Ben-Gurion University of the Negev

Faculty of Engineering Sciences

Department of Software and Information systems Engineering

Deep Learning
Assignment 3

The purpose of the assignment

Enabling students to experiment with building a recurrent neural net and using it on a real-world dataset. In addition to practical knowledge in the "how to" of building the network, an additional goal is introducing the students to the challenge of integrating different sources of information into a single framework.

Submission instructions:

a) The assignment due date: 14.01.2026
b) The assignment is to be carried out using PyTorch.
c) Submission in **pairs** only. Only **one copy** of the assignment needs to be uploaded to Moodle.
d) Plagiarism of any kind (e.g., GitHub) is forbidden.
e) The entire project will be submitted as a single zip file, containing both the report (in PDF form only) and the code. It is the students' responsibility to make sure that the file is valid (i.e. can be opened correctly).
f) The report accompanying the code needs to be a **.docx file** (use Calibri font, text size 12, margins of 2.5cm). The report itself will not exceed **6 pages** of text. Figures (graphs, tables, etc) can be placed an appendix, and must be referenced in the main body of report.

Introduction

In class we covered the topic of automatic sentence completion/generation. In addition to the completion of "regular" sentences, this technique can also be applied to other domains such as lyrics and melodies generation. In this task you will train a neural net to generate lyrics based on the provided melody.

During the training of the model you will have access both to the lyrics of a song and its melody. The melodies are stored in .mid (MIDI files) and contain various types of information – notes, the instruments used etc. You are encouraged to experiment with various methods to incorporate this information with the lyrics. During the test phase, you are required to automatically generate lyrics for a provided melody.

Please note that this assignment cannot be measured using objective (i.e., absolute) performance measures. Instead, we will be evaluating your approach to the solution, the implementation, and your analysis of your model's performance.

Instructions

1. Please download the following:
    a. A .zip file containing all the MIDI files of the participating songs
    b. the .csv file with all the lyrics of the participating songs (600 train and 5 test)
    c. Pretty_Midi, a python library for the analysis of MIDI files
2. Implement a recurrent neural net (LSTM or GRU) to carry out the task described in the introduction.
    a. During each step of the training phase, your architecture will receive as input one word of the lyrics. Words are to be represented using the Word2Vec representation that can be found online (300 entries per term, as learned in class).
    b. The task of the network is to predict the next word of the song's lyrics. Please see Figure 1 for an illustration. You may use any loss function
    c. In addition to this textual information, you need to include information extracted from the MIDI file. The method for implementing this requirement is entirely up to your consideration. Figure 1 shows one of the more simplistic options – inserting the entire melody representation at each step.
    d. Note that your mechanism for selecting the next word should not be deterministic (i.e., always select the word with the highest probability) but rather be sampling-based. The likelihood of a term to be selected by the sampling should be proportional to its probability.
    e. You may add whatever additions you want to the architecture (e.g., regularization, attention, teacher forcing)
    f. You may create a validation set. The manner of splitting (and all related decisions) are up to you.
    g. You need to set "guidelines" (either fixed rules, preferably through the loss function) regarding the length of the generated text, the maximal number of words per line, etc. The goal is to make your lyrics look like those of an actual song.

3. The Pretty_Midi package offers multiple options for analyzing .mid files. Figures 2-4 demonstrate the types of information that can be gathered.
4. You can add whatever other information you consider relevant to further improve the performance of your model.

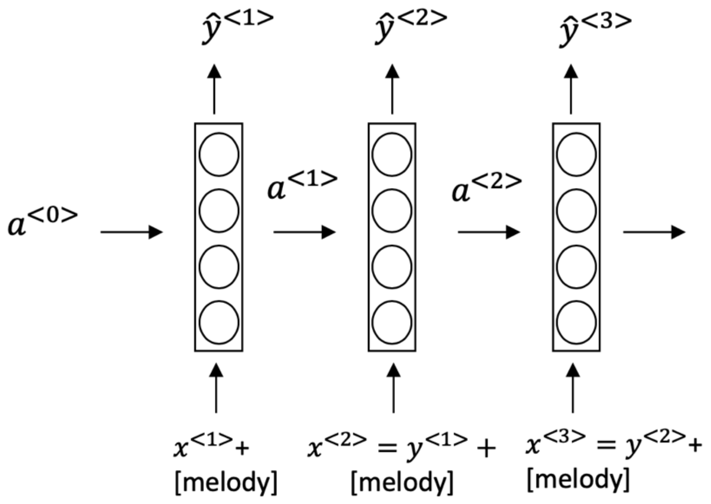
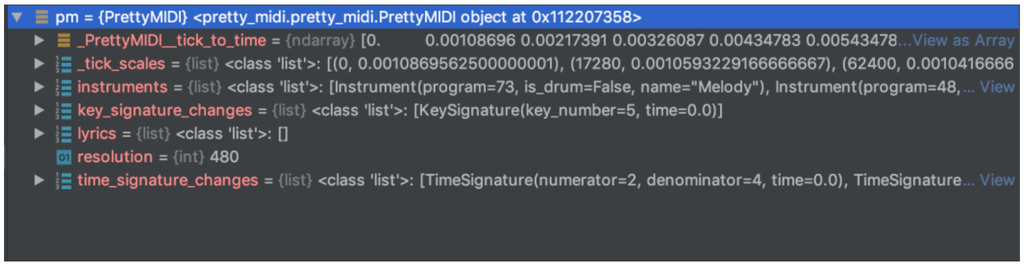


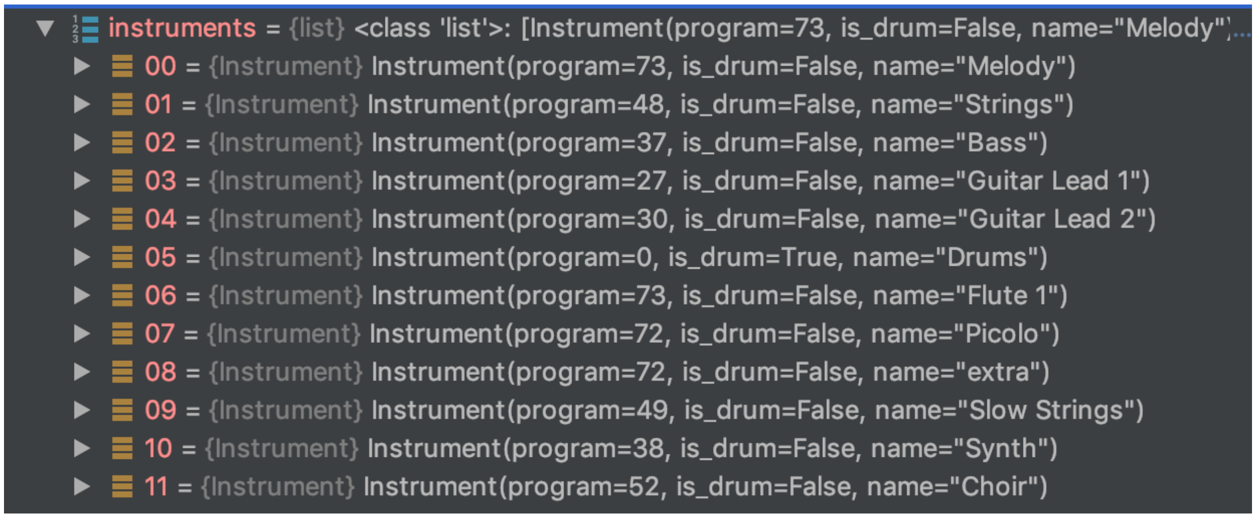
Figure 1: an example of a simplistic way of combining the melody and lyrics

5. You are to evaluate two approaches for integrating the melody information into your model. The two approaches don't have to be completely different (one can build upon the other, for example), but please refrain from making only miniature changes.



```
▼ pm = {PrettyMIDI} <pretty_midi.pretty_midi.PrettyMIDI object at 0x112207358>
  ▶ _PrettyMIDI__tick_to_time = {ndarray} [0.        0.00108696 0.00217391 0.00326087 0.00434783 0.00543478...View as Array
  ▶ _tick_scales = {list} <class 'list'>: [(0, 0.0010869562500000001), (17280, 0.0010593229166666667), (62400, 0.0010416666
  ▶ instruments = {list} <class 'list'>: [Instrument(program=73, is_drum=False, name="Melody"), Instrument(program=48,... View
  ▶ key_signature_changes = {list} <class 'list'>: [KeySignature(key_number=5, time=0.0)]
  ▶ lyrics = {list} <class 'list'>: []
    resolution = {int} 480
  ▶ time_signature_changes = {list} <class 'list'>: [TimeSignature(numerator=2, denominator=4, time=0.0), TimeSignature... View
```

Figure 2: general .mid file information



```
▼ instruments = {list} <class 'list'>: [Instrument(program=73, is_drum=False, name="Melody")...
   ▶ 00 = {Instrument} Instrument(program=73, is_drum=False, name="Melody")
   ▶ 01 = {Instrument} Instrument(program=48, is_drum=False, name="Strings")
   ▶ 02 = {Instrument} Instrument(program=37, is_drum=False, name="Bass")
   ▶ 03 = {Instrument} Instrument(program=27, is_drum=False, name="Guitar Lead 1")
   ▶ 04 = {Instrument} Instrument(program=30, is_drum=False, name="Guitar Lead 2")
   ▶ 05 = {Instrument} Instrument(program=0, is_drum=True, name="Drums")
   ▶ 06 = {Instrument} Instrument(program=73, is_drum=False, name="Flute 1")
   ▶ 07 = {Instrument} Instrument(program=72, is_drum=False, name="Picolo")
   ▶ 08 = {Instrument} Instrument(program=72, is_drum=False, name="extra")
   ▶ 09 = {Instrument} Instrument(program=49, is_drum=False, name="Slow Strings")
   ▶ 10 = {Instrument} Instrument(program=38, is_drum=False, name="Synth")
   ▶ 11 = {Instrument} Instrument(program=52, is_drum=False, name="Choir")
```

Figure 3: Instrument information of the analyzed file

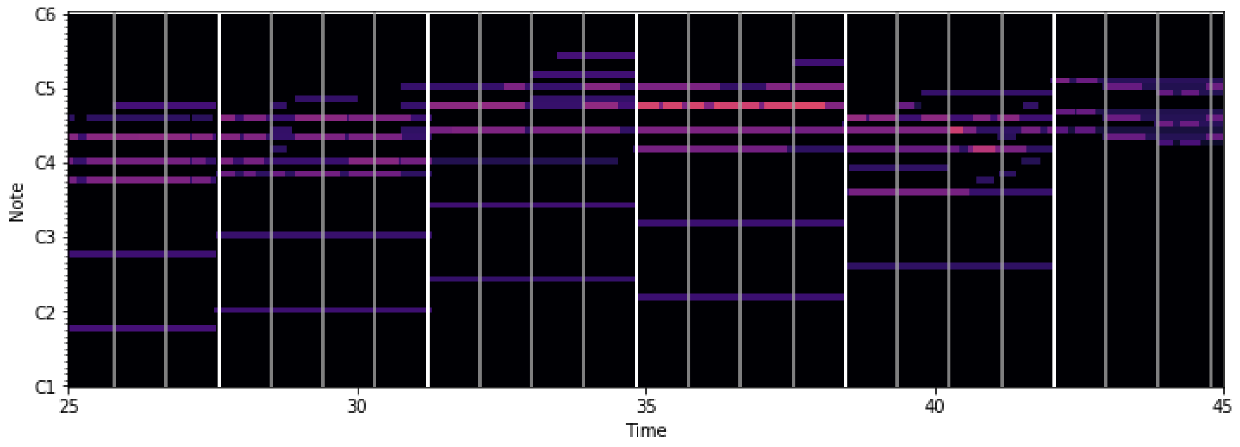


Figure 4: timing information of the analyzed file

6. Please include the following information in your report regarding the training phase:
   a. The chosen architecture of your model
   b. A clear description of your approach(s) for integrating the melody information together with the lyrics
   c. TensorBoard graphs showing the training and validation loss of your model.
7. Please include the following information in your report regarding the test phase:
   a. For each of the melodies in the test set, produce the outputs (lyrics) for each of the two architectural variants you developed. The input should be the melody and the initial word of the output lyrics. Include all generated lyrics in your submission.
   b. For each melody, repeat the process described above three times, with different words (the same words should be used for all melodies).

c. Attempt to analyze the effect of the selection of the first word and/or melody on the generated lyrics.

Good Luck!