבוהה. העובדה שהיה עליי ללמוד את הנושא על מנת להכין פרויקט בתחום חייב אותי ללמוד מאתרים אלו וכך לשפר את האנגלית שלי.

בעקבות חקר העבודה וכתיבתה, אני לוקח איתי את המידע הרב שצברתי במהלך העבודה כולה, אשר נמשכה לאורך תקופה ארוכה. תחום הבינה המלאכותית ולמידת המכונה הינו תחום מרתק ומעניין מאוד,חדשני ואני מאמין שיש עוד המון חידושים שהוא יכול לתרום בהם. על כן, אני שמח שהייתה לי הזכות להיחשף אליו ולהכין פרויקט גמר בנושא, ויותר מזה אני בטוח שאעבוד על פרויקט נוסף בקיץ, יותר מתוחכם ויותר יפה שיתווסף לתיק העבודות שלי.

כמו כן, משום שכל חבריי להתמחות נחשפו לתחום הזה לראשונה, למדנו רבות יחד ואף העברנו הרצאות במהלך השיעורים עם דינה מורתנו. אני מרגיש כי הדבר ליכד אותנו וגרם לכך שהייתה לנו מטרה משותפת.
נוצרה עזרה ההדדית בין חברי הכיתה והיה לי כיף להיות נוכחת בכך.

לסיכום, מאחר ביצעתי את הפרויקט ולמדתי תוך כדי הכנתו, התנסיתי בגרסאות שונות כשבכל גרסה שיפרתי משהו לקראת הגרסה הסופית, אני חושב שהצלחתי בסופו של דבר להגיע למטרה שלי ואני גאה בתוצר הסופי שלי.

ספר פרויקט למידת מכונה

ביבליוגרפיה:

https://stackoverflow.com/questions/40651891/get-length-of-wav-file-in-samples-with-python

https://www.programiz.com/python-programming/methods/built-in/zip

https://stackoverflow.com/questions/69079608/how-to-make-multiple-pages-in-tkinter-gui-app

https://write.corbpie.com/downloading-youtube-videos-and-playlists-with-yt-dlp/

https://deepchecks.com/how-to-test-machine-learning-models/

https://elitedatascience.com/overfitting-in-machine-learning