AGC – Audio Genre Classification

פרויקט גמר
"ניתוח נתונים ולמידת מכונה"

שנת הלימוד: 2021
ביתה: י
מגיש: עידו שרון

קישור פרויקט:

https://colab.research.google.com/drive/1ix0kdv3DY3b0EQipRB9XI78PKL2qc8t6?usp=sharing

תודה

ברצוני להודות לנועם פרנקו, חבר מכיתה י״א שלמד בעצמו למידת מכונה ועזר לי בפרויקט כאשר נתקלתי בקשיים, ולימד אותי חומר מתקדם שבלעדיו המודל לא היה מצליח להגיע לאחוזי דיוק כאלו גבוהים.

תוכן עניינים

מבוא

מטרת הפרויקט

הפרויקט מתמקד בקבצי סאונד וכיצד ניתן להבדיל ביניהם לז׳אנרים שונים. כידוע במוזיקה ישנם ז׳אנרים שונים של מוזיקה (ג׳אז, פופ, רוק וכו' ...). בפרויקט זה ברצוני להיות מסוגל, בעזרת למידת מכונה, להבדיל בין 5 ז׳אנרים שונים (קלאסי, פופ, ג׳אז, קנטרי, רוק) על ידי שימוש בקטעי מוזיקה באורך 10 שניות. כלומר נוכל להבדיל בין קטע קול של מוזיקת רוק לזה של ג׳אז למוזיקה קלאסית וכו׳.

הרקע לפרויקט

בחיי היום יום שלי יוצא לי הרבה מאד פעמים לשמוע שיר שאני לא מכיר ברדיו ולהשתמש באפליקציית ״שאזאם״ שמסוגלת לזהות בתוך שניות ספורות על איזה שיר מדובר רק בעזרת קטע סאונד קצר שלו. תמיד התעניינתי כיצד המערכת הזו עובדת, ולכן החלטתי לאתגר את עצמי בפרויקט לנסות לייצר מערכת דומה, אמנם פשוטה יותר מכיוון שאין באמצעותי את היכולות וכמות כזו גדולה של מידע לשירים אקטואליים כמו שאזאם. ולכן הסתפקתי בזיהוי רק הז׳אנר של השיר ללא שם השיר עצמו.

תיאור תכולת הפרויקט

הפרויקט דרש ממני ללמוד הרבה ולחקור באינטרנט, מחיפוש אחר מאגר נתונים טוב ועד קריאה על גלי סאונד והמאפיינים שלהם. בנוסף היה צריך לקרוא על השימוש בספריות השונות כמו librosa.
בסך הכל הפרויקט מחולק לשלושה חלקים מרכזיים: הראשון הוא המרת כל קטעי הקול לתמונות (ראה הסבר בהמשך) וטעינתם לתוך טבלת pandas אחת, לאחר מכן נרמול כל הנתונים והמרת התגיות למספרים. החלק השלישי והאחרון הוא בניית המודלים ללמידת מכונה. המודלים שבחרתי לאמן הם mlp עם TensorFlow ו- knn עם sklearn.

סביבת עבודה

את הפרויקט נכתוב בשפת פייתון, מכיוון שהשפה מאד אינטואיטיבית ומצוידת בספריות רבות בין היתר גם עבור למידת מכונה. את הקוד החלטתי לכתוב בסביבת העבודה google colab שמאפשרת לכתוב מחברת פיתון, היתרון הראשי במחברת על פני תוכנת קוד אחת היא שניתן לחלק את הקוד בה לבלוקים שונים שניתן להריץ בנפרד וכך אין צורך להריץ בכל פעם את הקוד מחדש לאחר כל שינוי.
בפרויקט נעשה שימוש בספריות רבות: ספריית הפייתון librosa, מאפשרת קריאה, חיתוך וניתוח של קבצי סאונד. הספרייה pandas מאפשרת לארגן את הנתונים בטבלאות וביצוע פעולות שונות עליהן. הספרייה NumPy מאפשרת ביצוע פעולות מתמטיות רבות ביחד עם מערכים יותר חכמים. הספריות sklearn, TensorFlow ללמידת מכונה, חילוק הנתונים לאימון ולמידה, נרמול הנתונים וכו׳. ועוד ספריות רבות (תיאור על כל הספריות נמצא במחברת הקוד).

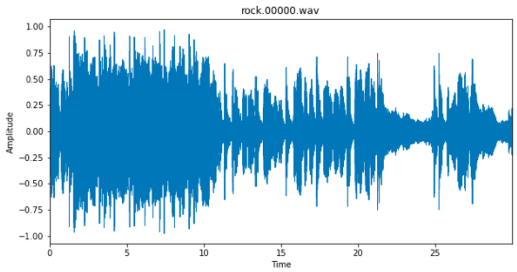
תכנון והכנת הנתונים

חיפוש ובחירה

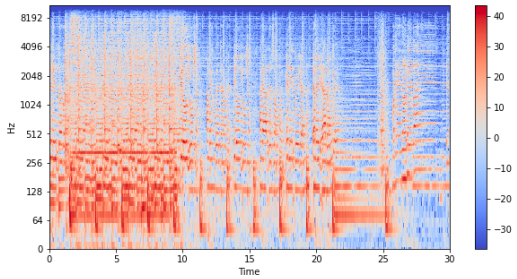
על מנת למצוא נתונים עבור אימון המודל ידעתי שעלי למצוא מאגר נתונים שיכיל כמות מספקת של הקלטות מכל ז׳אנר שונה. לאחר חיפוש באתר kaggle נתקלתי במאגר בשם - "GTZAN Dataset - Music Genre Classification", המכיל כאלף קטעי קול שונים באורך של לא פחות מ 30 שניות, בשביל תאימות בנתונים ניקח רק את ה 30 שניות של הקובץ, זה אינו הבדל משמעותי מכיוון שמדובר בדרך כלל בשבריר שנייה יותר (מעתה והלאה, לא אציין זאת אבל יש להתייחס לקבצים באורך 30 שניות בדיוק).
החלטתי לא לקחת את כל הז׳אנרים ולהסתפק רק בז׳אנרים: קלאסי, היפ-הופ, ג׳אז, פופ, רוק. זאת מכיוון שכמות גדולה של כל עשרת הז׳אנרים, כלומר 1000 קטעי קול באורך 30 שניות כל אחד גרמה לקריסה של ה RAM בקלואב, בנוסף כמות גדולה של הז׳אנרים מקשה מאד על למידת המכונה בשל ז׳אנרים דומים (כמו רוק ומטאל).

טיוב הנתונים

לפני שאוכל להתחיל לפרט על טיוב הנתונים עלי לתת הקדמה מסוימת לרעיון מאחוריו, הפרויקט אמנם נועד עבור קבצי סאונד אבל נוכל לתרגם את הפרויקט לזיהוי של תמונות ולאחר מכן חזרה לסאונד, כיצד? ובכן לכל קטע סאונד ישנו גרף (או תמונה) בשם "ספקטרוגרמה" מתאים,
ספקטרוגרמה הוא ייצוג חזותי של אופן שינוי התדרים של קובץ הסאונד לאורך זמן, כלומר כגרף אשר ציר ה- x תואם את הזמן, ציר ה- y מייצג את התדר באותה נקודת זמן ונוסיף ציר שלישי שמיוצג על פי עוצמת צבע הנקודה ומייצג את האמפליטודה[1] של התדר באותו רגע זמן. כך תיראה לדוגמא ספקטרוגרמה של קטע הסאונד של רוק הנ"ל:



גרף שינוי האמפליטודה ("הרגיל")



גרף הספקטרוגרמה המתאים

אין צורך להעמיק לתוך הנושא[2], אלא פשוט לזכור כי לכל קובץ סאונד תתאים תמונה אחת ויחידה של ספקטרוגרמה, כלומר אם ניתן להבדיל בין תמונות הספקטרוגרמה ניתן להבדיל גם בין קבצי הסאונד. מעתה והלאה נתייחס בפשטות לכל קובץ סאונד כתמונת הספקטרוגרמה המתאימה לו

[1] אמפליטודה - https://he.wikipedia.org/wiki/משרעת. תדירות - https://en.wikipedia.org/wiki/Frequency.
[2] ראה - https://iw.vvikipedla.com/wiki/Spectrogram אם ברצונך להרחיב בנושא.

ונעבוד על חיזוי ז׳אנר מתוך תמונת ספקטרוגרמה מתאימה. למעשה שיטה זו הרבה יותר יעילה מאשר חיזוי על פי הקובץ סאונד שהוא מערך של ערכים בין -1 ל 1 כפי שמוצג בגרף מעל, מכיוון שאנו מוסיפים עוד תכונות לכל קובץ כמו התדר ובכך מדייקים את המכונה להבחין יותר טוב בין הז׳אנרים.

כעת נחזור לדון בטיוב הנתונים. כפי שציינתי כל הקלטה במאגר הינה באורך של כ- 30 שניות ואין הקלטות ריקות. ולכן לאחר קריאה והמרה של קטע הסאונד לספקטרוגרמה בעזרת librosa נקבל מערך דו-ממדי שבו כל שורה מייצגת את ערכי האמפליטודה השונים לאורך זמן עבור כל תדר. ולכן במערך הדו מימדי יהיו 1025 מערכים, ולכל מערך או שורה יהיו 1292 ערכים. ולכן בסך הכל מדובר ב $1{,}324{,}300 = 1292 \cdot 1025$ ערכים שונים, כלומר לכל קטע קול יהיו $1{,}324{,}300$ תכונות.

כמות כזו של תכונות מעבר לכך שיקריס את המחשב בכל פעם, תגרום לאוברפיטינג בגלל היחס למספר הדגימות המועט (300), ולכן נוכל לבצע את המניפולציה הבאה: נחלק כל קטע קול של 30 שניות ל-3 קטעים של 10 שניות (0-10 שניות לחלק הראשון, 10-20 חלק שני, וחלק שלישי: 20-30 שניות) באופן הזה גם נרחיב את כמות הדוגמאות שלנו פי שלוש, וגם נפחית את מספר המאפיינים המוגזם, ואחרי הכל יש סיבה שבחיי היום-יום גם להקליט הקלטה באורך 30 שניות יהיה מעיק מאד.

כעת, מכל קטע סאונד קיבלנו שלושה חלקים ובכל חלק $441{,}775$ ערכים שונים. בסך הכל כעת יהיו בקירוב[3] 1500 דגימות ביחס ל- 500 שהיו בהתחלה. לאחר שביצענו את כל זה נוכל להמיר את המערך למערך חד-מימדי ולהכניס אותו לתוך המערך הכולל של דוגמאות, במקביל נשים במערך הז׳אנרים את שם התיקייה של הקטע סאונד, כלומר הז׳אנר שלו.

נרמול נתונים

כפי שניתן לראות מהגרף שהוצג בדוגמא הקודמת כאשר טוענים עם librosa את קטע הקול וממירים אותו לגרף ספקטרוגרמה נקבל מערך דו-מימדי של שורות עם ערכים בין ערך האמפליטודה המינימלי שיהיה שלילי וערך האמפליטודה המקסימלי שיהיה חיובי, לדוגמא זהו מערך דו-מימדי של אותו קטע רוק מהדוגמה הקודמת:

```
[[ 8.415056, -4.335116 , 3.4518528, ...,  -5.285228, -29.825329, -2.4258187],
 [ 10.844802, 6.950298 , -11.926927,...,  -4.514557,-5.240502, -5.6513453],
 [ 13.8021965, 7.2432895, 1.342082, ...,  -6.306215,-4.183678,  -7.483863 ],
    ...,
 [-32.175125, -36.48182, -36.48182, ..., -36.48182, -36.48182, -36.48182],
 [-32.320312, -36.48182, -36.48182, ..., -36.48182, -36.48182, -36.48182],
 [ -32.47715, -36.48182, -36.48182, ..., -36.48182, -36.48182, -36.48182]]
```

לכן על מנת לנרמל את הנתונים יהיה עלינו להעביר את כל הערכים להיות בטווח שבין 1- ל- 1 ולכן נשתמש ב StandardScaler של ספריית sklearn.
כעת כל שנותר לעשות יהיה להתאים כל ז׳אנר למספר על מנת שהמכונה תוכל ללמוד, את זה נעשה בעזרת LabelEncoder של ספריית sklearn, ולכן נקבל את ההתאמה:

```
label_dic = { 0: "classical", 1: "hiphop", 2: "jazz", 3: "pop", 4: "rock" }
```

[3] בקירוב מכיוון שיכול להיות שחלק משלושת החלקים של אחד הקטעים אירעה שגיאה בניסיון להכניס אותו למערך. אך מדובר בכמות מזערית של עד 5 חלקים מתוך 1500 ולכן נוכל פשוט לפסוח ולהשתמש ב- try...except על מנת לא לשבור את רצף הריצה. במקרה של שגיאה תודפס לדוגמא ההודעה: `hiphop.00032.wav 20  - had an error` וכך נדע למה לא כל הקבצים נטענו.

ניתוח הנתונים

בחירת שדות למודל

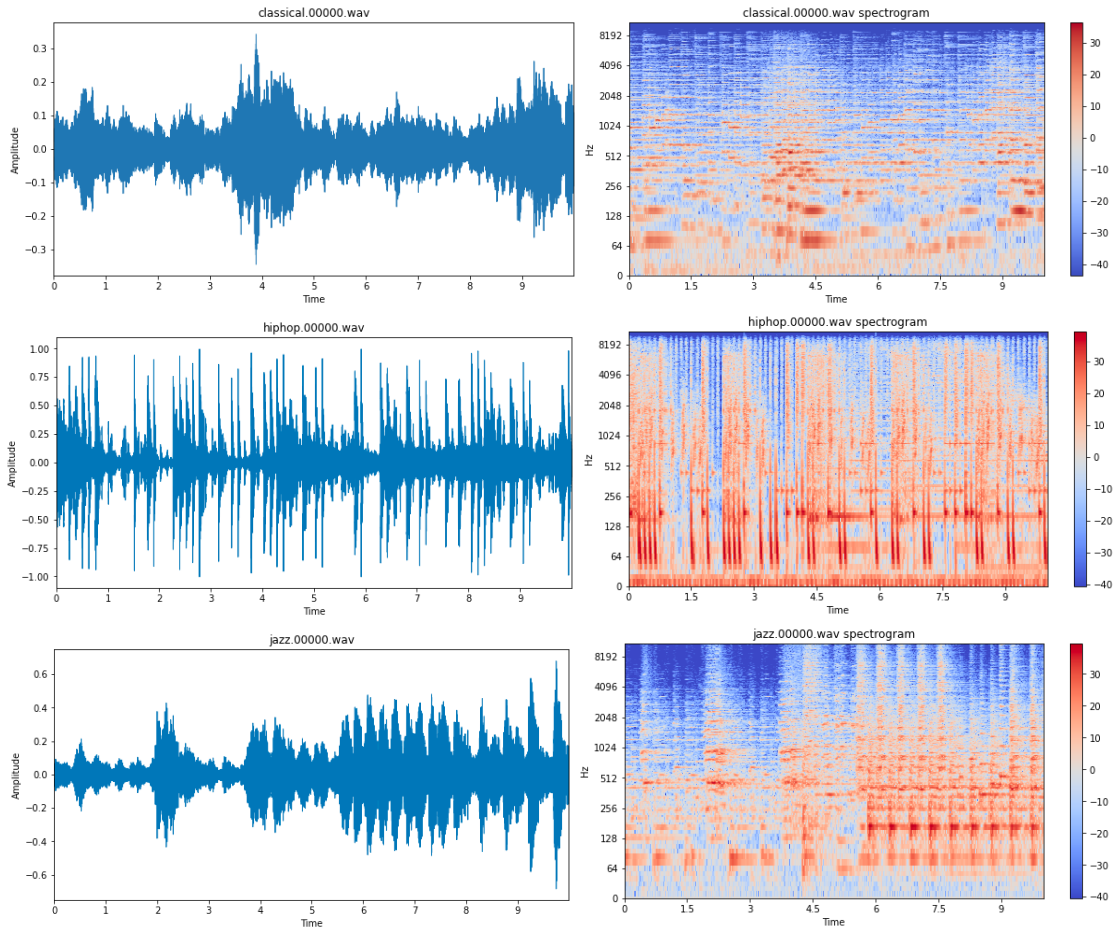
לאחר התנסות עם אחוזים שונים החלטתי לחלק את שדות למודל כאשר 80% מהערכים ילכו לאימון המודל ו- 20% לבדיקה שלו. השתמשתי בפונקציה train_test_split בספריית sklearn על מנת לחלק את הנתונים באופן הבא:

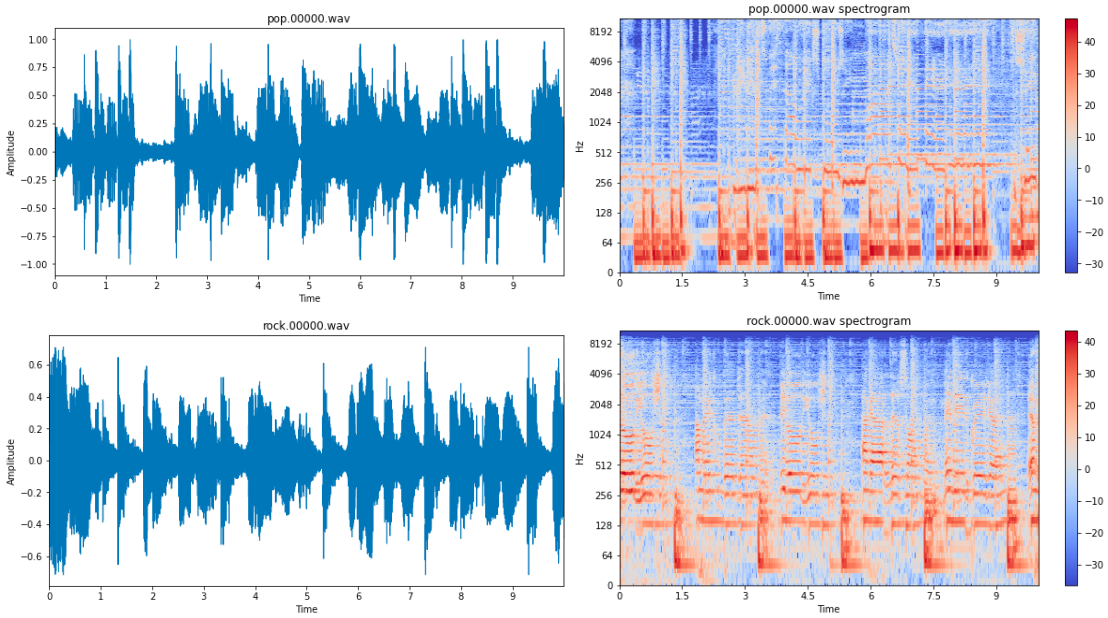
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 1)
```

בחרתי לקחת באופן רנדומלי את הנתונים על מנת שלא ליצור מצב בו המודל מפספס חלק מהז׳אנרים מכיוון שהם מסודרים בסדר אלפביתי וכך לשפר את הדיוק שלו.

הצגת מגמות וקשרים

במקרה שלנו לא תהיה כל כך אפשרות לקשור קשרים רבים בין התכונות של כל דוגמא לבין הז׳אנר שלה, מכיוון שהתכונות הם בעצם הערכים השונים שלה בטווח הזמנים של 10 השניות ואלו כאמור - 441,775 ערכים שונים, כלומר יהיה זה ווקטור בגודל 441,775 עבור כל תכונה. כל שאוכל לעשות הוא יהיה להציג את הספקטרוגרמה השונה המתקבלת לז׳אנרים השונים (כזכור מדובר רק ב 10 שניות הראשונות מתוך הקובץ הכולל):

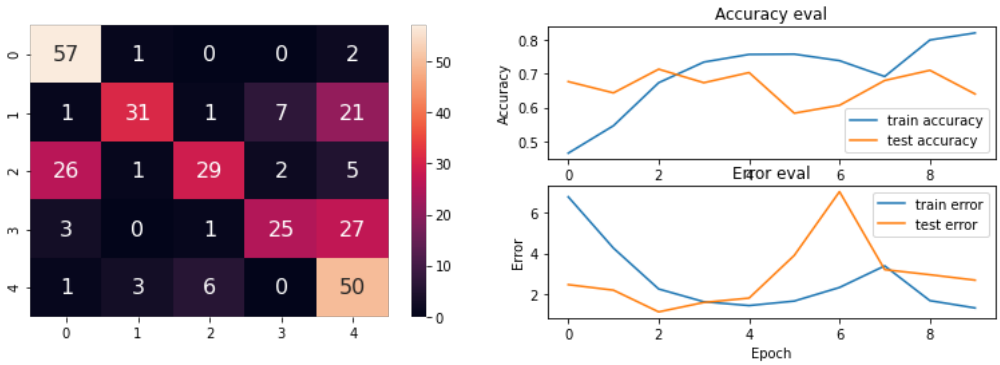




ואכן ניתן לראות כי הספקטרוגרמה אכן מאפיינת את הז׳אנר המתאים לו וקיים הבדל משמעותי בין ז׳אנרים שונים, לדוגמא פופ יותר ״רועש״ מקלאסי. בעצם זהו ההבדל שאותו אנחנו מנסים ללמוד בעזרת המכונה.

תוצאות הרצה

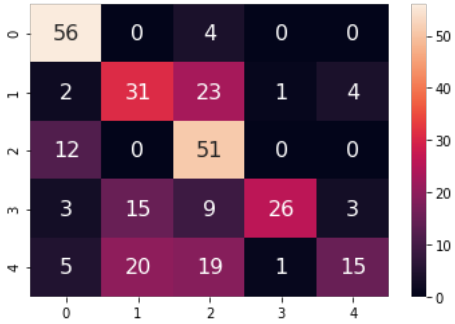
המודלים שבחרתי כזכור הם: TensorFlow mlp, Sklearn KNN.
MLP של TensorFlow - תחילה כתבתי מודל פשוט יחסית ואחר כך שיפרתי אותו. תחילה השתמשתי רק בשכבות Dense, כלומר שכבות נוירונים רגילות עם פונקציית אקטיבציה של relu[4] שידוע כיותר יעילה מ sigmoid. בשכבה האחרונה שמתי 5 נוירונים כמספר התגיות, עם אקטיבציה של SoftMax[5] שגם כן יותר טובה עבור השכבה האחרונה. בנוסף לכך השתמשתי בשכבה Dropout שמוציאה אחוז מסוים מהנוירונים מהרשת ובאופן הזה מונעת אוברפיטינג. בסך הכל אחוז דיוק המודל היה: 0.6400. להלן גרף הלמידה ומטריצת הבלבול של המודל:



ניתן לראות שההפרש בין אחוז הדיוק של ה train לזה של ה test גדול מאד משמע המודל נמצא באוברפיטינג, ולכן ננסה לטפל בבעיה זו בהמשך החלק באופטימיזציה, וכך לשפר את האחוזים של המודל.

[4] ראה - https://he.wikipedia.org/wiki/רשת_עצבית_מלאכותית - פונקציית_אקטיבציה
[5] ראה - 4

KNN של sklearn - נבנה מודל KNN שבו k=5 ולכן הציון יהיה 0.416. ומטריצת הבלבול של המודל תהיה:

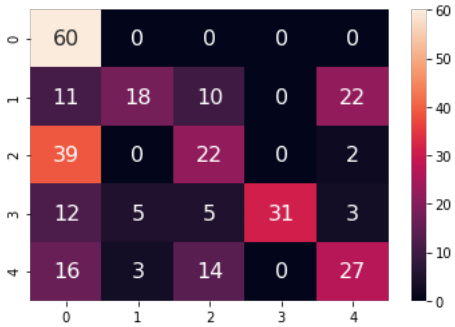


אופטימיזציה

בחלק זה אציג דרכים בהם שיפרתי את שני המודלים.
Knn – על מנת לשפר את המודל היה עלי לבדוק בעבור איזה k נקבל ציון מקסימלי, לכן כתבתי קוד שיבדוק את הציונים בעבור כל ערכי k בין הערכים 1 ל- 8 והתוצאה שהתקבלה היא:

k: 1, score: 0.526
k: 2, score: 0.446
k: 3, score: 0.43
k: 4, score: 0.41
k: 5, score: 0.416
k: 6, score: 0.41
k: 7, score: 0.426
k: 8, score: 0.42
best k: 1 , max score: 0.526

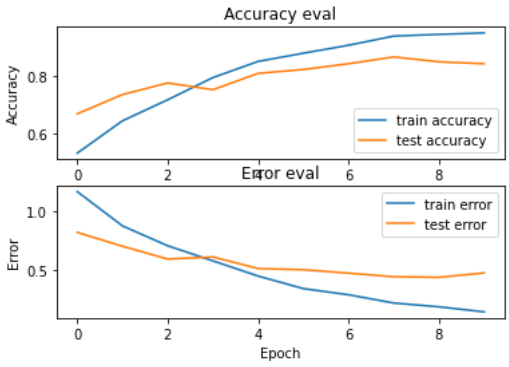
ולכן הציון הטוב ביותר מתקבל בעבור k=1 והוא 0.526 ומטריצת הבלבול של המודל תהיה:



בזאת תם הניסיון לשפר את מודל ה- Knn.

MLP – כזכור המודל האחרון שאימנו הגיע לאחוזי דיוק של 0.8264 ב- train, 0.6400 ב- test.
כלומר המודל היה באוברפיטינג, על מנת לפתור בעיה זו יהיה עלינו להשתמש בכלים מתקדמים יותר שיש ל TensorFlow להציע, כגון: קונבולוציה, שהמטרה העיקרית בה היא לצמצם באופן חכם את מספר המשתנים וכך למנוע אוברפיטינג. המודל המשופר יפורט לפרטי פרטים בחלק "הסבר על קוד נבחר", אך כרגע רק אציג את התוצאות שלו. בציון המודל קיבל: 0.8433.

ולהלן מטריצת הבלבול וגרף הלמידה של המודל:



מסקנות וממצאים

המסקנה המתקבלת משני המודלים היא שהמודל השני הרבה יותר מדויק. לסיכום, ברצוני לסכם את העבודה כעבודה שהעשירה אותי בהרבה ידע, מנושא גלי הקול ועד קונבולוציות במודל. העבודה אתגרה אותי לא מעט כפי שרציתי, אמשיך לעבוד על הפרויקט בתכנון להכין אפליקציה שתאפשר להשתמש במודל.

קוד הפרויקט

קישור למחברת קולאב –
https://colab.research.google.com/drive/1ix0kdv3DY3b0EQipRB9XI78PKL2qc8t6?usp=sharing

הסבר על קוד נבחר

בחרתי בקטע קוד של בניית ואימון המודל mlp השני של TensorFlow מכיוון שלפי דעתי הוא החלק המסובך והכי חשוב בפרויקט, ולכן דורש הסבר מפורט יותר בעברית. להלן הקטע קוד לבניית המודל:

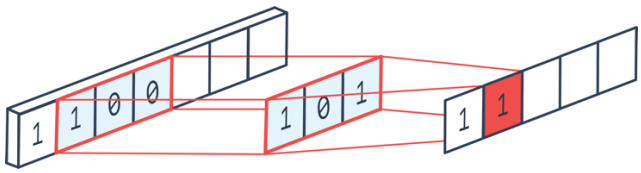
```
model = tf.keras.models.Sequential([

  # Reshape
  tf.keras.layers.Reshape((441775, 1)),

  # Conv 1
  tf.keras.layers.Conv1D(32,6,strides=2),
  tf.keras.layers.MaxPool1D(4),

  # Conv 2
  tf.keras.layers.Conv1D(32,6,strides=2),
  tf.keras.layers.MaxPool1D(4),

  # Flatten
  tf.keras.layers.Flatten(),

  # regular perceptrons layers
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(160, activation='relu'),

  # dropout
```

```
  tf.keras.layers.Dropout(0.2),

  # regular perceptrons layers
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(160, activation='relu'),

  # final softmax layer
  tf.keras.layers.Dense( num_labels, activation='softmax')
])
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.00001),
              loss = loss_fn,
              metrics = ['accuracy'])

hist = model.fit(x_train, y_train , validation_data = (x_test, y_test),
epochs=10, batch_size=1024)
plot_history(hist)
```

כעת אני אסביר את השכבות השונות של המודל:

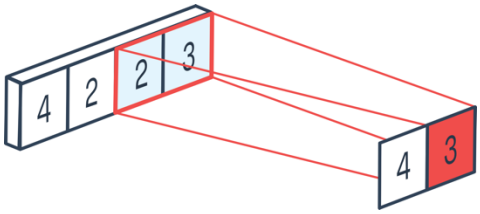
Reshape – בדיוק כפי שמבצעים Reshape עם NumPy כך גם נוכל לשים במודל עצמו, במודל שלנו אנו לוקחים את המערך החד-ממדי מהצורה (441775,) שמתקבל כקלט למודל והופכים אותו למערך דו ממדי מצורה (441775,1), כלומר בעצם מכניסים אותו לתוך מערך חדש, אמנם זה נראה חסר משמעות אבל יש לבצע את זה בעבור השכבות הבאות מכיוון שהן מקבלות כקלט רק מערך דו ממדי.

Conv1D – השכבה הזו היא שכבת קונבולוציה, ישנה הגדרה כללית יותר בעבור קונבולוציה מתמטית אבל במקרה שלנו השכבה מקבלת מטריצה מהשכבה הקודמת לה (ולכן עשינו Reshape למערך דו-ממדי) ומחזירה מטריצה חדשה מצומצמת יותר, כלומר אנחנו עושים "צמצום חכם" של המערך. דבר זה נועד למנוע אוברפיטינג ולאפשר שימוש ביותר נוירונים בשכבות ה Dense.
הקונבולוציה עוברת על המערך שהיא מקבלת בעזרת חלון שנקבע על ידי הפרמטר kernel שהיא מקבלת, בכל פעם היא לוקחת את האיברים באותו חלון מכפילה בפרמטרים שמשתנים כל פעם בדומה לנוירונים וסוכמת, וזהו האיבר במערך המצומצם שהיא תוציא. לאחר מכן היא "מדלגת" מספר צעדים קבוע (strides) ומבצעת את התהליך שוב. כמות הפעמים בהם נבצע קונבולוציה על אותו מערך ייקבע על פי ה filter.
בסך הכל הפרמטרים שהשכבה מקבלת על פי הסדר הם: הפילטר – כמות הפעמים בהם נבצע קונבולוציה על אותו מערך. Kernel – גודל החלון איתו נרוץ על המערך. Strides – המרווחים שנדלג בין חלון לחלון. דוגמא וויזואלית לביצוע קונבולוציה על מערך:



במקרה שלנו יהיו 32 פילטרים, גודל החלון יהיה 6 איברים, ונדלג בכל שלב 2 איברים.

MaxPool1D – פעולה דומה קונבולוציה אך פשוטה בהרבה, הפעולה מקבלת רק kernel ובכל פעם לוקחת את המספר הגדול ביותר מהחלון למערך המצומצם החדש. דוגמה וויזואלית לביצוע MaxPooling על מערך:



במקרה שלנו גודל החלון יהיה 4 איברים מתוכם ניקח את הגדול ביותר.

Flatten – משטחת את המערך הדו ממדי שהיא מקבלת בחזרה למערך חד ממדי.
Dense – שכבת נוירונים רגילה. ופונקציית אקטיבציה relu שיותר יעילה מאשר sigmoid. ו SoftMax שיותר יעיל לשכבה אחרונה.

Dropout – מוציא אחוז מסוים מהנוירונים מהשכבה הקודמת, מונע אוברפיטינג.

לאחר בניית המודל נאמן לאורך 10 epochs ונשתמש ב- validation_data על מנת להציג בכל פעם את אחוז הדיוק גם על נתוני הבדיקה ואת batch_size נגדיר ל 1024 על מנת להימנע מקריסה של ה RAM תוך כדי אימון. בסך הכל, כאמור אחוזי הדיוק שיתקבלו יהיו:
ומטריצת הבלבול וגרף הלמידה שנקבל יהיו:

פירוט בדיקות

שם הבדיקה	מטרת הבדיקה	מה נדרש לבצע	מתי	השפעה ומסקנות אופרטיביות
בניית מודל MLP ראשוני	קבלת הבנה כיצד לבנות מודל MLP	כתיבת מודל MLP ב TansorFlow	29.4	מסקנות שמודל פשוט יוביל לאוברפיטינג
בניית מודל MLP שני	שיפור המודל MLP הקודם	הוספת שכבות ושינוי פרמטרים מהמודל הקודם	30.4	גם לאחר שכבות נוספת של dropout המודל מגיע לאוברפיטינג
בניית מודל KNN ראשוני, k=5	כתיבת מודל KNN	לכתוב מודל KNN של sklearn	2.5	מסקנה שעלינו לבדוק באופן שיטתי את כל ערכי k על מנת להגיע לציון מקסימלי
בדיקה שיטתית של מודל KNN	למצוא k בעבורו ציון המודל יהיה מקסימלי	יצירת לולאה בה בכל פעם מגדילים את ערך ה k ומוצאים את הערך עבורו הציון מקסימלי	2.5	מסקנה שבעבור k=1 מתקבל ציון מקסימלי
כתיבת מודל MLP מתקדם	כתיבת מודל MLP סופי בעל אחוז דיוק של לפחות 80%	הוספת קונבולוציות ומניפולציות כאמור בחלק הקודם	20.5	קבלת מודל MLP אופטימלי שמוציא ציון מצוין.
בדיקת המודל MLP בעבור הקלטות מהחיים האמתיים	לבדוק האם המודל צודק גם בהקלטות שלא הגיעו מהמאגר נתונים	הקלטתי הקלטות מהרדיו של קטעים מז׳אנרים שונים ובדקתי אם המודל חוזה נכון	20.5	בעבור כל ההקלטות שנתתי המודל צדק בז׳אנר.