**tonnenz\_dn\_v2.py**

Данный скрипт представляет собой программу, которая обучает нейронную сеть на основе архитектуры LSTM и автоэнкодера для работы с музыкальными данными в формате тоннетц. Для работы программе необходимо указать некоторые параметры и файлы в качестве входных данных.

Программа принимает следующие аргументы командной строки:

* 'vocab': путь к файлу с сохраненным словарём (pickle файл) для автоэнкодера в формате тоннетц.
* 'train': путь к файлу с обучающими данными (pickle файл) для LSTM.
* '--valid': опциональный путь к файлу с данными для валидации.
* '--test': опциональный путь к файлу с данными для тестирования.
* '--CNNepoch': опциональное количество эпох обучения для автоэнкодера с использованием сверточных слоёв.
* '--LSTMepoch': опциональное количество эпох обучения для LSTM.

Затем программа загружает необходимые параметры и данные из указанных файлов, и начинает обучение модели.

Автоэнкодер используется для предварительного обучения данных тоннетц. В этой части кода задаётся структура автоэнкодера, состоящего из трёх скрытых слоёв и одного выходного слоя. Для каждого слоя определены веса и смещения, а также функции активации (в данном случае используется ReLU). Автоэнкодер обучается на обучающих данных путём минимизации ошибки между входными и выходными данными с помощью оптимизатора Adam.

После завершения обучения автоэнкодера, программа переходит к обучению модели LSTM. LSTM используется для последовательного анализа данных тоннетц и предсказания классов. Структура LSTM включает в себя несколько слоёв LSTM, инициализацию состояния, определение входных и выходных данных, а также веса и смещения. Модель LSTM обучается на обучающих данных с использованием оптимизатора Adam и минимизации кросс-энтропийной потери между предсказанными и истинными классами.

Обучение происходит в течение заданного числа эпох. После каждой эпохи происходит проверка на валидационных данных, и выводятся значения потери и точности модели. По окончании обучения модель сохраняется для последующего использования.

**tonnenz\_generate\_v2.py**

Данный скрипт выполняет генерацию музыкальных последовательностей с использованием предварительно обученной модели. Вот более подробное описание его работы:

1. Программа начинает с импорта необходимых библиотек и объявления аргументов командной строки с помощью модуля argparse.
2. Затем программа загружает словарь и файл тестовых данных из указанных файлов.
3. После этого задаются параметры для моделей LSTM и автоэнкодера, включая размеры пакета, количество шагов, количество классов и размер состояния LSTM.
4. Далее определяются функции для генерации пакетов данных для обучения LSTM и автоэнкодера. Функции gen\_epochs\_cnn и gen\_batch\_cnn используются для генерации пакетов данных для обучения автоэнкодера. Функции gen\_epochs и gen\_batch используются для генерации пакетов данных для обучения LSTM.
5. Затем программа определяет функцию generate\_characters, которая выполняет генерацию музыкальных последовательностей. Внутри этой функции восстанавливаются сохранённые переменные и граф модели из файла чекпоинта.
6. Далее происходит цикл генерации последовательностей. В каждой итерации цикла данные передаются в модель LSTM, и производится генерация последовательности предсказаний. Все предсказания собираются в выходной список.
7. После завершения генерации последовательностей, список предсказаний сохраняется в текстовый файл и в pickle файл.
8. Затем программа преобразует предсказанные данные в формате тоннетц в формат нот. Для каждого фрейма предсказаний, программа анализирует вероятности каждой ноты в заданном диапазоне и определяет, какие ноты будут включены в итоговую последовательность на основе заданной политики и порогового значения.
9. В результате, программа сохраняет сгенерированные последовательности в текстовом и pickle файлах.
10. Наконец, программа вызывает функцию generate\_characters с указанием пути к сохранённой модели LSTM для генерации музыкальных последовательностей.

Функция tonnetz2note\_batch используется вместо tonnetz2note для обработки пакета фреймов предсказаний, а не только отдельного фрейма.

Функция tonnetz2note принимает один фрейм предсказаний и возвращает список нот, основываясь на заданной политике и пороговом значении. Она выполняет эту операцию для одного фрейма и возвращает результат.

С другой стороны, функция tonnetz2note\_batch принимает пакет фреймов предсказаний и выполняет операцию tonnetz2note для каждого фрейма в пакете. Затем она собирает результаты для каждого фрейма в список и возвращает этот список. Это позволяет обрабатывать несколько фреймов одновременно и получать список нот для каждого из них.

Таким образом, функция tonnetz2note\_batch используется, когда требуется обработать пакет фреймов предсказаний, в то время как функция tonnetz2note используется для обработки только одного фрейма.

**Расчёт и сборка датасета**

Данный код выполняет следующие действия:

1. Определяет шаблон templ, который представляет собой матрицу тоннетц из статьи. Этот шаблон используется в функции compute\_tonnetz для вычисления тоннетц-вектора.
2. Импортирует необходимые библиотеки, такие как numpy, os, pickle и pretty\_midi.
3. Определяет функцию compute\_tonnetz, которая принимает значение pitch и матрицу pitch\_matrix. Функция вычисляет тоннетц-вектор на основе заданного pitch и pitch\_matrix.
4. Определяет функцию process\_midi\_file, которая принимает путь к файлу MIDI и матрицу pitch\_matrix. Функция считывает файл MIDI, извлекает информацию о нотах, разбивает её на интервалы времени и вычисляет тоннетц-векторы для каждого интервала времени на основе pitch\_matrix.
5. Задает pitch\_matrix с использованием значения templ.
6. Определяет путь к папке, содержащей файлы MIDI.
7. Получает список файлов MIDI в указанной папке.
8. Разделяет список файлов на тренировочный, валидационный и тестовый наборы данных.
9. Обрабатывает каждый файл MIDI в тренировочном наборе, вызывая функцию process\_midi\_file для каждого файла, и сохраняет полученные данные в тренировочный набор данных train\_dataset и словарь уникальных паттернов train\_vocab.
10. Аналогично обрабатывает файлы MIDI в валидационном и тестовом наборах данных, сохраняя полученные данные в соответствующие переменные.
11. Сохраняет тренировочный набор данных и словарь в отдельные файлы с использованием модуля pickle.
12. Аналогично сохраняет валидационный и тестовый наборы данных и словари в отдельные файлы.

· **compute\_tonnetz(pitch, pitch\_matrix)**: Эта функция используется для вычисления тоннетц-вектора на основе заданного значения высоты тона и матрицы высоты тона. Входные параметры функции:

· pitch: значение высоты тона, для которого нужно вычислить тоннетц-вектор.

* pitch\_matrix: матрица высоты тона, в которой каждый элемент соответствует конкретному интервалу и конкретной ноте.

Функция начинает с создания нулевого вектора tonnetz размером 288. Затем она проходит по каждому элементу матрицы pitch\_matrix и, если значение в матрице соответствует заданному pitch, устанавливает соответствующий элемент вектора tonnetz равным 1. Затем функция применяет пороговое значение, заменяя все значения вектора tonnetz, превышающие 1, на 1. В итоге функция возвращает вычисленный тоннетц-вектор.

· **process\_midi\_file(file\_path, pitch\_matrix)**: Эта функция обрабатывает файл MIDI и вычисляет тоннетц-векторы для каждого интервала времени на основе заданной матрицы высоты тона. Входные параметры функции:

* · file\_path: путь к файлу MIDI, который нужно обработать.
* pitch\_matrix: матрица высоты тона, используемая для вычисления тоннетц-векторов.

Функция начинает с чтения данных из файла MIDI с использованием библиотеки pretty\_midi. Затем она инициализирует пустые списки dataset и unique\_patterns. Далее функция проходит по каждой ноте в первом инструменте файла MIDI. Для каждой ноты она получает высоту тона, начальное и конечное время, а также длительность интервалов времени. Затем функция проходит по каждому интервалу времени и извлекает ноты, находящиеся в этом интервале. Если в интервале есть ноты, функция вызывает compute\_tonnetz для вычисления тоннетц-вектора для этого набора нот. Полученные тоннетц-векторы добавляются в список dataset, а уникальные тоннетц-векторы добавляются в список unique\_patterns. В конце функция возвращает dataset (список тоннетц-векторов для каждого интервала времени) и unique\_patterns (список уникальных тоннетц-векторов, используемых в файле MIDI).

· **train\_test\_split(files\_list)**: Эта функция используется для разделения списка файлов MIDI на тренировочный, валидационный и тестовый наборы данных. Входной параметр функции:

· files\_list: список файлов MIDI, которые нужно разделить на наборы данных.

Функция использует метод train\_test\_split из модуля sklearn.model\_selection, который случайным образом разделяет список файлов на тренировочный и тестовый наборы. Далее она вызывает этот метод с входным списком файлов и задает размеры тестового и валидационного наборов данных (40% и 50% соответственно). Результат разделения возвращается как трехэлементный кортеж, содержащий тренировочный набор, валидационный набор и тестовый набор файлов.

Таким образом, данный код предназначен для обработки файлов MIDI, вычисления тоннетц-векторов и сохранения обработанных данных в файлы для последующего использования.

Функция load\_predictions():

* 1. Загружает предсказания из файла Testpred\_majority0.3.pickle.
  2. Открывает файл в двоичном режиме чтения с помощью open() и использует pickle.load() для загрузки данных из файла.
  3. Возвращает загруженные предсказания.

Функция remove\_repeated\_lists(predictions):

* 1. Принимает список предсказаний predictions в качестве входного параметра.
  2. Создаёт пустой список unique\_predictions, который будет содержать уникальные предсказания.
  3. Использует переменную count для подсчета повторяющихся предсказаний.
  4. Проходит по списку predictions и проверяет, является ли текущее предсказание одинаковым с предыдущим предсказанием. Если да, то увеличивает счетчик count.
  5. Если count меньше 2, то добавляет текущее предсказание в unique\_predictions, чтобы избежать повторов.
  6. Если предсказание отличается от предыдущего или count достигает значения 2, то сбрасывает счетчик count и добавляет текущее предсказание в unique\_predictions.
  7. Возвращает список уникальных предсказаний unique\_predictions.

Функция tonnetz\_to\_midi(tonnetz\_predictions):

* 1. Принимает список предсказаний tonnetz\_predictions в качестве входного параметра.
  2. Создаёт объект midi\_file класса mido.MidiFile, представляющий MIDI-файл.
  3. Создаёт трек track класса mido.MidiTrack и добавляет его в midi\_file.
  4. Проходит по временным шагам и аккордам в tonnetz\_predictions.
  5. Для каждого временного шага и аккорда:
     1. Вычисляет время в тиках на основе текущего временного шага и параметров времени.
     2. Если аккорд содержит ноты, то для каждой ноты:
        1. Создаёт сообщение "note\_on" с указанием высоты тона, скорости и времени начала и добавляет его в трек.
        2. Создаёт сообщение "note\_off" с указанием высоты тона, скорости и времени окончания и добавляет его в трек.
     3. Если аккорд пустой (отсутствуют ноты), то создаёт паузу с сообщением "note\_off" для нулевой высоты тона и добавляет её в трек.
  6. Возвращает созданный объект midi\_file.

Функция process\_midi\_with\_fluidsynth(midi\_path, soundfont\_path, output\_path):

* 1. Принимает пути к MIDI-файлу (midi\_path), SoundFont файлу (soundfont\_path) и пути для сохранения выходного аудиофайла (output\_path).
  2. Создаёт команду cmd, используя путь к программе fluidsynth и переданные пути к файлам.
  3. Запускает команду с помощью subprocess.run(), выполняя преобразование MIDI-файла в аудиофайл формата WAV с использованием SoundFont файла.
  4. Результат сохраняется по указанному пути output\_path.

Код после определения функций:

* 1. Загружает предсказания с помощью load\_predictions() и сохраняет их в переменную tonnetz\_predictions.
  2. Удаляет повторяющиеся списки из предсказаний с помощью remove\_repeated\_lists(). Уникальные предсказания сохраняются в переменную tonnetz\_predictions\_unique.
  3. Создаёт MIDI-файл с помощью tonnetz\_to\_midi() на основе уникальных предсказаний и сохраняет его по указанному пути output\_midi\_path.
  4. Указывает путь к SoundFont файлу в переменной soundfont\_path.