Report

רקע

בתרגיל זה התבקשנו לבנות רשת נוירונים שיודעת לסווג כל קובץ wav למילה שנאמרה באותו הקובץ ע"י שימוש בספריות שונות כרצוננו וב- PyTorch framework.

הדאתא שהתקבל חולק מראש לשלושה סטים:

- בסט האימון התקבלו 30,000 תצפיות, כאשר כל תצפית מיוצגת כקובץ wav בה נאמרת מילה כלשהי.
- בסט הולידציה התקבלו 6,789 תצפיות.
- בסט המבחן התקבלו 6,835 דוגמאות חדשות, אשר נרצה לחזות את התיוגים עבורם.

קבצי הקול שהתקבלו בסט האימון והולידציה היו ממוינים ל- 30 קטגוריות שונות, כאשר כל קטגוריה הופרדה בתת תיקייה משלה של קבצי קול שונים שבהם נשמעת שם הקטגוריה שהיא למעשה התיוג. כל קובץ קול הוא באורך של שניה.

בתרגיל זה ניתנה לנו אפשרות לבנות מודל כרצוננו לפי הכלים השונים שלמדנו DNN, CNN וכ'ו.

Pre-Processing

כחלק מטעינת הדאתא קיבלנו קובץ הממיר את כל אחת מקבצי הקול לספקטוגרמה שאומרת איזה תדרים היו משמעותיים בכל נקודת זמן (ציר x מציין זמן וציר y את התדרים). ניתן להתייחס לספקטוגרמה בתור תמונה לכל דבר ולעבוד על רשת CNN חד/דו מימדית או על רשת RNN. בדרך זו הומרו שלושת הסטים של הנתונים, כך שכל הדאתא יהיה מיוצג בפורמט של תמונה. יתר על כן, האתחול של וקטור המשקולות נעשה באמצעות שימוש ב- xavier לשם שיפור ביצועים.

אופן הרצת הקוד וקליטת הנתונים

שלושת התיקיות שהכילו את הסטים של הנתונים: train, valid and test נשמרו יחדיו בתיקייה בשם gcommands. כמו כן, יצרתי תת תיקייה חדשה בתיקייה test שגם נקראה test והעברתי אל תת התיקייה את כלל קבצי הקול שהיו בתיקייה המקורית של test, כך שתהיה התאמה לאופן שמירת הקבצים בתיקיות שהכילו את הסטים האחרים- train & valid.

ההרצה של הקוד דרך הטרמינל מתבצעת לאחר הזנת השורה: python ex_5.py.

לאחר הכנה מקדימה זו יכולתי לבצע קליטה של הנתונים בקוד באופן הבא:

```
datasetOfTrain = GCommandLoader(r'gcommands/train')
train_loader = torch.utils.data.DataLoader(datasetOfTrain, batch_size = 10,
                                           shuffle = True, pin_memory = True)
```

כאשר GCommandLoader היא פונקציה מובנית שקיבלנו בקוד שצורף לתרגיל, ובקליטה של סט המבחן נקבע shuffle = False.

עבור כל סט נתונים שקלטתי בשיטה זו אנו מקבלים tuple המכיל **ארבעה פרמטרים**:

batch, channels, height, width.

המודל שנבחר

כתיבת המודל כללה מספר שלבים: הגדרת המודל והשכבות בו, כתיבת פונקציה לשלב האימון, כתיבת פונקציה לשלב הולידציה וכתיבת פונקציה לשלב המבחן ויצירת קובץ הטקסט הנדרש המציג את הפרדיקציות של המודל.

המודל שבנינו מכיל בחלקו הראשון שכבות של CNN ובחלקו האחרון שכבות Fully Connected.
להלן הסבר על בניית המודל

- החלק של CNN בנוי מ- 8 שכבות קונבולוציה באופן הבא:
Conv-Conv-Pooling-Conv-Conv- Pooling-Conv-Conv- Pooling-Conv-Conv-Pooling.
הוחלט להעמיק את הרשת על מנת לקבל ביצועים יותר המתבטאים באחוזי דיוק גבוהים.
- לאחר כל שכבת קונבולוציה הופעל BatchNorm2d ואחריו הופעלה פונקצית האקטיבציה ReLU.
- לאחר כל שתי שכבות קונבולוציה נוספה שכבת Max Pooling.
- בחלק של Fully Connected קיימות שלוש שכבות בגדלים [15360,100,50,30], אשר בין כל שתי שכבות הופעל BatchNorm1d ולאחריו הופעלה פונקצית האקטיבציה ReLU.

קביעת היפר פרמטרים ובחירות נוספות עבור המודל

1) **שכבות הרשת**-

Conv-Conv-Pooling-Conv-Conv- Pooling-Conv-Conv- Pooling-Conv-Conv-Pooling-Fully Connected-Fully Connected-Fully Connected.

Layers_size for Fully connected part = [15360,100,50,30]

כאשר השכבה האחרונה בגודל של כמות הסיווגים האפשריים.

2) **batch_size** שנבחר עבור המודל הוא 10 תצפיות עליהם נעבור כל פעם.

3) **Channels**- התחלנו עם ערוץ בגודל אחד בשכבה הראשונה של הקונבולוציה ובפלט של אותה השכבה קיבלנו שהערוץ גדל פי 2. את גודל הערוץ שקיבלנו לקחנו כמספר הערוצים עבור הקלט של שכבת הקונבולוציה הבאה וכן הלאה ליתר שכבות הקונבולוציה.

4) **הפילטרים** נקבעו להיות בגודל 3*3 עבור כלל שכבות הקונבולוציה, ואילו בכלל שכבות ה- Max Pooling הפילטרים היו בגודל 2*2.

5) **Pooling**- בחרנו להשתמש ב- Max Pooling שהוא נחשב לאחד מסוגי ה- Pooling השכיחים.

6) **Stride**- נקבע להיות 1 עבור כלל שכבות הקונבולוציה, ואילו בכלל שכבות ה- Max Pooling נקבע לו הערך 2.

7) **Padding**- נקבע להיות הערך 1 עבור כלל שכבות הקונבולוציה.

8) **מספר האפוקים**- נקבע להיות 10. לאחר 10 אפוקים ניתן לראות התכנסות עם תוצאות של 88-93 אחוזי דיוק על ה- validation set עבור קצב הלמידה שנקבע.

9) **פונקציות האקטיבציה**- בחרנו להשתמש בפונקצית ReLU אשר הציגה ביצועים טובים יותר מאשר פונקצית Sigmoid עבור המודל שנבחר.

10) **Learning rate**- ע״י מעבר על הערכים המקובלים [0.1, 0.01, 0.001] עבור המודל, נבחר ערך קצב הלמידה 0.001 שנתן אחוזי דיוק יותר גבוהים ופונקציית loss יותר נמוכה. בנוסף עשינו fine tuning לפי התוצאות שהתקבלו על ה- train & validation.

11) **Optimizers**- שלב האופטימיזציה בו נעשה עדכון לפרמטרים של המודל תחילה נעשה ע״י שימוש ב- SGD optimizer, אך לאחר מספר בדיקות שנעשו התברר ש- Adam optimizer הביא לביצועים טובים בהרבה משאר האפשריות.

12) **Loss function**- בחרנו להשתמש בפונקציית nll_loss.