Санкт-Петербургский государственный университет

ШАЛЕВА Анна Сергеевна

Выпускная квалификационная работа

Разработка системы взаимодействия человек-машина по фрагментам устной русскоязычной речи

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2015 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Исследование и проектирование систем управления и обработки сигналов»

Научные руководители:
доцент, кафедра компьютерного моделирования и многопроцессорных систем, к.т.н.
Гришкин Валерий Михайлович

ассистент, кафедра компьютерного моделирования и многопроцессорных систем,
Якушкин Олег Олегович

Рецензент:
доцент, кафедра компьютерных технологий и систем, к.ф.-м.н.
Погожев Сергей Владимирович

Санкт-Петербург

2019

Оглавление

Введение

С возрастающей потребностью в диалоговых интерфейсах для мобильных и VR-девайсов, IoT, теле- и радио- вещания, активное развитие происходит в области понимания устной речи [27; 45; 50]. Высокопроизводительные системы преобразования речи в текст и текста в речь представляют два наиболее важных аспекта таких интерфейсов, поскольку большинство вычислительных алгоритмов разработаны для текстового ввода и вывода [10; 28]. Не менее важной составляющей разговорных интерфейсов является определение ключевых слов (Keyword Spotting, KWS) – обнаружение заданных ключевых слов в непрерывном потоке аудио для управления последующим переходом между вычислительными состояниями связанной системы [17].

Долгое время для построения систем обработки и распознавания речи успешно применялись последовательные алгоритмы статистического моделирования: скрытые марковские модели (HMM) [31; 52] и условные случайные поля (CRF) [13; 14]. Подобные алгоритмы, не смотря на их широкую распространенность, имели свои недостатки: большая размерность пространства признаков, требование явных предположений о независимости наблюдений и необходимость специфических знаний о задаче для проектирования включаемых состояний для HMM; зависимость выбора факторов-признаков от специфики конкретных данных для CRF [15]. Перечисленные особенности подразумевали под собой активное вмешательство человека в процесс создания модели.

С развитием вычислительных мощностей, графических ускорителей и технологий параллельных вычислений [22] широкое распространение получили глубинные нейронные сети (DNN), проявившие себя в задачах классификации изображений [21; 29] и распознавания речи [43; 53]. Отличительной особенностью DNN в контексте сравнения с нейронными сетями является наличие более чем трех слоев (т. е. более одного скрытого

слоя), что значительно увеличивает количество параметров модели и является причиной высоких вычислительных затрат. Развитие ускорительных технологий позволило осуществлять вычисление взвешенных сумм, необходимое DNN для обучения и предсказания, за обозримое время.

На смену HMM пришли гибридные системы, использующие DNN для предварительной обработки и классификации признаков, сокращая тем самым размерность задачи для HMM [19]. Позже были разработаны гибридные DNN-CRF системы, использующие в качестве признаков для вероятностной модели вектора, полученные с помощью DNN, превосходящие по качеству DNN-HMM в области обработки речи [34]. В отличие от своих предшественников, разработанные системы позволили применить «end-to-end» обучение, ограничивающее вмешательство человека в процесс настройки параметров модели, но требующее взамен большое количество данных для обучения и проектирование архитектуры модели.

Системы распознавания речи, базирующиеся на технологиях DNN, быстро завоевали лидирующее место в организации взаимодействия человека и электронных девайсов, например, Amazon Echo, Google Home. Тем не менее, непрерывно работающая система распознавания речи не является энергоэффективной и может вызывать перегрузки сети, передавая непрерывный поток аудио в облачный сервис. Кроме того, облачные решения добавляют задержку на отклик приложения, что негативно сказывается на пользователях. Чтобы избежать подобных недостатков, для активации полноценной системы распознавания речи девайсу нужно выделить определенные ключевые слова, например, “Алекса”, “Ok Google” – данная задача относится к области KWS. Наиболее очевидным и часто используемым способом реализации таких систем является постоянная поддержка KWS модуля в работающем режиме, для чего идеально подходят микроконтроллеры – недорогие и энергоэффективные процессоры. Однако развертывание KWS, базирующихся на DNN, на микроконтроллерах

сталкивается с двумя основными проблемами: ограниченный объем памяти (обычные микроконтроллеры имеют несколько сотен КВ доступной памяти, которая должна вместить в себя нейронную сеть вместе со входом, выходом и параметрами) и ограниченные вычислительные ресурсы (поскольку KWS система постоянно поддерживается в рабочем состоянии, объем вычислительных операций, выполняемых за один цикл предсказания, должен укладываться в рамки реального времени).

Высокие требования к точности, скорости срабатывания и надежности систем KWS с одной стороны, и ограниченные вычислительные и энергетические ресурсы – с другой, поддерживают актуальной задачу разработки *бережливой* архитектуры DNN. Несмотря на активные исследования и коммерческую адаптацию вышеописанных методов, в настоящий момент существует дефицит систем с отрытым исходным кодом, удовлетворяющих обоим ограничениям. Особенно мало систем определения ключевых слов с поддержкой русского языка.

В данной работе изложен процесс разработки системы определения ключевых слов в русскоязычной речи с использованием технологий глубинного обучения, а также применение этой системы для организации взаимодействия человек-машина. Описан процесс построения архитектуры нескольких моделей глубинных нейронных сетей. Приведен разработанный метод для автоматического сбора и создания корпуса обучающих данных. Описаны процесс оптимизации гиперпараметров моделей с использованием инструмента с открытым исходным кодом Microsoft NNI и процесс обучения моделей на собранном корпусе данных. Проведена сравнительная характеристика обученных моделей. Описан принцип работы системы распознавания ключевых слов, основанной на применении лучшей из обученных моделей.

Постановка задачи

Целями данной работы являются:

1. Разработка метода для автоматического сбора русскоязычных аудио данных, применимых для обучения нейронной сети распознаванию ключевых слов
2. Разработка end-to-end системы распознавания ключевых слов в устной русскоязычной речи при помощи обучения глубинной нейронной сети на примерах *(аудио, класс ключевого слова, подкласс ключевого слова)*
3. Разработка голосовой системы взаимодействия человек-машина

Система распознавания ключевых слов должна классифицировать и распознавать короткие (менее 1 секунды) голосовые команды, содержащие ключевые слова из заранее определенного множества и иметь уровень точности распознавания более 75%.

Система взаимодействия человек-машина должна уметь обрабатывать поданные голосом и не ограниченные по длительности команды человека, позволяя идентифицировать в голосовой команде ключевые сущности заранее заданных классов, интерпретировать полученный набор ключевых сущностей в терминах машины и организовывать взаимодействие путем передачи команд к исполнению машине.

Для оценки точности распознавания ключевых слов используется метрика, называемая категориальной точностью. Данная метрика вычисляется, как отношение количества примеров с верно предсказанными категориями к общему количеству примеров:

$$categorical\_accuracy = \frac{\sum i}{n},$$

$$i: argmax(y_i^{true}) == argmax(y_i^{predict}), i = \overline{1,n}$$

Где y_i^{true} – вектор, представляющий действительный класс $i^{-\text{го}}$ примера, $y_i^{predict}$ – вектор, представляющий вероятностное распределение предсказанных классов для $i^{-\text{го}}$ примера, n – общее количество примеров. Алгоритм кодирования классов представлен в [3.2].

Обзор литературы

Возвращаясь к недостаткам скрытых марковских моделей, описанных во введении, следует также отметить, что подобные системы, хотя и достигают значительной точности, требуют высоких вычислительных мощностей как во время обучения, так и во время предсказания, поэтому рассмотрим далее техники создания систем распознавания ключевых слов, базирующихся на глубинных нейронных сетях.

Основой KWS моделей, использующих обычные DNN [48], является последовательность полносвязных и нелинейных активационных слоев. На вход такой DNN поступает матрица векторизованных признаков, извлеченных из аудио, которую далее обрабатывают d скрытых полносвязных слоев, состоящих каждый из n нейронов. Функция активации ReLU [1], обычно следующая после каждого полносвязного слоя, решает проблему исчезающего градиента DNN, когда во время обучения модели методом обратного распространения ошибки, для каждого последующего скрытого слоя величина градиента экспоненциально уменьшается. Выходной слой является линейным с функцией активации Softmax [3], генерирующей итоговые вероятности для k ключевых слов. Описанная в [48] DNN показывает 45% улучшение точности распознавания ключевых слов по сравнению с конкурентоспособной HMM, обладая при этом необходимым быстродействием. Более того, техники низкоранговой аппроксимации [39], используемые для сжатия весов обученной DNN модели, позволяет достичь близкой к исходной точности с меньшими аппаратными затратами [12; 39]. Главным недостатком DNN является неспособность эффективно моделировать локальные временные и спектральные корреляции во входных признаках аудио [27].

KWS, базирующиеся на сверточных нейронных сетях (CNN) [16], способны улавливать локальные временные корреляции, вследствие чего демонстрируют более высокую точность распознавания по сравнению с DNN [27], в том числе – меньшее количество ложных срабатываний [16]. CNN

рассматривают набор входных признаков, зависящих от времени и спектра аудио, как двумерную матрицу, и осуществляют двумерные сверточные операции над ней. За сверточными слоями обычно следует пакетная нормализация (batch-normalization) [7], призванная сократить ковариантный сдвиг в данных и ускорить процесс сходимости. Также используются функция активации ReLU и Max\Average Pooling слои [42], которые сокращают размерность пространства признаков. В некоторых случаях в целях сокращения количества параметров и ускорения процесса обучения [20] между сверточным слоем и dense слоем добавляется линейный полносвязный слой с низкой размерностью. Недостатком CNN в моделировании временных рядов (например, речи) является тот факт, что они игнорируют сильно удаленные друг от друга по времени зависимости признаков [27].

Рекуррентные нейронные сети отлично проявили себя в задачах моделирования временных последовательностей, особенно – распознавании речи [18; 35], построении языковых моделей [44], переводе [46]. RNN не только способны выделять локальные временные корреляции во входящем сигнале, но также улавливать удаленные друг от друга зависимости, используя механизм ворот (gates). В отличие от CNN, где входной сигнал представляется в виде двумерного массива, RNN оперируют T моментами времени, где каждому моменту времени t соответствует вектор спектральных признаков, конкатенированный с выходным значением, полученным в предыдущий момент времени.

Для RNN особенно важен контекст состояния на каждом временном шаге. Одним из способов сохранения памяти о предыдущих предсказаниях в подобных сетях является использование в нейроне ячеек с памятью (LSTM) [32] вместо простых функций активации. Такие ячейки способны сохранять аналоговые значения и имеют ворота на входе и выходе, которые контролируют возможность входного сигнала изменять значение в ячейке и возможность значения в ячейке модифицировать выходной сигнал

соответственно. Кроме того, существуют «забывающие» ворота (forget gate), контролирующие убывание значения, хранящегося в памяти ячейки. Таким образом, пока входные и забывающие ворота закрыты, значение памяти в ячейке остается неизменным.

Другим зарекомендовавшим себя способом решения проблемы исчезающего градиента для RNN является использование управляемых рекуррентных блоков (GRU) [23]. В отличие от LSTM, GRU не содержат выходных ворот, вследствие чего имеют меньшее количество параметров по сравнению с LSTM.

Поскольку веса модели, используемые вдоль всех T моментов времени, могут использоваться повторно для длинных временных последовательностей, RNN модели обычно имеют меньшее количество параметров, нежели CNN. Несмотря на то, что методы с использованием RNN значительно превосходят по точности распознавания KWS, базирующиеся на HMM, их слабой стороной является высокая длительность распознавания [27].

В сверточных рекуррентных нейронных сетях (CRNN) [17] сочетаются преимущества CNN и RNN. KWS, базирующиеся на CRNN способны улавливать как локальные временные и пространственные корреляции, используя сверточные слои, так и глобальные временные зависимости в речевых признаках, используя рекуррентные слои. CRNN модель состоит из сверточного слоя, за которым следуют RNN для кодирования сигнала и полносвязный слой для отображения информации в конечную форму. В статье [8] используется двунаправленный рекуррентный слой, имеющий несколько состояний, что повышает способность сети к запоминанию. Основной ячейкой рекуррентного слоя являются GRU - такая архитектура позволяет использовать меньше параметров, чем LSTM, и дает лучшую сходимость [27]. Кроме того, CRNN проявляют устойчивость к зашумленным данным [17].

Depthwise Separable Convolutional Neural Network (DS-CNN) была впервые представлена в качестве эффективной альтернативы стандартным сверточным сетям [55] и использовались для достижения большей компактности архитектуры в области компьютерного зрения [47; 38]. Стандартный сверточный слой одновременно учитывает как пространственную информацию (корреляцию соседних точек внутри одного канала), так и межканальную информацию, поскольку свертка применяется ко всем каналам сразу. В свою очередь, DS-CNN строятся на предположении о том, что эти два вида информации можно обрабатывать последовательно, без ущерба для качества работы сети. DS-CNN декомпозируют стандартные трехмерные свертки в последовательность двумерных и одномерных, осуществляя на первом этапе свертку каждого канала входа во входную карту признаков с отдельными двумерными фильтрами (spatial convolution), и выполняя точечные свертки 1×1 (pointwise convolution) для комбинирования выходов вглубь на втором этапе. При этом DS-CNN существенно выигрывают по количеству параметров и операций по сравнению с обычными CNN, что делает возможным расширение и увеличение глубины модели даже в условиях ограниченных ресурсов. После успешного применения в задачах компьютерного зрения [29], подход, использованный в DS-CNN, был применен в задачах распознавания речи, в том числе – в задаче распознавания ключевых слов [27], где показал превосходство DS-CNN над рекуррентными и гибридными моделями.

Принимая во внимание успешный пример переноса подхода к решению задач из области компьютерного зрения в область распознавания речи в [27], стоит также отметить публикацию [40], описывающую многоуровневый процесс извлечения особенностей входных данных в задаче классификации временных последовательностей. В данной статье описывается Multi-Scale Convolutional Neural Network (M-CNN), объединяющая в себе два компонента: извлечение дискриминативных признаков и классификацию, а также

учитывающая факт наличия признаков в разных временных и частотных масштабах входных данных. Другими словами, формирование итогового результата происходит с учетом не только последнего скрытого слоя, но также и данных, полученных понижением размерности входной матрицы признаков с помощью нескольких скрытых сверточных слоев. Данный подход, расширенный в [57], стал лидирующим по точности распознавания в области детектирования объектов в режиме реального времени, что позволяет сделать предположение о его возможном успешном применении в задаче распознавания ключевых слов.

Наиболее важные публикации, описывающие процесс развития глубинных нейронных сетей в области решения задач распознавания ключевых слов, отображены в сравнительной таблице 1.

Таблица 1. Наиболее важные публикации, описывающие развитие глубинных нейронных сетей применительно к решению задач KWS

Название статьи	Год публикации	Основные результаты исследования
Small-footprint keyword spotting using Deep Neural Networks [48]	2014	Разработаны две модели для решения задачи KWS: основанная на DNN, состоящей из полносвязных слоев, и базирующаяся на HMM. Показано 45% улучшение точности распознавания модели, использующей глубинную нейронную сеть по сравнению с системой, использующей скрытые марковские модели.
Deep Speech: Scaling up end-to-end speech recognition [18]	2014	Разработана архитектура RNN для решения задачи распознавания речи, использующая двунаправленный рекуррентный слой и прорывной для

		задач данного класса CTC подход. Показано превосходство в точности над существующими на момент проведения исследования моделями.
Convolutional Neural Networks for Small-footprint Keyword Spotting [16]	2015	Разработаны несколько CNN моделей для решения задачи KWS, а также описаны несколько способов уменьшения размера моделей путем сокращения числа параметров и количества операций. Показано 27-44% превосходство по критерию ложных отклонений по сравнению с конкурентоспособными DNN моделями.
Xception: Deep Learning with Depthwise Separable Convolutions [55]	2016	Разработан и описан принцип работы Depthwise Separable CNN, примененный для решения задач в области компьютерного зрения. Данный подход позволил сократить количество параметров модели и достичь улучшения точности путем более эффективного способа их использования.
Multi-Scale Convolutional Neural Networks for Time Series Classification [40]	2016	Разработана модель Multi-Scale CNN, объединяющая в себе два компонента: извлечение дискриминативных признаков и классификацию, а также учитывающая факт наличия признаков в разных временных и частотных

		масштабах входных данных. Данная архитектура успешно применена для решения задачи классификации временных последовательностей.
Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting [17]	2017	Приводится описание и сравнение показателей точности нескольких Convolutional Recurrent NN моделей в зависимости от размера, типа и количества скрытых слоев. Предлагается несколько стратегий обучения, направленных на совершенствование показателей точности моделей, а также описываются способы аугментации обучающих данных.
Hello Edge: Keyword Spotting on Microcontrollers [27]	2018	Подход, использующий Depthwise Separable CNN для решения задач из области компьютерного зрения, применен для решения задачи KWS. Разработаны несколько вариантов архитектур моделей глубокого обучения с учетом ограничений на параметры и количество операций в моделях, произведена оценка точности обученных моделей на едином наборе данных [49] и показано преимущество DS-CNN над другими DNN в случае указанных ограничений.

Принимая во внимание описанную в данной части работы и представленную в таблице историю развития моделей глубинного обучения в контексте решения задачи распознавания ключевых слов, можно утверждать, что одним из перспективных направлений в решении задачи является использование сверточных нейронных сетей. Данная работа основывается на трех наиболее значимых и многообещающих в плане дальнейшего развития в задаче KWS подходах, описанных в статьях [16; 27; 40], а именно, использование CNN, Depthwise Separable CNN и Multi-Scale CNN.

Глава 1 Выбор инструментов для реализации задачи

Целью данной главы является сравнение и подбор оптимальных средств для решения поставленных задач.

В данной главе приводится обоснование выбора языка программирования для реализации каждого модуля системы, аппаратного обеспечения и данных для обучения модели распознавания, а также средства оптимизации параметров модели.

1.1 Выбор языка программирования

Одним из широко используемых языков программирования в области науки о данных является скриптовый язык Python. Его преимуществом является широкий выбор предметно-ориентированных программных пакетов и фреймворков в области работы с данными, таких как numpy, scipy, и, в частности, в области машинного обучения - TensorFlow и Keras. Версия 3.6 – одна из более поздних версий языка, постепенно заменяющая Python 2.7. Python 3.6 поддерживает последние обновления стандартных библиотек, а также большинство современных проектов.

1.2 Выбор аппаратного обеспечения

Поскольку решение поставленной задачи предполагает обучение глубоких нейронных сетей для распознавания ключевых слов, неотъемлемым компонентом процесса разработки становятся высокие вычислительные затраты. Большое количество параметров и выполняемых операций при обучении DNN требует аппаратного обеспечения, включающего графические процессоры, для выполнения поставленной задачи за обозримое время. Кроме того, для сбора и хранения корпуса обучающих данных необходимо наличие достаточного объема свободного дискового пространства на используемой ЭВМ.

Требованиям, перечисленным выше, удовлетворяет ЭВМ ресурсного центра «Вычислительный центр Санкт-Петербургского государственного университета». Данная ЭВМ имеет следующие характеристики:

- **ЦПУ:** 2 x Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60 ГГц
- **ОЗУ:** 256 ГБ
- **Графические процессоры:** 2 x Nvidia Tesla P100, по 16 ГБ видеопамяти

1.3 Выбор источников данных для обучения модели

Для успешного обучения системы распознавания ключевых слов, основанной на глубокой нейронной сети, требуется более 1000 примеров аудио семплов для каждого ключевого слова [49]. Рассмотрим различные способы создания корпуса данных.

1.3.1 Открытый корпус данных

Одним из немногих корпусов русскоязычных аудиоданных с транскриптами, находящихся в открытом доступе, является набор речевых данных, описанный в [9]. Корпус содержит 11.5 и 650 часов транскрибированной русскоязычной речи, собранных с сайта VoxForge.org и видеохостинга YouTube соответственно. Минусом описанного источника данных является длина семплов – от нескольких до десятка секунд (3 - 8 слов в одном семпле). Учитывая специфику данных, необходимых для задачи распознавания ключевых слов, для использования материалов описанного корпуса данных необходимо точно локализовать местонахождение искомого ключевого слова в аудио для его последующей обработки, что является трудоемкой задачей.

Следует также отметить, что готовых наборов русскоязычных данных, ориентированных на непосредственное использование в решении задач распознавания ключевых слов в речи, в открытом доступе не было найдено.

1.3.2 Аудиокниги

Данный источник содержит записи книг, прочитанных дикторами, а также транскрипты к ним. Преимуществом является большой объем доступного материала, а также практически точное совпадение письменного текста и записанной речи. Минусом данного источника, как и предыдущего, является необходимость временного выравнивания аудио относительно текста для локализации ключевых слов в аудио.

1.3.3 Видео-хостинг YouTube

Данный источник данных представляет собой видео-хостинговый сайт, содержащий миллионы часов аудио и видео контента. Большая часть записей сопровождается субтитрами, предоставленными пользователями, либо автоматически-сгенерированными с помощью алгоритмов автоматического распознавания речи от Google.

Пользовательские субтитры, хотя и обладают большей степенью смысловой достоверности, имеют меньшую точность сопоставления во времени транскрипта и аудиодорожек. Автоматически-сгенерированные субтитры, в отличие от пользовательских, доступны для большего количества видео. Кроме того, автоматически-сгенерированные субтитры предоставляют время начала и конца каждого распознанного слова. Их минусом является неточная локализация слов во времени.

Принимая во внимание тот факт, что видео с данного хостинга имеют разное качество, содержат в себе множество различных голосов говорящих с разными акцентами, записаны в разных шумовых условиях и с помощью различных девайсов, использование данных с YouTube в качестве обучающих позволит адаптировать модель к описанным условиям.

1.3.4 Программные сервисы синтеза речи

Благодаря развитию методов моделирования речевого сигнала, в настоящее время существует широкий выбор так называемых Text-to-Speech

сервисов (TTS services), способных синтезировать близкую к человеческой по звучанию речь, используя печатный текст. Синтезированные семплы можно использовать в качестве дополнительных примеров в корпусе данных, содержащем записи реальной речи.

Наиболее популярными TTS сервисами являются Google Text-to-Speech [25], TTS из пакета Microsoft Azure Cognitive Speech Services [36], Yandex SpeechKit [56]. TTS сервис от Microsoft обладает большей вариативностью дикторов по сравнению с другими сервисами, а также предоставляет бесплатный пробный период для знакомства с сервисом.

1.3.5 Выбранные источники

Наиболее эффективными способами для составления корпуса данных являются использование видео-хостинга YouTube и Microsoft Azure Cognitive Speech Services. Большой объем информации, который можно получить с помощью видео-хостинга, будет полезен при составлении обучающего набора данных, а синтезированные с помощью Azure TTS семплы позволят дополнить и разнообразить корпус данных.

1.4 Выбор средства оптимизации гиперпараметров модели

Задача проектирования архитектуры модели и выбора ее параметров включает в себя элемент эмпирических решений и их последующего сравнения. Чтобы автоматизировать данный процесс, существуют инструменты автоматической настройки гиперпараметров моделей.

В ноябре 2017 года исследователями Microsoft был представлен активно развиваемый и поддерживаемый инструмент с открытым исходным кодом «Neural Network Intelligence» [37]. NNI позволяет пользователям автоматизировать проведение экспериментов по машинному обучению, генерируя с помощью оптимизационного алгоритма архитектуру и\или набор параметров модели из заданного множества и запуская процесс обучения построенной модели, а также предоставляет интерфейс визуализации данных

и результатов. От аналогичных инструментов (Talos [51], H2O.ai [26]) NNI отличается широким выбором поддерживаемых фреймворков машинного обучения, оптимизирующих алгоритмов (вплоть до возможности создания уникального алгоритма) и обучающих сервисов, а также простотой использования.

Глава 2 Разработка архитектуры end-to-end модели распознавания ключевых слов

Целью данной главы является проектирование архитектуры сверточной нейронной сети для решения задачи распознавания ключевых слов из входного аудио сигнала.

Наиболее значимыми и многообещающими в плане дальнейшего развития научными публикациями в области распознавания ключевых слов с использованием сверточных нейронных сетей, проанализированными в части «Обзор литературы» являются [16; 27; 40]. В данной главе формализуются требования к модели распознавания ключевых слов, указанные в постановке задачи, описывается процесс разработки трех сверточных нейронных сетей, основой для которых послужили модели CNN, Depthwise Separable CNN и Multi-Scale CNN, описанные в [16; 27; 40]. Также для каждой модели приводятся особенности составляющих компонент и детали построения.

2.1 Формализация условий задачи

Исходя из поставленной задачи, исходными данными являются аудиосигналы постоянной длины, однако непосредственная обработка звуковых сигналов во временной области не является эффективной [17]. С помощью последовательного применения дискретного преобразования Фурье, оконной функции и дискретного косинусного преобразования, будем извлекать из сигнала последовательность векторов признаков $\boldsymbol{x}_i \in \mathbb{R}^m$, $i = \overline{1,n}$ мел-кепстральных коэффициентов (MFCC) [54]. Данные признаки численно описывают отрезок сигнала, обеспечивая более высокое «разрешение» детализации в области интересных нам частот, имея при этом меньшую по сравнению со спектрограммой сигнала размерность. Таким образом в качестве входных данных модели будем использовать матрицу признаков $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, где m – размерность каждого вектора-коэффициента, n – задаваемое количество коэффициентов.

Выходными данными для каждого аудио сигнала являются вектор класса ключевого слова $\boldsymbol{y} \in \mathbb{R}^k$, представляющий распределение вероятностей принадлежности ключевого слова к классу i, $i = \overline{1,k}$, $k = |\boldsymbol{C}|$, и вектор $\boldsymbol{z} \in \mathbb{R}^l$ представляющий распределение вероятностей принадлежности ключевого слова к подклассу j, $j = \overline{1,l}$, $l = |\boldsymbol{S}|$; $\boldsymbol{C}$, $\boldsymbol{S}$ – множества классов и подклассов ключевых слов соответственно.

Определим сверточную нейронную сеть со входом размерности $m \times n$, двумя выходами размерности k и l соответственно, и вектором весов ω, определяющим отображение $\boldsymbol{F_\omega}$ из множества последовательностей входных векторов $\boldsymbol{x}$ в множества последовательностей выходных векторов $(\boldsymbol{y}, \boldsymbol{z})$.

$$\boldsymbol{F_\omega}: \mathbb{R}^{m \times n} \longrightarrow (\mathbb{R}^k, \mathbb{R}^l)$$

2.2 Описание архитектуры

2.2.1 Convolutional Neural Network

Данная модель включает в себя 5 скрытых слоев. Данными для входного слоя являются тензоры размерности $m \times n$, представляющие собой наборы векторов мел-кепстральных коэффициентов.

Первыми четырьмя скрытыми слоями являются сверточные слои. Параметры сверточного слоя представляют собой набор обучаемых фильтров, каждый из которых имеет размерность $w \times h \times d$ – ширина, высота и глубина фильтра соответственно. Обычно размеры ядра свертки много меньше размеров входного тензора. Во время прямого прохождения данного слоя осуществляется операция свертки путем вычисления поэлементного произведения между участками значений входного тензора и весами фильтра вдоль каждой пространственной размерности с шагом, задаваемым параметром $stride = 1$. Таким образом на выходе формируется карта активационных признаков. По мере прохождения сверточных слоев, карты активационных признаков накапливаются в глубину, увеличивая тем самым количество каналов, что позволяет осуществлять переход от детектирования

низкоуровневых локальных признаков входного сигнала к локализации их комбинаций – более сложных характеристик.

Для регулирования пространственных размерностей тензора для каждого сверточного слоя задается параметр $padding$, с учетом которого осуществляется окаймление входного тензора элементами с нулевыми значениями. В данном эксперименте используется значение $padding$, позволяющее сохранить пространственную размерность входного тензора

После каждого сверточного слоя применяется пакетная нормализация (batch-normalization) [7], призванная сократить ковариантный сдвиг в данных и ускорить процесс сходимости модели.

Далее для борьбы с проблемой исчезающего градиента применяется функция активации ReLU (Rectifier Linear Unit), определяемая как:

$$ReLU(x) = \max(0, x)$$

Для сокращения пространственных размерностей выходных тензоров после каждого сверточного слоя с пакетной нормализацией и активационной функцией ReLU применяется операция субдискретизации – нелинейного уплотнения карты признаков, с использованием функции максимума (MaxPooling).

Ко всем четырем слоям также применяется dropout. Применение dropout с уровнем $dropout\_rate$ исключает выбранный с вероятностью $dropout\_rate$ нейрон из итерации процесса обучения, что позволяет предотвратить переобучение модели [42].

Далее полученный результат «вытягивается» в вектор и передается для обработки двум слоям:

1. Выходному полносвязному слою с $k = |\boldsymbol{C}|$ нейронами и функцией активации Softmax, где значения на выходе каждого из нейронов

отражают вероятность появления соответствующего класса ключевого слова

2. Полносвязному слою с p нейронами и функцией активации ReLU, после которого следует выходной полносвязный слой с $l = |\boldsymbol{S}|$ нейронами, выходные значения которых пропорциональны вероятности появления соответствующего подкласса ключевого слова.

Схематичное изображение описанной сверточной нейронной сети представлено на рисунке 1.

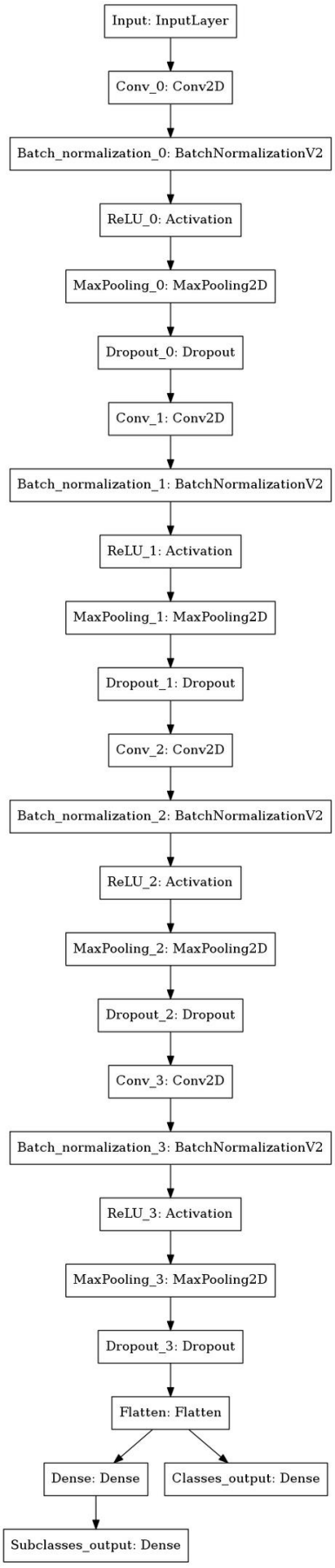


Рис. 1. Схематичное изображение модели CNN

2.2.2 Depthwise Separable Convolutional Neural Network

Depthwise Separable Convolutional Neural Network (DS-CNN) использует принцип разделения свертки на пространственную и точечную, вследствие чего достигается более эффективное использование параметров модели.

DS-CNN имеет 5 скрытых слоев и, подобно модели CNN, описанной в части 2.2.1, единственный вход, принимающий матрицу MFCC размерностью $m \times n$. Первым скрытым слоем является описанный в части 2.2.1 сверточный слой, поскольку в случае, когда входные данные имеют глубину = 1, Depthwise Separable Convolution слой вырождается в двумерный сверточный слой. Следующие три скрытых слоя являются Depthwise Separable Convolution слоями (DS-C). Данный слой принимает на вход тензор размерности $p_1 \times q_1 \times d_1$, после чего, в отличие от обычного сверточного слоя, для каждого входного канала *в отдельности* осуществляется так называемая глубинная пространственная свертка (depthwise spatial convolution) с фильтром размера $w \times h \times 1$. Количество выходных каналов не изменяется, и результатом применения данной операции является тензор размерности $p_2 \times q_2 \times d_1$. После к полученному тензору применяется точечная свертка (pointwise convolution) с d_2 фильтрами, имеющими размерность ядра 1×1. Результатом данной операции является тензор $p_2 \times q_2 \times d_2$. Авторы статьи [34] утверждают, что в общем случае порядок выполнения операций глубинной пространственной и точечной сверток не имеет значения. Число весов в DS-C слое $c_1 = d_1 * 1 * 1 * d_2 + w * h * d_2$, что в $\frac{d_1 * w * h}{d_1 + w * h}$ раз меньше количества весов в аналогичном сверточном слое размерности $w \times h \times d_2$: $c_2 = d_1 * w * h * d_2$.

В двумерных сверточных операциях используется значение $padding$, описанное в части 2.2.1, пакетная нормализация и ReLU в качестве активационной функции, после чего к выходному тензору применяется

операция MaxPooling и dropout. Дальнейший процесс обработки полученного результата совпадает с пунктами 1 и 2 алгоритма, описанными в части 2.2.1.

Схематичное изображение данной DS-CNN представлено на рисунке 2.

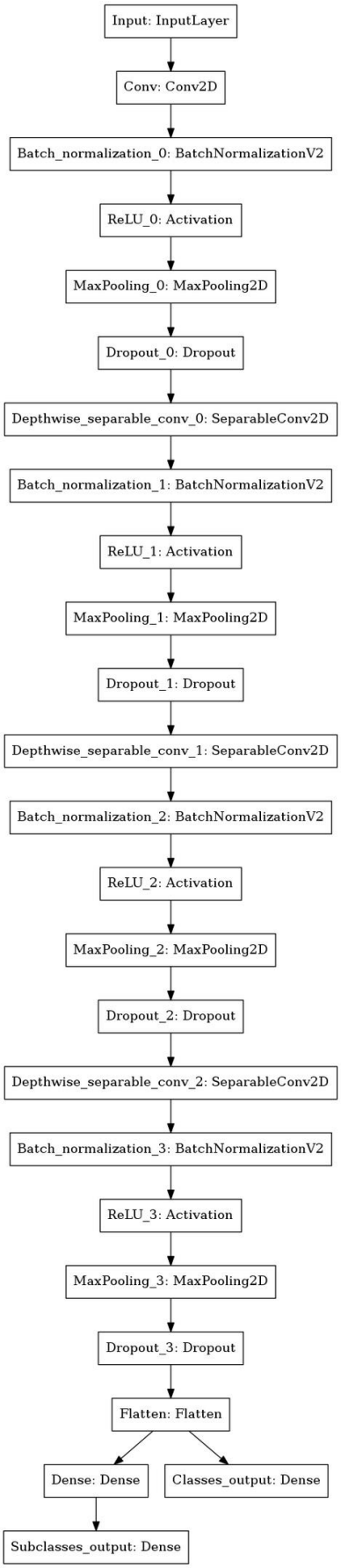


Рис. 2. Схематичное изображение модели DS-CNN

2.2.3 Multi-Scale Convolutional Neural Network

Multi-Scale Convolutional Neural Network (M-CNN) осуществляет локализацию признаков не только на конечном этапе распознавания, но также и в промежуточном скрытом сверточном слое, включая информацию, полученную после свертки, в последний скрытый слой.

Данная сверточная сеть имеет 5 скрытых слоев и принимает на вход матрицу MFCC признаков размерностью $m \times n$. Первые два скрытых слоя являются сверточными с применением пакетной нормализации, активационной функцией ReLU, MaxPooling и dropout, описанными в части 2.2.1. Результатом применения сверточных слоев ко входной матрице будет являться тензор x_1, имеющий размерность $m_1 \times n_1 \times k_1$. Далее следуют два скрытых сверточных слоя с пакетной нормализацией, активационной функцией ReLU и dropout, но без операции MaxPooling, что необходимо для сохранения пространственных размеров результирующего тензора x_2 размерности $m_1 \times n_1 \times k_2$. Для включения промежуточных состояний в итоговый тензор далее осуществляется операция конкатенации x_1 и x_2 в результирующий тензор размерности $m_1 \times n_1 \times (k_1 + k_2)$. Полученный тензор затем «вытягивается» и передается двум слоям, описанным в пунктах 1 и 2 части 2.2.1.

Схематичное изображение данной M-CNN представлено на рисунке 3.

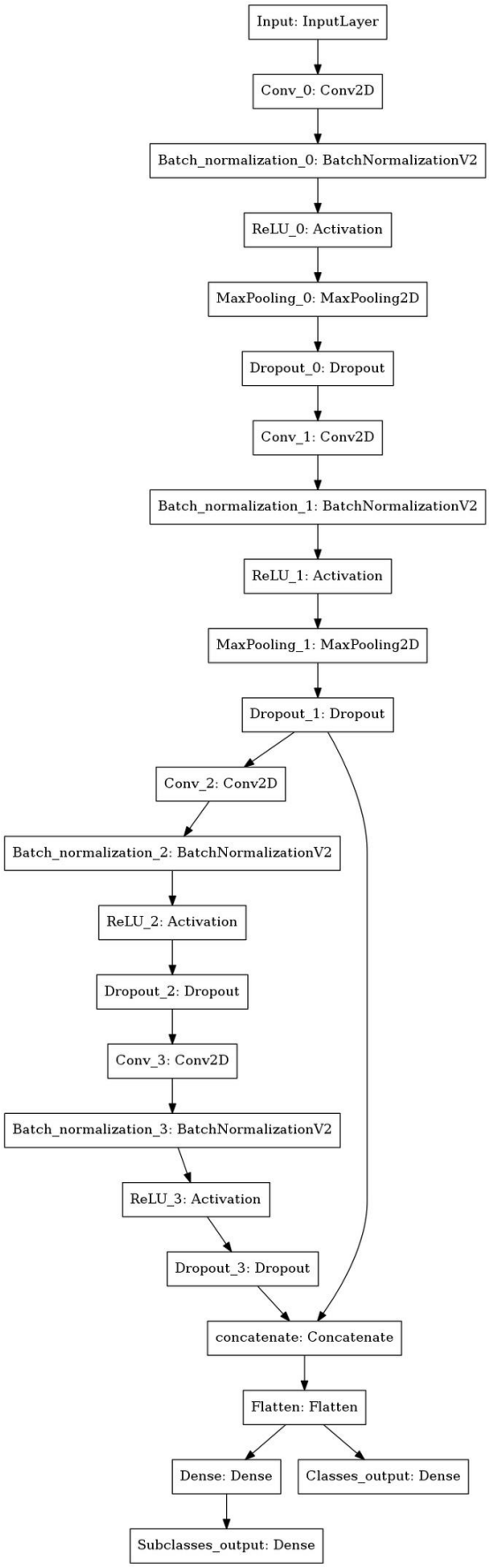


Рис. 3. Схематичное изображение модели M-CNN

2.3 Функция потерь

Для обучения модели методом обратного распространения ошибки необходимо ввести функцию, вычисляющую ошибку для входной матрицы MFCC $\boldsymbol{X}$ и предсказанного моделью $\boldsymbol{F_\omega}$ вектора распределения вероятностей принадлежности $\boldsymbol{X}$ к классам $\boldsymbol{y} \in \mathbb{R}^k$. Будем использовать в качестве такой функции ошибки категориальную перекрестную энтропию, определяемую в дискретном случае как:

$$H(p,q) = -\sum_x p(x) \log q(x)$$

Где p и q – распределения над вероятностным пространством принадлежности ключевого слова к k классам, причем p является истинным вероятностным распределением, q – вычисленное моделью распределение вероятностей. В данном случае, поскольку каждый обучающий пример истинно может принадлежать к единственному классу c, p определяется как:

$$p(c_i) = \begin{cases} 1, & \boldsymbol{X} \in c_i \\ 0, & \boldsymbol{X} \notin c_i \end{cases}, \qquad i = \overline{1,k}$$

Учитывая тот факт, что предсказанный моделью вектор классов $\boldsymbol{y}$ представляет собой распределение вероятностей принадлежности ключевого слова к k классам, т. е. $\boldsymbol{y}^i = q[\boldsymbol{X} \in c_i]$, формулу для определения категориальной кросс-энтропии можно записать в виде:

$$H(p,q) = -\sum_{i=1}^{k} p[\boldsymbol{X} \in c_i] \log q[\boldsymbol{X} \in c_i] = -\sum_{i=1}^{k} p[\boldsymbol{X} \in c_i] \log \boldsymbol{y}^i$$

Обобщая на случай N примеров, получим:

$$H(p,q) = -\frac{1}{N} \sum_{r=1}^{N} \sum_{i=1}^{k} p[\boldsymbol{X_r} \in c_i] \log \boldsymbol{y}_r^i$$

С учетом наличия двух целевых вероятностных распределений для классов и подклассов ключевых слов соответственно, обобщим функцию ошибки следующим образом:

$$H = -\frac{1}{N}\sum_{r=1}^{N}\left[\sum_{i=1}^{k} p[\boldsymbol{X_r} \in c_i]\log \boldsymbol{y}_r^i + \sum_{j=1}^{l} p\left[\boldsymbol{X_r} \in s_j\right]\log \boldsymbol{z}_r^j\right]$$

Где N – общее количество обучающих примеров;

$\boldsymbol{X_r}$ – матрица MFCC признаков для $r^{-\text{го}}$ примера, $r = \overline{1, N}$;

c_i – $i^{-\text{й}}$ класс ключевого слова, $i = \overline{1, k}$;

$\boldsymbol{y_r} \in \mathbb{R}^k$ – вектор распределения вероятностей принадлежности $\boldsymbol{X_r}$ к классам ключевых слов, определенный моделью;

s_j – $j^{-\text{й}}$ подкласс ключевого слова, $j = \overline{1, l}$;

$\boldsymbol{z_r} \in \mathbb{R}^l$ – вектор распределения вероятностей принадлежности $\boldsymbol{X_r}$ к подклассам ключевых слов, определенный моделью.

Данная функция является дифференцируемой по переменным выходного слоя нейронной сети $\boldsymbol{y^i}$ и $\boldsymbol{z^j}$, что делает возможным ее использование в итеративном алгоритме обучения модели методом обратного распространения ошибки.

2.4 Процесс обучения модели

В качестве обучающих данных для каждой из описанных моделей используются тройки *(аудио, класс, подкласс)*. Корпус данных для обучения разбивается на три непересекающиеся выборки: *train dataset*, *validation dataset* и *test dataset* в соотношении 60:20:20 соответственно. *Train dataset* используется для обучения весов модели. *Validation dataset* используется во время обучения для объективной оценки степени соответствия модели обучающему набору данных и предотвращения переобучения модели. *Test*

dataset используется для независимого вычисления метрик качества уже обученной модели.

Обучение модели осуществляется с помощью метода обратного распространения ошибки. Одна итерация состоит из следующих шагов:

1. Из обучающих аудио семплов извлекаются MFCC и подаются на вход модели
2. С помощью весов нейронной сети над полученной матрицей признаков осуществляются преобразования, в результате которых на выходе нейронной сети получаем кортеж *(класс, подкласс)*, где каждый из векторов отражает распределение вероятностей принадлежности анализируемого семпла к соответствующему классу (подклассу).
3. Для вычисления ошибки между полученными и реальными классами в качестве функции потерь используется категориальная кросс-энтропия. Далее с помощью метода стохастического градиентного спуска (SGD) либо его модификации (Adam) [4] происходит корректировка весов модели.

Процесс обучения продолжается до тех пор, пока значение функции потерь на выборке *validation dataset* не начинает увеличиваться на протяжении нескольких эпох обучения.

2.5 Выводы

1. Описана математическая формулировка задачи распознавания заданного множества ключевых слов из аудио постоянной длины.
2. Разработаны три архитектуры модели, удовлетворяющие условиям описанной задачи.
3. Выбрана и описана функция потерь для использования при обучении модели.

4. Описан процесс обучения модели методом обратного распространения ошибки с помощью выбранной функции потерь.

Глава 3 Сбор данных для обучения моделей

Для обучения KWS модели на основе DNN требуется большое количество размеченных данных. Ввиду отсутствия в открытом доступе достаточно больших корпусов русскоязычных данных, пригодных для решения задачи распознавания ключевых слов, было решено собрать данные самостоятельно. Оптимальным решением задачи сбора данных является разработка метода для автоматического сбора данных из открытых источников, поскольку ручные подходы требуют высоких трудозатрат и большого количества времени.

Целью данной главы является создание и тестирование корпуса ключевых слов для построения моделей, способных распознавать произнесенное слово из набора ключевых слов с наименьшим уровнем ошибки, которое возможно при наличии в семплах фонового шума и сторонней речи.

В данной главе описываются методы автоматического сбора русскоязычных аудио данных, а также их применение при создании корпуса ключевых слов для последующего использования в обучении модели распознавания ключевых слов.

3.1 Обоснование требований к корпусу данных

Для успешного решения задачи активации девайса по распознанному ключевому слову, описанной выше, необходимо принять во внимания следующие предположения: использование в качестве обучающих данных аудиофайлов, записанных в студии звукозаписи с микрофонами высокого качества, не даст положительного результата, поскольку в таких аудио будут отсутствовать посторонние шумы, качество звукозаписывающего оборудования конечных пользователей в большинстве случаев не сравнится с профессиональным, а манера произношения «искусственных» семплов будет отличаться от разговорной. Кроме того, корпус должен основываться на

русскоязычных данных, что существенно сужает круг источников подходящих данных. Наконец, корпус данных должен содержать как можно больше голосов и произношений различных людей, поскольку модель распознавания ключевых слов, не ориентированная на конкретного человека, более функциональна.

В различных источниках [49; 11] для задачи KWS в качестве обучающих данных используются аудио, длина которых варьируется от 0.6 до 1 секунды. Ограничим длину каждого семпла промежутком в 0.8 секунды, поскольку данное значение позволит полностью включить в семпл ключевое слово вместе с небольшой окрестностью, что позволит устранить недостаток неточного выравнивания ключевого слова в заданном временном промежутке. Более того, выбранное значение исключает слишком длинные слова, что также является преимуществом, поскольку в качестве ключевых слов обычно выбираются короткие слова.

Для систем KWS используется ограниченный набор слов. С одной стороны, модель должна удовлетворять аппаратным ограничениям и поддерживать легковесность процесса распознавания, что определяет верхнюю границу количества слов в наборе. С другой стороны, корпус данных должен обладать достаточной разнообразностью для обеспечения применения обученной модели в различных приложениях, что определяет нижнюю границу количества ключевых слов. Таким образом, исходя из назначения KWS модели, в множество ключевых слов были включены цифры от 0 до 9, слова «Да», «Нет», «Влево», «Вправо», «Вперед», «Назад», «Север», «Юг», «Восток», «Запад», «Иди», «Начать», «Стоп». Также для включения в множество ключевых слов были выбраны 5 женских и 5 мужских распространенных русских имен: «Иван», «Николай», «Егор», «Антон», «Вадим», «Мария», «Любовь», «Оксана», «Наталия», «Карина».

Одной из наиболее трудных проблем в задаче KWS является способность модели игнорировать речь, которая не содержит слова-триггеры

– для этого в множество ключевых слов необходимо включить семплы, содержащие сторонний шум, а также слова, не входящие ни в один из вышеуказанных подклассов ключевых слов. В качестве таких слов были выбраны «Степь», «Лук», «Дерево». Обоснованием для выбора проверочных слов послужила их созвучность с некоторыми словами, принадлежащими множеству ключевых слов.

Для расширения функциональности KWS модели, элементы из множества ключевых слов объединены в классы способом, указанным в таблице 2.

Таблица 2. Разбиение ключевых слов на классы

Класс	Подкласс (ключевое слово)
«Цифра»	«Ноль», «Один», «Два», «Три», «Четыре», «Пять», «Шесть», «Семь», «Восемь», «Девять»
«Согласие»	«Да», «Нет»
«Направление»	«Влево», «Вправо», «Вперед», «Назад», «Север», «Юг», «Восток», «Запад»
«Действие»	«Иди», «Начать», «Стоп»
«Имя»	«Иван», «Николай», «Егор», «Антон», «Вадим», «Мария», «Любовь», «Оксана», «Наталия», «Карина»
«Без категории»	-

3.2 Формат данных

Набор данных представляет собой совокупность директорий, соответствующих классам ключевых слов. Каждая директория содержит в

себе поддиректории, соответствующие ключевым словам в подклассе и содержащие аудиофайлы – фрагменты русскоязычной речи в формате wav. Аудиофайлы формата wav имеют один канал (моно), частоту дискретизации 16000 Гц, длину 0.8 секунды и закодированы с шириной 2 байта для каждого значения.

В контексте поставленной задачи мы имеем дело с двумя признаками данных: классом («Цифра», «Согласие», «Направление», «Действие», «Имя», «Без категории») и подклассом («Ноль», «Один», «Два», «Три», «Четыре», «Пять», «Шесть», «Семь», «Восемь», «Девять», «Да», «Нет», «Влево», «Вправо», «Вперед», «Назад», «Север», «Юг», «Восток», «Запад», «Иди», «Начать», «Стоп», «Иван», «Николай», «Егор», «Антон», «Вадим», «Мария», «Любовь», «Оксана», «Наталия», «Карина», «Без категории»). Поскольку классы и подклассы ключевых слов могут принимать значения из конечного и заранее заданного множества, удобно кодировать их с помощью метода One-Hot encoding: каждому уникальному значению признака класса и признака подкласса соответствует вектор, состоящий из единицы на позиции, соответствующей нужному классу (подклассу) и нулей на оставшихся позициях. Для разбиения данных на обучающую, тестовую и проверочную выборки в python скрипте из всех аудио извлекаются MFCC, закодированные классы и подклассы, затем данные случайным образом перемешиваются, после чего разделяются в заранее заданном соотношении 60:20:20 для обучающей, тестовой и проверочной выборок соответственно.

Таким образом, структурной единицей набора данных является тройка (X, Y_class, Y_subclass), где X представляет собой двумерную матрицу признаков аудио размерностью (20, 494), Y_class – вектор класса размерностью (6, 1), Y_subclass – вектор подкласса размерностью (10, 1).

3.3 Использование аудио и автоматически-сгенерированных субтитров из видеофайлов сервиса YouTube для создания корпуса ключевых слов

На первом этапе процесса получения аудио семплов ключевых слов производится скачивание аудио и автоматически сгенерированных субтитров YouTube в форматах wav и vtt соответственно. Для получения идентификаторов плейлистов осуществляется итерирование по текстовому файлу, содержащему список URL. Этот файл заполняется ссылками на наиболее релевантные плейлисты вручную. Скачивание субтитров и аудио производится при помощи утилиты youtube-dl.

Второй этап заключается в обработке и индексации скаченных субтитров. Автоматически-сгенерированные субтитры в формате vtt содержат время начала и окончания каждого распознанного слова. В python скрипте производится выделение слова и соответствующих ему временных границ, после чего производится индексация результата в экземпляре свободной программной поисковой системы ElasticSearch в формате {filename, start, end, text} для дальнейшей обработки данных.

На третьем шаге в python скрипте производится итерирование по списку ключевых слов из заполненного вручную файла и поиск соответствий ключевых слов и транскриптов. Поскольку при определении ключевых слов необходимо учитывать их разнообразные формы (падеж, число и т. д.), для каждого ключевого слова осуществляется запрос к экземпляру ElasticSearch по поиску неточных соответствий (Fuzzy-query). В качестве критерия соответствия найденного и ключевого слов берется расстояние Левенштейна [33] между этими словами. Пороговое значение расстояния редактирования, при котором найденное слово вносится в корпус данных равно 1.

Для каждого найденного соответствия, согласно временным границам, производится обрезка аудио длиной в 0.8 секунды, начиная с определенного в

субтитрах начала слова, нормализация громкости к уровню -10 децибел, обработка полосовым частотным фильтром с границами 200 Гц и 3000 Гц, и сохранение полученного семпла в нужную директорию в заданном формате с помощью утилиты ffmpeg. Кроме того, для цифр отдельно осуществляется поисковые запросы, содержащие не прописные, а численные значения, поскольку автоматически-сгенерированные субтитры YouTube содержат оба варианта написания цифр.

Полученные фрагменты сохраняются в соответствующую директорию, согласно описанной в части 3.2 структуре. Таким образом, после окончания всех итераций мы получаем готовый набор аудио фрагментов ключевых слов.

3.4 Корректировка и выравнивание временных границ ключевых слов, полученных из автоматически-сгенерированных субтитров

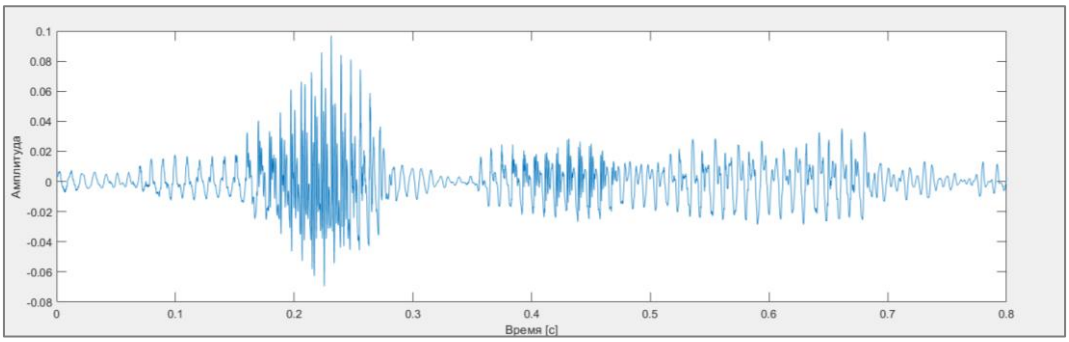
Путем выборочной ручной проверки соответствия ключевых слов и собранных аудио семплов было обнаружено, что автоматически сгенерированные субтитры YouTube в большинстве случаев предоставляют неточное время начала слова, из-за чего в аудио семпл частично попадает ненужная информация и, что более значимо, отрезок аудио длиной 0.8 секунды от начала, взятого из автоматически-сгенерированных субтитров, может содержать менее половины ключевого слова. Таким образом, возникла необходимость в выравнивании временных границ ключевых слов и их отцентровки относительно указанной длительности семпла.

Первым способом выделения точных границ слова является поиск окружающих промежутков тишины. С помощью Voice Activity Detector из open-source проекта WebRTC локализировались ближайшие к предполагаемому началу слова-триггера промежутки тишины в пределах 0.5 секунды. Однако, после нескольких экспериментов с различными классами ключевых слов было установлено, что данных метод плохо работает на

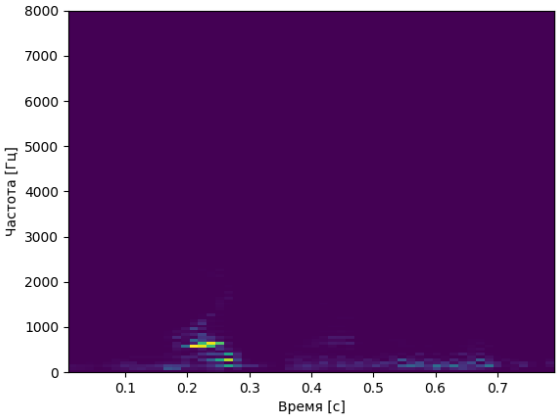
цифрах, поскольку обычно в речи за цифрами следуют единицы измерения, например, «два часа», «три дня», произносимые без ярко выраженной паузы, вследствие чего промежуток аудио с определенными границами содержит в себе два слова.

Иным подходом к уточнению границ ключевого слова стал анализ спектрограммы аудио. Поскольку слово разговорной речи чаще всего имеет длину менее 0.8 секунды, для решения задачи выделения ключевого слова из потока речи по неточным границам достаточно установить местоположения центра слова во временном ряде x_c и вырезать из исходного аудиофайла отрезок $[\, x_c - 0.4;\ x_c + 0.4\,]$. Для определения центра слова в окрестности 0.5 секунды от его предполагаемого начала с помощью последовательных оконных преобразований Фурье вычисляется спектрограмма отрывка аудио. Спектрограмма показывает зависимость спектральной плотности мощности сигнала от времени. Опытным путем было установлено, что локальные максимумы спектрограммы соответствуют частям слова, наиболее выделяемым голосом при произношении, таким образом, правильно подобрав пороговое значение, получается вычислить количество слов в анализируемом промежутке аудио, а также получить маркеры их местоположения во временном ряде.

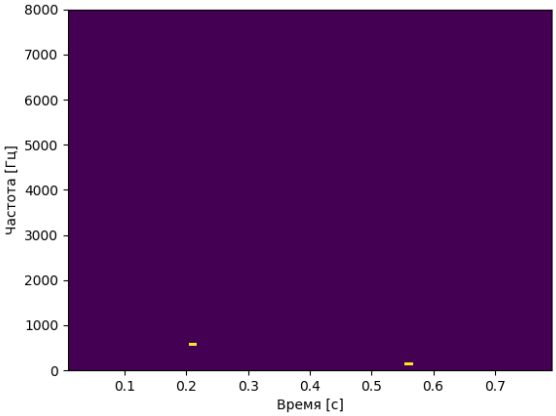
Этапы выделения локальных максимумов спектрограммы на примере произношения фразы «два, наверное» приведены на рисунке 4.



А) Исходный семпл



Б) Спектрограмма семпла



В) Локальные максимумы спектрограммы

Рис. 4. Этапы выделения локальных максимумов спектрограммы аудио

После нахождения локальных максимумов спектрограммы производится их фильтрация по частоте с верхним пороговым значением 3500 Гц, что удовлетворяет верхней границе речевой полосы частот, используемой в телефонии. Таким образом, из полученного множества маркеров ключевых слов выбирается ближайший к известному времени начала слова, полученному из автоматически-созданных субтитров. Найденный маркер считается центром слова x_c, после чего из исходного аудио вырезается фрагмент $[\, x_c - 0.4;\ x_c + 0.4\,]$ и сохраняется в соответствующую директорию корпуса данных.

3.5 Использование TTS сервиса для синтеза данных

Для автоматического синтеза аудио семплов ключевых слов был использован TTS сервис из пакета Microsoft Azure Cognitive Speech Services.

Из предоставленных сервисом были выбраны 40 дикторов, 3 из которых ориентированы непосредственно на русский язык.

Ввиду ограниченного количества доступных дикторов, первым этапом подготовки синтезированный данных стало составление списка синонимов, близких по звучанию, для каждого ключевого слова. Для дикторов, поддерживающих русский язык, список синонимов был составлен с использованием кириллицы; для остальных были использованы наиболее близкие по произношению к русскоязычным словам латинские транскрипты. Примером множества синонимов к ключевому слову «Влево» является набор: «Налево», «Левый», «Слева» в кириллическом написании и «Nalevo», «Vlevo», «Sleva», «Leviy» в латинском написании. Затем список синонимов был озвучен при помощи выбранного TTS сервиса, после чего аудио семплы были сохранены в соответствующие директории корпуса данных.

Необходимо отметить, что полученные семплы не содержат сторонних шумов, в то время как зашумленные обучающие данные позволяют модели адаптироваться к реальным условиям, поэтому следующим этапом подготовки синтезированных данных стало наложение белого гауссовского шума (часть 3.7.1).

3.6 Промежуточные результаты

С использованием методов, описанных в частях 3.3-3.5 за несколько дней удалось собрать набор аудио семплов в описанном формате, содержащий в себе 11801 примеров произношений ключевых слов различными людьми, а также синтезированных автоматически. Собранные семплы вошли в корпус данных «yt_tts_clean».

Сравнивая размер полученного набора данных с существующими корпусами данных, находящимися в открытом доступе, обучение на которых дает хорошие результаты по точности [6; 49], можно сделать вывод, что полученных данных недостаточно для дальнейшего успешного продолжения

работы. Для получения точности распознавания в задаче KWS, сопоставимой с результатами, демонстрируемыми в [27] на корпусе данных «The Speech Commands dataset» [49], необходимо, чтобы корпус данных содержал не менее 1000 примеров для каждого ключевого слова.

Ввиду ограниченного количества времени альтернативным методом расширения собранного корпуса данных является модификация уже собранных семплов.

3.7 Аугментация корпуса данных

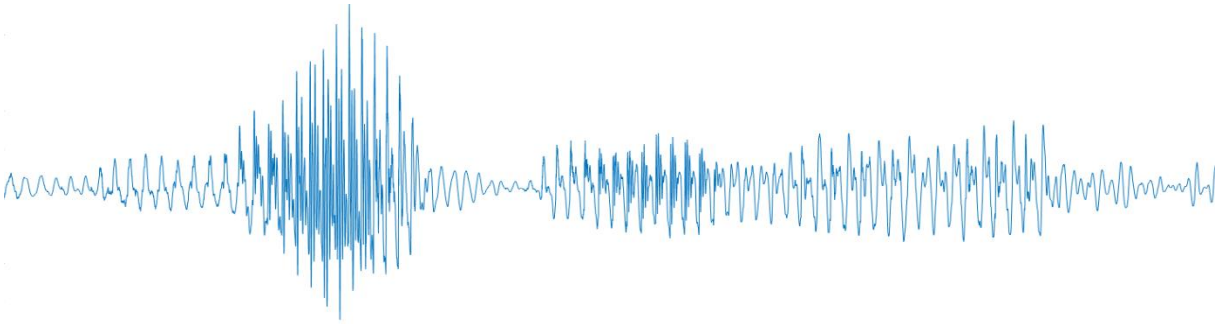
Принимая во внимание подходы, описанные в статье [30], можно выделить несколько методов модификации аудиоданных для искусственного увеличения количества обучающих примеров: изменение скорости аудио и наложение аддитивного шума. Данные методы позволяют увеличить размер корпуса данных в несколько раз и зарекомендовали себя в практических задачах распознавания речи, их применение целесообразно в случае нехватки оригинальных данных для обучения. Рассмотрим подробнее каждый из методов и процесс его применения к собранному корпусу данных.

3.7.1 Наложение белого гауссовского шума

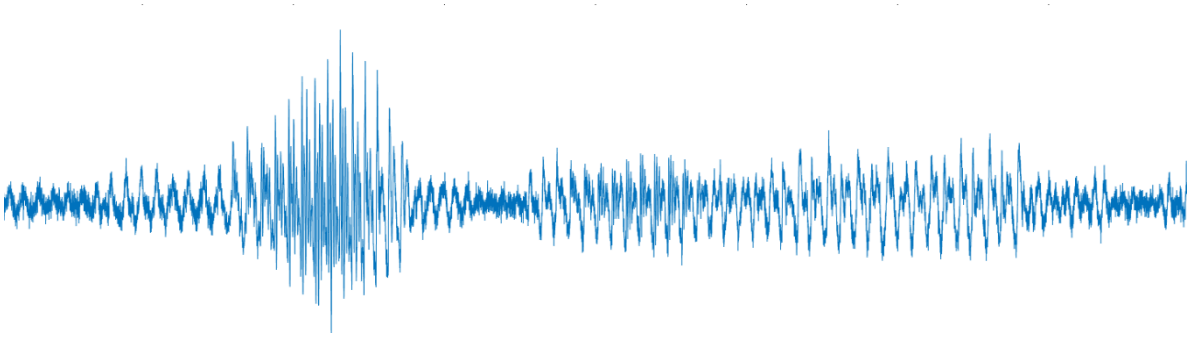
Аддитивный белый гауссовский шум представляет собой мешающее воздействие (в данном случае - на аудио сигнал) и характеризуется равномерной спектральной плотностью мощности, нормально распределенными временными значениями и аддитивным способом воздействия на сигнал. Добавление шума с подобранными параметрами к аудио позволяет предотвратить переобучение модели и повысить ее устойчивость к изменению входных данных.

Для добавления шума в python скрипте для каждого семпла вычисляется абсолютное среднее амплитуды входного сигнала $\bar{x}$. В качестве параметров для Гауссовского распределения значений шума выбираются математическое ожидание $\mu = 0$ и среднеквадратическое отклонение $\sigma = \sqrt{k * \bar{x}}$, где k –

коэффициент степени зашумления. В данном случае использовалось эмпирически выбранное значение $k = 10$. После генерации шум добавляется с полезным сигналом и модифицированный семпл сохраняется в соответствующую директорию. Применение данного метода позволило увеличить объем корпуса данных в 2 раза. Рисунок 5 иллюстрирует процесс добавления шума к аудио.



А) Исходный семпл



Б) Семпл с наложением аддитивного белого гауссовского шума

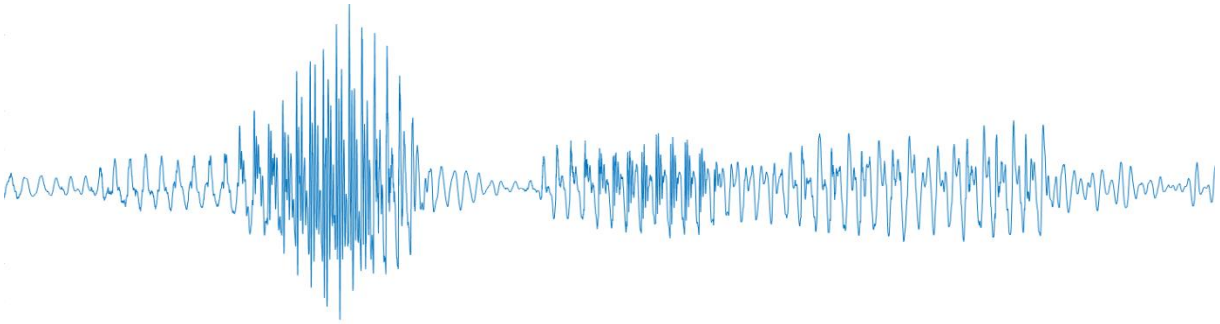
Рис. 5. Аугментация данных путем наложения шума

3.7.2 Изменение скорости

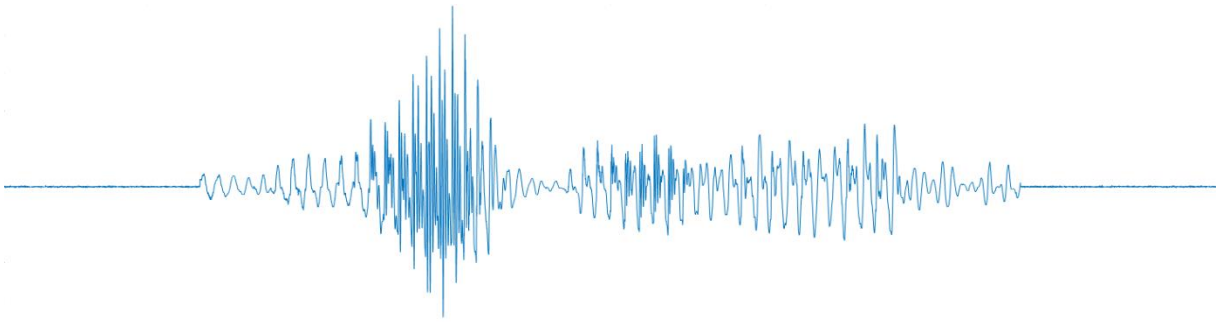
Увеличение скорости аудио без изменения частоты его дискретизации достигается путем усреднения значений сигнала в заданном диапазоне семплов. Увеличение скорости входного аудио сигнала позволяет адаптировать модель к различным темпам речи и качеству обучающих данных.

В python скрипте с помощью утилиты ffmpeg для каждого имеющегося семпла производится увеличение его скорости в 1.5 раза. К полученному

фрагменту слева и справа симметрично добавляется аддитивный белый гауссовский шум для восстановления исходной длительности семпла в 0.8 секунды. Данный метод позволил увеличить объем корпуса данных в 2 раза. Рисунок 6 иллюстрирует процесс ускорения семпла.



А) Исходный семпл



Б) Ускоренный семпл с добавлением шума по краям

Рис. 6. Аугментация данных путем ускорения аудио

3.8 Результирующие наборы данных

После применения вышеописанных методов были собраны 2 набора данных для обучения KWS модели: «yt_tts_clean» и «yt_tts_augmented», содержащие 11801 и 47204 примеров соответственно.

В таблице 3 указано количество примеров для каждого ключевого слова, а также общее количество примеров для каждого класса в собранных наборах данных.

Таблица 3. Результирующее количество примеров в наборах данных «yt_tts_clean» и «yt_tts_augmented»

Класс	Общее количество примеров		Подкласс	Количество примеров	
	«yt_tts_clean»	«yt_tts_augmented»		«yt_tts_clean»	«yt_tts_augmented»
«Цифра»	4767	19068	«Ноль»	423	1692
			«Один»	537	2148
			«Два»	548	2192
			«Три»	531	2124
			«Четыре»	453	1812
			«Пять»	428	1712
			«Шесть»	565	2260
			«Семь»	493	1972
			«Восемь»	378	1512
			«Девять»	411	1644
«Направление»	2598	10392	«Вперед»	453	1812
			«Назад»	383	1532
			«Влево»	306	1224
			«Вправо»	305	1220
			«Север»	231	924
			«Юг»	245	980
			«Восток»	317	1268
			«Запад»	358	1432
«Согласие»	585	2340	«Да»	303	1212
			«Нет»	282	1128
«Действие»	577	2308	«Стоп»	125	500
			«Иди»	177	708
			«Начать»	275	1100

«Имя»	2688	10752	«Иван»	307	1228
			«Николай»	215	860
			«Егор»	377	1508
			«Антон»	288	1152
			«Вадим»	276	1104
			«Мария»	275	1100
			«Любовь»	257	1028
			«Оксана»	245	980
			«Наталия»	220	880
			«Карина»	228	912
«Без категории»	586	2344	-	586	2344

3.9 Выводы

1. Описан и обоснован формат корпуса данных для обучения нейронной сети распознаванию ключевых слов.
2. Разработан метод автоматического сбора обучающих данных из видеохостинга YouTube с использованием анализа спектрограммы аудио.
3. С помощью описанного метода, а также синтеза данных с использованием TTS сервиса составлен набор данных «yt_tts_clean», состоящий из 11801 семпла для 34 различных ключевых слов, принадлежащих 6 классам.
4. Изучены и применены несколько способов аугментации аудио данных, с помощью которых получен набор данных «yt_tts_augmented», содержащий 47204 семпла.

Глава 4 Обучение моделей

Целью данной главы является обучение разработанных в главе 2 моделей распознавания ключевых слов на собранных наборах данных и их сравнение.

В данной главе приводится обзор гиперпараметров разработанных моделей и обучения, описывается алгоритм их оптимизации, а также результаты обучения моделей на наборах данных «yt_tts_clean» и «yt_tts_augmented». Обученные модели сравниваются между собой с учетом показателей точности распознавания и количества обучаемых параметров.

4.1 Выбор и оптимизация гиперпараметров моделей и обучения

4.1.1 Обзор гиперпараметров моделей и обучения

На качество обучения и дальнейшую точность обученной модели влияет совокупность гиперпараметров:

1. **learning_rate** – коэффициент скорости обучения, контролирующий величину коррекции весов на каждой итерации обучения. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров.
2. **epoch** – число эпох, или количество полных итераций оптимизационного алгоритма по обучающим данным. Для итерации алгоритма оптимизации гиперпараметров выбирается достаточно большое число = 150, позволяющее отследить тенденцию процесса обучения, поскольку количество эпох до момента начала увеличения значения функции потерь у моделей с разными гиперпараметрами может сильно варьироваться.
3. **batch_size** – число примеров в пакете при обучении, валидации и тестировании модели. Выбирается в зависимости от размера набора

данных и доступного объема видеопамяти. Используется стандартное значение = 32.

4. **dropout_rate** – вероятность случайного выбора нейрона для его исключения из итерации процесса обучения. Обычно варьируется в пределах от 0.2 до 0.4. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров.
5. **optimizer** – алгоритм оптимизации, применяющийся для корректировки весов модели во время обучения. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров из двух хорошо зарекомендовавших себя алгоритмов: SGD – стохастического градиентного спуска, и его модификации - Adam [4].
6. **conv_size** – количество фильтров в сверточном слое. Значение данного параметра зависит от объема и разнообразия обучающих данных. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров.
7. **separable_conv_size** – количество фильтров в DS-C слое на этапе точечной свертки. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров.
8. **dense_size** – количество нейронов в скрытом полносвязном слое. Зависит от вариативности обучающих данных, а также от размерности следующего за ним выходного слоя. Выбирается в соответствии с алгоритмом оптимизации гиперпараметров.

4.1.2 Алгоритм оптимизации гиперпараметров

Для индивидуальной оптимизации рассмотренных гиперпараметров, т. е. выбора значений, обеспечивающих наибольшую точность распознавания для каждой из моделей, был использован инструмент Microsoft Neural Network Intelligence (NNI). NNI позволяет автоматизировать процесс подбора оптимальных параметров для модели глубинного обучения путем решения задачи нахождения экстремума целевой функции многих переменных

(например, максимизации функции точности модели в зависимости от ее гиперпараметров), выполняя следующие шаги:

1. В файл, содержащий описание модели глубинного обучения, разработчиком вносятся изменения, позволяющие использовать в качестве гиперпараметров модели набор переменных, полученный с помощью NNI, а также передающие результат обучения модели в оптимизирующий алгоритм NNI.
2. Разработчиком также задается файл конфигурации эксперимента, содержащий в себе такие параметры, как путь к модели, максимальную длительность эксперимента, оптимизирующий алгоритм, максимальное количество его итераций, а также параметры аппаратного выполнения эксперимента и другие.
3. Из пространства гиперпараметров модели[1], задаваемого разработчиком в отдельном файле, оптимизирующим алгоритмом выбирается набор гиперпараметров.
4. В соответствии с выбранными гиперпараметрами и заданным описанием конструируется модель глубинного обучения, после чего осуществляется ее обучение на указанном наборе данных и передача результата оптимизирующему алгоритму.
5. На основе полученного результата оптимизирующим алгоритмом определяется следующий набор гиперпараметров модели таким образом, чтобы максимизировать (минимизировать) целевую функцию.
6. Шаги 3-5 повторяются до тех пор, пока не будут достигнуты максимальное количество итераций алгоритма либо максимальное время выполнения эксперимента.

[1] Пространство гиперпараметров модели – дискретное или непрерывное множество допустимых значений гиперпараметров модели

Таким образом итеративно происходит оптимизация параметров модели глубинного обучения.

Ниже приведены результаты экспериментов по оптимизации гиперпараметров для трех разработанных CNN моделей, включающие в себя описание параметров конфигурации эксперимента и пространства гиперпараметров модели, а также графики протекания процесса оптимизации и комментарии к ним.

4.1.3 Конфигурации экспериментов

Настройки параметров эксперимента указываются в конфигурационном файле формата *yaml*. В таблице 4 приведены конфигурационные параметры, использованные в файле конфигурации экспериментов, описано их назначение, а также указаны значения для проводимых экспериментов.

Таблица 4. Параметры конфигурации эксперимента NNI

Параметр	Описание	Значение
authorName	Автор эксперимента	Anna
experimentName	Название эксперимента	CNN / DS-CNN / MI-CNN
trialConcurrency	Максимальное количество итераций оптимизационного алгоритма, выполняемых одновременно	1
maxExecDuration	Максимальная длительность эксперимента	20h
maxTrialNum	Максимальное количество итераций	200

			оптимизационного алгоритма	
trainingServicePlatform			Платформа, на которой запускается эксперимент	local
searchSpacePath			Путь к файлу, описывающему пространство гиперпараметров модели	searchspace.json
tuner	builtinTunerName		Оптимизационный алгоритм	TPE[2]
	classArgs	optimize_mode	Режим оптимизации метрики	maximize
trial	command		Команда для запуска итерации	python3 model.py
	codeDir		Директория файлов эксперимента	.
	gpuNum:		Номер GPU для запуска основного процесса	2

Все три эксперимента имеют одинаковую конфигурацию.

4.1.4 Описание структуры экспериментов

Далее в пунктах 4.1.5, 4.1.6 и 4.1.7 для разработанных моделей CNN, DS-CNN и M-CNN соответственно приводятся таблицы, содержащие множества

[2] Tree-structured Parzen Estimator [42], алгоритм оптимизации гиперпараметров общего назначения, особенно хорошо выступающий в случаях ограниченных вычислительных ресурсов и превосходящий RandomSearch в случае большого количества экспериментов.

допустимых значений гиперпараметров модели и обучения. Совокупность гиперпараметров, указанных в каждой таблице, представляет собой пространство поиска и хранится в файле *searchspace.json*.

Для каждой модели проводится процесс оптимизации гиперпараметров по описанному в части 4.1.2 алгоритму с конфигурацией эксперимента, указанной в 4.1.3 и пространством поиска, заданным соответствующей таблицей (таблицы 5, 7, 9). Во время оптимизации для обучения и оценки точности моделей используется часть набора данных «yt_clean».

Для каждого эксперимента также приводится график, отображающий процесс оптимизации гиперпараметров модели и обучения (рисунки 7, 9, 11). По оси абсцисс отмечены номера итераций оптимизационного алгоритма, по оси ординат – значения метрики, используемой при оптимизации, т. е. значения категориальной точности распознавания подклассов ключевых слов.

Кроме того, для каждого эксперимента приводится график зависимости значений целевой метрики от набора гиперпараметров (рисунки 8, 10, 12). Каждый набор гиперпараметров определяется кривой, проходящей через соответствующие значения из множества допустимых значений гиперпараметра. Цвет кривых соответствует категориальной точности, получаемой при данном наборе гиперпараметров. Максимальное значение точности в рамках одного эксперимента соответствует красному цвету; минимальное – зеленому.

Для формирования вывода по результатам эксперимента для каждой модели приводится таблица, содержащая наборы гиперпараметров, соответствующие пяти лучшим и пяти худшим показателям целевой метрики во время процесса оптимизации гиперпараметров (таблицы 6, 8, 10). Строки таблиц, соответствующие лучшим и худшим показателям метрики помечены красным и зеленым цветами соответственно.

В итоге каждого эксперимента приводится краткий анализ результатов эксперимента, а также выводы об оптимальном наборе гиперпараметров для рассматриваемой модели.

4.1.5 Оптимизация параметров модели CNN

Таблица 5. Множество допустимых значений гиперпараметров модели CNN

Гиперпараметр	Множество допустимых значений
learning_rate	0.0001, 0.0002, 0.0005, 0.001
optimizer	Adam, SGD
conv0_size[3]	16, 32, 64, 128, 256
conv1_size	16, 32, 64, 128, 256
conv2_size	16, 32, 64, 128, 256
conv3_size	16, 32, 64, 128, 256
dense_size	32, 64, 128, 256, 512
dropout_rate	0.2, 0.3

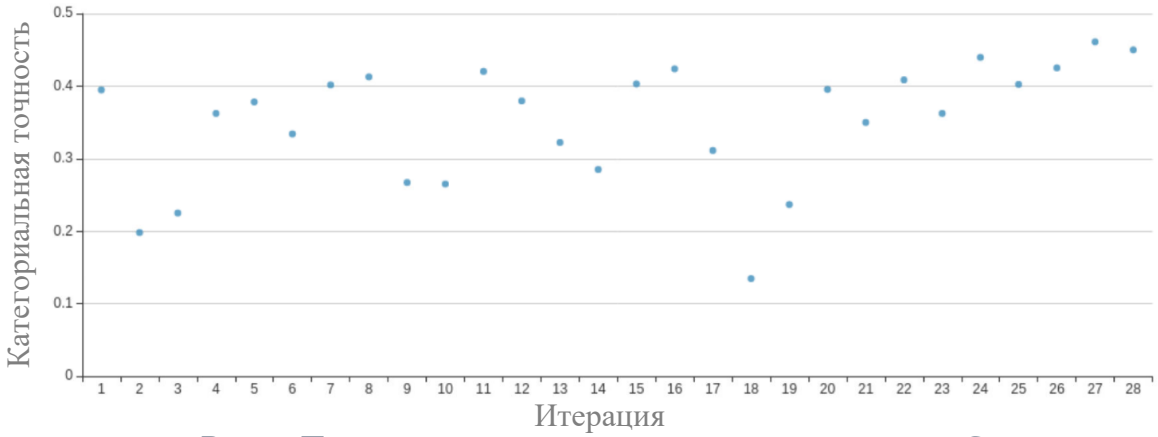


Рис. 7. Процесс оптимизации гиперпараметров модели CNN

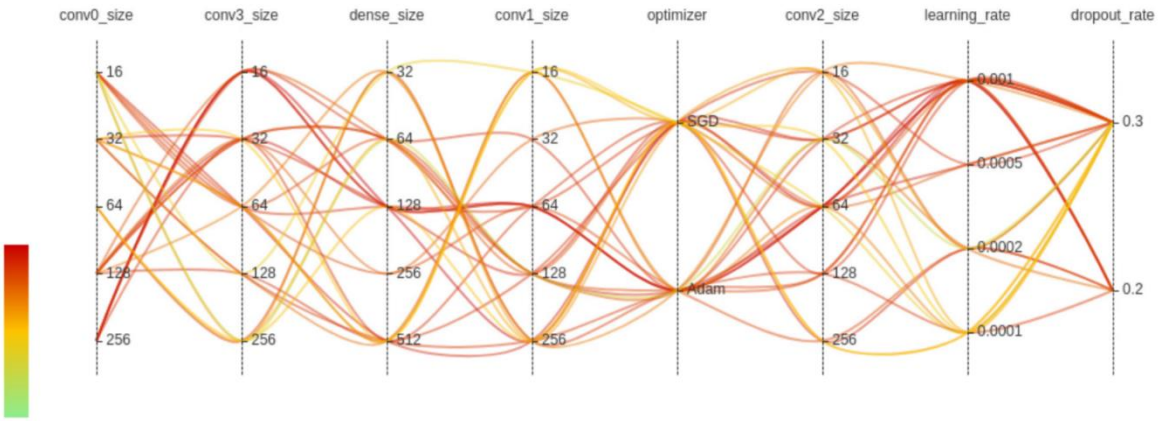


Рис. 8. График зависимости результирующей категориальной точности модели CNN от наборов гиперпараметров

[3] Здесь и далее conv[*i*]_size соответствует гиперпараметру conv_size для i-го слоя модели.

Таблица 6. Наборы параметров модели CNN, соответствующие пяти лучшим и пяти худшим значениям целевой метрики

Номер итерации	Гиперпараметры								Категориальная точность
	learning_rate	optimizer	conv0_size	conv1_size	conv2_size	conv3_size	dense_size	dropout_rate	
27	0.001	SGD	256	256	64	16	128	0.2	0.461
28	0.001	Adam	256	64	64	16	128	0.2	0.45
24	0.001	Adam	128	64	64	32	128	0.3	0.4396
26	0.0005	Adam	256	64	64	16	128	0.3	0.4251
16	0.001	SGD	16	256	32	64	512	0.3	0.4237
18	0.0002	Adam	16	128	32	256	64	0.3	0.1346
2	0.0001	SGD	16	16	64	128	32	0.3	0.1981
3	0.0001	SGD	32	16	32	32	512	0.3	0.225
19	0.0001	SGD	64	256	256	256	128	0.3	0.2367
10	0.0001	SGD	32	16	16	64	512	0.3	0.265

По результатам эксперимента можно сделать вывод о том, что в итерациях, обеспечивающих лучшие показатели точности, прослеживается тенденция к уменьшению количества фильтров в сверточных слоях по мере увеличения числа слоев (итерации 27, 28, 25, 26 таблицы 6). Неизменное значение размера скрытого полносвязного слоя на протяжении нескольких итераций с лучшей точностью говорит об оптимальности данного гиперпараметра. Кроме того, из графика зависимости значений целевой метрики от набора гиперпараметров, представленного на рисунке 8, видно, что увеличение уровня обучения в совокупности с уменьшением значения гиперпараметра dropout_rate для обоих алгоритмов оптимизации закономерно дает лучшую сходимость за время, отведенное на каждую итерацию.

Таким образом, для дальнейшего обучения модели CNN, описанной в части 2.2.1, будем использовать набор гиперпараметров, полученный в итерации 27 оптимизационного алгоритма и отображенный в таблице 6.

4.1.6 Оптимизация параметров модели DS-CNN

Таблица 7. Множество допустимых значений гиперпараметров модели DS-CNN

Гиперпараметр	Множество допустимых значений
learning_rate	0.0001, 0.0002, 0.0005, 0.001
optimizer	Adam, SGD
conv_size	16, 32, 64, 128, 256
separable_conv0_size[4]	16, 32, 64, 128, 256
separable_conv1_size	16, 32, 64, 128, 256
separable_conv2_size	16, 32, 64, 128, 256
dense_size	32, 64, 128, 256, 512
dropout_rate	0.2, 0.3

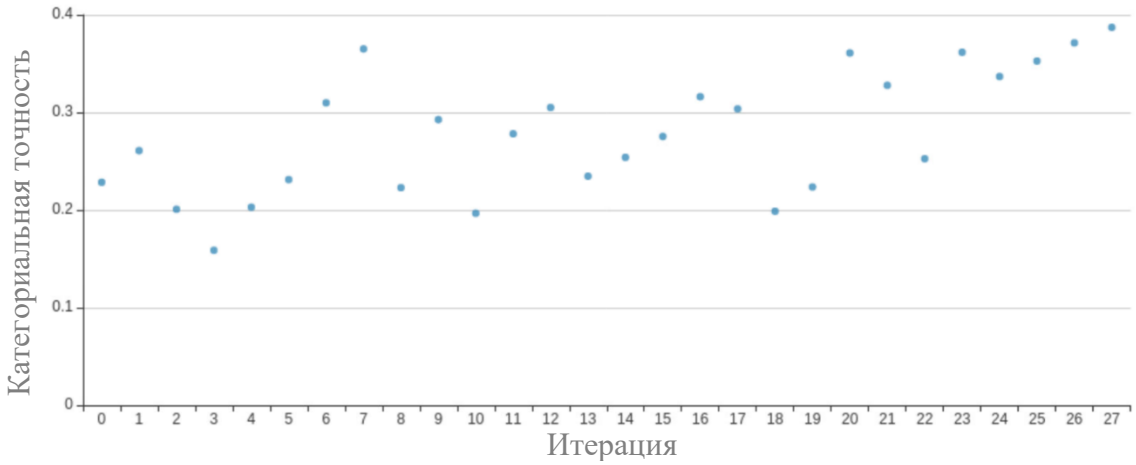


Рис. 9. Процесс оптимизации гиперпараметров модели DS-CNN

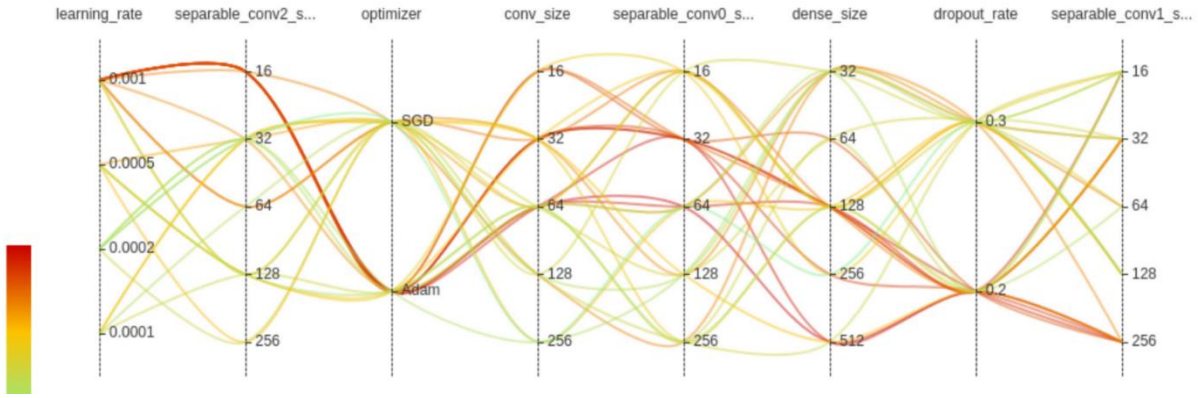


Рис. 10. График зависимости результирующей категориальной точности модели DS-CNN от наборов гиперпараметров

[4] Здесь и далее separable_conv[*i*]_size соответствует гиперпараметру separable_conv_size для i-го слоя модели.

Таблица 8. Наборы параметров модели DS-CNN, соответствующие пяти лучшим и пяти худшим значениям целевой метрики

Номер итерации	Гиперпараметры								Категориальная точность
	learning_rate	optimizer	conv_size	separable_conv0_size	separable_conv1_size	separable_conv2_size	dense_size	dropout_rate	
27	0.001	Adam	64	64	256	16	512	0.2	0.3871
26	0.001	Adam	64	32	256	16	512	0.2	0.3713
7	0.001	Adam	64	64	16	16	128	0.2	0.3651
23	0.001	Adam	32	32	256	16	128	0.2	0.3616
20	0.001	Adam	32	32	32	16	128	0.2	0.3609
3	0.0002	SGD	256	64	128	32	256	0.3	0.1587
10	0.0002	Adam	256	128	16	32	32	0.3	0.1967
18	0.0005	Adam	64	256	64	128	128	0.2	0.1988
2	0.0002	Adam	64	64	16	32	32	0.2	0.2001
4	0.0001	SGD	128	128	32	64	32	0.3	0.2029

Из результатов эксперимента видно, что наименьшее из допустимых значений уровня обучения в совокупности с наименьшим значением гиперпараметра dropout_rate и использованием алгоритма оптимизации Adam обеспечивают лучшие значения целевой метрики (итерации 27, 26, 7, 23, 20, отображенные в таблице 8). Увеличение размерности полносвязного слоя положительно влияет на показатель точности обученной модели, позволяя улавливать большее число корреляций между признаками входных данных и категориями ключевых слов (рисунок 10). Также из рисунка 10 видно, что в итерациях с более высоким значением целевой метрики использованы наименьшие значения размерности последнего DS-C слоя. Кроме того, меньшие по сравнению с CNN моделью показатели точности говорят о более медленной сходимости данной модели, что будет учтено впоследствии при задании количества эпох для обучения данной модели.

Таким образом, для дальнейшего обучения модели DS-CNN, описанной в части 2.2.2, будем использовать набор гиперпараметров, полученный в итерации 27 оптимизационного алгоритма и отображенный в таблице 8.

4.1.7 Оптимизация параметров модели M-CNN

Таблица 9. Множество допустимых значений гиперпараметров модели M-CNN

Гиперпараметр	Множество допустимых значений
learning_rate	0.0001, 0.0002, 0.0005, 0.001
optimizer	Adam, SGD
conv0_size	16, 32, 64, 128, 256
conv1_size	16, 32, 64, 128, 256
conv2_size	16, 32, 64, 128, 256
conv3_size	16, 32, 64, 128, 256
dense_size	32, 64, 128, 256, 512
dropout_rate	0.2, 0.3

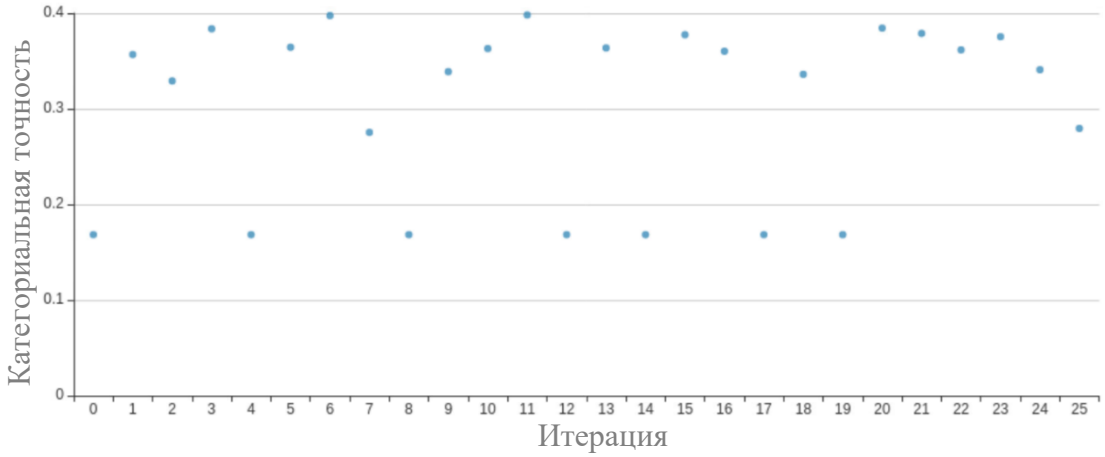


Рис. 11. Процесс оптимизации гиперпараметров модели M-CNN

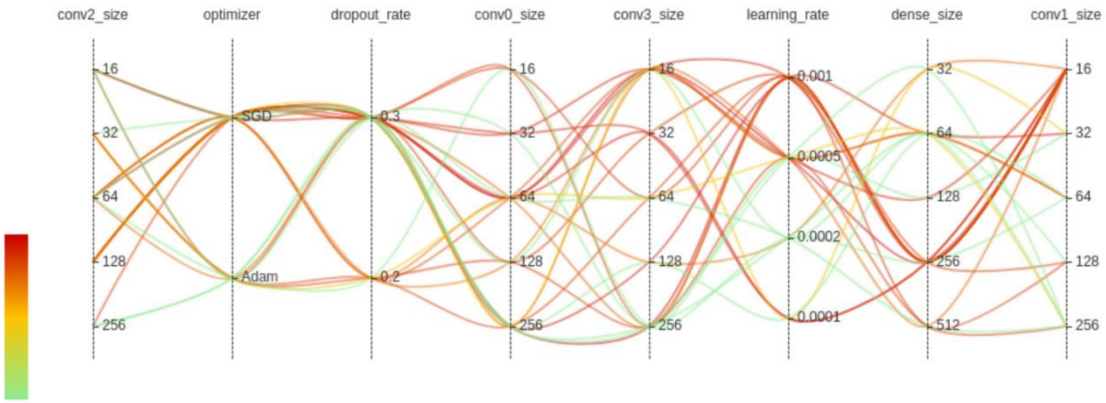


Рис. 12. График зависимости результирующей категориальной точности модели M-CNN от наборов гиперпараметров

Таблица 10. Наборы параметров модели M-CNN, соответствующие пяти лучшим и пяти худшим значениям целевой метрики

Номер итерации	Гиперпараметры								Категориальная точность
	learning_rate	optimizer	conv0_size	conv1_size	conv2_size	conv3_size	dense_size	dropout_rate	
11	0.0005	SGD	64	16	128	16	256	0.3	0.3982
6	0.001	SGD	256	16	64	256	256	0.3	0.3975
20	0.0001	SGD	64	16	128	32	256	0.3	0.3844
3	0.001	SGD	256	32	16	128	64	0.3	0.3837
21	0.001	SGD	64	16	128	16	256	0.3	0.3789
0	0.0002	SGD	128	256	64	16	64	0.3	0.1684
4	0.0001	Adam	256	64	256	128	64	0.3	0.1684
8	0.0005	Adam	32	32	16	256	128	0.3	0.1684
12	0.0005	SGD	256	256	16	256	32	0.3	0.1684
14	0.0002	Adam	16	64	64	256	256	0.2	0.1684

Проанализировав результаты эксперимента, можно утверждать, что данная архитектура более чувствительна к выбору значения параметра уровня обучения, и, например, результаты итераций 3 и 11, отображенные в таблице 10, показывают, что уменьшение значения данного параметра увеличивает итоговое значение целевой метрики при использовании стохастического градиентного спуска в качестве алгоритма оптимизации. Анализ графика на рисунке 12 показывает, что увеличение размерности полносвязного слоя позволяет учитывать большее количество признаков входных данных, что положительно сказывается на результате. Кроме того, лучшие показатели точности данной модели во время оптимизации параметров не превосходят лучших показателей точности модели CNN, что также говорит о более медленной сходимости данной модели.

Таким образом, для дальнейшего обучения модели M-CNN, описанной в части 2.2.3, будем использовать набор гиперпараметров, полученный в итерации 11 оптимизационного алгоритма и отображенный в таблице 10.

4.2 Описание корпуса данных для обучения

Поскольку конечные показатели целевой метрики модели сильно зависят от данных, на которых производится обучение и тестирование, необходимо привести описание корпуса обучающих данных.

Для обучения разработанных моделей использовались два набора данных: «yt_tts_clean» и «yt_tts_augmented». Набор «yt_tts_clean» состоит из 11801 аудио семпла ключевых слов, собранных из видеороликов YouTube, записанных вручную, а также синтезированных с помощью Microsoft Azure TTS сервиса. Набор «yt_tts_augmented» содержит в себе 47204 аудио семпла ключевых слов и представляет собой аугментированный с помощью наложения белого гауссовского шума и изменения скорости семплов набор «yt_tts_clean». Процесс сбора указанных наборов данных, а также сводная таблица, описывающая их детальное содержимое, представлены в главе 3.

4.3 Описание структуры экспериментов

Для каждой из разработанных моделей CNN, DS-CNN и M-CNN были установлены наборы гиперпараметров, полученные с помощью алгоритма оптимизации гиперпараметров в части 4.1. Каждая из моделей была обучена и протестирована на двух наборах данных: «yt_tts_clean» и «yt_tts_augmented». Обучение моделей проводилось до момента, когда значение функции потерь не начинало возрастать на протяжении десяти эпох – данное событие служит индикатором начала переобучения модели.

Для каждого эксперимента приводятся:

а) график изменения значения функции ошибки на обучающей и тестовой выборках во время обучения (рисунки 13, 16, 19, 22, 25, 28)

б) график изменения значения категориальной точности для классов ключевых слов на тестовой выборке во время обучения (рисунки 14, 17, 20, 23, 26, 29)

в) график изменения значения категориальной точности для подклассов ключевых слов на тестовой выборке во время обучения (рисунки 15, 18, 21, 24, 27, 30)

Также в таблице приводятся результирующие значения категориальной точности для классов и подклассов ключевых слов, вычисленные на проверочной выборке (таблицы 11-16).

4.4 Эксперименты

4.4.1 Обучение модели CNN на наборе данных «yt_tts_clean»

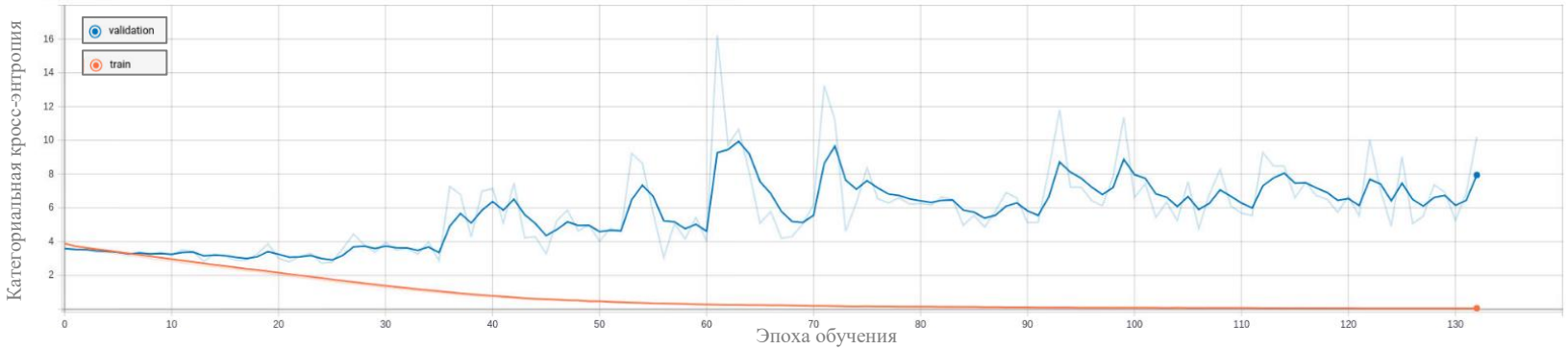


Рис.13. График изменения функции ошибки модели CNN на обучающей и тестовой выборках «yt_tts_clean»

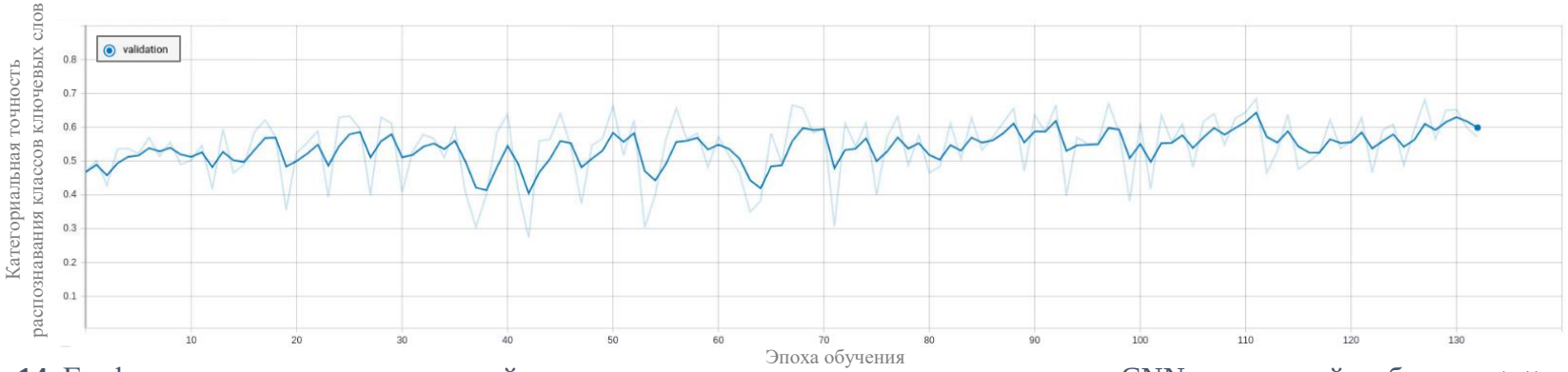


Рис. 14. График изменения категориальной точности для классов ключевых слов модели CNN на тестовой выборке «yt_tts_clean»

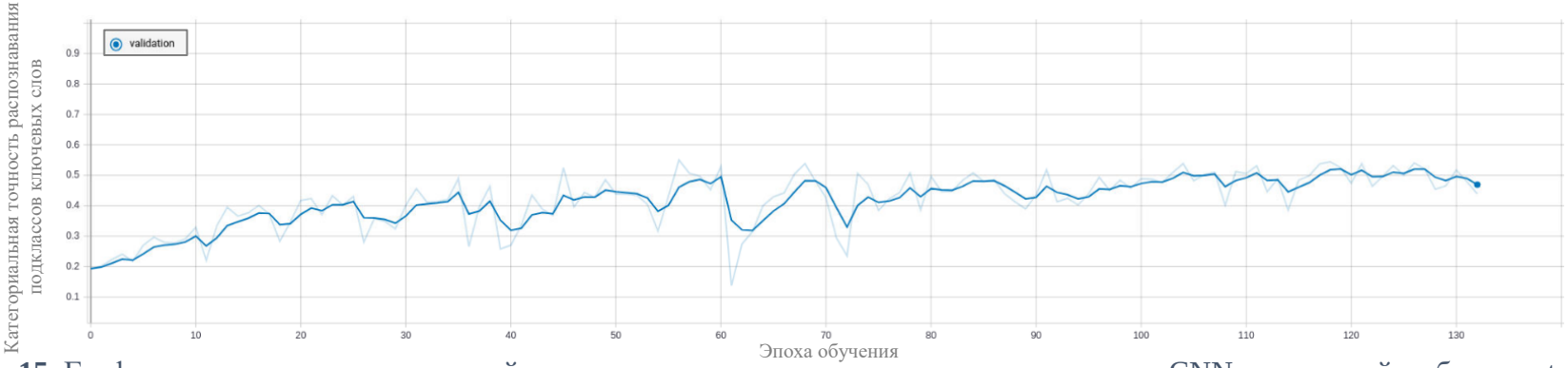


Рис. 15. График изменения категориальной точности для подклассов ключевых слов модели CNN на тестовой выборке «yt_tts_clean»

Таблица 11. Результаты обучения модели CNN на наборе данных «yt_tts_clean»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
60.24%	45.64%

В результате обучения CNN модели на наборе данных «yt_tts_clean» были достигнуты максимальные значения категориальной точности 60.24% для классов ключевых слов и 45.64% для подклассов ключевых слов (таблица 11). Несмотря на то, что переобучение модели началось уже после десятой эпохи обучения, показатели точности на тестовой выборке продолжали расти до конца обучения, что можно объяснить использованием dropout в архитектуре модели. Большая разница в точности распознавания между классами и подклассами ключевых слов объясняется меньшей размерностью множества классов ключевых слов.

4.4.2 Обучение модели CNN на наборе данных «yt_tts_augmented»

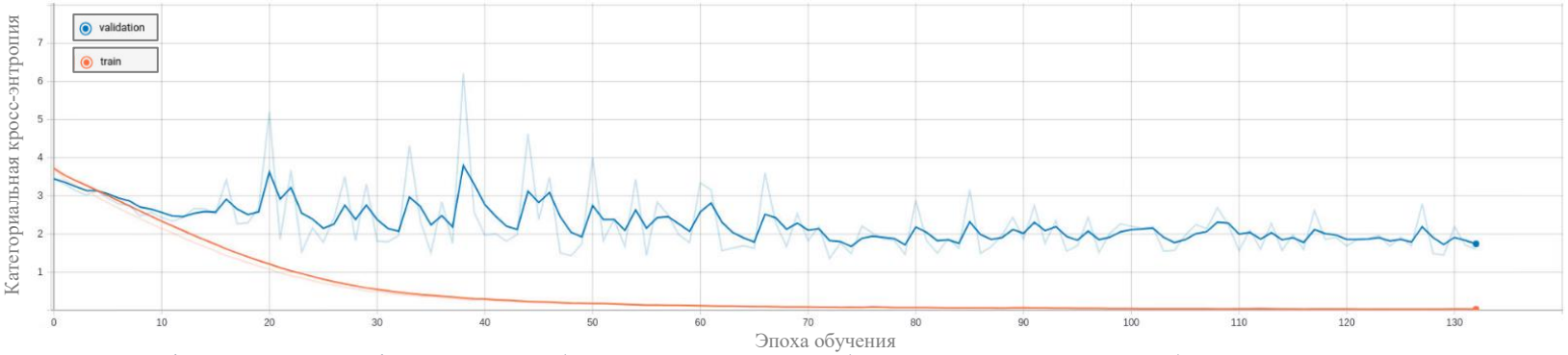


Рис. 16. График изменения функции ошибки модели CNN на обучающей и тестовой выборках «yt_tts_augmented»

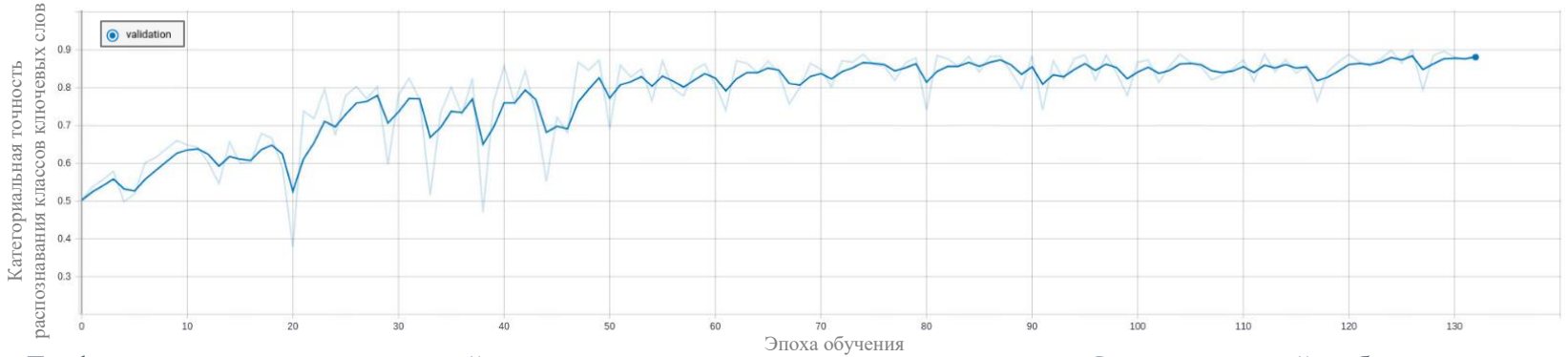


Рис. 17. График изменения категориальной точности для классов ключевых слов модели CNN на тестовой выборке «yt_tts_augmented»

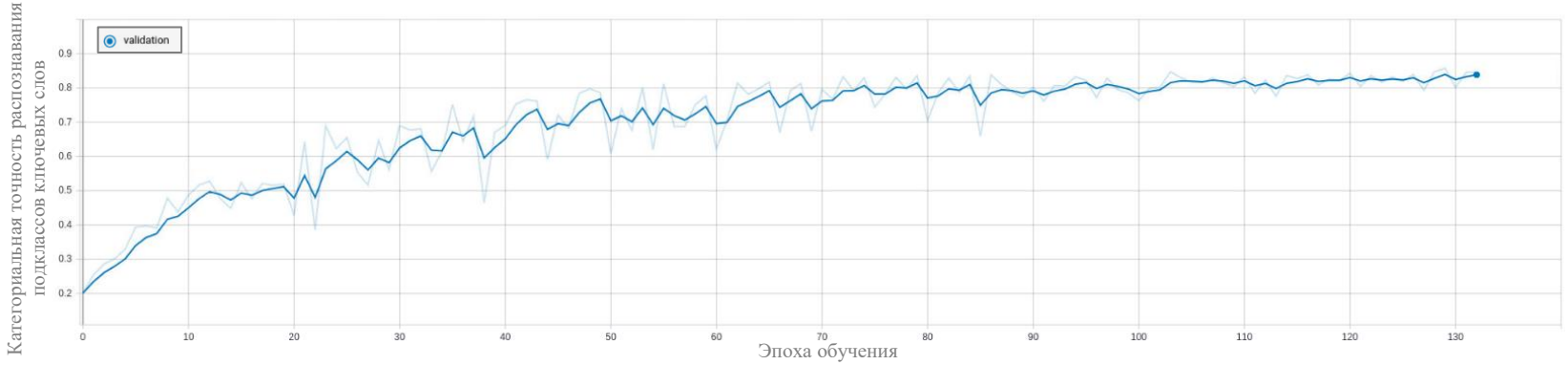


Рис. 18. График изменения категориальной точности для подклассов ключевых слов модели CNN на тестовой выборке «yt_tts_augmented»

Таблица 12. Результаты обучения модели CNN на наборе данных «yt_tts_augmented»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
87.52%	84.09%

В результате обучения модели CNN на наборе данных «yt_tts_augmented» была достигнута наибольшая категориальная точность распознавания 87.52% для классов ключевых слов и 84.09% для подклассов ключевых слов (таблица 12). Заметно значительное увеличение точности распознавания по сравнению с моделью CNN, обученной на наборе данных «yt_tts_clean», что связано с б*о*льшим объемом обучающего набора данных.

4.4.3 Обучение модели DS-CNN на наборе данных «yt_tts_clean»

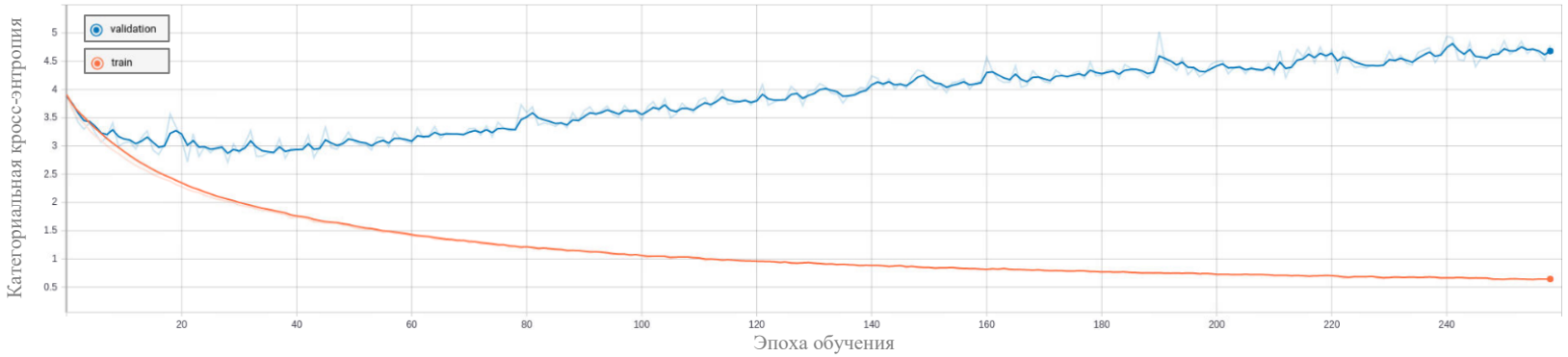


Рис. 19. График изменения функции ошибки модели DS-CNN на обучающей и тестовой выборках «yt_tts_clean»

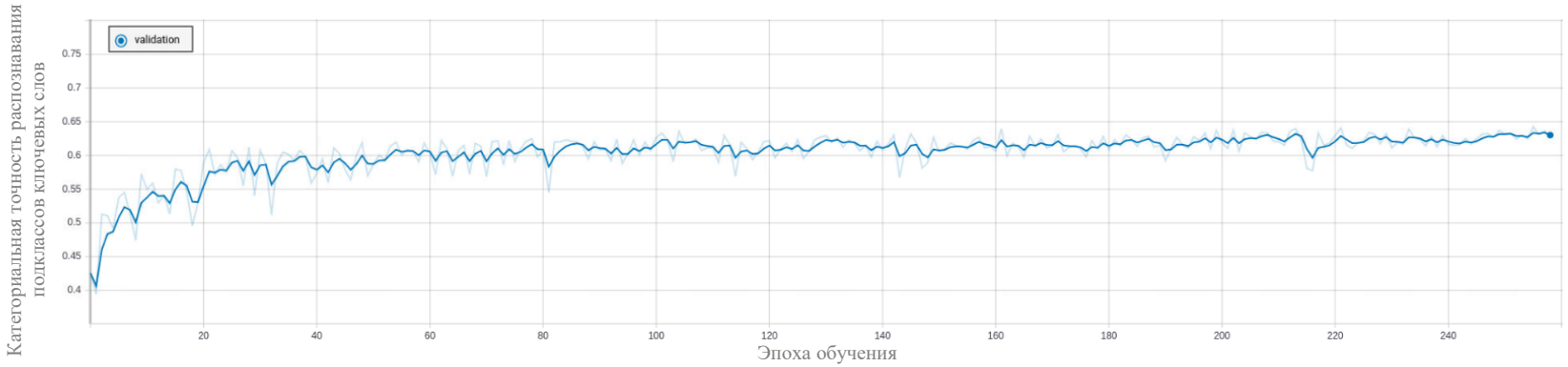


Рис. 20. График изменения категориальной точности для классов ключевых слов модели DS-CNN на тестовой выборке «yt_tts_clean»

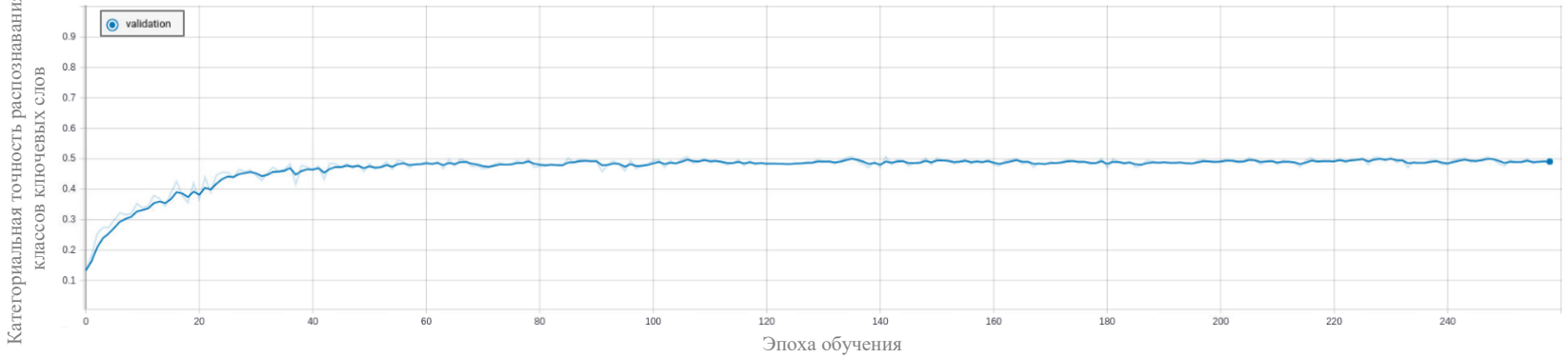


Рис. 21. График изменения категориальной точности для подклассов ключевых слов модели DS-CNN на тестовой выборке «yt_tts_clean»

Таблица 13. Результаты обучения модели DS-CNN на наборе данных «yt_tts_clean»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
62.42%	48.94%

В результате обучения модели DS-CNN на наборе данных «yt_tts_clean» удалось достичь максимальных показателей категориальной точности 62.42% для классов ключевых слов и 48.94% для подклассов ключевых слов (таблица 13). Небольшое увеличение данных показателей по сравнению с CNN моделью, обученной на том же наборе данных, можно объяснить использованием DS-C слоев, а также более подходящим набором гиперпараметров модели. Кроме того, сравнивая графики зависимости значения функции потерь от продолжительности обучения на рисунках 19 и 13, можно сделать вывод о более медленной сходимости DS-CNN модели, что является следствием использования DS-C слоев, а также полносвязного слоя большей размерности.

4.4.4 Обучение модели DS-CNN на наборе данных «yt_tts_augmented»

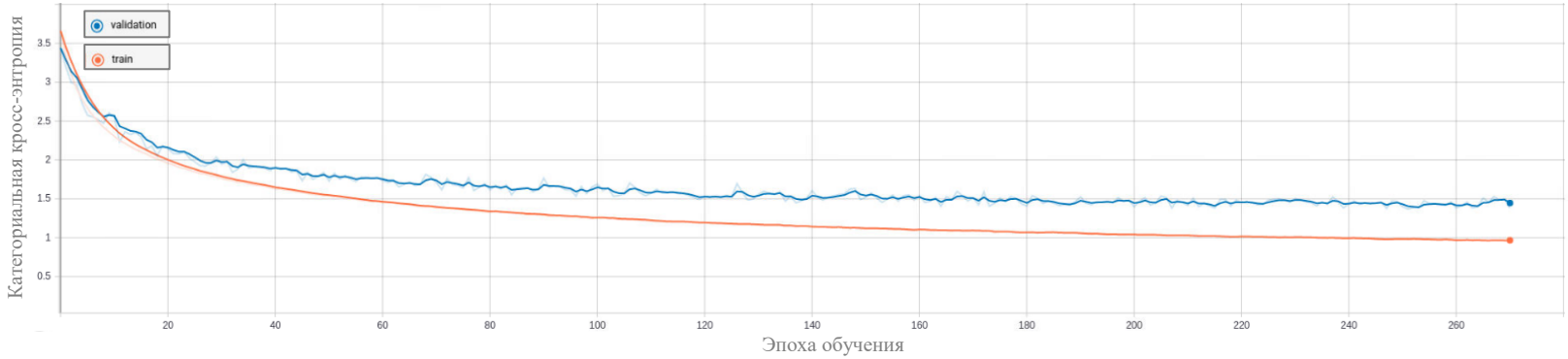


Рис.22. График изменения функции ошибки модели DS-CNN на обучающей и тестовой выборках «yt_tts_augmented»

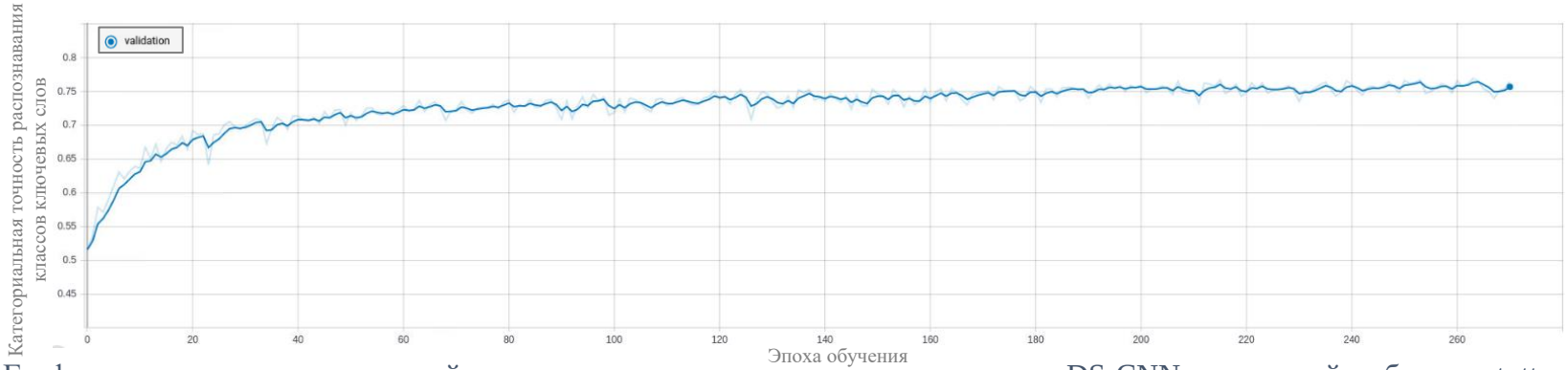


Рис. 23. График изменения категориальной точности для классов ключевых слов модели DS-CNN на тестовой выборке «yt_tts_augmented»

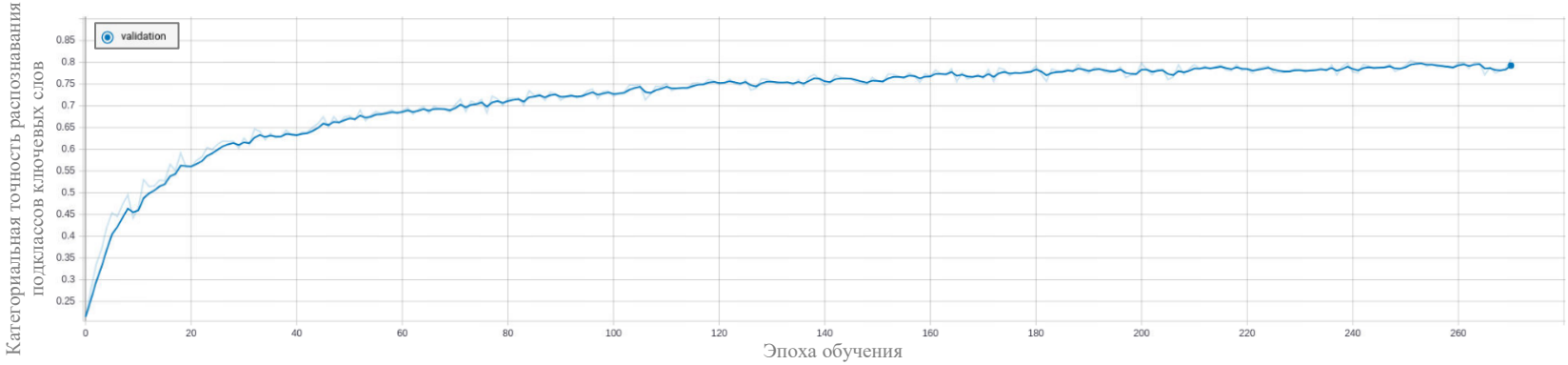


Рис. 24. График изменения категориальной точности для подклассов ключевых слов модели DS-CNN на тестовой выборке «yt_tts_augmented»

Таблица 14. Результаты обучения модели DS-CNN на наборе данных «yt_tts_augmented»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
74.86%	79.28%

В результате обучения модели DS-CNN на наборе данных «yt_tts_augmented» удалось достичь максимальных показателей категориальной точности 74.86% для классов ключевых слов и 79.28% для подклассов ключевых слов (таблица 14). В данном эксперименте наблюдается существенное увеличение результирующих значений категориальной точности по сравнению с результатами модели, обученной на наборе данных «yt_tts_clean», что объясняется большим объемом обучающего набора данных. Наблюдается уменьшение показателей точности по сравнению с моделью CNN, обученной на том же наборе данных, что может быть связано с меньшим количеством обучаемых параметров в данной модели.

4.4.5 Обучение модели M-CNN на наборе данных «yt_tts_clean»

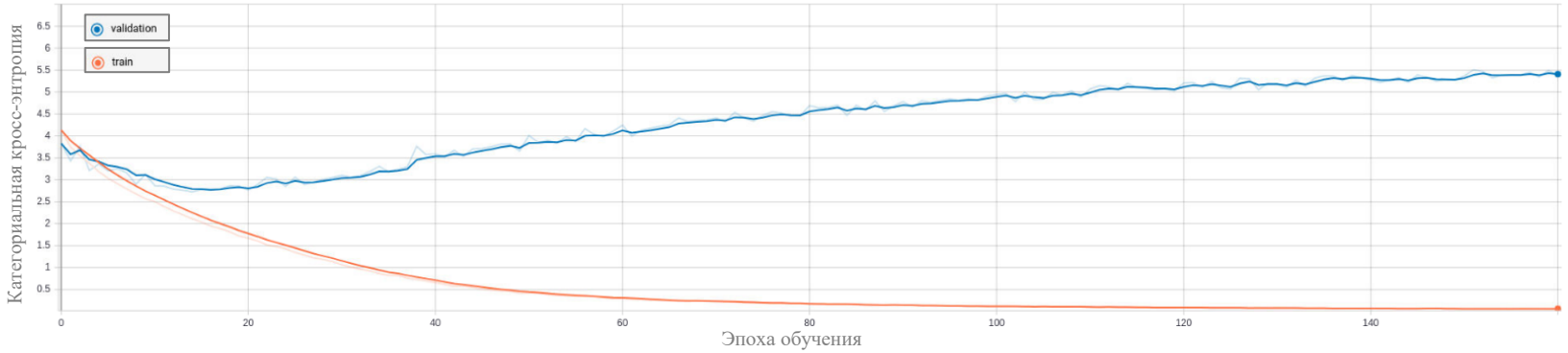


Рис. 25. График изменения функции ошибки модели M-CNN на обучающей и тестовой выборках «yt_tts_clean»

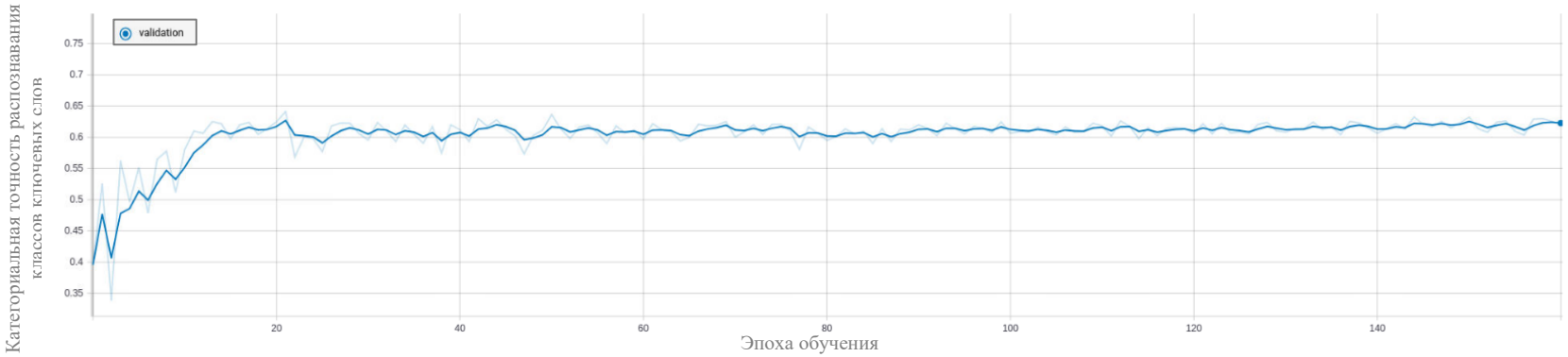


Рис. 26. График изменения категориальной точности для классов ключевых слов модели M-CNN на тестовой выборке «yt_tts_clean»

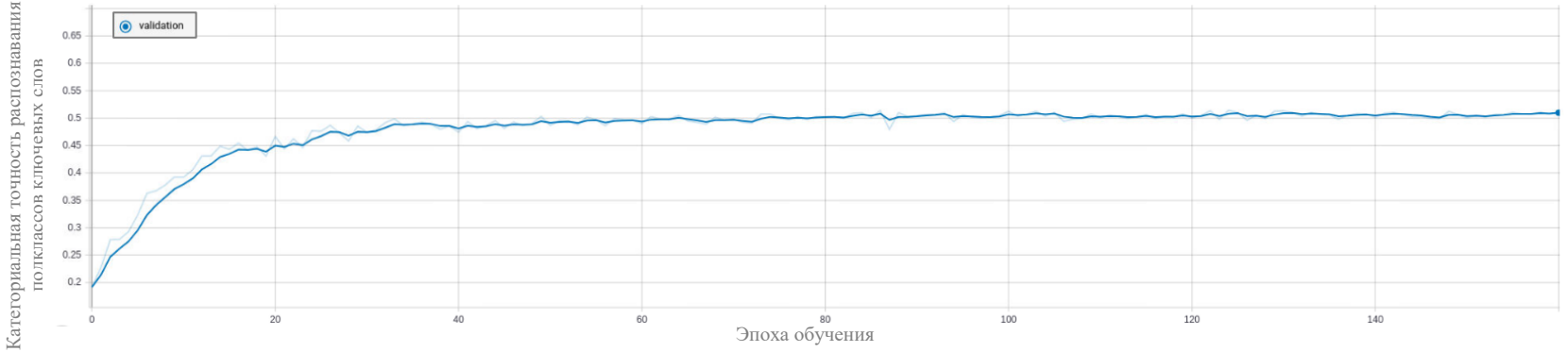


Рис. 27. График изменения категориальной точности для подклассов ключевых слов модели M-CNN на тестовой выборке «yt_tts_clean»

Таблица 15. Результаты обучения модели M-CNN на наборе данных «yt_tts_clean»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
62.81%	50.37%

В результате обучения модели M-CNN на наборе данных «yt_tts_clean» удалось достичь максимальных показателей категориальной точности 62.81% для классов ключевых слов и 50.37% для подклассов ключевых слов (таблица 15). Увеличение категориальной точности распознавания данной модели по сравнению с CNN и DS-CNN достигается путем использования при вычислении результата дополнительной информации из промежуточного скрытого слоя, т. е. двухуровневой локализации признаков, следствием чего является лучшая способность модели улавливать и запоминать признаки входного сигнала.

4.4.6 Обучение модели M-CNN на наборе данных «yt_tts_augmented»

Рис. 28. График изменения функции ошибки модели M-CNN на обучающей и тестовой выборках «yt_tts_augmented»

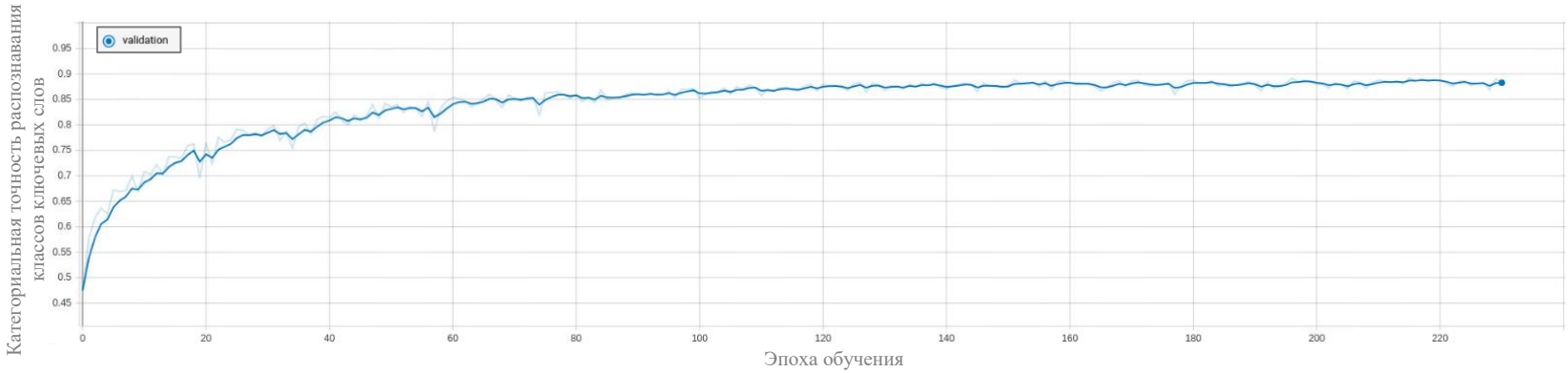


Рис. 29. График изменения категориальной точности для классов ключевых слов модели M-CNN на тестовой выборке «yt_tts_augmented»

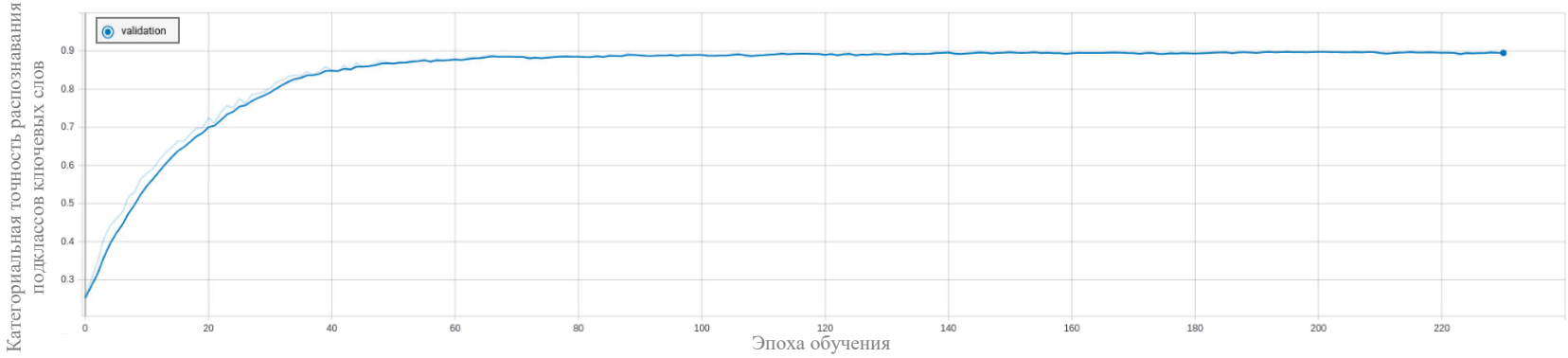


Рис. 30. График изменения категориальной точности для подклассов ключевых слов модели M-CNN на тестовой выборке «yt_tts_augmented»

Таблица 16. Результаты обучения модели M-CNN на наборе данных «yt_tts_augmented»

Категориальная точность на проверочной выборке	
Классы ключевых слов	Подклассы ключевых слов
87.61%	89.24%

В результате обучения модели M-CNN на наборе данных «yt_tts_augmented» были достигнуты максимальные показатели категориальной точности 87.61% для классов ключевых слов и 89.24% для подклассов ключевых слов (таблица 16). Достигнута сопоставимая с моделью CNN точность распознавания классов ключевых слов. Как и в эксперименте 4.4.5, значительное увеличение точности распознавания подклассов ключевых слов по сравнению с моделями CNN и DS-CNN достигается за счет использования при формировании окончательного результата признаков, локализованных в промежуточном скрытом сверточном слое. За счет малого значения уровня обучения наблюдается более плавная сходимость модели.

4.5 Сравнение моделей

В сводной таблице 17 приведены значения целевой метрики, вычисленные моделями CNN, DS-CNN и M-CNN на проверочных выборках наборов данных «yt_tts_clean» и «yt_tts_augmented», а также количество параметров данных моделей.

Таблица 17. Сводная таблица результатов обучения моделей

Модель	Набор данных				Количество параметров	
	«yt_tts_clean»		«yt_tts_augmented»			
	Категориальная точность		Категориальная точность			
	Классы	Подклассы	Классы	Подклассы	Обучаемых	Необучаемых
CNN	60.24%	45.64%	87.52%	84.09%	2942912	1184
DS-CNN	62.42%	48.94%	74.86%	79.28%	553856	800
M-CNN	62.81%	50.37%	87.61%	89.24%	5270192	448

Анализируя результаты экспериментов, отраженные в таблице 17, можно отметить несколько закономерностей. Достигаемая на наборе данных «yt_tts_clean» точность каждой модели сильно меньше точности моделей, обученных на наборе «yt_tts_augmented», что можно объяснить разницей объемов наборов данных. Увеличение количества обучающих примеров позволило достичь большей точности распознавания. Кроме того, при увеличении объема обучающей выборки в экспериментах 4.4.4 и 4.4.6 становится видно превосходство точности распознавания подклассов ключевых слов над точностью распознавания классов, что является следствием использования дополнительного полносвязного слоя перед выходным слоем подклассов.

Результаты проведенных экспериментов показывают, что наилучшие значения категориальной точности как для классов, так и для подклассов ключевых слов, показала модель M-CNN. Данная модель будет использована в дальнейшем при разработке системы взаимодействия человек-машина. Несмотря на хорошую точность распознавания, необходимо сделать замечание: недостатком данной модели является большое количество параметров, поэтому для решения задач распознавания ключевых слов в условиях ограниченных аппаратных ресурсов может быть использована модель DS-CNN, хотя и уступающая в точности распознавания ключевых слов, но являющаяся более легковесной.

4.6 Выводы

1. Описаны гиперпараметры моделей и обучения, заданы множества их допустимых значений, а также описан алгоритм оптимизации гиперпараметров.
2. Проведены эксперименты по оптимизации гиперпараметров моделей CNN, DS-CNN, M-CNN; выбраны наборы гиперпараметров, обеспечивающие лучшие значения метрики точности распознавания.

3. Описан и проанализирован процесс обучения моделей с выбранными гиперпараметрами на наборах данных «yt_tts_clean» и «yt_tts_augmented». Достигнуты наибольшие уровни точности 62.81% для классов и 50.37% для подклассов ключевых слов на наборе «yt_tts_clean», 87.61% для классов и 89.24% для подклассов на наборе «yt_tts_augmented».
4. Проведена сравнительная характеристика обученных моделей. Выбрана M-CNN модель для реализации системы взаимодействия человек - машина.

Глава 5 Разработка системы взаимодействия человек-машина через голосовые команды

Одним из возможных применений обученной модели распознавания ключевых слов является встраивание ее в качестве модуля системы с голосовым интерфейсом, обеспечивающей взаимодействие пользователя и машины. Такая система, получив голосовую команду от пользователя, содержащую набор заранее заданных ключевых слов, должна уметь интерпретировать ее и передать распознанную команду машине для дальнейшего исполнения.

Целью данной главы является создание системы взаимодействия человек-машина по фрагментам устной русскоязычной речи с использованием обученной модели распознавания ключевых слов M-CNN, разработанной и обученной в предыдущих главах.

В данной главе приведено детальное описание основных компонентов разработанной системы, связей между ними и принципа их работы. Также изложены детали реализации каждого модуля.

5.1 Модули системы

Разработанная система состоит из трех модулей: голосового модуля клиента, серверного модуля распознавания речевых команд и модуля-интерпретатора девайса. На рисунке 31 представлено схематичное изображение системы.

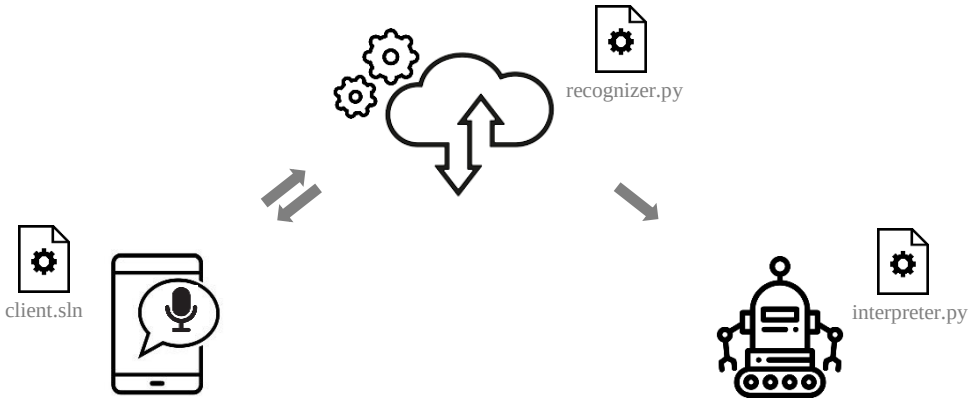


Рис. 31. Схематичное изображение системы взаимодействия человек-машина

Рассмотрим каждую компоненту системы.

5.1.1 Голосовой модуль клиента

Голосовой модуль представляет собой мобильное приложение с графическим интерфейсом для записи аудио любой длины, передачи записанного аудио на сервер с модулем распознавания ключевых слов и отображения результатов распознавания команд. Для демонстрации возможностей системы, набор доступных команд был ограничен классами «Цифра» и «Направление». Пример команды: «вправо на семь единиц, вперед на две единицы».

5.1.2 Модуль распознавания ключевых слов

Модуль распознавания основывается на разработанной модели распознавания ключевых слов M-CNN и представляет собой серверное приложение, которое, получая от голосового модуля записанное аудио с командами пользователя, дополняет его справа белым шумом до длительности, кратной 0.8 секундам, разделяет его на части длиной 0.8 секунды каждая, после чего для каждого семпла осуществляется нормализация громкости к уровню -10 децибел, обработка полосовым частотным фильтром с границами 200 Гц и 3000 Гц и предсказание его категории с помощью обученной модели. Полученный вектор классов $(y_1^{start}, \dots, y_n^{start})$ и подклассов $(z_1^{start}, \dots, z_n^{start})$ ключевых слов для последовательных отрезков голосовой команды сравнивается затем с результатами предсказания для аудио, полученного сдвигом исходного семпла на 0.4 секунды $(y_1^{start+0.4}, \dots, y_m^{start+0.4})$ и $(z_1^{start+0.4}, \dots, z_m^{start+0.4})$. Совпадающие классы и подклассы, находящиеся на одной позиции в векторах и имеющие наибольшие вероятностные значения принадлежности к определенному классу, последовательно преобразуются в вектор команд $(w_1, \dots, w_k)$, доступных для исполнения машиной. Различающиеся классы и подклассы, находящиеся на одной позиции в векторах, добавляются к результирующему вектору команд в порядке $(y_i^{start}, z_i^{start})$, затем

$(y_i^{start+0.4}, z_i^{start+0.4})$. Полученный результат преобразуется в лист относительных координат и сохраняется в файл для дальнейшей обработки модулем-интерпретатором девайса.

Распознанный вектор команд передается пользователю в качестве ответа на запрос.

5.1.3 Модуль-интерпретатор девайса

Модуль-интерпретатор девайса представляет собой десктопное приложение с графическим интерфейсом, отображающее траекторию и местоположение точки в системе координат O_{xy} на плоскости. Набор возможных команд включает в себя перемещение точки на n единиц $n = \overline{1,9}$ (ключевые слова класса «Цифра») в положительном и отрицательном направлениях оси Ox, соответствующих ключевым словам «Вправо» и «Влево»; перемещение точки на n единиц $n = \overline{1,9}$ в положительном и отрицательном направлениях оси Oy, соответствующее ключевым словам «Вперед» и «Назад».

Модуль-интерпретатор производит последовательное чтение распознанных и записанных серверным модулем относительных координат из файла и отображает график траектории точки с учетом заданных направлений. Отсчет ведется от начала координат.

5.2 Детали реализации

Система взаимодействия человек-машина была реализована на языках Python 3.6 и C# 7.0.

Для реализации мобильного голосового клиента был использован Xamarin. Xamarin является фреймворком для кросс-платформенной разработки мобильных приложений, которые легко могут быть масштабированы на iOS, Android и Windows платформы. Еще одним плюсом данного фреймворка является наличие возможности использовать нативные

для каждой платформы средства разработки и элементы пользовательского интерфейса. Таким образом, использование данного инструмента для реализации клиентского приложения позволило с минимальными затратами расширить набор целевых платформ. Код реализации главного класса голосового клиента, ориентированного на платформу Android 7.1 Nougad, а также макет главного экрана приложения прилагается в приложении 6.1.

Для реализации серверного модуля распознавания ключевых слов использовался Python-фреймворк Flask. Наиболее значимым преимуществом данного фреймворка по сравнению с другими вариантами, например, Django, является его легковесность – Flask реализуется с минимальными надстройками, которые всецело предоставлены разработчику. Код реализации модуля распознавания прилагается в приложении 6.2.

Для реализации модуля-интерпретатора использовалась библиотека для визуализации данных matplotlib языка Python, предоставляющая решения для построения и анимации графиков. Модуль-интерпретатор располагается в директории проекта модуля распознавания и в качестве входных данных использует данные из файла, записываемого модулем распознавания. Код реализации модуля-интерпретатора прилагается в приложении 6.3.

5.3 Выводы

1. Описаны требования и разработана архитектура системы взаимодействия человек-машина.
2. Построенная модель распознавания ключевых слов M-CNN использована при реализации модуля распознавания описанной системы.
3. Создан голосовой клиент – мобильное приложение с графическим интерфейсом для взаимодействия пользователя с модулем распознавания.

4. Создан модуль-интерпретатор девайса для взаимодействия машины с модулем распознавания.

Заключение

Результаты выполненной работы

1. Изучены важнейшие публикации и исследования в области обработки речи и, в частности, распознавания ключевых слов с использованием моделей глубинного обучения.
2. На основе сравнительного анализа изученной литературы выявлены наиболее перспективные методы построения глубинных моделей, с опорой на которые разработаны 3 модели сверточных нейронных сетей для распознавания ключевых слов: CNN, DS-CNN и M-CNN.
3. Исследовано несколько способов автоматического сбора данных для составления обучающего корпуса данных ключевых слов.
4. Собраны два набора данных, содержащие русскоязычные аудио семплы 34 ключевых слов из 6 различных классов: «yt_tts_clean» и «yt_tts_augmented», содержащие 11801 и 47204 примеров соответственно и пригодные для обучения глубинных моделей при решении задачи распознавания ключевых слов.
5. Проведены эксперименты по оптимизации гиперпараметров разработанных моделей и обучения; получены наборы гиперпараметров, обеспечивающие лучшую точность распознавания.
6. Проведены эксперименты по обучению моделей с использованием собранных наборов данных; достигнуты наибольшие уровни точности 62.81% для классов и 50.37% для подклассов ключевых слов на наборе «yt_tts_clean», 87.61% для классов и 89.24% для подклассов на наборе «yt_tts_augmented».
7. На основе модели M-CNN разработана система взаимодействия человек-машина с использованием устных русскоязычных команд.

Способы применения результатов работы

Разработанный метод автоматического сбора русскоязычных аудио данных может быть применен для создания корпуса обучающих данных для

других языков, поскольку не включает в себя использование информации, специфичной для конкретного языка. Разработанные архитектуры моделей и описанный метод оптимизации гиперпараметров также могут быть использованы при создании модели для распознавания ключевых слов для других языков, поскольку существует литература, описывающая успешные способы применения моделей со схожей архитектурой для распознавания, например, англоязычных ключевых слов [27].

Обученные модели распознавания русскоязычных ключевых слов могут быть использованы при разработке встроенных голосовых помощников, поскольку одна из разработанных моделей (DS-CNN) обладает достаточной для данной задачи легковесностью и малым количеством параметров. Также разработанные модели могут использоваться в качестве модулей систем управления группами роботов, что становится возможным благодаря наличию класса «Имя» в обучающих наборах данных и выходных слоях моделей.

Направления дальнейших исследований

1. Исследование более эффективных и точных способов автоматического сбора обучающих данных для систем распознавания ключевых слов; расширение набора классов и подклассов используемых ключевых слов.
2. Изучение и применение способов сокращения числа параметров и количества операций для разработанных моделей с целью увеличения производительности и уменьшения время отклика системы для дальнейшего встраивания ее в девайсы с ограниченными аппаратными ресурсами.
3. Разработка модели распознавания ключевых слов с использованием рекуррентных слоев с LSTM ячейками; сравнение результатов точности распознавания моделей.
4. Разработка системы управления несколькими роботами на основе обученной модели, показывающей лучшую точность распознавания.

Список используемых сокращений

CNN – Convolutional Neural Network

CRF – Conditional Random Field

CTC - Connectionist Temporal Classification

DNN – Deep Neural Network

DS-CNN – Depthwise Separable Neural Network

GRU - Gated Recurrent Units

HMM – Hidden Markov Model

IoT - Internet of Things

KWS – Keyword Spotting

LSTM - Long Short-Term Memory

M-CNN - Multi-Scale Convolutional Neural Network

MFCC - Mel-Frequency Cepstral Coefficients

NNI - Neural Network Intelligence

ReLU - Rectified Linear Unit

RNN - Recurrent Neural Network

SGD - Stochastic Gradient Descent

TTS – Text to Speech

URL - Uniform Resource Locator

VR - Virtual Reality

ОЗУ - Оперативное Запоминающее Устройство

ЦПУ - Центральное Процессорное Устройство

ЭВМ - Электронно-Вычислительная Машина

Список литературы

1. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units / Quoc V. Le [и др.] // arXiv preprint arXiv:1504.00941 – 2015
2. Acoustic similarity scores for keyword spotting / Veiga A. [и др.] // Computational Processing of the Portuguese Language. – 2014
3. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning / Chigozie Nwankpa [и др.] // arXiv preprint arXiv:1811.03378 – 2018
4. Adam: A Method for Stochastic Optimization / Diederik P. Kingma [и др.] // arXiv preprint arXiv:1412.6980 – 2014
5. Algorithms for Hyper-Parameter Optimization / James Bergstra [и др.] // Advances in Neural Information Processing Systems Conference – 2011
6. AVA-Speech: A Densely Labeled Dataset of Speech Activity in Movies / Sourish Chaudhuri [и др.] // Google AI research – 2018 – URL: https://ai.google/research/pubs/pub47336
7. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / Sergey Ioffe [и др.] // arXiv preprint arXiv:1502.03167 – 2015
8. Bidirectional Recurrent Neural Networks / Mike Schuster [и др.] // IEEE Transactions on Signal Processing – IEEE. 1997 – С. 2673-2681
9. Building Corpora of Transcribed Speech from Open Access Sources / O. O. Iakushkin [и др.] // Proceedings of the 8th International Conference Distributed Computing and Grid-technologies in Science and Education – CEUR. 2018 – URL: http://ceur-ws.org/Vol-2267/475-479-paper-91.pdf
10. Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx) / Gamal Bohouta [и др.] // Int. Journal of Engineering Research and Application – 2017 – С. 20-24
11. Comparison of Keyword Spotting Approaches for Informal Continuous Speech / Igor Szöke [и др.] //Proceedings of the 9th European Conference on Speech Communication and Technology. – 2005 – С. 633-636.

12. Compressing Deep Neural Networks using a Rank-Constrained Topology / Preetum Nakkiran [и др.] // Proceedings of 16th Annual Conference of the International Speech Communication Association, INTERSPEECH – ISCA. 2015 – С. 1473-1477
13. Conditional Random Fields in Speech, Audio, and Language Processing / Eric Fosler-Lussier [и др.] // Proceedings of the IEEE. – IEEE. 2013 – С. 1054-1075.
14. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data / John D. Lafferty [и др.] // Proceedings of the Eighteenth International Conference on Machine Learning – ICML. 2002 – С. 282-289
15. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks / A. Graves [и др.] // Proceedings of the 23rd international conference on Machine learning. – ACM. 2006 – С. 369-376.
16. Convolutional Neural Networks for Small-footprint Keyword Spotting / Tara N. Sainath [и др.] // 16th Annual Conference of the International Speech Communication Association, INTERSPEECH. – 2015
17. Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting / Sercan O. Arik [и др.] // 18th Annual Conference of the International Speech Communication Association, INTERSPEECH. – 2017
18. Deep Speech: Scaling up end-to-end speech recognition / Awni Hannun [и др.] // arXiv preprint arXiv:1412.5567 – 2014
19. Dimensionality reduction methods for HMM phonetic recognition / Hongbing Hu [и др.] // Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, INTERSPEECH. – ICASSP. 2010 – С. 4854-4857.
20. Do Deep Nets Really Need to be Deep? / Lei Jimmy Ba [и др.] // arXiv preprint arXiv:1312.6184 – 2014
21. Dual Path Networks / Yunpeng Chen [и др.] // arXiv preprint arXiv:1707.01629 – 2017

22. Efficient Processing of Deep Neural Networks: A Tutorial and Survey / Vivienne Sze [и др.] // Proceedings of the IEEE 105(12) – IEEE. 2017 – С. 2295-2329
23. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling / Junyoung Chung [и др.] // arXiv preprint arXiv:1412.3555 – 2014
24. Experimental, Limited Vocabulary, Speech Recognizer / Charles F. Teacher // IEEE Transactions on Audio and Electroacoustics 15(3) – IEEE. 1967 – С. 127-130
25. Google Cloud Text-to-Speech – Accessed: 2019-05-17. URL: https://cloud.google.com/text-to-speech/
26. H2O.ai – Accessed: 2019-03-14. URL: https://www.h2o.ai/
27. Hello edge: Keyword spotting on microcontrollers / Yundong Zhang // arXiv preprint arXiv:1711.07128 – 2017
28. High quality text-to-speech synthesis: A comparison of four candidate algorithms / T. Dutoit // Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing – IEEE. 1994 – С. 565-568
29. ImageNet Classification with Deep Convolutional Neural Networks / Alex Krizhevsky [и др.] // Advances in neural information processing systems 25(2) – 2012
30. Improving speech recognition using data augmentation and acoustic model fusion / Ilyes Rebai [и др.] // Procedia Computer Science 112 – 2017 – С. 316-322
31. Keyword spotting using Hidden Markov Models / Şevket Duran // Graduate Study in Science and Engineering Bosphorus University. – 1997
32. Learning Precise Timing with LSTM Recurrent Networks / Felix Gers [и др.] // Journal of Machine Learning Research 3(1) – 2002 – С. 115-143
33. Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach / Rishin Haldar [и др.] // arXiv preprint arXiv:1101.1232 – 2011

34. Log-Linear Models, Extensions, and Applications / Alexandr Aravkin [и др.] // MIT – 2018
35. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling / H. Sak [и др.] // Proceedings of the 15th Annual Conference of the International Speech Communication Association, INTERSPEECH. – ISCA. 2014 – С. 338-342
36. Microsoft Azure Speech Services – Accessed: 2019-05-17. URL: https://azure.microsoft.com/ru-ru/services/cognitive-services/speech-services/
37. Microsoft Neural Network Intelligence – Accessed: 2019-03-14. URL: https://github.com/microsoft/nni
38. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications / Andrew G. Howard [и др.] // arXiv preprint arXiv:1704.04861 – 2017
39. Model Compression Applied to Small-Footprint Keyword Spotting / George Tucker [и др.] // Proceedings of the17th Annual Conference of the International Speech Communication Association, INTERSPEECH – ISCA. 2016 – С. 1878-1882
40. Multi-Scale Convolutional Neural Networks for Time Series Classification / Zhicheng Cui [и др.] // arXiv preprint arXiv:1603.06995 – 2016
41. Online Keyword Spotting with a Character-Level Recurrent Neural Network / Kyuyeon Hwang [и др.] // arXiv preprint arXiv:1512.08903. – 2015.
42. Recent Advances in Convolutional Neural Networks / Jiuxiang Gu [и др.] // arXiv preprint arXiv:1512.07108 – 2017
43. Recent Advances in Deep Learning for Speech Research at Microsoft / Li Deng [и др.] // Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on – ICASSP. 2013 – С. 8604-8608
44. Recurrent Neural Network Based Language Model / Tomas Mikolov [и др.] // Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010. – ISCA. 2010 – С. 1045-1048

45. Russian-Language Speech Recognition System Based on Deepspeech / O. O. Iakushkin [и др.] // Proceedings of the 8th International Conference Distributed Computing and Grid-technologies in Science and Education – CEUR. 2018 – С. 470-474
46. Sequence to Sequence Learning with Neural Networks / Ilya Sutskever [и др.] // arXiv preprint arXiv:1409.3215 – 2014
47. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices / Xiangyu Zhang [и др.] // arXiv preprint arXiv:1707.01083 – 2018
48. Small-Footprint Keyword Spotting using Deep Neural Networks / Guoguo Chen [и др.] // Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing – ICASSP. 2014 – С. 4087-4091
49. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition / Pete Warden // arXiv preprint arXiv:1804.03209 – 2018
50. Spoken language understanding: Systems for extracting semantic information from speech / Renato De Mori [и др.] // WILEY.2011
51. Talos – Accessed: 2019-03-14. URL: https://github.com/autonomio/talos
52. The Application of Hidden Markov Models in Speech Recognition / Mark Gales [и др.] // Foundations and Trends® in Signal Processing 1(3) – 2007 – С. 195-304
53. The Microsoft 2017 Conversational Speech Recognition System / W. Xiong [и др.] // Microsoft AI and Research Technical Report MSR-TR-2017-39 – 2017
54. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques / Lindasalwa Muda [и др.] // arXiv preprint arXiv:1003.4083 – 2010
55. Xception: Deep Learning with Depthwise Separable Convolutions / Francois Chollet // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) – IEEE. 2017 – С. 1800-1807
56. Yandex SpeechKit – Accessed: 2019-05-17. URL: https://cloud.yandex.ru/services/speechkit

57. YOLOv3: An Incremental Improvement / Joseph Redmon [и др.] // arXiv preprint arXiv:1804.02767 – 2018

Приложения

6.1 Реализация голосового модуля клиента для системы взаимодействия человек-машина через голосовые команды

Листинг 1. Реализация макета экрана для голосового модуля системы взаимодействия человек-машина через голосовые команды на расширяемом языке разметки приложений XAML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Record area"
        android:textSize="24sp"/>
    <Button
        android:id="@+id/buttonRecord"
        android:text="Record"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button
        android:id="@+id/buttonStopRecord"
        android:text="Stop Record"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send area"
        android:textSize="24sp"/>
    <Button
        android:id="@+id/buttonSend"
        android:text="Send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Листинг 2. Реализация голосового модуля клиента для системы взаимодействия человек-машина через голосовые команды на языке C#

```
// Подключение пространств имен
using Android.App;
using Android.OS;
using Android.Support.V7.App;
using Android.Runtime;
using Android.Widget;
using Android.Media;
using Android;
using Android.Support.V4.App;
using Android.Content.PM;
using System;
using Environment = Android.OS.Environment;
using Android.Util;
using System.Net.Http;
using System.IO;
using System.Threading.Tasks;

namespace client
{
    // Класс приложения
    [Activity(Label = "@string/app_name", Theme ="@style/
      Theme.AppCompat.Light.NoActionBar" , MainLauncher = true)]
    public class MainActivity : AppCompatActivity
    {
        private const int REQUEST_PERMISSION_CODE = 1000;
        private static readonly string UPLOAD_SERVICE_ADRESS =
          "http://168.61.86.72:8093/process_command";

        // Объявление переменных
        private Button btnRecord, btnStopRecord, btnSend;
        private string pathSave = "";
        private MediaRecorder mediaRecorder;
        private bool isGrantedPermission = false;

        // Метод проверки разрешения доступа
        public override void OnRequestPermissionsResult(int requestCode,
          string[] permissions, [GeneratedEnum] Permission[] grantResults)
        {
            switch(requestCode)
            {
                case REQUEST_PERMISSION_CODE:
                    {
                        if (grantResults.Length > 0 && grantResults[0] ==
                    Permission.Granted)
                        {
                            Toast.MakeText(this, "Granted",
                    ToastLength.Short).Show();
                            isGrantedPermission = true;
                        }
                        else
                        {
                            Toast.MakeText(this, "Not granted",
                    ToastLength.Short).Show();
```

```
48                          isGrantedPermission = false;
49                      }
50                  }
51                  break;
52              }
53          }
54
55          // Метод, вызываемый при создании формы
56          protected override void OnCreate(Bundle savedInstanceState)
57          {
58              base.OnCreate(savedInstanceState);
59
60              // Установка шаблона формы
61              SetContentView(Resource.Layout.activity_main);
62
63              // Запросы на наличие необходимых разрешений
64              if(CheckSelfPermission
                  (Manifest.Permission.WriteExternalStorage) !=
                  Android.Content.PM.Permission.Granted
65                  && CheckSelfPermission(Manifest.Permission.RecordAudio) !=
                      Android.Content.PM.Permission.Granted)
66              {
67                  ActivityCompat.RequestPermissions(this, new string[] {
68                      Manifest.Permission.WriteExternalStorage,
69                      Manifest.Permission.RecordAudio
70                  }, REQUEST_PERMISSION_CODE);
71              }
72              else
73              {
74                  isGrantedPermission = true;
75              }
76
77              // Инициализация переменных кнопок
78              btnRecord = FindViewById<Button>(Resource.Id.buttonRecord);
79              btnStopRecord = FindViewById<Button>
                  (Resource.Id.buttonStopRecord);
80              btnSend = FindViewById<Button>(Resource.Id.buttonSend);
81
82              btnStopRecord.Enabled = false;
83              btnSend.Enabled = false;
84
85              // Событие при нажатии на кнопку "Record"
86              btnRecord.Click += delegate
87              {
88                  RecordAudio();
89              };
90
91              // Событие при нажатии на кнопку "Stop record"
92              btnStopRecord.Click += delegate
93              {
94                  StopRecorder();
95              };
96
```

```
97              // Событие при нажатии на кнопку "Send"
98              btnSend.Click += delegate
99              {
100                 SendLastRecord();
101             };
102
103         }
104
105         // Инициализация экземпляра MediaRecorder
106         private void SetupMediaRecorder()
107         {
108             mediaRecorder = new MediaRecorder();
109             mediaRecorder.SetAudioSource(AudioSource.Mic);
110             mediaRecorder.SetOutputFormat(OutputFormat.ThreeGpp);
111             mediaRecorder.SetAudioEncoder(AudioEncoder.AmrNb);
112             mediaRecorder.SetOutputFile(pathSave);
113         }
114
115         // Метод для отправки аудио на сервер
116         private async void SendLastRecord()
117         {
118             btnSend.Enabled = true;
119             btnRecord.Enabled = true;
120             btnStopRecord.Enabled = false;
121
122             Toast.MakeText(this, "Uploading file to server...",
                  ToastLength.Short).Show();
123
124             var content = new MultipartFormDataContent();
125             var fileBytes = File.ReadAllBytes(pathSave);
126             ByteArrayContent byteContent = new ByteArrayContent(fileBytes);
127             content.Add(byteContent);
128
129             try
130             {
131                 using (HttpClient client = new HttpClient())
132                 {
133                     HttpResponseMessage response = await client.PostAsync
                        (UPLOAD_SERVICE_ADRESS, content);
134                     Task<string> result = response.Content.ReadAsStringAsync
                        ();
135                     Toast.MakeText(this, result.Result,
                        ToastLength.Short).Show();
136                 }
137             }
138             catch (Exception e)
139             {
140                 Log.Debug("DEBUG", e.Message);
141             }
142         }
143
144         // Метод для начала записи аудио
145         private void RecordAudio()
```

```
146         {
147             if (isGrantedPermission)
148             {
149                 pathSave =
                      Environment.ExternalStorageDirectory.AbsolutePath.ToString
                      () + '/' + new Guid().ToString() + "_audio.3gp";
150                 SetupMediaRecorder();
151                 try
152                 {
153                     mediaRecorder.Prepare();
154                     mediaRecorder.Start();
155                     btnRecord.Enabled = false;
156                     btnStopRecord.Enabled = true;
157                     btnSend.Enabled = false;
158                 }
159                 catch (Exception e)
160                 {
161                     Log.Debug("DEBUG", e.Message);
162                 }
163                 Toast.MakeText(this, "Start recording...",
                      ToastLength.Short).Show();
164             }
165         }
166
167         // Метод для остановки записи аудио
168         private void StopRecorder()
169         {
170             mediaRecorder.Stop();
171             btnStopRecord.Enabled = false;
172             btnSend.Enabled = true;
173             btnRecord.Enabled = true;
174
175             Toast.MakeText(this, "Stop recording...",
                  ToastLength.Short).Show();
176         }
177     }
178 }
```

6.2 Реализация серверного модуля распознавания ключевых слов для системы взаимодействия человек-машина через голосовые команды

Листинг 3. Реализация серверного модуля распознавания для системы взаимодействия человек-машина через голосовые команды на языке Python 3.6

```
#Импорт библиотек и функций
import os
import sys

current_dir_path = os.path.dirname(os.path.realpath(__file__))
project_root_path = os.path.join(current_dir_path, os.pardir, os.pardir)
sys.path.insert(0, project_root_path)

from flask import Flask, request, jsonify
from werkzeug import secure_filename
from tensorflow.keras.models import load_model
import numpy as np

from audio_utils import convert_audio_to_wav
from audio_utils import apply_bandpass_filter
from audio_utils import normalize_volume
from audio_utils import audio_file_to_input_vector
from audio_utils import split_audio
from commands_utils import match_labels_with_command
from commands_utils import convert_commands_to_relative_coordinates
import constants as const

#Создание инстанса Flask
app = Flask(__name__)

#Загрузка обученной модели
model = load_model(const.MODEL_PATH)

#Задание маршрутов Flask
@app.route('/process_command', methods=['Post'])
def process_command():
    try:
        #Сохранение полученного аудио-файла
        audio = request.files['file']
        audio_path = os.path.join(const.AUDIO_SOURCE_PATH, secure_filename(audio.filename))
        audio.save(audio_path)

        #Предобработка аудио
        preprocessed_audio_path = preprocess_audio(audio_path)

        #Выделение команд из аудио-файла
        commands = get_commands(preprocessed_audio_path)

        #Преобразование команд в последовательность относительных координат точки
        relative_coordinates = convert_commands_to_relative_coordinates(commands)

        #Запись относительных координат в файл
        with open(const.COORDINATES_SOURCE_PATH, 'a+') as f:
```

```
49              f.write('\n'.join('%s %s' % x for x in
    relative_coordinates))
50
51          #Формирование ответа клиенту
52          resp = jsonify({'commands': commands})
53          resp.status_code = 200
54
55          return resp
56
57      except Exception as e:
58          raise e
59
60  #Задание обработчика ошибок Flask
61  @app.errorhandler(Exception)
62  def exception_handler(error=None):
63      message = {
64                  'status': 500,
65                  'message': 'Internal server error: ' + str(error),
66              }
67      resp = jsonify(message)
68      resp.status_code = 500
69
70      return resp
71
72  #Функция предобработки аудио
73  def preprocess_audio(audio_path):
74      #Преобразование в .wav формат с заданными характеристиками
75      wav_audio_path = os.path.join(const.DATA_PATH, 'audio.wav')
76      convert_audio_to_wav(source=audio_path, sample_rate=16000,
    n_channels=1, byte_width=2, dest=wav_audio_path)
77
78      #Нормализация громкости аудио
79      normalized_audio_path = os.path.join(const.DATA_PATH,
    'normalized_audio.wav')
80      normalize_volume(source=wav_audio_path, level=-10,
    dest=normalized_audio_path)
81
82      #Применение полосового частотного фильтра
83      filtered_audio_path = os.path.join(const.DATA_PATH,
    'filtered_audio.wav')
84      apply_bandpass_filter(source=normalized_audio_path, lower_bound=250,
    upper_bound=3000, dest=filtered_audio_path)
85
86      return filtered_audio_path
87
88  #Функция выделения команд из аудио
89  def get_commands(audio_path):
90      #Разделение аудио-файла на семплы длиной 0.8 секунды без сдвига
91      original_samples_paths = split_audio(source=audio_path, shift=.0,
    duration=const.SAMPLE_LENGTH)
92
93      #Формирование входной последовательности матриц MFCC
94      X_original = []
95      for filename in original_samples_paths:
96          X_original.append(audio_file_to_input_vector(source=filename,
    numcep=const.N_INPUT, numcontext=const.N_CONTEXT))
97
98      #Предсказание классов и подклассов для входной последовательности
99      Y_class_original, Y_subclass_original = model.predict(X_original)
100
101     #Разделение аудио-файла на семплы длиной 0.8 секунды со сдвигом 0.4
    секунды от начала аудио
```

```
102       shifted_samples_paths = split_audio(source=audio_path, shift=0.4,
      duration=const.SAMPLE_LENGTH)
103
104       #Формирование входной последовательности матриц MFCC
105       X_shifted = []
106       for filename in shifted_samples_paths:
107           X_shifted.append(audio_file_to_input_vector(source=filename,
      numcep=const.N_INPUT, numcontext=const.N_CONTEXT))
108
109       # Предсказание классов и подклассов для входной последовательности
      со сдвигом
110       Y_class_shifted, Y_subclass_shifted = model.predict(X_shifted)
111
112       #Формирование листа команд
113       commands = []
114       for i in range(len(X_original)):
115           # Выбор классов и подклассов с лучшей точностью
116           original_class_label = np.argmax(Y_class_original[i])
117           original_subclass_label = np.argmax(Y_subclass_original[i])
118           shifted_class_label = np.argmax(Y_class_shifted[i])
119           shifted_subclass_label = np.argmax(Y_subclass_shifted[i])
120
121           #Добавление команд в лист
122           commands.append(match_labels_with_command(original_class_label,
      original_subclass_label))
123           if (original_class_label != shifted_class_label) |
      (original_subclass_label != shifted_subclass_label):
124               commands.append(match_labels_with_command(
      shifted_class_label, shifted_subclass_label))
125
126       return commands
127
128
129   if __name__ == '__main__':
130       #Запуск веб-приложения
131       app.run(host='0.0.0.0', port = 5000, debug = True)
```

6.3 Реализация модуля-интерпретатора команд для системы взаимодействия человек-машина через голосовые команды

Листинг 4. Реализация модуля-интерпретатора команд для системы взаимодействия человек-машина через голосовые команды на языке Python 3.6

```
#Импорт библиотек и функций
import os
import sys

current_dir_path = os.path.dirname(os.path.realpath(__file__))
project_root_path = os.path.join(current_dir_path, os.pardir, os.pardir)
sys.path.insert(0, project_root_path)

import matplotlib.pyplot as plt
import matplotlib.animation as animation

import constants as const

#Определение функции анимации графика
def animate(i):
    #Чтение файла с относительными координатами
    with open(const.COORDINATES_SOURCE_PATH, 'r') as data:
        lines = data.read().split('\n')

        #Последовательное формирование точек графика по данным из файла
        X = [0]
        Y = [0]
        for line in lines:
            if(len(line) > 1):
                delta_x, delta_y = line.split(',')
                X.append(X[-1] + int(delta_x))
                Y.append(Y[-1] + int(delta_y))

        #Обновление графика
        main_plot.clear()
        main_plot.plot(X, Y)


if __name__ == '__main__':
    #Задание графика
    fig = plt.figure()
    main_plot = fig.add_subplot(1, 1, 1)
    plt.xlabel('x')
    plt.ylabel('y')

    #Задание функции анимации для графика с периодом обновления 1 секунда
    ani = animation.FuncAnimation(fig, animate, interval=1000)

    #Отображение графика
    plt.show()
```