Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия
« _____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Генерация музыки при помощи машинного обучения

Руководитель _____ ст. преп. каф. ВТ И. В. Матковский
подпись, дата

Выпускник _____ Н. В. Белый
подпись, дата

Нормоконтролер _____ И. В. Матковский
подпись, дата

Красноярск 2021

РЕФЕРАТ

Выпускная квалификационная работа по теме «Генерация музыки при помощи машинного обучения» содержит 20 страниц текстового документа, 11 изображений, 3 формул, 5 использованных источников.

АВТОЭНКОДЕР, НЕЙРОСЕТЬ, MIDI, PYTHON, ЦЕПИ МАРКОВА, НЕЙРОН, СИНАПС.

Цель – создание приложения для генерации музыки.

Задачи:

- изучение предметной области;
- рассмотрение аналогов;
- выбор средств разработки;
- рассмотрение методов реализации;
- разработка приложения.

Были рассмотрены аналоги, а также выделены основные особенности алгоритмов генерации музыки. В результате выполнения работы было получена нейросеть и приложение для генерации музыки. Мелодия генерируется в формате MIDI или WAW. Также добавлены инструменты для редактирования мелодии в реальном времени.

Данная программа предоставляет простой и быстрый доступ к генерации мелодий. Описание данной работы демонстрирует возможности генерации музыки, что может помочь при разработке похожих приложений.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

По мере того, как всё больше людей проводят время дома, создание, прослушивание и использование музыки в различных проектах приобретает большую значимость в их жизнях. Первые успехи в создании, производстве и редактировании музыки с помощью искусственного интеллекта ошеломляют и ещё сильнее ускорят эту тенденцию.

Автоматическая генерация музыки — довольно сложная задача по многим причинам. Больше всего мешает то, что простая трехминутная песня, которую группа людей может легко запомнить, содержит слишком много переменных для компьютера. Кроме того, пока не существуют идеального способа, как обучить искусственный интеллект быть музыкантом.

Весь звук внутри компьютера является цифровым сигналом – сигналом, который можно представить в виде последовательности цифровых значений. Благодаря этому звук можно создавать, либо видоизменять при помощи программного обеспечения. Процесс создания звука на программном обеспечении называется звуковым синтезом.

В современном мире музыка используется в самых различных отраслях. Почти любая музыка имеет синтезированные звуки, каждый фильм, видеоигра, реклама, даже техника – всё это имеет звук, благодаря чему различные бренды приобретают уникальность. Такой генератор может очень сильно упростить работу композиторов.

Данная работа посвящена разработке генератора музыки. На основании этого были поставлены следующие задачи выпускной квалификационной работы:

1. Построение алгоритма генерации музыки;
2. Преобразование обучающих данных в приемлемую для программы структуру для дальнейшего обучения;
3. Создание генератора и проигрывателя с настройкой композиции.

1 Аналитическая часть

1.1 Аналог

Больших, коммерческих проектов на данный момент не так уж и много, однако стоит выделить проект AIVA, генерирующий классическую и симфоническую музыку в MIDI формате на основе огромного количества обучающего материала. Искусственный автор в состоянии создавать собственные паттерны музыки и предсказывать, как будет развиваться музыкальная тема в том или ином паттерне дальше. Причем AIVA якобы может безошибочно вычищать плагиат из собственных сочинений и делать развитие своих музыкальных тем эмоциональным (На деле это не совсем так). Из недостатков стоит выделить цену данного продукта, его не завершенность, а также однообразность генерируемой музыки.

Принимая во внимание недостатки данного проекта, пробуем разработать программу, лишенную этих проблем.

1.2 Особенности задачи и необходимые компоненты

В современном мире существует очень много форматов для записи и генерации музыкальных произведений. Лучшим решением для генератора является формат MIDI:

MIDI, расшифровывается как Musical Instrument Digital Interface, – это технический стандарт, описывающий протокол связи, цифровой интерфейс и электрические разъемы, которые соединяют широкий спектр электронных музыкальных инструментов, компьютеров и связанных аудиоустройств для воспроизведения, редактирования и записи музыки.

Такой формат как нельзя лучше подходит для данного проекта, т.к. звук уже оцифрован и его сразу же можно использовать в программе.

По результатам изучения источников можно выделить следующие наиболее распространенные методы для генерации музыки:

- Цепи Маркова
- Нейросети

Далее кратко разберем методы и выберем один из них за основу нашей программы.

1.3 Цепи Маркова

Цепь Маркова описывает псевдослучайный процесс перехода из одного состояния в другое без запоминания предыдущего состояния (такой переход называется «марковостью»). Короче говоря, переход из одного состояние в другое – это случайный процесс, носящий вероятностный характер. Все это хорошо ложится на алгоритмизацию музыкальных композиций.

Ноты – это вероятностные состояния. Для реализации всего процесса будет использоваться цепь Маркова второго порядка, а это означает, что следующее состояние системы будет строиться на основании двух предыдущих (нот). Все вероятности хранятся в матрице.

На входе синтезатор получает два целых числа выступающих в качестве начальных нот. На их основании алгоритм вычисляет/генерирует следующую ноту и продолжает процесс вычисления до бесконечности (до тех пор, пока вы его не остановите).

Для корректной работы такого подхода необходима матрица весов, которую получают с использованием трех нот. Для каждой комбинации из трех нот первые две всегда являются «начальным» состоянием, а третья «конечным»; результатом всегда является инкрементация соответствующего поля в матрице весов.

Далее матрица весов «нормализуется» (или конвертируется) в матрицу

переходов путем замены целых чисел на их процентное отношение к сумме всех значений в строке.

1.4 Нейросети

Для данного проекта, исходя из анализа источников, наиболее подходящей нейросетью является автоэнкодер.

Автоэнкодеры — это нейронные сети прямого распространения, которые восстанавливают входной сигнал на выходе. Внутри у них имеется скрытый слой, который представляет собой код, описывающий модель. Автоэнкодеры конструируются таким образом, чтобы не иметь возможность точно скопировать вход на выходе. Обычно их ограничивают в размерности кода (он меньше, чем размерность сигнала) или штрафуют за активации в коде. Входной сигнал восстанавливается с ошибками из-за потерь при кодировании, но, чтобы их минимизировать, сеть вынуждена учиться отбирать наиболее важные признаки.

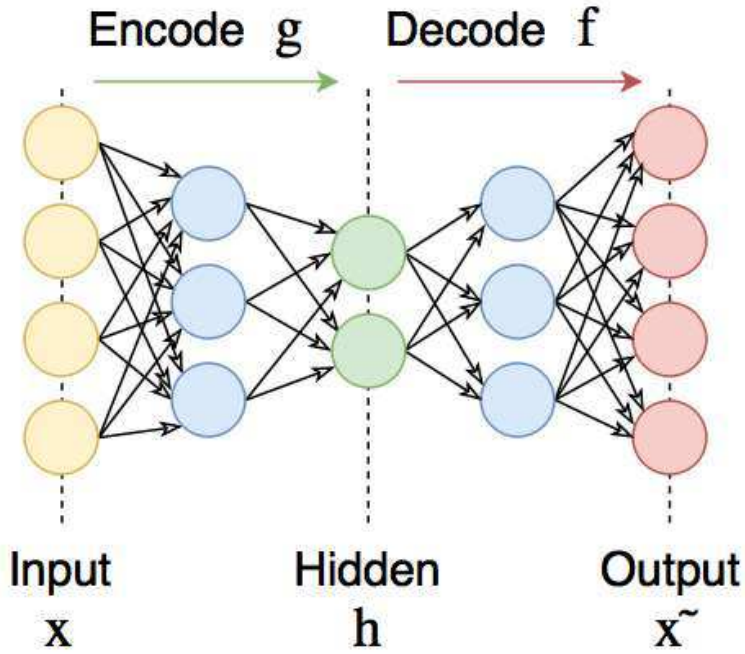

Рисунок 1 – Схема автоэнкодера

Автоэнкодеры состоят из двух частей: энкодера g и декодера f. Энкодер переводит входной сигнал в его представление (код):

$$h = g(x) , \quad (1)$$

а декодер восстанавливает сигнал по его коду:

$$x = f(h) . \quad (2)$$

Автоэнкодер, изменяя g и f, стремится выучить тождественную функцию

$$x = f(g(x)) , \quad (3)$$

минимизируя какой-то функционал ошибки. При этом семейства функций энкодера g и декодера f как-то ограничены, чтобы автоэнкодер был вынужден отбирать наиболее важные свойства сигнала.

Сама по себе способность автоэнкодеров сжимать данные используется редко, так как обычно они работают хуже, чем вручную написанные алгоритмы для конкретных типов данных вроде звуков или изображений. А также для них критически важно, чтобы данные принадлежали той генеральной совокупности, на которой сеть обучалась. Обучив автоэнкодер на цифрах, его нельзя применять для кодирования чего-то другого (например, человеческих лиц).

Является наиболее подходящей нейронной сетью т.к. выделяются лишь главные параметры. Это очень важно, ведь, как выше было сказано, музыка имеет очень много параметров, а при помощи такой нейросети возможно сократить число параметров, пусть и взамен некоторой потери качества.

1.5 Вывод

Для данного проекта оба метода являются приемлемыми, однако, выберем нейронную сеть ввиду большей гибкости.

2 Описание выбранного метода

Под нейронными сетями принято понимать вычислительные системы, имеющие способности к самообучению, постепенному повышению своей производительности. Основными элементами структуры нейронной сети являются:

- Искусственные нейроны, представляющие собой элементарные, связанные между собой единицы.
- Синапс – это соединение, которые используется для отправки-получения информации между нейронами.
- Сигнал – собственно информация, подлежащая передаче.

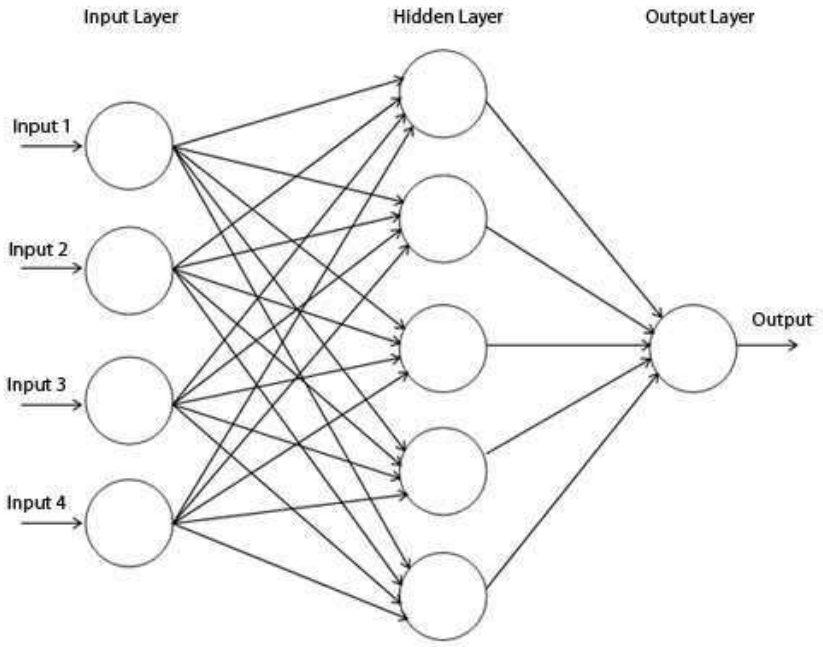

Рисунок 2 – Схема простой нейронной сети

Нейросети на данный момент используются в следующих областях:

- Машинное обучение, представляющее собой разновидность искусственного интеллекта. В основе его лежит обучение ИИ на примере миллионов однотипных задач. В наше время машинное обучение активно

внедряют поисковые системы Гугл, Яндекс, Бинг, Байду. Так на основе миллионов поисковых запросов, которые каждый день вводятся в Гугле, данные алгоритмы учатся показывать наиболее релевантную выдачу, чтобы найти именно то, что нужно.

- В робототехнике нейронные сети используются в выработке многочисленных алгоритмов для железных «мозгов» роботов.
- Архитекторы компьютерных систем пользуются нейронными сетями для решения проблемы параллельных вычислений.

Нейроны — это вычислительная единица, которая обрабатывает поступающую информацию и производит над ней вычисления и передает следующим нейронам.

Нейроны делят на 3 основных типа: входные нейроны, скрытие нейроны и выходные нейроны. Объединение большого количество нейронов называют слоем.

У нейронов также присутствуют параметры: входные и выходные. У входного нейрона нет разницы между входными и выходными параметрами, когда в остальных вся информация из выходных данных предыдущего слоя передается на вход следующему. После передачи информации информация нормализуется (срабатывает функция активации нейрона) и через выходной параметр отправляется к следующему слою. Из этого мы делаем вывод, есть входной слой, на который поступает информация, выходной слой, выдающий результаты работы, и между ними скрытие слои, которые обрабатывают и модифицируют информацию для вывода.

Синапсы – это связь двух нейронов. Синапс имеет только 1 параметр – вес. Допустим, 5 нейронов соединены со следующим нейроном. И для каждого нейрона есть синапс с определенным весом. В зависимости от «тяжести» веса синапса данные передаваемые им будут важнее чем остальные.

Далее рассмотрим необходимые слои для данной нейронной сети

2.1 Входной слой

Входные данные в данном случае — это музыка в формате MIDI. Если мелодия будет большого размера вычислительная сложность увеличится, т.е. ответ от нейронной сети будет медленнее, но если она будет небольшой, то нейросети будет сложно найти ключевые параметры.

2.2 Полносвязный слой

Нейроны в полносвязном слое связаны со всеми активационными нейронами предыдущего слоя. Основной задачей полносвязного слоя является моделирование сложной нелинейной функции, используемой в данном случае для генерации.

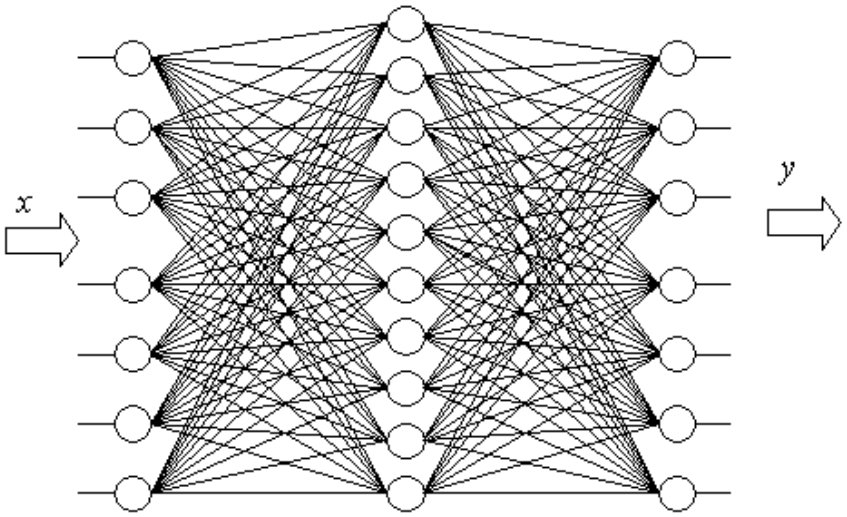

Рисунок 3 – Полносвязные слои нейронной сети

2.3 Выходной слой

Все нейроны последнего слоя связаны с выходным слоем.

Число этих нейронов равно количеству генерируемых классов (в данном случае - 1).

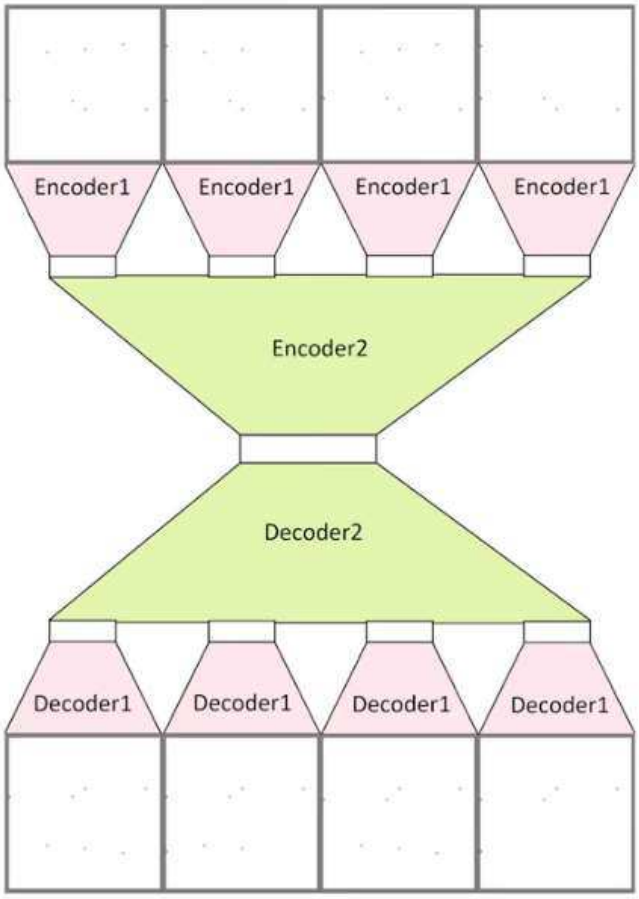

Рисунок 4 – Примерная схема нейросети «автоэнкодер»

2.4 Обучение нейросети

Методом обучения в данной работе был метод градиентного спуска.

Градиентный спуск — метод нахождения минимального значения функции потерь (существует множество видов этой функции). Минимизация любой функции означает поиск самой глубокой впадины в этой функции.

Функция используется, чтобы контролировать ошибку в прогнозах модели машинного обучения. Поиск минимума означает получение наименьшей возможной ошибки или повышение точности модели. Мы увеличиваем точность, перебирая набор учебных данных при настройке параметров нашей модели (весов и смещений).

Итак, градиентный спуск нужен для минимизации функции потерь.

Суть алгоритма – процесс получения наименьшего значения ошибки.

В дальнейшем, чтобы найти самую низкую ошибку в функции потерь (по отношению к одному весу), нужно настроить параметры модели. Как их настроить? В этом поможет математический анализ. Благодаря анализу мы знаем, что наклон графика функции – производная от функции по переменной. Это наклон всегда указывает на ближайший минимум

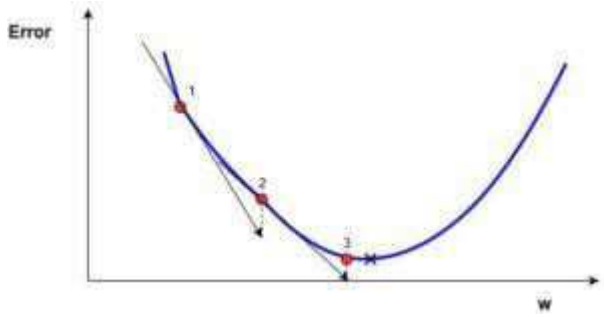

Рисунок 5 – Интерфейс проигрывателя

На рисунке 5 изображен график функции потерь с одним весом. Теперь, если вычислить наклон функции потерь относительно одного веса, то получается направление, в котором нужно двигаться, чтобы достичь локальных минимумов.

3 Этапы проектирования

Конечный продукт должен иметь следующие функции:

- Разбор MIDI – файлов на ноты и длины
- Обучение нейросети
- Генерация музыки
- Обработка и сохранение музыки

Функция генерации музыки будет реализована нейронной сетью. Самая популярная библиотека для создания нейросетей - TensorFlow

3.1 TensorFlow

TensorFlow — открытая программная библиотека для машинного обучения, созданная Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

TensorFlow пользуется концепцией графического представления вычислительных задач. Подобных расклад разрешает пользователям определять математические операции в качестве составляющих графов данных, переменных и операторов. Поэтому нейронные сети, по сути, и считаются графами данных и математических операций, TensorFlow очень хорошо подходит для машинного обучения.

3.2 Выбор язык программирования

Для работы был выбран язык программирования Python, так как основная API для работы TensorFlow реализована именно для Python.

3.3 Python

Python является одним из самых популярных языков программирования сегодня - и на то есть веские причины. Он бесплатный, с открытым исходным кодом, надежный и безопасный, но при этом обеспечивает легкую масштабируемость и гибкость при необходимости. Он также поддерживается большим и полезным онлайн-сообществом.

Python также является одним из лучших языков программирования для проектов машинного обучения и искусственного интеллекта. Простой синтаксис Python позволяет быстро разрабатывать проекты по искусственному интеллекту, не тратя время и силы на изучение более сложного языка программирования.

По сравнению с другими языками, Python в 5-10 раз быстрее по времени разработки, однако медленный при выполнении программ. Он обеспечивает расширенные возможности управления процессами и объектно-ориентированный дизайн, помогая как в скорости, так и в производительности. Упрощенный контекст и удобные для пользователя структуры данных позволяют разработчикам легко читать и писать. Сокращенные сроки, связанные с Python, также означают меньшие затраты на разработку для компаний-разработчиков Python и их клиентов.

Учитывая сказанное выше Python как нельзя лучше подходит для этого проекта.

3.4 Сценарий работы программы

Программа может располагаться на любом ПК, необходимо лишь установить необходимые пакеты. При запуске скрипта появится окно проигрывателя, где и происходит генерация и общая обработка сигнала. Пользователь сразу же может генерировать мелодии (по нажатию на «горячую

клавишу» R). После генерации понравившейся мелодии пользователь может произвести начальную обработку при помощи слайдеров в верхней части окна и ползунков в нижней части и сохранить данную мелодию в одном из двух доступных форматов: MIDI или WAW.

3.5 Интерфейс приложения

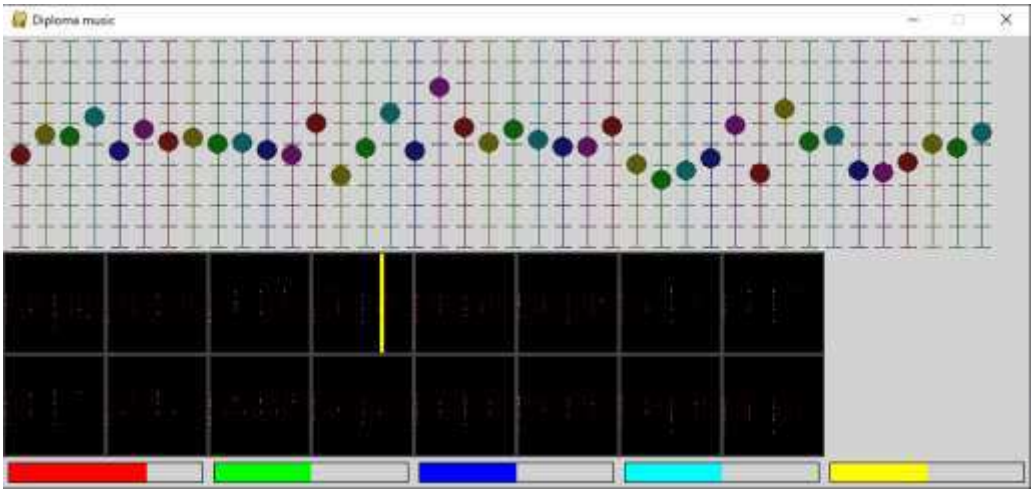

Рисунок 6 – Интерфейс проигрывателя

Интерфейс проигрывателя представляет собой окно со множеством слайдеров (верхняя часть), визуализацией сгенерированной музыки (центральная часть) и 5 ползунками основных настроек (нижняя часть).

Слайдеры привязаны к ключевым параметрам нейросети, поэтому их назначение определить не так просто.

По центру располагается 16 картинок с красными и белыми точками. Одна такая картинка обозначает музыкальную фразу, точки на этой картинке – ноты, белые ноты проигрываются, красные – нет.

Красный слайдер снизу – интенсивность мелодии. Отвечает за соотношение красных и белых нот. Зеленый слайдер – скорость проигрывания.

Синий – громкость. Бирюзовый – баланс нот (низкие ноты громче чем высокие и наоборот). Желтый – продолжительность нот.

3.6 Формирование обучающей выборки

Обучающая выборка формируется из мелодий формата MIDI с единственным инструментом – пианино. Желательно чтобы музыка имела один и тот же или похожий жанр (классическая музыка и джаз, например). Качество работы обученной нейросети напрямую зависит от количества файлов, выбранных для обучения.

4 РЕАЛИЗАЦИЯ

Для начала необходимо разобраться с наборами данных для обучения нейросети и их обработкой. Для этого мы разбиваем входные мелодии на фрагменты. Такие фрагменты будут использоваться для обучения нейросети.

```
try:
    samples = midi_utils.midi_to_samples(path)
except Exception as e:
    print("ERROR ", path)
    print(e)
    failed += 1
    continue
```

Рисунок 7 – функция разбиения MIDI файлов на фрагменты

Теперь можно создавать модель нейросети типа автоэнкодер с несколькими слоями.

```
x_in = Input(shape=embedding_input_shape)
print((None,) + embedding_input_shape)

x = Embedding(embedding_shape, latent_space_size, input_length=1)(x_in)
x = Flatten(name='encoder')(x)
```

Рисунок 8 – создание энкодера (входной слой)

```
x = Dense(1600, name='decoder')(x)
x = BatchNormalization(momentum=batchnorm_momentum)(x)
x = Activation('relu')(x)
if dropout_rate > 0:
    x = Dropout(dropout_rate)(x)
print(K.int_shape(x))
```

Рисунок 9 – создание декодера

Начинаем тренировать данную модель.

```
model = models.create_autoencoder_model(input_shape=y_shape[1:],
                                        latent_space_size=LATENT_SPACE_SIZE,
                                        dropout_rate=DROPOUT_RATE,
                                        max_windows=MAX_WINDOWS,
                                        batchnorm_momentum=BATCHNORM_MOMENTUM,
                                        use_vae=USE_VAE,
                                        vae_b1=VAE_B1,
                                        use_embedding=USE_EMBEDDING,
                                        embedding_input_shape=x_shape[1:],
                                        embedding_shape=x_train.shape[0])
```

Рисунок 10 – Создание модели

```
decoder = K.function([model.get_layer('decoder').input, K.learning_phase()], [model.layers[-1].output])
encoder = Model(inputs=model.input, outputs=model.get_layer('encoder').output)

random_vectors = np.random.normal(0.0, 1.0, (NUM_RAND_SONGS, LATENT_SPACE_SIZE))
np.save('data/interim/random_vectors.npy', random_vectors)
```

Рисунок 11 – Функция тренировки модели

После тренировки модель уже можно использовать в программе.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы была построена модель нейронной сети, которая генерирует мелодии. При разработке были изучены составляющие части нейронных сетей и её структура, и был рассмотрен способ её обучения. Параллельно были разработаны алгоритмы для работы с готовой моделью нейронной сети.

К недоработкам проекта можно отнести, некоторую однообразность мелодии, их одинаковую длину.

Разработанная модель и программа могут быть использована для генерации случайных мелодий с их последующей доработкой и использовании в различных проектах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A brief Introduction to MIDI // Wayback Machine URL: https://web.archive.org/web/20120830211425/http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/ (дата обращения: 20.04.21).
2. Шилов В. Л. Практический словарь иностранных музыкальных терминов // М.: Композитор. – 2003.
3. Почему стоит выбрать python URL: https://zen.yandex.ru/media/itgap/pochemu-stoit-vybirat-python-pri-razrabotke-svoih-proektov-5f13e9e3b7ecf524cfc7a2aa
4. Градиентный спуск URL: https://neurohive.io/ru/osnovy-data-science/gradient-descent/
5. Как работает вариационный автоэнкодер URL: https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«10» 06 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Генерация музыки при помощи машинного обучения

Руководитель _____ 8.06.21 ст. преп. каф. ВТ И. В. Матковский
подпись, дата

Выпускник _____ 8.06.21 Н. В. Белый
подпись, дата

Нормоконтролер _____ 8.06.21 И. В. Матковский
подпись, дата

Красноярск 2021