פרויקט גמר
רשתות נוירונים ולמידה עמוקה

מנחה: ד"ר בני בורנפלד

מגיש:
רותם צלישר 203773601

מבוא:

בפרוייקט זה אחקור את בעיית הקלסיפיקציה של קטעי אודיו לכדי אחד מעשרה סגנונות מוסיקליים. הבסיס לפרוייקט זה הוא בסיס נתונים המכיל 1000 דגימות אודיו באורך 30 שניות ממגוון של עשרה סגנונות מוסיקליים שונים.

קישור למחברת Kaggle: https://www.kaggle.com/code/aasimahmed04/music-genre-classifier

המטרה: לקחת מודל קיים ולשפר את רמת הדיוק שלו באמצעות טכניקות שונות.

שלבים:

1. היכרות עם הדאטא: אציג את הדאטא, את רקע לחקר הפיצ'רים ואת תהליך בניית מרחב הפיצ'רים.
2. הצגת מודל הבסיס מKaggle
3. חקר של מגוון של ארכיטקטורות שונות וביצועיהן

היכרות עם הדאטא:

בסיס הנתונים בבעיה זו הוא טבלה, המכילה פיצ'רים מרכזיים שעובדו ממאגר נתונים של 1000 דוגמיות סאונד באורך 30 שניות ותיוגן.

הפיצר'ים המרכזיים:

- תוחלת ושונות של כרומוגרמה (כרומוגרמה היא תמונה המכילה את התו המתנגן לכל יחידת זמן בממוצע, כמו ספקטרוגרמה רק שציר התדר מתחלק ע"פ החלוקה לתווים)
- תוחלת ושונות של אנרגיה הממוצעת האגורה באות (RMS)
- תוחלת ושונות של התדר שמתחתיו מתרכזת 85% מהאנרגיה האגורה באות (או יותר נכון, בפריים נתון מתוך האות. Roll Off Frequency)
- תוחלת ושונות לרוחב פס של האות (Spectral Bandwidth)
- תוחלת ושונות של "מרכז המסה" של פיזור האנרגיה בתדר (Spectral Centroid)
- MFCC's – הסבר עליהם יכול להיות מצורף בספר פרוייקט משל עצמו. בקצרה – הMFCC היא ספקטרוגרמה של ספקטרוגרמה (בלומר, מדד למחזוריות של התמרת פורייה של האות).

יחד עם הטבלה, מגיע גם מאגר הדוגמיות שממנה יצרו את הטבלה. באחת הארכיטקטורות שמימשתי גם ביצעתי עיבוד למאגר המידע הזה בעצמי (על מנת להתאים אותו לכניסת הארכיטקטורה) בעזרת ספריית Librosa הנועדה לעיבוד אודיו.

הדאטא עצמו מגיע מאוזן, לכן לא היו הרבה בעיות של עיבוד מקדים. עבודה שחוזרת על עצמה הרבה באינטרנט היא חלוקת כל דוגמית של 30 שניות ל10 דוגמיות של 3 שניות וחילוץ הפיצ'רים משם, על מנת לעבות את מאגר המידע. לאחר שראיתי שהעבודה המקדימה הזו חוזרת על עצמה הרבה, מימשתי אותה גם.

על מנת לנסות לשמור על מסמך זה תמציתי לגבי הפרוייקט, אני מדלג פה על הצגת הדאטא עצמו. להרחבה, ניתן להיכנס למחברות Feature Generation,Data Visualisation בו צירפתי גרפים של כל אחד מהפיצ'רים עצמו.

ארכיטקטורת בסיס:

ארכיטקטורת הבסיס היא רשת הנוירונים המובאת בקישור:

https://www.kaggle.com/code/aasimahmed04/music-genre-classifier

מבנה הרשת:

-

```
In [12]: #Creating a Neural Network
         model = Sequential()

         model.add(Flatten(input_shape=(58,)))
         model.add(Dense(256, activation='relu'))
         model.add(BatchNormalization())
         model.add(Dense(128, activation='relu'))
         model.add(Dropout(0.3))
         model.add(Dense(10, activation='softmax'))

         model.summary()
```

חלוקת הנתונים לסדרת אימון, ולידציה ומבחן:

- החלוקה במחברת עצמה בKaggle הייתה מבולבלת ולא מספקת (השתמשו בסדרת מבחן בלבד בלי ולידציה, ולמרות זאת שלחו את אותה הסדרה גם כ"ולידציה" באימון וגם למבחן עצמו) – לכן לקחתי על עצמי את חלוקת הדאטא הנכונה.
- הדאטא מחולק ל680 דגימות לאימון, 170 דגימות לולידציה ו150 דגימות למבחן.

Split the data to train and test:

```
In [10]: X_train, X_test, y_train, y_test = train_test_split(scaled_data, labels, test_size=0.15,shuffle=True, random_state=1)
         X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,shuffle=True, random_state=1)

In [11]: print("Train data size: ", X_train.shape);
         print("Validation data size: ",X_val.shape);
         print("Test data size: ", X_test.shape);

         Train data size:  (680, 58)
         Validation data size:  (170, 58)
         Test data size:  (150, 58)
```

קימפול והרצת הלמידה:

Compiling and fitting:

```
In [13]: #Compiling & Fitting the Model
         adam = keras.optimizers.Adam(learning_rate=1e-4)
         model.compile(optimizer=adam,
                       loss='sparse_categorical_crossentropy',
                       metrics=['accuracy'])
         hist = model.fit(X_train, y_train, validation_data= (X_test, y_test),
                          epochs=100,
                          batch_size=32)
```

תוצאות ריצת האימון:

```
Epoch 100/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9075 - loss: 0.3250 - val_accuracy: 0.7667 - val_loss: 0.8645
```

בחינת הרשת על סט המבחן:

```
In [14]: y_predict = np.argmax(model.predict(X_test),axis=1)

         5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step

In [15]: #Test Accuracy
         acc = np.mean(y_predict == y_test)*100
         print("test accuracy = %.2f"%acc, "%");

         test accuracy = 72.00 %
```

ניתן לראות כי הרשת מגיעה לדיוק של 72 אחוז מול 10 מחלקות שונות (לא רע בכלל!).

שיפור ארכיטקטורת הבסיס (ע"י ניסיון למזער התאמת יתר):

בחלק זה אני בוחן מספר אפשרויות למלחמה בהתאמת יתר. לבסוף אסכם ואציג את הרשת המוצלחת ביותר.

<u>המחברת:</u>

Simple Neural Network - Based on Kaggle's aasimahmed04 - L1 Regularization

במחברת זו הוספתי רגולריזציה מסוג L1. שיטה זו היא אחת משתיים לרגולריזציה על קצב שינוי הפרמטרים של הרשת. ההבדל העיקרי בשיטה זו היא ששיטה זו יכול להגיע למצב שהיא "משתקת" נוירון (מגיעה לאפס). הרשת:

Improving The Network:

```
In [17]: #Creating a Neural Network
         model = Sequential()

         model.add(Flatten(input_shape=(58,)))
         model.add(Dense(256,kernel_regularizer=regularizers.l1(0.0005), activation='relu'))
         model.add(BatchNormalization())
         model.add(Dense(128,kernel_regularizer=regularizers.l1(0.0005), activation='relu'))
         model.add(Dropout(0.3))
         model.add(Dense(10, activation='softmax'))

         model.summary()
```

דיוק האימון:

```
Epoch 100/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9299 - loss: 1.5153 - val_accuracy: 0.7667 - val_loss: 1.9878
```

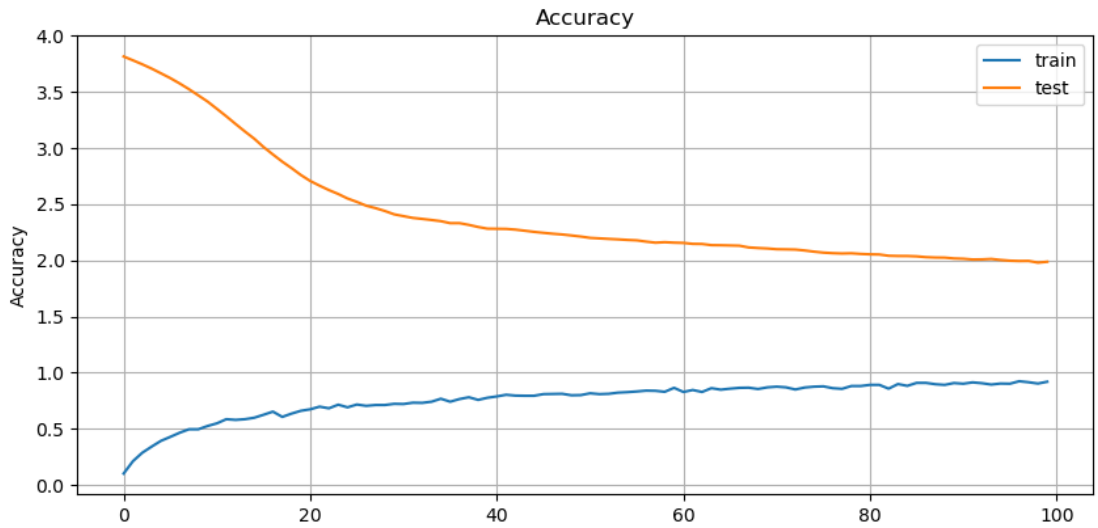
דיוק המבחן:

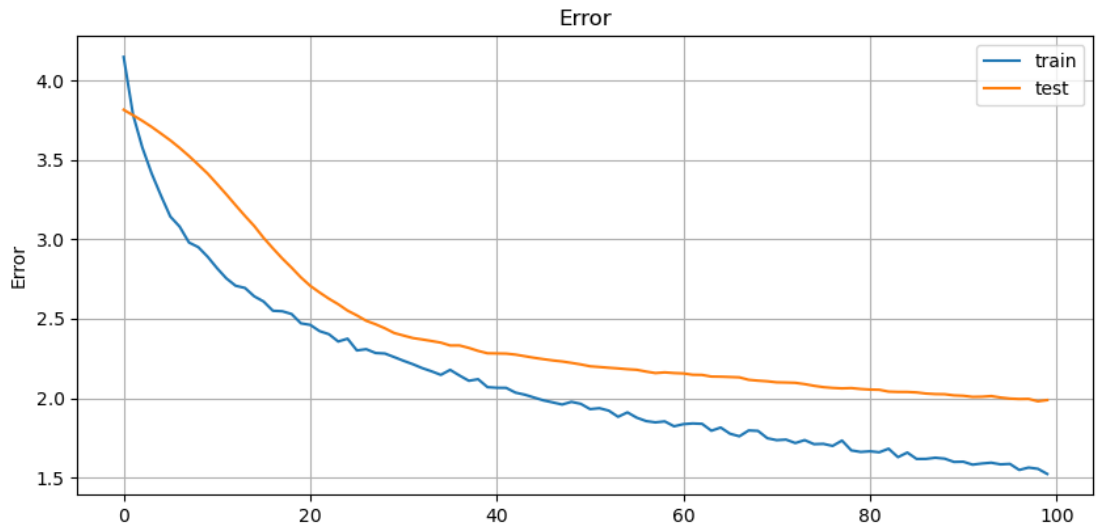
```
In [19]: y_predict = np.argmax(model.predict(X_test),axis=1)

         5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step

In [20]: #Test Accuracy
         acc = np.mean(y_predict == y_test)*100
         print("test accuracy = %.2f"%acc, "%");

         test accuracy = 76.67 %
```

התקבל שיפור של כ4.6 אחוז!

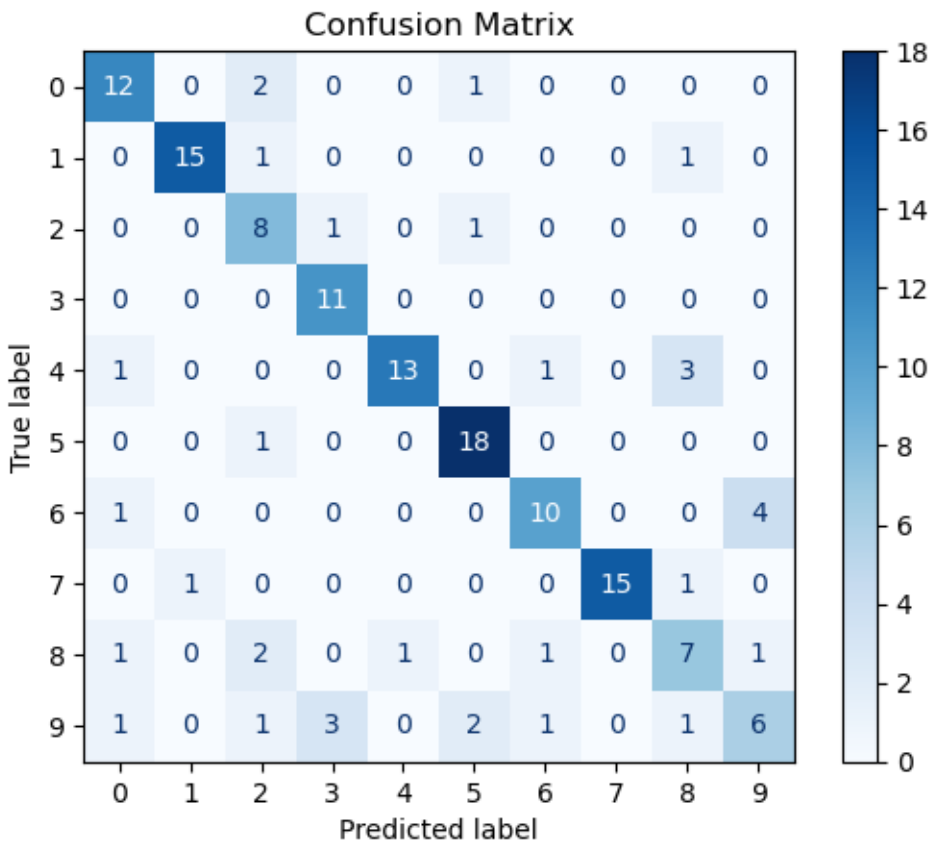
גרפים:

דיוק מבחן לעומת אימון:





מטריצת ערבול:



ניתן לראות ממטריצת הערבול כי אין מחלקה מסויימת שממופה באופן מסיבי למחלקה ספציפית אחרת. השגיאות מפוזרות לאורך המחלקות השונות.

<u>סיכום:</u> התקבל שיפור משמעותי ביותר בביצועי הרשת. הרשת עצמה עדיין לא מצליחה לשבור את מחסום ה-80 אחוז דיוק ולכן המשכתי לבחון ארכיטקטורות אפשריות נוספות.

<u>המחברת:</u>

Simple Neural Network - Based on Kaggle's aasimahmed04 - L2 Regularization

רגולריזציה זו לא יכולה "לשתק" אף נוירון. הרשת:

Improving The Network:

In [19]:
```
#Creating a Neural Network
model = Sequential()

model.add(Flatten(input_shape=(58,)))
model.add(Dense(256,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
model.add(BatchNormalization())
model.add(Dense(128,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(10, activation='softmax'))

model.summary()
```

הרצת האימון:

```
Epoch 100/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9391 - loss: 0.4095 - val_accuracy: 0.7400 - val_loss: 0.9812
```

הרצת המבחן:

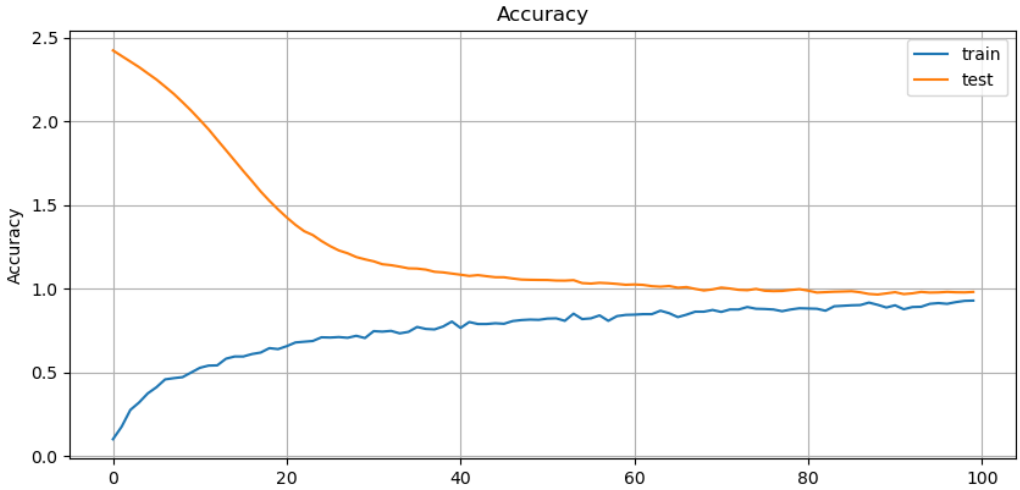
Test set:

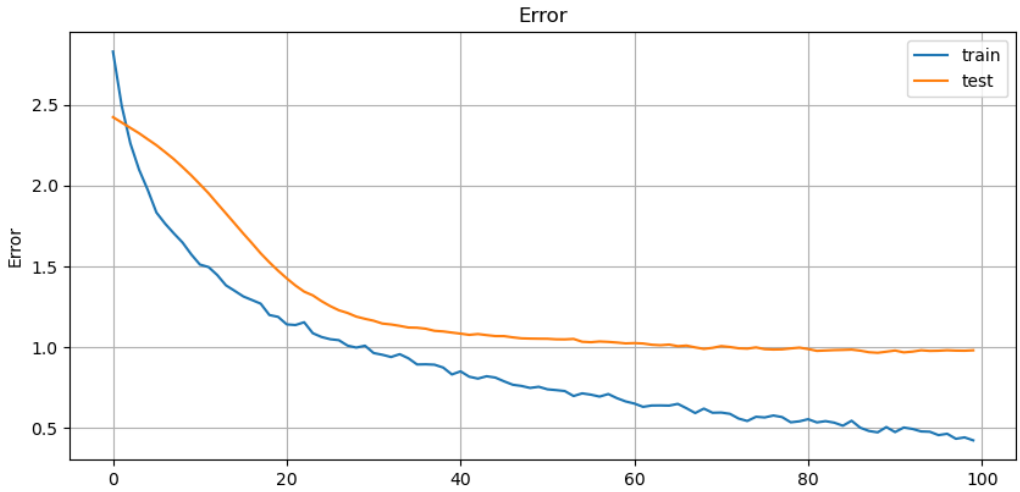
```
In [21]: y_predict = np.argmax(model.predict(X_test),axis=1)

5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step

In [22]: acc = np.mean(y_predict == y_test)*100
         print("test accuracy = %.2f"%acc, "%");

test accuracy = 74.00 %
```

קיבלתי דיוק של 74 אחוז. ביצועים מעט פחות טובים מהשיפור הקודם שעשיתי, אך עדיין שיפור ביחס למודל הבסיס.

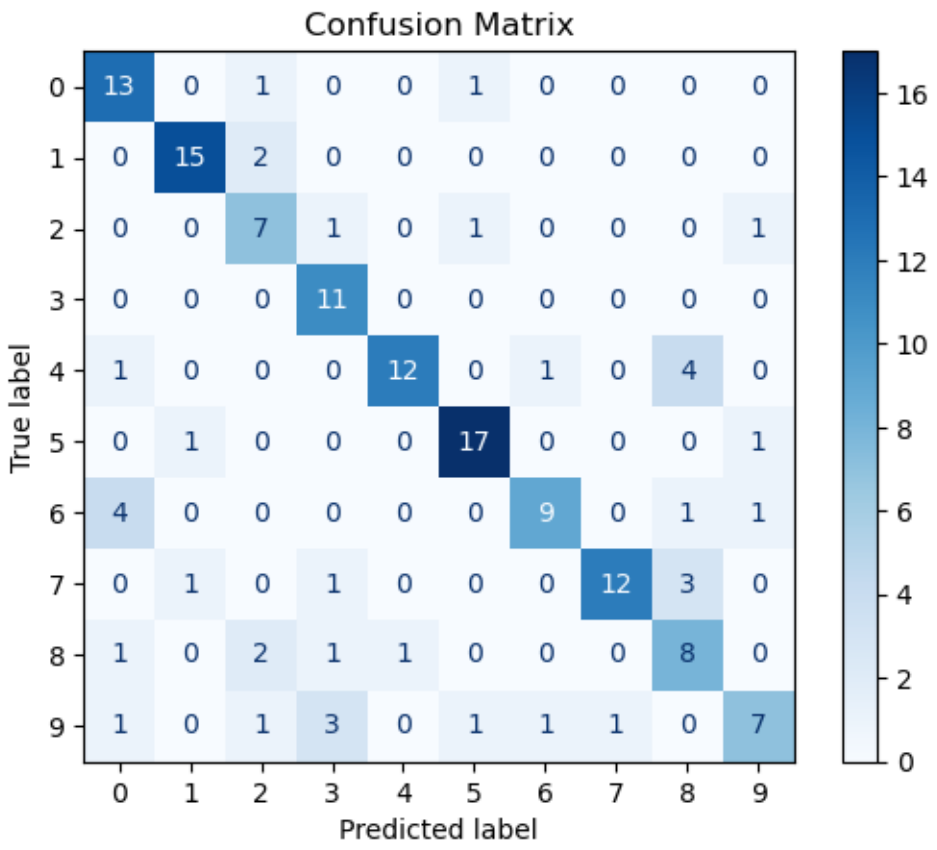
גרפים:

דיוק מבחן לעומת אימון:





מטריצת ערבול:



ניתן לראות באותו האופן, כי הטעויות מתפזרות על מספר יחסית גדול של מחלקות ולא ממחלקות ספציפיות למחלקות ספציפיות.

<u>המחברת</u>:

Simple Neural Network - Based on Kaggle's aasimahmed04 - L2 Regularization SECOND STAGE ONLY

במחברת זו, מימשתי את הרגולריזציה על השכבה השניה בלבד. מתוך קו המחשבה ש**הדאטא המוצג במחברת Data Visualization אינו מייצר עקומי החלטה מורכבים מידי במישור הדאטא.** לכן, ארצה לייצר רשת זהירה יותר בשלב השני לעומת השלב הראשון.

הרשת:

Improving The Network:

```
In [19]: #Creating a Neural Network
         model = Sequential()

         model.add(Flatten(input_shape=(58,)))
         model.add(Dense(256, activation='relu'))
         model.add(BatchNormalization())
         model.add(Dense(128,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
         model.add(Dropout(0.3))
         model.add(Dense(10, activation='softmax'))

         model.summary()
```

אימון הרשת:

```
Epoch 100/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9139 - loss: 0.4019 - val_accuracy: 0.7471 - val_loss: 0.7761
```

סדרת המבחן:

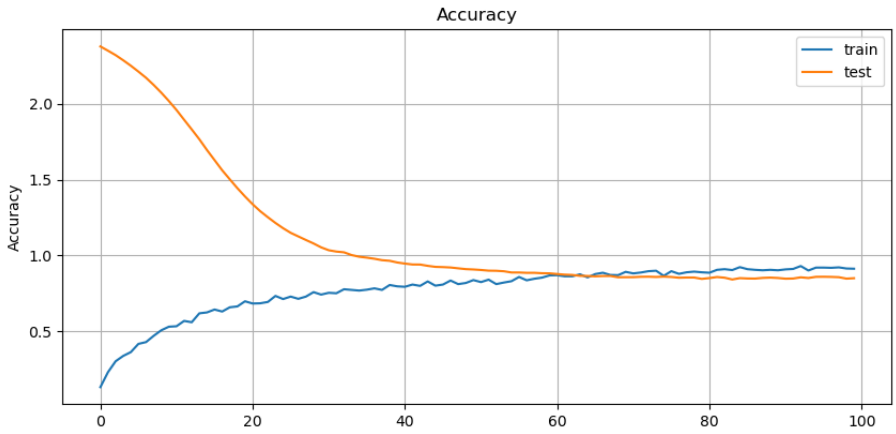
Test set:

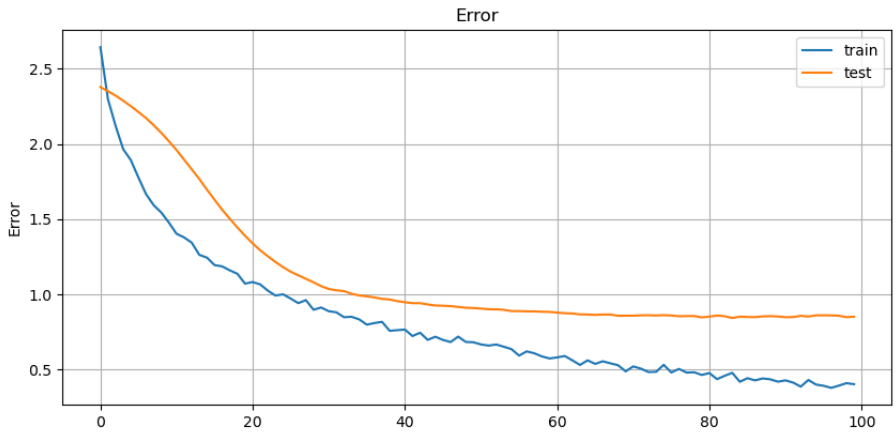
```
In [21]: y_predict = np.argmax(model.predict(X_test),axis=1)

5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step

In [22]: #Test Accuracy
         acc = np.mean(y_predict == y_test)*100
         print("test accuracy = %.2f"%acc, "%");

test accuracy = 75.33 %
```

קיבלתי דיוק של 75.3 אחוז בסדרת המבחן. דיוק טוב יותר משל הרשת בעלת רגולריזציה מסוג L2 לשתי השכבות, אך פחות טוב משל הרשת בעלת רגולריזציה מסוג L1.

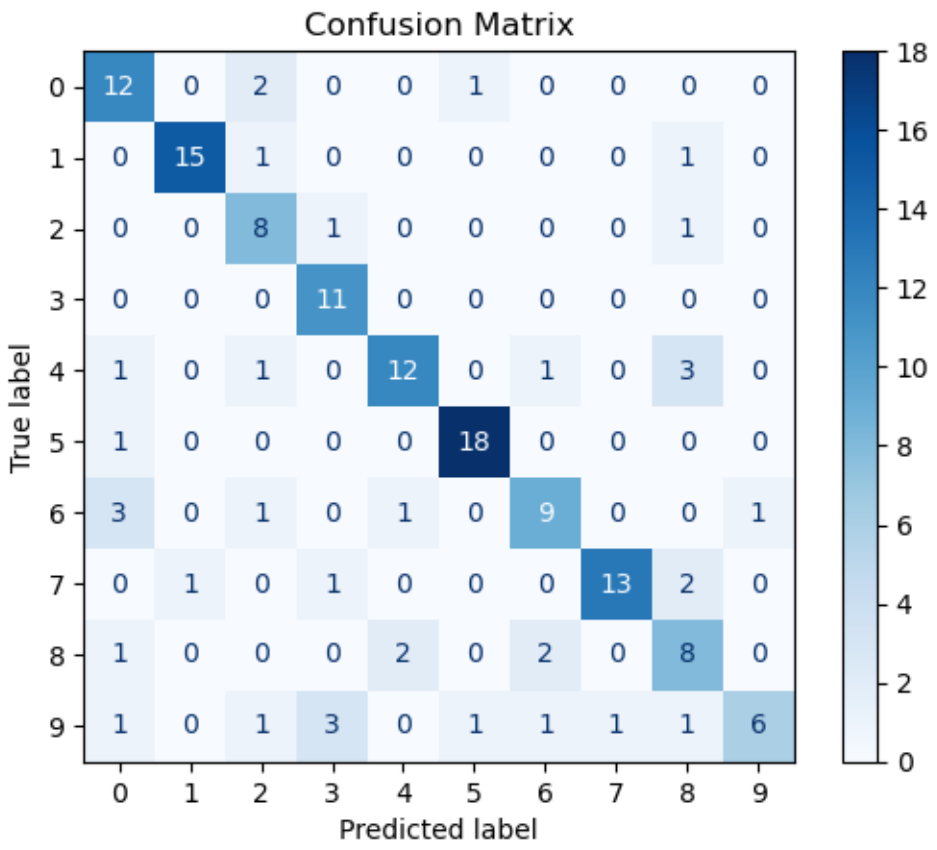
גרפים:

דיוק אימון לעומת מבחן:





מטריצת ערבול:



המסקנות מאוד זהות. השגיאות מפוזרות בין מגוון של מחלקות.

למידת העברה - VGGish:

אחת הבעיות העיקריות בפרוייקט זה היה הרצון לרתום למשימה גם רשת בארכיטקטורת CNN. בעיה שהתעוררה היא העובדה שהמידע המעובד מקטעי האודיו הוא לא תמונה או מידע דו מימדי כלשהו. המידע לכל דוגמת אודיו מוחזק בצורת וקטור (של ערכי ממוצע, שונויות וערכים נוספים).

בהמשך, הראיתי גם ביצוע של רשת CNN על המידע (ע"י סידור צורת הכניסה לרשת ומימוש רשת המבצעת קונבולוציות חד ממדיות) – כל זאת בהמשך.

תוך כדי קריאה וחשיבה על מודל VGG שלמדנו עליו, הסתבר לי שגוגל התמודדו עם הבעיה הזו בעבר. גוגל ייצרו מודל בשם VGGish, שנוצר על מנת לתת את המענה לרשת קונבולוציה שיודעת להתמודד עם המידע שמוחזק עבור קטעי אודיו.

רשת זו מצפה לכניסה ברזולוציה של 16 קילו-הרץ ובאורך של שניה, כלומר, 16 אלף דגימות. מוצא הרשת הוא וקטור באורך 128. במימוש שלי, מוצא המערכת של גוגל ייכנס לClassifier משלי (כלומר, אשתמש בעיבוד המקדים של רשת הקונבולוציה כסוג של Embedding לקטע האודיו).

הורדת המודל והעברת קטעי האודיו בVGGish מובאת במחברת:

Transfer Learning - VGGish (Google Model) - Preprocessing And Arranging the Data (Notebook 1 of 2)

Loading the architecture:

```
In [3]: # Load VGGish feature extractor from TensorFlow Hub
        vggish = hub.load("https://tfhub.dev/google/vggish/1")
```

מוצא הVggish נשמר במידע טבלאי שיוזן למסווג שמעליו.

Create the .csv file out of the data outside the VGGish network:

This step is in order to not have to process the data through the VGGish model each run (the process is very slow)

```
In [8]: # Convert to DataFrame
        df_features = pd.DataFrame(X)
        df_labels = pd.DataFrame(y, columns=['label'])

        # Combine features and labels into one DataFrame
        df = pd.concat([df_features, df_labels], axis=1)

        # Save to CSV
        csv_file_path = 'vggish_features_labels.csv'
        df.to_csv(csv_file_path, index=False)

        print(f"Dataset saved to {csv_file_path}")

        Dataset saved to vggish_features_labels.csv
```

המחברת (של המסווג):

Transfer Learning - VGGish (Google Model) - Preprocessing And Arranging the Data (Notebook 2 of 2)

במחברת זו מימשתי את המסווג מעל רשת הקונבולוציה:

```
In [6]: # Build a simple classifier on top of VGGish features
        model = Sequential()

        model.add(Flatten(input_shape=(X_train.shape[1],)))
        model.add(Dense(128,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
        model.add(BatchNormalization())
        model.add(Dense(64,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(16,kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(10, activation='softmax'))

        model.summary()
```

ריצת האימון:

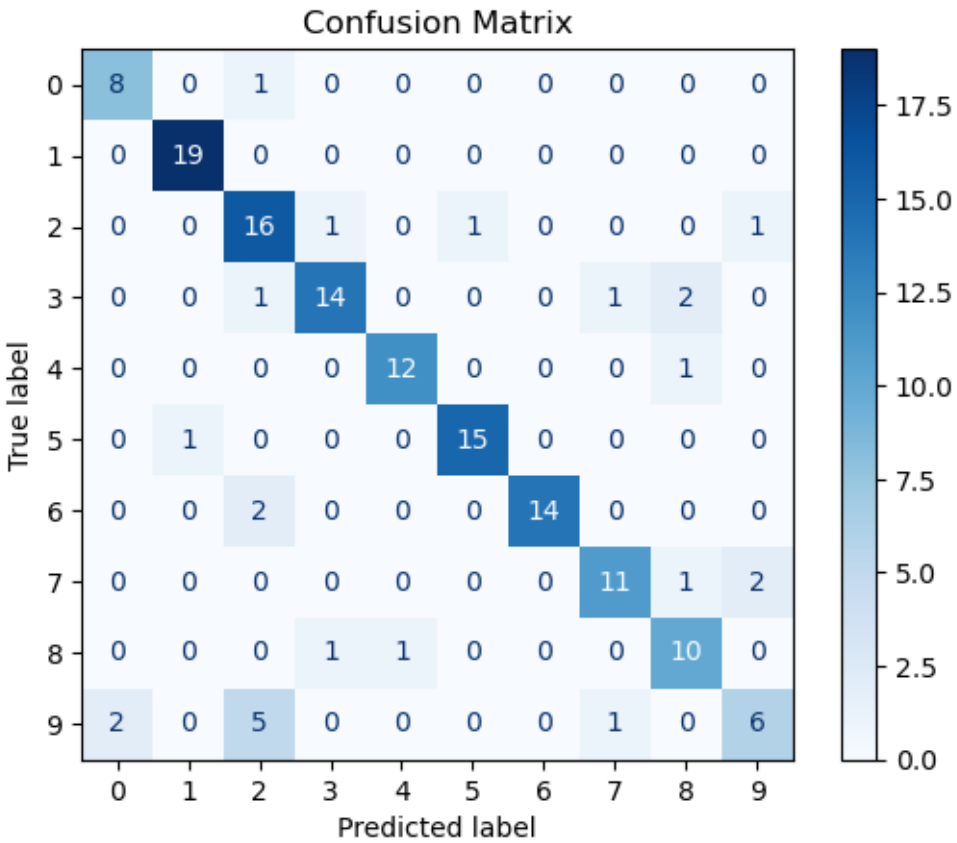
```
Epoch 100/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7958 - loss: 0.7165 - val_accuracy: 0.7941 - val_loss: 0.7923
```

ריצת המבחן:

```
In [8]: y_predict = np.argmax(model.predict(X_test),axis=1)

        5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step

In [9]: acc = np.mean(y_predict == y_test)*100
        print("test accuracy = %.2f"%acc, "%");

        test accuracy = 83.33 %
```

הגעתי ל83.3 אחוז דיוק! שיפור של 7 אחוז מהביצוע הטוב ביותר בחלק הקודם.

מטריצת ערבול:



השגיאה של תיוג 2 לעומת תווית אמיתית 9 קצת מעצבנת אבל כל השאר מפוזרות יחסית.

בנקודה זו חשבתי איך אפשר לרתום את נושא הגדלת בסיס הנתונים. אין הרבה משמעות למיסוך תדר לספקטרוגרמות או כרומוגרמות מכיוון שהערך שנכנס לטבלת הנתונים הוא ממוצע או שונות של תדר לאורך כל הקטע (איפוס עמודה לא ישנה בצורה דרסטית את הממוצע הזה).

קראתי קצת בדיונים בKaggle וראיתי שהרבה מימשו את הרעיון של חלוקת קטעי האודיו של 30 שניות ל10 קטעים של 3 שניות. מסתבר שהביצועים על הדאטא הזה היו כל כך טובים שהפתרון הזה המשיך לחזור על עצמו ובסופו של דבר הטבלאות האלה מצאו את עצמן בתוך הדאטא שמוצג עם הבעיה. (המשך בעמ' הבא)

המחברת:

Transfer Learning - VGGish (Google Model) - Preprocessing And Arranging EXTENDED DATASET (Notebook 1 of 2)

- במחברת זו חזרתי על אותו העיבוד והעברת המידע ברשת VGGish אך חילקתי כל קטע אודיו ל-10 קטעים של 3 שניות.

המחברת:

Transfer Learning - VGGish (Google Model) - Classifier EXTENDED DATASET (Notebook 2 of 2)

המסווג היושב מעל הרשת VGGish הוא בדיוק אותו המסווג שישב על הרשת הקודמת.

הרצת האימון:

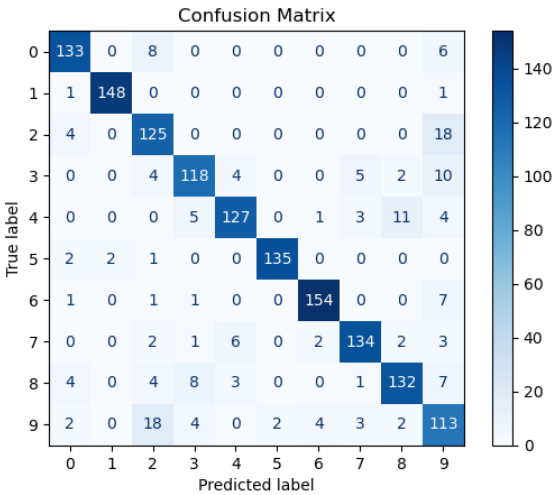
```
Epoch 100/100
213/213 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9141 - loss: 0.3466 - val_accuracy: 0.8629 - val_loss: 0.5197
```

הרצת המבחן:

```
In [8]: y_predict = np.argmax(model.predict(X_test),axis=1)

        47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step

In [9]: acc = np.mean(y_predict == y_test)*100
        print("test accuracy = %.2f"%acc, "%");

        test accuracy = 87.99 %
```

הצלחתי להגיע לסיווג של 88 אחוז! הביצוע הכי טוב (בפער) עד כה.

מטריצת ערבול:



השגיאה של פרדיקציה של 2 לעומת תווית 9 חוזרת על עצמה שוב פה. מסתמן שבעבור ארכיטקטורה זו הבלבול בין המחלקות האלו ספציפית הוא מאוד בעייתי (גם פרדיקציה של 9 לעומת תווית 2).

קיימת מחברת נוספת מצורפת להגשה, המציגה מימוש של רשת CNN המבוססת על הרשת שקיבלנו במטלות בית (קיים קישור לרשת המקורית במחברת), אך במקום שכבות קונבולוציה דו מימדית אני ממש שכבות חד מימדיות. הרשת הזאת הניבה ביצועים נוראיים ואני לא מצרף סקירה שלה פה, ניתן לפתוח את המחברת להריץ ולראות.

Music Genre Classifier - CNN Model (1D , Based on Network from HW)

מסקנות:

1. **שימוש ברשת CNN מסוג VGGish (בחלק של למידת העברה) הניב את התוצאות הכי טובות, גם עבור בסיס נתונים רגיל וגם עבור בסיס הנתונים המורחב.** ניתן לחשוב על הרשת הזו כסוג של "מקודדת" את המאפיינים השונים באודיו לכדי וקטור מאפיינים באורך 128, ולהשתמש במסווג על וקטור מאפיינים זה. (Embedding).
2. **שימוש ברשתות קונבולוציה** צריך לעבור אדפטציה משמעותית לניתוח קטעי אודיו. לעתיד ניתן יהיה לבדוק מה יהיו ביצועים של רשת קונבולוציה על תמונות (של ממש) המייצגות אודיו (כמו ספקטרוגרמה, או ספסטרום).
3. **שימוש ברשת רגילה** נותן ביצועים טובים בארכיטקטורה בעלת רגולריזציה מסוג L1. במעבר חופשי והשמעה אקראית של דגימות מתוך בסיס הנתונים, אני לא בטוח שאדם אקראי היה מצליח לזהות סגנונות באחוזים יותר טובים (למשל, במחברת Data Visualization מוצגים שני קטעי אודיו. אני לא בטוח בכלל שאדם אקראי שישמע את השיר המובא ב Hip Hop Sample לא יחשוב שזו דגימת רוק בכלל..)

נהניתי מאוד מהקורס, התוכן והפרויקט. תודה !

מקורות לVGGish:

1. CNN ARCHITECTURES FOR LARGE-SCALE AUDIO CLASSIFICATION – מצורף להגשה (PDF)
2. https://www.mathworks.com/help/audio/ref/vggishfeatures.html
3. https://colab.research.google.com/drive/1TbX92UL9sYWbdwdGE0rJ9owmezB-Rl1C