МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА
технический директор
ООО "Компания СГ-групп"
Д.В. Гапонюк
“___”_____ 2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2018 г.

РАЗРАБОТКА ПРОГНОЗИРУЮЩЕЙ НЕЙРОННОЙ СЕТИ ДЛЯ ОБРАБОТКИ ДВИЖЕНИЙ ЧЕЛОВЕКА НА ОСНОВЕ СКЕЛЕТИЗАЦИИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2018.115-016.ВКР

Научный руководитель
кандидат физ.-мат. наук,
доцент кафедры СП
_____В. Голодов

Автор работы,
студент группы КЭ-401
_____И.А. Нарыжный

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2018 г.

Челябинск-2018

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ 4
ВВЕДЕНИЕ 5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7
1.1. Разработка прогнозирующей нейронной сети для обработки движений человека на основе скелетизации 7
1.2. Обзор аналогичных проектов и существующих решений 8
2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 16
2.1. Нейронная сеть. Обучение FORCE 16
3. ПРОЕКТИРОВАНИЕ 19
3.1. Варианты использования 20
3.2. Диаграмма последовательности 21
4. РЕАЛИЗАЦИЯ 22
4.1. Средства реализации 22
4.2. Основные методы нейронной сети 23
4.3. Реализация графического интерфейса пользователя 24
5. ТЕСТИРОВАНИЕ 27
ЗАКЛЮЧЕНИЕ 30
ЛИТЕРАТУРА 31
ПРИЛОЖЕНИЯ 33
Приложение 1. Награды 33

ГЛОССАРИЙ

Рекуррентные нейронные сети (англ. Recurrent neural network) [5] — вид нейронных сетей, где связи между элементами образуют направленную последовательность, благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки.

Скелетизация [13] — процедура, целью которой является извлечение формы объекта на основе области, представляющей общий вид объекта

Motion capture [11] — технологии для цифровой записи движений человека (актера) и перевода их в компьютерно-анимированные изображения.

Генеративно-состязательная сеть (Generative adversarial network, GAN) [3] — алгоритм машинного обучения без учителя, построенный на комбинации из двух нейронных сетей, одна из которых (сеть G) генерирует образцы, а другая (сеть D) старается отличить правильные («подлинные») образцы от неправильных. Таким образом, сети G и D имеют противоположные цели — создать образцы и отбраковать образцы.

Прогнозирование [2] — исследование конкретных перспектив дальнейшего развития какого-либо процесса. Прогнозирование направлено на определение тенденций динамики конкретного объекта или события на основе ретроспективных данных, т.е. анализа его состояния в прошлом и настоящем. Таким образом, решение задачи прогнозирования требует некоторой обучающей выборки данных.

ВВЕДЕНИЕ

Актуальность

Задача обработки и анализа движений человека существует давно и имеет множество реализаций.

Применение анализа движения можно найти в разнообразных областях, таких как наблюдение, при исследованиях автомобильной безопасности при краш-тестах, в спортивной биомеханике.

Анализ походки (или ходьбы) также используется в спорте для оптимизации спортивных результатов или для идентификации движений, которые могут привести к травме или деформациям.

Биомеханика ходьбы здорового человека в естественных условиях является основой для изучения всех биомеханических наук. Ходьбу рассматривают с позиции единства биомеханических и нейрофизиологических процессов, которые определяют функционирование локомоторной системы человека.

Таким образом, тема анализа и обработки движений человека в наши дни является актуальной.

Цель и задачи исследования

Целью данной работы является разработка десктоп-приложения, которое использует нейронную сеть для выделения шаблонов из последовательностей значений (наборов данных) с функцией генерации данных на основе полученных шаблонов.

Для достижения поставленной цели необходимо решить следующие задачи:

1) произвести анализ предметной области (научные и программные аналоги);

2) спроектировать архитектуру нейронной сети;

3) провести обучение и тестирование нейронной сети;

4) реализовать и протестировать приложение, с функцией выделения шаблона и последующей генерации данных по полученному шаблону.

Структура и объем работы

Работа состоит из глоссария, введения, 5 глав, заключения, библиографии и приложения. Объем работы составляет 33 страницы, объем библиографии составляет 18 источников, объем приложений 1 страница.

Краткое содержание

В первой главе осуществляется анализ предметной области и обзор существующих решений в области генеративных нейронных сетей.

Во второй главе содержится описание архитектуры нейронной сети и проектирование программной структуры приложения.

В третьей главе описывается процесс реализации системы.

В четвертой главе содержится описание основных результатов тестирования приложения.

В пятой главе содержатся результаты тестирования системы.

В заключении работы описаны полученные в ходе выполнения работы результаты.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Разработка прогнозирующей нейронной сети для обработки движений человека на основе скелетизации

Для прогнозирования значений на основе входных данных необходимо подготовить данные для обучения сети, разработать топологию и определить параметры сети. Лучше всего для прогнозирования подходят хаотические нейронные сети.

Хаотическая рекуррентная нейронная сеть [16] – нейронная сеть, в которой хаос составляет базовую форму коллективной нейронной активности для всех процессов и функций восприятия. Он играет роль контролируемого источника шума с целью обеспечения непрерывного доступа к предыдущим запомненным образам и запоминанию новых. Хаос позволяет системе быть всегда активной, избавляя ее от необходимости пробуждаться или входить в устойчивое состояние каждый раз, когда изменяется входное воздействие.

Резервуарные вычисления

Сложность и неэффективность существующих алгоритмов обучения РНС вынуждает искать новые подходы, и стратегии. Одним из таких подходов стало новое направление – резервуарные вычисления(РВ) [1]. Основная идея РВ заключается в использовании РНС как резервуара с богатой динамикой и мощными вычислительными возможностями. При этом резервуар формируется случайным образом, что исключает необходимость проводить его обучение. При подаче на вход резервуара непрерывного сигнала он попадает в определенный динамический режим или состояние. Это состояние зависит от входного сигнала. Резервуар формируется таким образом, чтобы для похожих входных сигналов это состояние было похожим, а для разных – разным. Выход резервуара соединен со специальными устройствами – считывателями, которые по состоянию резервуара решают поставленную задачу – производят классификацию, предсказание, кластеризацию.

1.2. Обзор аналогичных проектов и существующих решений

Создание диалоговых систем с использованием генеративных иерархических моделей нейронных сетей (Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models)

В работе [12] была изучена задача построения диалоговых систем на основе больших наборов данных для исследований с использованием моделей генеративных сетей. Генеративные модели производят запросы, которые автономно генерируются по одному слову, что дает возможность для реалистичных, гибких взаимодействий. Для работы системы была улучшена иерархическая рекуррентная нейронная сеть – кодер-декодер, а также было продемонстрированно, что эта модель является конкурентоспособной с самыми современными моделями нейронных сетей.

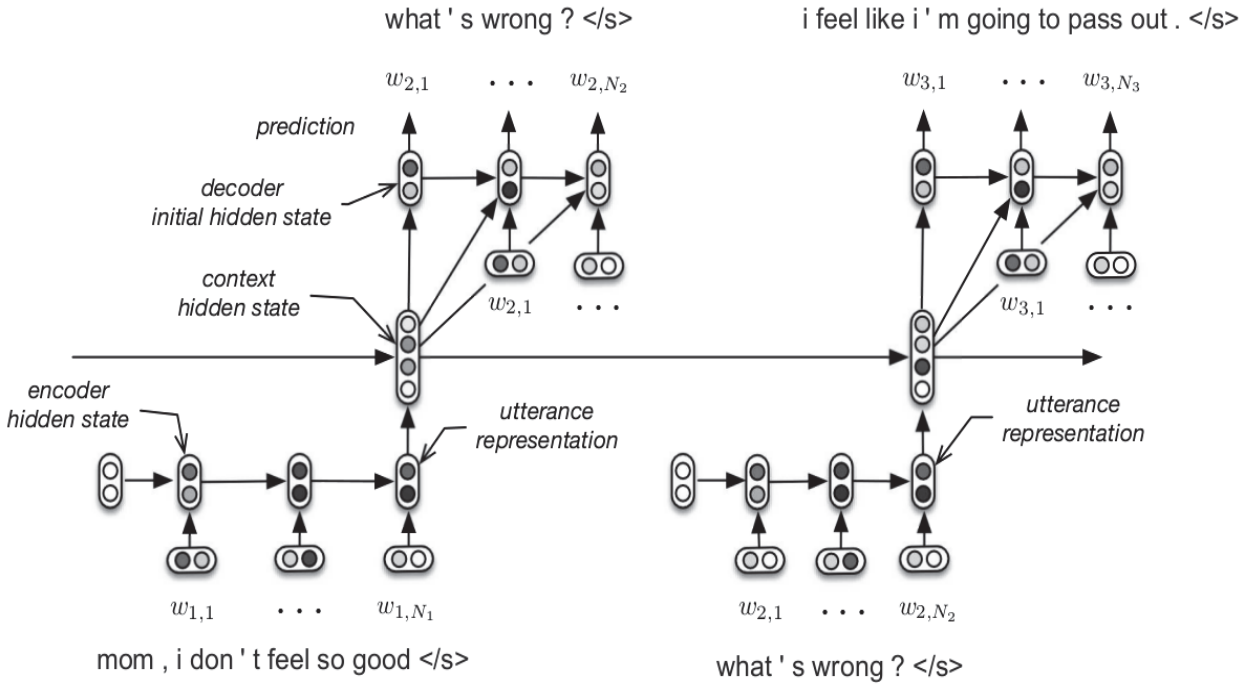

Рис. 1. График архитектуры HRED для диалога

В работе используется иерархическая рекуррентная архитектура кодеровщика-декодера (HRED), предложенного для веб-запросов. В исходной структуре HRED прогнозирует следующий веб-запрос с учетом запросов, уже представленных пользователем. История прошлых отправленных запросов рассматривается как двухуровневая последовательность: после-

довательность слов для каждого веб-запроса и последовательность запросов. HRED моделирует эту иерархию последовательностей с двумя RNN: один на уровне слов и один на уровне запросов. В работе сделано аналогичное предположение, что диалог можно рассматривать как последовательность высказываний, которые, в свою очередь, являются последовательностями токенов. График архитектуры HRED для диалога изображен на Рис. 1. График архитектуры HRED для диалога

Было обнаружено два важных компонента для повышения производительности: использование большой внешней выборки для инициализации вложений слов и использование большой связанной не диалоговой выборки. Это указывает на необходимость создания более крупных наборов данных диалогов.

В итоге было продемонстрировано, что генерирующая модель иерархической рекуррентной нейронной сети может превзойти как модели на основе n-граммов, так и базовые модели линейных нейронных сетей для задачи моделирования высказываний и речевых актов.

DRAW: Рекуррентная нейронная сеть для генерации изображений

В работе [4] была представлена архитектура нейронной сети Deep Recurrent Att Attive Writer (DRAW) для генерации изображений. Сети DRAW используют механизм пространственного внимания, который имитирует центральную ямку(лат. fovea centralis) человеческого глаза, с последовательной вариационной системой автоматического кодирования для итеративной конструкции сложных изображений. Система существенно улучшает современное состояние для генеративных моделей на MNIST(База данных «Mixed National Institute of Standards and Technology»), а при обучении в наборе данных Num Street House Numbers генерирует изображения, которые нельзя отличить от реальных данных невооруженным глазом. Примеры сгенерированных чисел изображены на рис. 2. и рис. 3.

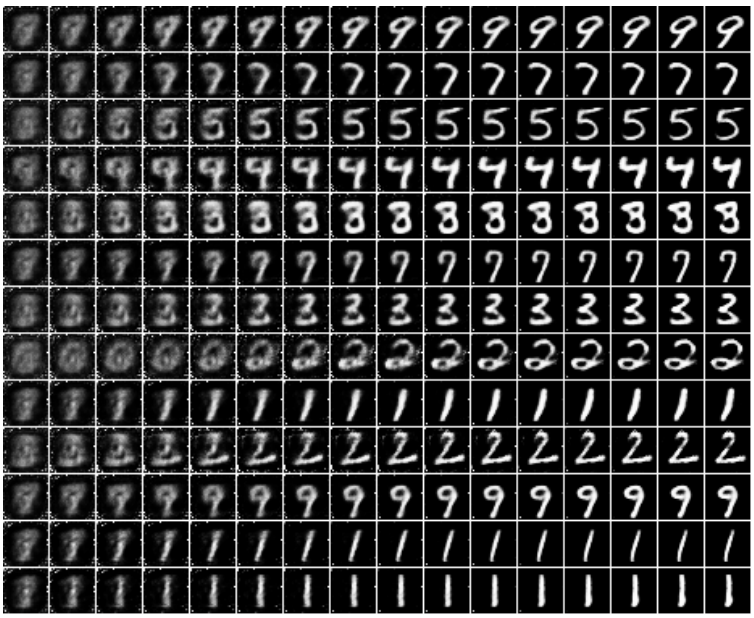

Рис. 2. Одиночные символы MNIST сгенерированные сетью DRAW

Рис. 3. Числа с двумя символами сгенерированные сетью DRAW

Базовая структура сети DRAW аналогична структуре других вариационных автокодеров: сеть кодировщика определяет распределение по скрытым кодам, которые фиксируют важную информацию о входных данных; сеть декодера получает образцы из дистрибутива кода и использует их для проверки собственного распределения по изображениям. Однако есть три ключевых отличия. Во-первых, как кодер, так и декодер являются

рекуррентными сетями в DRAW, так что между ними происходит обмен последовательностей кода; кроме того, кодер подключен к предыдущим выходам декодера, позволяя ему адаптировать коды, которые он отправляет, в соответствии с поведением декодера до сих пор. Во-вторых, выходы декодера последовательно добавляются к распределению, которое в конечном итоге генерирует данные, в отличие от излучения этого распределения за один шаг. И в-третьих, динамически обновляемый механизм оценки используется для ограничения как входной области, наблюдаемой кодером, так и области вывода, измененной декодером. Проще говоря, сеть решает свести к минимуму стохастический градиентный спуск.

В статье [4] была представлена архитектура нейронной сети Deep Recurrent Attentive Writer (DRAW) и продемонстрирована ее способность генерировать высокореалистичные естественные изображения, такие как фотографии номеров домов, а также улучшение наилучших результатов для генерации бинаризованной информации MNIST. Также было установлено, что двумерный дифференцируемый механизм восприятия, встроенный в DRAW, выгоден не только для генерации изображений, но и для классификации изображений.

Modeling Human Motion Using Binary Latent Variable

В статье [17] предлагается нелинейная генеративная модель для данных движения человека, которая использует неориентированную модель с бинарными скрытыми переменными и вещественными «видимыми» переменными, которые представляют собой совместные углы. Скрытые и видимые переменные на каждом временном шаге получают ориентированные соединения от видимых переменных на последних временных шагах. Такая архитектура делает онлайн вывод эффективным и позволяет нам использовать простую примерную процедуру обучения. После обучения модель находит один набор параметров, которые одновременно захватывают несколько разных видов движения. Авторы демонстрируют мощь подхода, синтезируя различные последовательности движения и выполняя онлайн-

заполнение данных, потерянных во время захвата движения.

В данной работе используется ограниченная машина Больцмана (RBM) для моделирования распределения вероятностей вектора наблюдения на каждом временном интервале. Как правило, RBM используют двоичные логистические единицы как для видимых данных, так и для скрытых переменных, но в работе приложении данные (состоящие из совместных углов) являются непрерывными. Таким образом, авторы используют модифицированный RBM, в котором «видимые единицы» являются линейными, вещественными переменными, которые имеют гауссовский шум.

Данная нейронная сеть генерирует движения человека на основе суставных углов, схематичный пример генерации изображён на рис. 4.

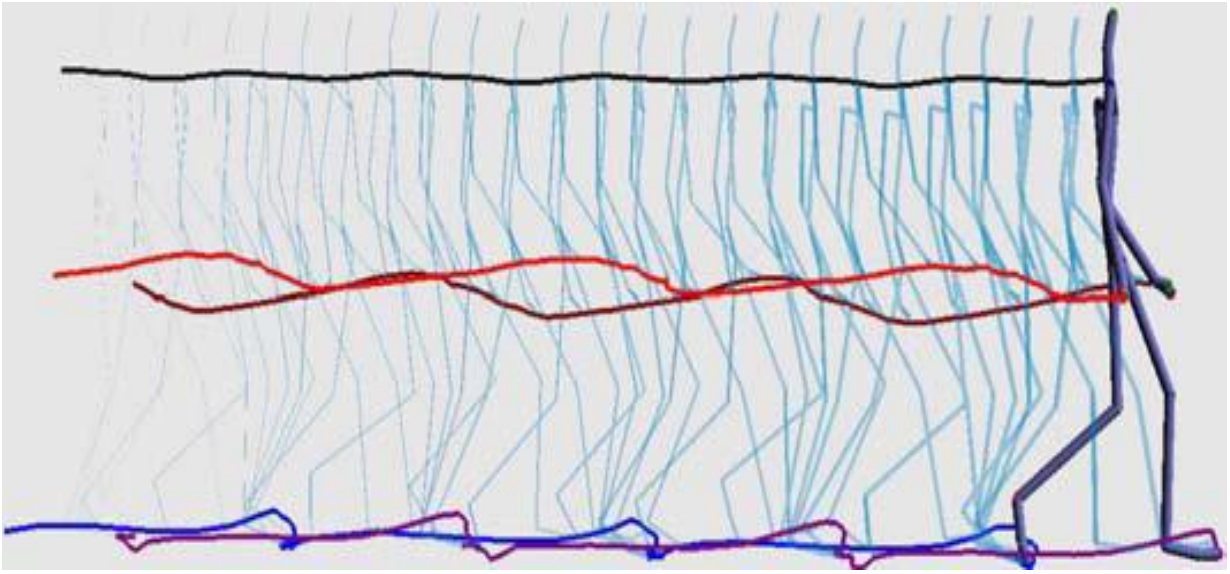

Рис. 4. Генерация движений человека

В данной работе была представлена генеративная модель человеческого движения, основанная на идее, что локальные ограничения и глобальная динамика могут быть эффективно изучены условной ограниченной машиной Больцмана. После обучения модели сетей способны эффективно захватывать сложные нелинейные данные без сложной предварительной обработки или уменьшения размерности. Модель сет была разработана с учетом человеческого движения, но может хорошо подходить к другим высокоразмерным временным рядам.

Skeleton-aided Articulated Motion Generation.

В работе [18] ученые делают попытку генерировать сочлененную последовательность движения человека с одного изображения. Для обучения сети используют два набора данных. В первом на вход подается информация о скелете человека, его движения и положение в пространстве. Во втором — образ человека в качестве ссылки на внешний вид, для создания новых кадров движения, основанных на инфраструктуре генеративно-состязательных сетей.

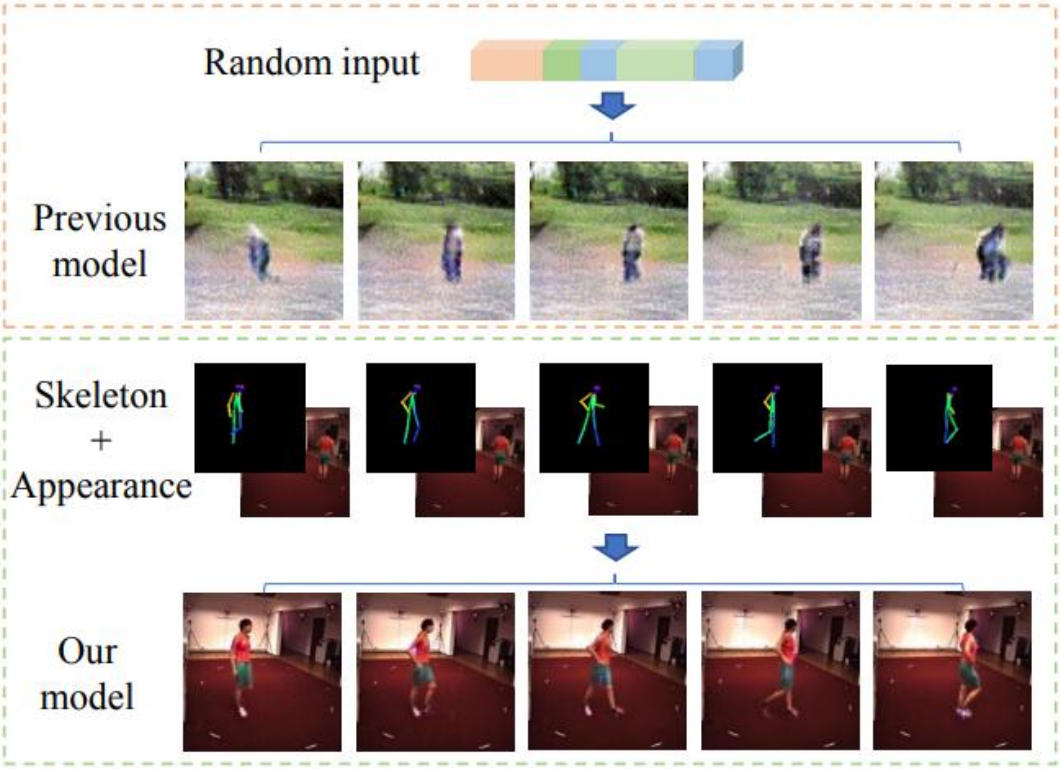

Рис. 5. Пример модели генеративно-состязательной сети

Использование этой техники позволяет в частности генерировать фотографии, которые человеческим глазом воспринимаются как натуральные изображения. Например, известна попытка синтезировать фотографии кошек, которые вводят в заблуждение эксперта, считающего ее естественными фото. Кроме того генеративно-состязательная сеть может использоваться для улучшения качества нечетких или частично испорченных фотографий. Пример модели генеративно-состязательной сети изображен на рис. 5.

Поскольку предлагаемая структура способна совместно использовать пространство внешнего вида изображения и сочлененное / кинематическое пространство движения, оно генерирует реалистичную сочлененную последовательность движения, в отличие от большинства способов генерации видео, которые дают эффекты размытого движения. Модель была протестирована на двух наборах данных о человеческих действиях, включая KTH и Human3.6M, и предлагаемая структура дает очень многообещающие результаты для обоих наборов данных.

Generating Coherent Patterns of Activity from Chaotic Neural Networks.

В работе [15] используется рекуррентная нейронная сеть с обучением FORCE, которая обеспечивает мощный алгоритм построения рекуррентных нейронных сетей, генерирующий сложные шаблоны деятельности. Данная технология способна работать как с входными данными, так и без них.

Обучение FORCE отличается от традиционного обучения нейронных сетей. Обычно суть обучения состоит в выполнении последовательной модификаций нейронов, в результате чего медленно уменьшаются изначально большие ошибки в выходном слое нейронной сети. При изучении FORCE ошибки всегда небольшие, даже с самого начала процесса обучения. В результате цель обучения — это не значительное уменьшение ошибок, а сокращение количества модификаций, необходимых для уменьшения ошибок. К концу периода обучения модификации больше не требуется, и сеть может генерировать желаемый вывод автономно.

С точки зрения машинного обучения предлагаемая нами процедура FORCE обеспечивает мощный алгоритм построения рекуррентных нейронных сетей, которые генерируют сложные и управляемые шаблоны деятельности либо в отсутствие, либо в ответ на ввод.

Для того чтобы сеть могла генерировать движения человека, она была обучена на данных, полученных от людей, которые двигались, нося ко-

стюм, позволяющий измерять такие переменные, как суставные углы. Эти данные были получены из библиотеки захвата движения CMU. Они состоят из 95 совместных углов, измеренных в течение сотен временных шагов.

Успешно обученные сети генерировали движения, демонстрируя, что одна хаотичная рекуррентная сеть может генерировать множественные, многомерные, непериодические шаблоны, которые напоминают сложные человеческие движения.

Обучение FORCE можно рассматривать или как модель для модификации нейронных сетей, вызванных обучением, или как метод построения моделей этих сетей.

По словам авторов статьи, обучение не требует никакой реконфигурации сети или изменений в ее динамике, кроме модификации синапсов. Наконец, сети, построенные с помощью обучения FORCE, более стабильны, чем в других подходах [15].

Выводы по главе 1

Для решения поставленной задачи поиска шаблонов и генерации данных была использована рекуррентная нейронная сеть с использованием процедуры FORCE, которая была модифицирована для решения поставленной задачи. Эта сеть была взята за основу, т.к. из всех рассмотренных вариантов она является наиболее гибким решением, что дает возможность модифицировать ее под конкретные задачи.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Архитектура

Разрабатываемая система генерации движений человека имеет следующую структуру.

Графический интерфейс, позволяющий выбирать данные для обучения нейронной сети, сохранять результат, полученный при обучении сети, а также предоставляющий возможность загрузить и отобразить результаты, полученные при предыдущих обучениях.

Рекуррентная нейронная сеть при обучении способна выделить шаблон из функции, что впоследствии позволяет генерировать сложные периодические и непериодические паттерны, например, движения человека, такие как бег или ходьба.

2.1. Нейронная сеть. Обучение FORCE

Алгоритм обучения, подходящий для обучения FORCE, должен быстро уменьшать величину разницы между фактическим и желаемым выводом до небольшого значения. Каждое обновление состоит из оценки вывода сети и определения того, насколько этот вывод отклоняется от целевой функции, и соответственно изменяя считываемые веса. $\Delta\, t$ представляет собой интервал времени между изменениями считываемых весов.

В момент времени t процедура обучения начинается с выборки выходного слоя, который задается в этой точке $w^T(t - \Delta\, t)\, \boldsymbol{r}(t)$. Сравнивая этот вывод с желаемым целевым выходом $f\ (t)$, мы определяем ошибку:

$$e_(t) = w^T(t - \Delta \mathrm{t}) r(t) - f(t).$$

Минус-индекс означает, что это ошибка до обновления веса в момент времени t. Сразу же после обновления веса выход сети будет $\boldsymbol{w^T}(t)\boldsymbol{r}(t)$, считая, что веса быстро изменяются в масштабе обучения сети. Таким образом, ошибка после обновления веса принимает следующее значение:

$$e_+(t) = w^T(t) r(t) - f(t).$$

Если процесс обучения инициализируется в момент времени $t = 0$, то первое обновление веса произойдет в момент времени $\Delta\, t$. Ошибка $|e_+(\Delta\, t)|$ после первого обновления веса должна быть небольшая, далее сохраняем $|e_-(\Delta\, t)|$ при медленном увеличении $e_+(\Delta\, t)/e_-(\Delta\, t) \to 1$. Учитывая небольшую величину $(e_+(\, t - \Delta\, t)), e_-(t)$, которая равна $e_+(\, t - \Delta\, t)$. Интервал обновления $\Delta\, t$ остается достаточно коротким. Это означает, что обучение может выполняться с ошибкой, которая с начала обучения остается небольшой.

Несколько правил модификации отвечают требованиям обучения FORCE, но рекурсивный алгоритм наименьших квадратов (RLS) особенно эффективен, и он используется в модификации RLS:

$$w(t) = w(t - \Delta\mathrm{t}) - e_(t)P(t)r(t).$$

где $P(t)$ — матрица $N \times N$,которая обновляется одновременно с весами согласно правилу

$$P(t) = P(t - \Delta\mathrm{t}) - \frac{P(t-\Delta\mathrm{t})\mathrm{r}(\mathrm{t})r^T(t)P(t-\Delta\mathrm{t})}{1+r^T(t)P(t-\Delta\mathrm{t})\mathrm{r}(\mathrm{t})}.$$

Алгоритм также требует начального значения для P, которое принято считать

$$P(0) = \frac{I}{\alpha},$$

где I — единичная матрица; α — постоянный параметр.

В этом алгоритме P представляет собой текущую оценку обратной корреляционной матрицы сетевых скоростей **r** плюс член регуляризации,

$$P = (\textstyle\sum_t r(t)r^T(t) + \alpha I)^{-1}.$$

Нетрудно показать, что правило RLS удовлетворяет условиям, необходимым для обучения FORCE. Во-первых, если мы предположим, что веса начального равны нулю (это не существенно), то из приведенных выше уравнений следует, что ошибка после первого обновления веса является такой:

$$e_{(\Delta\mathrm{t})} = -\frac{\alpha f(\Delta\mathrm{t})}{\alpha+r^T(t)r^T(\Delta\mathrm{t})\mathrm{r}(\Delta\mathrm{t})}.$$

Величина $r^T r$ имеет порядок N, число нейронов в сети, так что при $\alpha \ll N$ эта ошибка мала, и ее размер можно контролировать, регулируя α. Кроме того, в последствии из приведенных выше уравнений следует, что:

$$e_+(t) = e_-(t)(1 - r^T(t)P(t)r(t)).$$

Величина $r^T Pr$ изменяется в процессе обучения от 1 до значения, которое асимптотически приближается к 0 и всегда положительно. Это означает, что размер ошибки уменьшается по мере обновления веса, и в конечном счете $e_+(\Delta\, t)/e_-(\Delta\, t) \to 1$.

Параметр α, который действует как скорость обучения, должен корректироваться в зависимости от выбранной целевой функции. Малые значения α приводят к быстрому обучению, но иногда делают так быстро, что алгоритм становится неустойчивым. В этих случаях следует использовать большее значение α (при условии ограничения $\alpha \ll N$).

Выводы по главе 2

В данной главе был рассмотрен принцип работы рекуррентных нейронных сетей с использованием FORCE.

3. ПРОЕКТИРОВАНИЕ

Нейронная сеть

Для выделения шаблона из входных данных используется модель хаотической рекуррентной нейронной сети, где для выделения шаблона применяется процедура FORCE

Данная нейронная сеть состоит из:

1) входного слоя, размером 1000 нейронов;

2) скрытый слой для выделения шаблона;

3) выходной слой, размер которого равен размеру входного слоя;

Требования

Функциональные требования

Функциональные требования — это требования, охватывающие предполагаемое поведение системы, указывающие действия, которые система способна выполнять. В рамках разрабатываемой программной системы были выявлены следующие функциональные требования.

В ходе проектирования приложения были определены функциональные и нефункциональные требования к разрабатываемому приложению.

1. Программная система должна быть реализована в виде приложения с графическим интерфейсом.
2. Приложение должно предоставлять пользователю возможность выбирать данные для обучения и запускать процесс обучения спроектированной нейронной сети.
3. Приложение должно предоставлять пользователю возможность выбирать генерировать данные на основе обученной модели сети.
4. Приложение должно предоставлять пользователю возможность визуализации сгенерированных данных.
5. Приложение должно предоставлять пользователю возможность визуализации исходных данных.

Нефункциональные требования.

1. Система может быть реализована на языке Python.

3.1. Варианты использования

Рассмотрим взаимодействие актеров с системой генерации движений человека и варианты использования, которые актеры могут инициировать. Диаграмма вариантов использования изображена на рис. 6.

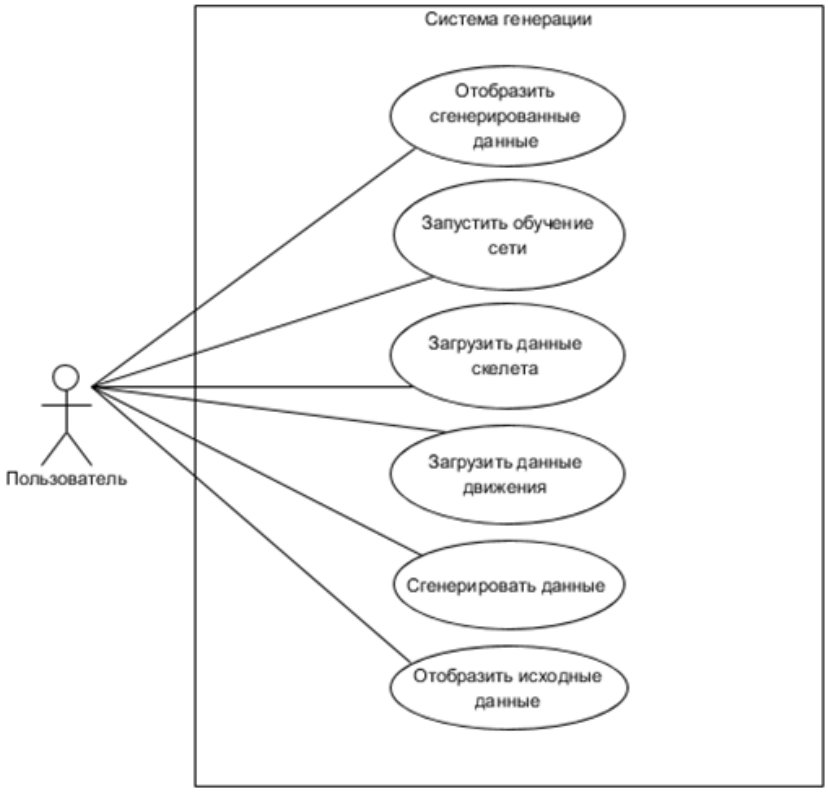

Рис. 6. Варианты использования системы

Пользователь представляет собой основного актера. Для пользователя существует шесть вариантов использования системы генерации:

1. Загрузить данные движения. Выбор производится среди наборов данных в каталоге data.
2. Загрузить данные скелета. Выбор производится среди наборов данных скелета в каталоге skel.
3. Запустить обучение сети. Система запускает сеть для выделения шаблона.
4. Сгенерировать данные. Данные генерируются на основе выделенных шаблонов.
5. Отобразить исходные данные. Запускается средство визуализации,

отображение происходит в автоматическом режиме.

6. Отобразить сгенерированные данные. Запускается средство визуализации. Отображаются сгенерированные данные.

3.2. Диаграмма последовательности

Отображает последовательность действий системы при генерации движений человека с помощью нейронной сети. Диаграмма последовательности изображена на рис. 7.

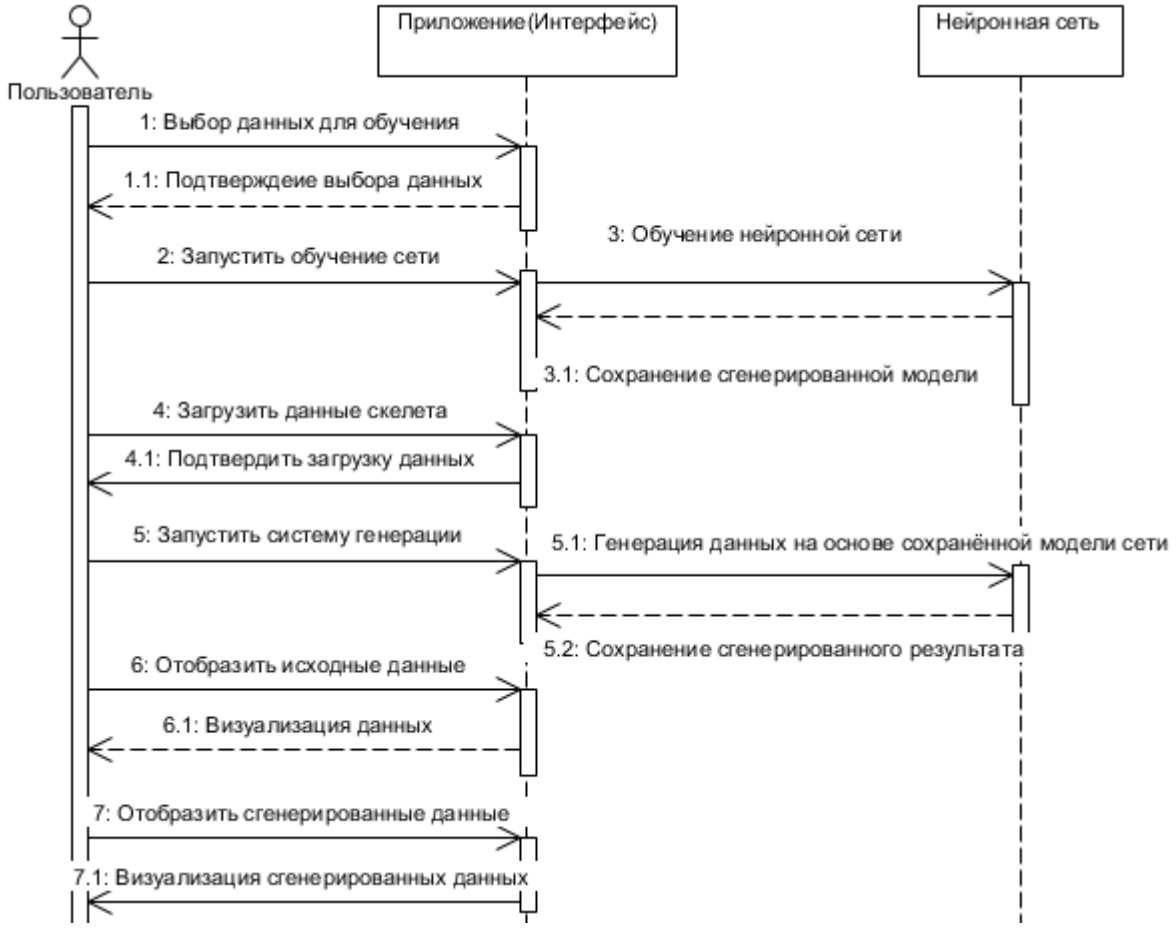

Рис. 7. Диаграмма последовательности

Выводы по главе 3

Для реализации системы в соответствии с требованиями была спроектирована система генерации движений на основе скелетизации.

4. РЕАЛИЗАЦИЯ

4.1. Средства реализации

MATLAB [7] – это высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов.

Для написания графического интерфейса и нейронной сети был выбран язык программирования Python версии 3.5.5 [10].

Этот выбор обусловлен тем, что стандартная библиотека включает большой объем полезных функций, а также существует множество библиотек и фреймворков для работы с математическими алгоритмами, построениями графиков и т.д.

PyQT [14] – для разработки графического интерфейса использовался кросс-платформенный фреймворк PyQT, который поддерживается Windows, MAC OS, Linux и другими UNIX-подобными операционными системами. Включает поддержку многих функций QT, а также включает в себя дизайнер графического интерфейса пользователя.

Pickle [9] – модуль pickle реализует мощный алгоритм сериализации и десериализации объектов Python. Применяется для сохранения и загрузки сложных объектов в Python.

Matplotlib [6] – библиотека на языке программирования Python для визуализации данных двумерной графикой. Используется для построения графиков.

NumPy [8] – open-source модуль для python, который предоставляет общие математические и числовые операции в виде прескомпилированных, быстрых функций.

Предобработка данных

Реализация представляет собой скрипт, написанный на языке программирования Matlab, принимающий в качестве входного параметра путь до файла данных захвата движений Acclaim Motion Capture data и возвращающая обработанные данные в виде матрицы значений double с расширением .mat.

Обучающая выборка представлена файлами данных в формате amc. Формат Acclaim состоит из двух файлов, файла скелета и файла движения. Это было сделано, из-за того что большую часть времени один скелет используется для отображения разных движений и вместо того, чтобы хранить один и тот же скелет в каждом из файлов движения, он должен храниться только один раз в отдельном файле. Файл скелета - это файл ASF (файл скелета Acclaim). Файл движения - это файл AMC (данные захвата движения Acclaim).

Файл amc преобразуется в матрицу данных. Модуль предобработки написан на языке программирования Matlab и интегрируется с помощью MATLAB Engine API для Python. Запуск функции Matlab в python отображен в листинге 1.

Листинг 1. Листинг вызова функции на языке Matlab

```
import matlab.engine
      eng = matlab.engine.start_matlab()
      data = eng.amc_to_matrix(amc_to_matrix.m)
```

4.2. Основные методы нейронной сети

Основная логика обучения нейронной сети содержится в методе train класса EHRule листинга 2.

Листинг 2. Метод train

```
class EHRule:
      def train(self, output, step, inp=None):
            alpha = 1 - self.__res.time_step / self.__t_avg
            out_size = output.shape[1]
            weights_out = np.zeros((self.__res.network_size, out_size))
            z = np.zeros((1, out_size))
            plp = 0
            zlp = np.zeros((1, out_size))
            out = np.zeros((output.size, out_size))

            for idx in xrange(0, output.shape[0], step):
                  r = self.__res.simulate(z, inp)
                  z = np.dot(r, weights_out) + np.random.rand(1, out_size)
- 0.5

                  p = -np.sum(np.power(z - output[idx], 2))
                  plp = alpha * plp + (1 - alpha) * p
                  zlp = alpha * zlp + (1 - alpha) * z

                  if p > plp:
                        eta = self.learn_rate_init / (1 + idx *
```

```
self.__res.time_step / self.decay_const)
                        dw = eta * np.multiply(np.transpose(z-zlp), r)
                        weights_out = weights_out + np.transpose(dw)
                  out[idx, :] = zlp

            model = FeedbackNet(self.__res, weights_out, 0.5)
            return model, out
```

Для генерации точек на основе выделенного шаблона используются методы simulate и simulateNoInput указанные в листинге 3.

Листинг 3. Методы simulate и simulateNoInput

```
class FeedbackNet:
def simulate(self, inp=None):
out = np.zeros((inp.shape[0], self.__outsize))
idx = 0

for i in inp:
r = self.res.simulate(i, self.feedback)
self.feedback = np.dot(r, self.weights) +
self.noise_level*(np.random.rand(1, self.__outsize)-0.5)
out[idx, :] = np.transpose(self.feedback)
idx = idx+1
self.feedback = self.feedback

return out

def simulateNoInput(self, steps):
out = np.zeros((steps, self.__outsize))

for k in xrange(0, steps):
r = self.res.simulate(self.feedback)
self.feedback = np.dot(r, self.weights) +
self.noise_level*(np.random.rand(1, self.__outsize)-0.5)
out[k, :] = self.feedback

return out
```

4.3. Реализация графического интерфейса пользователя

Рассмотрим основные элементы интерфейса пользователя (рис. 8):

1) возможность загрузить данные движения для отображения;

2) возможность загрузить данные скелета для отображения;

3) кнопка «Визуализировать» запускает средство визуализации для отображения исходных данных;

4) кнопка «Обучить на входных данных» приводит к запуску обучения сети;

5) возможность ввести количество точек для генерации;

6) кнопка «Генерировать данные» приводит к генерации результатов;

7) кнопка «Визуализировать сгенерированный результат» запускает средство визуализации и отображает сгенерированные данные;

8) возможность вывода лога операций;

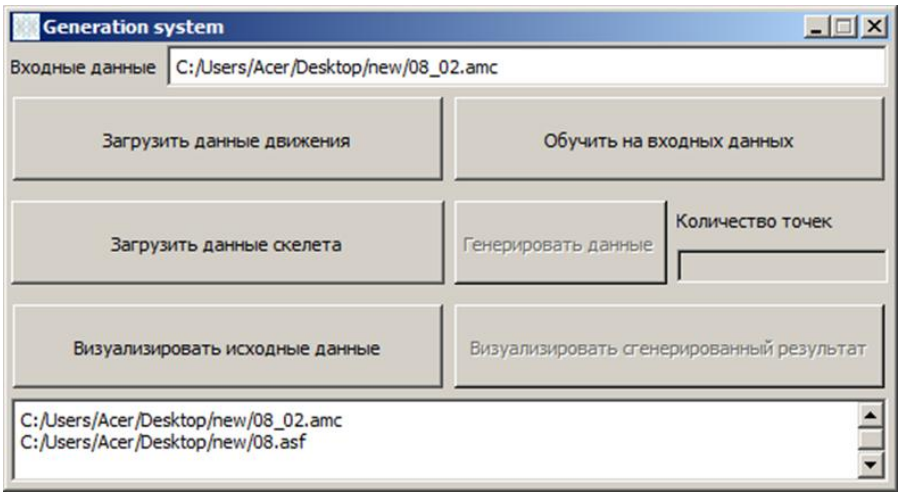

Рис. 8. Интерфейс пользователя

Средство визуализации данных.

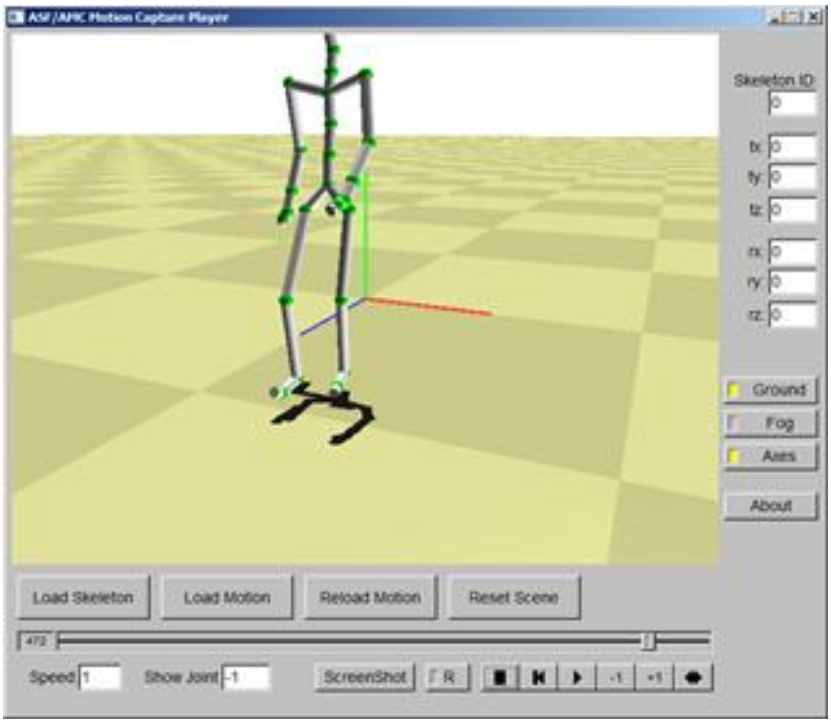

Рис. 9. ASF/AMC Motion Capture Player

В качестве средства визуализации используется приложение ASF/AMC Motion Capture Player для отображения данных в формате AMC и ACF (рис.9.)

Выводы по главе 4

В соответствии с требованиями была реализована система генерации движений на основе скелетизации, которая включает в себя нейронную сеть, графический интерфейс пользователя и средство визуализации.

5. ТЕСТИРОВАНИЕ

Тестирование — это процесс, который заключается в проверке соответствия программного продукта заявленным характеристикам и требованиям, требованиям эксплуатации в различных окружениях, с различными нагрузками, требованиями по безопасности и удобству использования.

Функциональное тестирование

Функциональное тестирование — это тестирование программного обеспечения, для проверки реализуемости функциональных требований, то есть способность программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Было проведено тестирование на основе функциональных требований. Ниже представлен протокол функционального тестирования.

Тест № 1.

Цель: Возможность выбора данных для обучения нейронной сети из набора данных.

Действие: Пользователь нажимает на кнопку «Загрузка данные движения»

Ожидаемый результат: Пользователь увидит окно выбора файла данных для обучения, отображаются файлы с необходимым расширением.

Тест пройден? Да.

Тест № 2.

Цель: Возможность выбора данных скелета для отображения.

Действие: Пользователь нажимает на кнопку «Загрузка данные скелета»

Ожидаемый результат: Пользователь увидит окно выбора файла данных для обучения, отображаются файлы только с необходимым расширением asf.

Тест пройден? Да.

Тест № 3.

Цель: Проверить возможность визуализации исходных данных.

Действие: Пользователь нажимает на кнопку «Визуализировать».

Ожидаемый результат: Загружается окно средства визуализации, затем происходит отображение ранее загруженных исходных данных в автоматическом режиме.

Тест пройден? Да.

Тест № 4.

Цель: Проверить возможность обучения сети.

Действие: Пользователь нажимает кнопку «Обучить на входных данных» с выбранным файлом данных для обучения.

Ожидаемый результат: Пользователь увидит сообщение о запуске процесса обучения. После обучения будет выведено сообщение об окончании обучения.

Тест пройден? Да.

Тест № 5. Цель: Проверить возможность генерации данных.

Действие: Пользователь нажимает кнопку «Генерировать данные».

Ожидаемый результат: Пользователь увидит сообщение о запуске процесса генерации. Запустится процесс генерации данных. После завершения процесса генерации будет выведено сообщение об успешно выполненной операции.

Тест пройден? Да.

Тест № 6. Цель: Проверить возможность визуализации сгенерированных данных.

Действие: Пользователь нажимает кнопку «Визуализировать сгенерированные данные».

Ожидаемый результат: Загружается окно средства визуализации, и происходит отображение сгенерированных данных в автоматическом режиме.

Тест пройден? Да.

Выводы по главе 5

В соответствии с функциональными требованиями было разработано и протестировано приложение. В ходе тестирования основного функционала программы ошибок выявлено не было.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано десктоп-приложение, которое использует нейронную сеть для выделения шаблонов из наборов данных с функцией генерации на основе полученных шаблонов.

При этом были решены следующие поставленные задачи:

1) проведен обзор литературы предметной области;

2) подготовлена обучающая выборка;

3) разработана архитектура нейронной сети;

4) произведено обучение и тестирование нейронной сети;

5) разработано приложение, с функцией выделения шаблона и последующей генерации данных по полученному шаблону;

6) произведено тестирование приложения.

ЛИТЕРАТУРА

1. Бендерская Е.Н. Рекуррентная нейронная сеть как динамическая система и подходы к ее обучению. // Научно-технические ведомости, 2001. − P. 29–40.

2. Data Mining. Прогнозирование и визуализация. [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/6/6/lecture/168 (дата обращения: 16.04.2018).

3. Goodfellow I.J. Generative Adversarial Networks. // arXiv preprint arXiv, 2014. − P. 1–9.

4. Gregor K. DRAW: A Recurrent Neural Network For Image Generation. // arXiv preprint arXiv, 2015 − P. 1–20.

5. Haykin S. Neural Networks and Learning Machines − United States of America, New Jersey: Prentice Hall 2008. − 906 p.

6. Matplotlib Documentation. [Электронный ресурс] URL: https://matplotlib.org/tutorials/index.html (дата обращения: 12.04.2018).

7. Matlab Documentation. [Электронный ресурс] URL: https://www.mathworks.com/help/matlab/ (дата обращения: 19.05.2018).

8. NumPy v1.14 Manual. [Электронный ресурс] URL: https://docs.scipy.org/doc/numpy (дата обращения: 23.04.2018).

9. Pickle — Python object serialization. [Электронный ресурс] URL: https://docs.python.org/3/library/pickle.html (дата обращения: 02.05.2018).

10. Python 3.5.5 documentation. [Электронный ресурс] URL: https://docs.python.org/3/ (дата обращения: 17.05.2018).

11. Ribeiro T.H. Motion Capture Technology — Benefits and Challenges. // International Journal of Innovative Research in Technology & Science(IJIRTS), 2016. − P. 2321–1156.

12. Serban I.V. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. // Proceedings of the Thirtieth {AAAI} Conference on Artificial Intelligence, 2016. − P. 3776–3784.

13. Skeletonization. [Электронный ресурс] URL: http://www.inf.u-

szeged.hu/~palagyi/skel/skel.html (дата обращения: 11.04.2018).

14.PyQt5 Tutorial Documentation. [Электронный ресурс] URL: https://pythonspot.com/pyqt5/ (дата обращения: 25.05.2018).

15. Sussillo D., Abbott L.F. Generating Coherent Patterns of Activity from Chaotic Neural Networks. // Neuron, 2009. − P. 544–557.

16. Sussillo D.C. Learning in Chaotic Recurrent Neural Networks. // PhD thesis, 2009. − 134 P.

17. Taylor G., Hinton G.E., Roweis S. Modeling human motion using binary latent variables. // Advances in neural information processing systems, 2007. – P. 1345.

18. Yan Y. Skeleton-aided Articulated Motion Generation. // arXiv preprint arXiv, 2017. – P. 1–9.

ПРИЛОЖЕНИЯ

Приложение 1. Награды

РОССИЙСКАЯ НАУЧНО-СОЦИАЛЬНАЯ ПРОГРАММА ДЛЯ МОЛОДЕЖИ И ШКОЛЬНИКОВ

ШАГ В БУДУЩЕЕ

XIII Уральская выставка научно-технического творчества молодежи «Евразийские ворота России»

Региональный филиал Российской молодежной научной и инженерной выставки «Шаг в будущее»

(г. Челябинск, Южно-Уральский государственный университет, 2-4 апреля 2018 г.)

ДИПЛОМ

ВТОРОЙ СТЕПЕНИ

за лучшую творческую работу по информационным технологиям

с вручением памятной медали выставки

НАГРАЖДАЕТСЯ

НАРЫЖНЫЙ Иван Александрович

г. Челябинск, ФГАОУ ВО «Южно-Уральский государственный университет» (НИУ), КЭ-401 (бакалавр)

Разработка прогнозирующей нейронной сети для обработки движений человека на основе скелетизации

Научный руководитель: Голодов Валентин Александрович, кандидат физ.-мат. наук, Южно-Уральский государственный университет

Ф.Л. Дегтярев
Президент
Южно-Уральской
торгово-промышленной
палаты, координатор
Уральской выставки НТТМ
«Евразийские ворота России»

А.А. Радионов
Проректор
по учебной работе
Южно-Уральского
государственного университета,
научный руководитель программы
«Шаг в будущее...»

Е.Н. Кузьмин
Член Центрального совета
Всероссийской программы для
молодежи «Шаг в будущее»,
координатор Южно-Уральской
программы для молодежи «Шаг
в будущее-Созвездие-НТТМ»

4 апреля 2018 г.

Рис. 10 Диплом второй степени