|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Метод повышения производительности рабочей***

***станции за счет вычислительного кластера***

Студентка ИУ7-41м **Новоженов В.А.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР  **Ковтушенко А.П.**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **Строганов Ю.В.**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

# **АННОТАЦИЯ**

**выпускной квалификационной работы**

Новоженова Владимира Алексеевича

на тему: «Метод повышения производительности рабочей станции за счет вычислительного кластера»

Объем ВКР: количество страниц 80, иллюстраций 25, таблиц 6, использованных источников 30, приложений 1.

Ключевые слова: повышение производительности, кластерные вычислительные системы, нейронные сети, распределенные алгоритмы, параллельные вычисления.

Объект исследования: вычислительные кластеры и методы повышения производительности при их использовании.

Цель работы: создание метода повышения производительности в задаче обучения нейронных сетей с использованием вычислительного кластера.

В ходе работы был создан метод повышения производительности вычислений, которые необходимо провести на рабочей станции. На рабочей станции развернута нейронная сеть, распознающая пользователей по клавиатурному почерку. В следствии, так называемого «concept drifting», появляется необходимость в дообучении нейронной сети. Дообучение нейронной сети является вычислительно сложной задачей, требующей много ресурсов и выполняющей большое число математических операций. Вследствие этого, не следует нагружать ресурсы рабочей станции задачей дообучения нейронной сети, а отправить задачу на вычислительный кластер. Само дообучение нейронной сети также следует реализовать в соответствии со спецификой кластера, а для повышения скорости получения результата и исключения дублей заданий, отправляемых на кластер следует использовать особую систему контроля заданий.

Результатом работы является сам метод повышения производительности при помощи высокопроизводительного кластера. Сам метод применим для обучения любого класса нейронных сетей, созданных на основе фреймворков TensorFlow и Pytorch.

Новизна работы состоит в использовании системы отслеживания заданий обучения нейронных сетей, отправляемых на вычислительный кластер для вычисления параллельно и распределенно.

Прогнозом развития метода является аппроксимация метода и системы контроля заданий под любой класс вычислительных математически-сложных задач, которые следует отправлять на выполнение на кластер и получать результат вычисления в другой среде, отличной от кластерной (например на некотором сервере или рабочей станции).

Автор ВКР «\_\_\_\_\_\_\_\_\_\_\_\_\_» «Новоженов Владимир Алексеевич»

**СОДЕРЖАНИЕ**

[**АННОТАЦИЯ** 2](#_Toc11867411)

[**ВВЕДЕНИЕ** 4](#_Toc11867412)

[**1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ** 6](#_Toc11867413)

[**1.1 Известные методы верификации подписи с использованием нейронных сетей** 8](#_Toc11867414)

[**1.2 Метод разблокировки смартфона Face ID** 16](#_Toc11867415)

[**1.3 Выводы** 19](#_Toc11867416)

[**2 КОНСТРУКТОРСКИЙ РАЗДЕЛ** 20](#_Toc11867417)

[**2.1 Архитектура алгоритма распознавания рукописного текста** 20](#_Toc11867418)

[**2.2 Математическая модель метода распознавания рукописного текста** 23](#_Toc11867419)

[**2.3 Состав ансамбля нейронных сетей** 39](#_Toc11867420)

[**2.4 Описание обучающей и тестовой выборок** 40](#_Toc11867421)

[**3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ** 43](#_Toc11867422)

[**3.1 Детали подготовки данных для обучения и тестирования системы** 43](#_Toc11867423)

[**3.2 Оценка качества работы метода** 48](#_Toc11867424)

[**3.3 Пути развития предложенного метода** 54](#_Toc11867425)

[**3.4 Выводы** 56](#_Toc11867426)

[**ЗАКЛЮЧЕНИЕ** 57](#_Toc11867427)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 58](#_Toc11867428)

[**ПРИЛОЖЕНИЕ A** 61](#_Toc11867429)

# 

# **ВВЕДЕНИЕ**

**Цель работы:** создание метода повышения производительности в задаче обучения нейронных сетей с использованием вычислительного кластера.

**Задачи:**

* Проанализировать методы повышения производительности в решении задачи обучения нейронных сетей. Проанализировать методы взаимодействия с вычислительным кластером по внешнему программному интерфейсу;
* Создать метод взаимодействия с кластером в рамках задачи распределенного обучения нейронных сетей с использованием среды контроля заданий;
* Протестировать разработанную систему в решении практических задач обучения нейронных сетей;

**Объект** – вычислительные кластеры и методы повышения производительности при их использовании.

**Проблема** – ресурсо-затратность и время-затратность в задаче обучения нейронных сетей без использования кластера или с использованием кластера без использования системы обработки заданий.

В настоящее время особую популярность в решении самых разных задач получили нейронные сети. С их помощью можно и определять наличие объектов на изображениях и выявлять паттерны и закономерности во временных рядах и обрабатывать и распознавать любой звук. Но перед использованием нейронных сетей их необходимо обучить на, зачастую, огромных массивах данных. Это могут быть тысячи изображений или миллионы строк данных. А для обучения нейронных сетей на них может потребоваться несколько десятков (а иногда даже тысячи) эпох (одна эпоха – полное прохождение всего набора данных в прямом и обратном направлении).

Основной проблемой является тот факт, что суммарное время обучения нейронной сети может достигать месяцев даже с использованием GPU-ускорителей, что в ряде задач может превышать время, по которому наступает «concept drifting» и не является приемлемым.

По этой причине целесообразно проводить вычисления не на машине, на которой уже развернута нейронная сеть и происходит обработка в реальном времени, а отправлять вычислительное задание на высокопроизводительный кластер. Также для контроля статуса выполнения заданий, устранения дублей заданий, подбора вычислительных ресурсов кластера, которые можно использовать для решения задания и соответствующей корректировки задания-скрипта обучения нейронной сети (который также необходимо реализовать и запускать соответствующим образом для использования всех доступных заданию мощностей и использования распределенных параллельных вычислений) целесообразно использовать систему планирования и управления заданиями, отправляемыми на кластер.

В работе проиллюстрировано взаимодействие приложения на рабочей машине пользователя, удостоверяющее его личность по клавиатурному почерку посредством отправки запросов на сервер, на котором развернута нейронная сеть. Сервер с нейронной сетью должен инициировать отправку задачи дообучения нейронной сети в систему управления заданиями на кластере. Из системы управления заданиями вычислительная задача отправляется на кластер для ускорения вычислений. Затем обученная нейронная сеть заменяет активную (по какому-либо методу разворачивания) на сервере.

# 

# **1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ**

## **1.1 Кластерные вычисления для ускорения вычислений. Виды кластерных систем.**

Существует множество вычислительно сложных задач. К ним можно отнести и компьютерное моделирование, в ходе которого решается ряд научно-технических или инженерных задач. Также к ним относятся и обучение нейронных сетей и классических алгоритмов машинного обучения. В ходе этих процессов производится огромное количество математических операций, для вычисления которых нужно большое количество ресурсов.

Обычные серверные или рабочие станции далеко не всегда обладают объемом вычислительных ресурсов, которые позволили бы быстро и эффективно решить какую-либо вычислительно сложную задачу.

Именно для решения таких задач конструируются вычислительные кластеры, состоящие из большого количества узлов (базовых инженерных объектов), связанных между собой на базе шинной архитектуры или коммутатора. С помощью использования кластера возможно производить вычисления не только параллельно, но и распределенно  
[ГОСТ Р ИСО/МЭК 10746-3-2001].

Также кластерные системы позволяют обеспечить повышенную надежность в решении ряда задач (например, производя вычисления одновременно на двух и более узлах, чтобы, в случае отказа одного из них, вычисления не были прерваны).

Кластерные системы обладают также большим потенциалом к масштабированию, которое реализуется путем увеличения количества вычислительных узлов.

По консистентности вычислительных систем, выделяют два вида кластеров:

* Гомогенные кластерные системы – системы, состоящие из одинаковых узлов;
* Гетерогенные кластерные системы – системы, состоящие из разнотипных узлов;

Отличительной особенностью кластерных вычислительных систем является высокий уровень доступности. Так как каждый отдельный вычислительный узел функционирует самостоятельно, на основе своей операционной системы, как правило, Unix-подобной (отсутствует единая операционная система), а также отсутствует совместно используемая память, это приводит к отсутствию проблемы когерентности кэшей. У каждого узла вычислительного кластера своя память. А результат вычислений на узле по некоторому интерфейсу отправляется на управляющую машину кластера, которая, как правило, агрегирует результаты, полученные на других узлах.

Для обеспечения высокого уровня надежности, доступности и бесперебойности на кластерных системах необходимо наличие специализированного программного обеспечения, которое организовывало бы бесперебойную работу при отказе одного или более узлов. Такое программное обеспечение осуществляет механизм передачи сообщений поверх стандартных сетевых протоколов.

При обмене информацией между узлами используются, как правило, два подхода:

* Передача сообщений (как правило, с помощью MPI – Message Passing Interface – то есть программный интерфейс для обмена сообщениями между процессами, выполняющими одну задачу);
* Совместно используемая память;

Неотъемлемая часть кластерных систем – программное обеспечение, организующее бесперебойную работу в случае отказа одного или более узлов. Такое программное обеспечение должно быть предустановлено на каждом узле вычислительного кластера. Это программное обеспечение способно перераспределить нагрузку в случае аварийной ситуации, а также восстановить вычисления в случае сбоя.

Также в кластерных системах необходимо программное обеспечение, которое могло бы контролировать работоспособность всех узлов. Контроль этот основан на периодической рассылке каждым узлом сигнала “keepalive” или “heartbeat”. В случае отсутствия сигнала в течение некотрого времени – узел считается неактивным и вышедшим из строя.

### [Источник: <http://www.nsc.ru/win/elbib/data/show_page.dhtml?77+858> Кластерные вычислительные системы]

### [Цилькер Б.Я. Организация ЭВМ и систем : Учебник для вузов / Б.Я. Цилькер, С.А. Орлов. - 2-е изд. - СПб.: Питер, 2011. - 688 с. - ISBN 978-5-49807-862-5.]

### Brewer E. Clustering: Multiply and Conquer // Data Communications, July, 1997.

## **1.2 Обучение нейронной сети как пример вычислительно сложной задачи, эффективно решаемую на кластере.**

Как уже было сказано ранее, одной из вычислительно сложных задач в настоящее время является обучение нейронных сетей.

Существует большое число задач, которые могли бы решаться при помощи нейронных сетей. Также в большом количестве задач требуется поддержание нейронных сетей в актуальном состоянии, вследствие “concept drifting”.

“Concept drift” – явление в статистической аналитике и машинном (а также глубоком) обучении, при котором статистические свойства целевой переменной, которые модель (к примеру, нейронная сеть или другая математическая модель) пытается предсказать – изменяются со временем некоторым непредвиденным образом. Это влечет понижение качества работы модели. Ярким примером является модель машинного обучения, распознающая голос конкретного человека. С годами голос человека склонен изменяться, что приведет к тому, что модель перестанет корректно распознавать этот голос. Другим примером может являться модель, распознающая человеческое лицо. Оно также склонно к изменениям, что также повлечет ухудшение распознавательной способности модели применительно к человеку, которого необходимо было распознавать. К другим примерам можно отнести и появление новых игроков на рынке ценных бумаг. Некоторая математическая модель, прогнозирующая курс котировок, станет менее точной вследствие изменения ситуации на рынке и появления других, неучтенных ранее факторов.

### “Concept drifting” может появляться во многих задачах, которые решаются с помощью машинного обучения и нейронных сетей. Поэтому следует учитывать его возможное появление в ряде задач, а в случае его появления – либо дообучать модель, либо обучать модель заново. [The problem of concept drift: definitions and related work , Alexey Tsymbal, 2004, <https://www.researchgate.net/publication/228723141_The_Problem_of_Concept_Drift_Definitions_and_Related_Work>]

### <https://towardsdatascience.com/concept-drift-and-model-decay-in-machine-learning-a98a809ea8d4>

Само по себе обучение представляет из себя большое число математических операций и преобразований, в ходе которых, в случае нейронных сетей, определяются веса между нейронами, содержащими некоторую функцию активации. Обучение проходит на обучающей выборке, качество обучения и способности решать поставленную задачу проверяется на тестовой выборке. Затем обученная нейронная сеть получает способность эффективно решать задачи на подобных объектах в реальном времени.

Для обучения нейронных сетей требуется большое количество информации на вход. Это могут быть сотни изображений, аудиофрагментов или миллионы строк табличных данных.

Обучение нейронных разделяют на следующие категории:

* Обучение с учителем;
* Обучение без учителя;
* Обучение с подкреплением;

Обучение с учителем предполагает наличие размеченных данных, содержащих как данные, которые должны идти на вход, так и данные, которые необходимо получить (например, набор изображений и текстовый файл со списком названий изображений и информацией, есть ли на этих изображениях некоторый объект для задачи распознавания объектов на изображениях). Такое обучение применяется в задачах классификации (к какому классу принадлежит объект, в том числе, к классу изображений с наличием чего-либо) и регрессии (задача выявления зависимости некоторой случайной величины от ряда других величин при помощи аппроксимации функции).

Обучение без учителя применяется в ряде задач, где либо нет размеченных данных, либо их необходимо, некоторым образом, разметить в ходе самой задачи. Примером таких задач может являться кластеризация. В некотором пространстве имеется ряд точек, которые необходимо объединить в кластеры из точек. Также может применяться для обнаружения аномалий на объектах, построения ассоциаций (например, для рекомендательных систем) и автокодировщиков (которые применяются в дальнейшем, например, для машинного перевода).

Обучение с подкреплением применяется для задач, в которых есть некоторая среда с конечным числом возможных состояний. Оно применяется, например, для обучения роботов совершать некоторые действия в ходе большого числа итераций.

Рассмотрим задачу обучения с учителем. Каждый объект при обучении сначала проходит через нейронную сеть, получает некоторый ответ, а затем происходит, так называемое, обратное распространение ошибки, в ходе которого веса нейронной сети корректируются в соответствии с градиентом ошибки.

В данной работе было применено обучение с учителем. Так как обучающие данные, на которых должна была обучаться нейронная сеть должны были быть размечены на метрики, полученные от корректного пользователя и метрики, которые были получены каким-либо другим образом (и при том не являются коррелирующими с метриками корректного пользователя).

Часто обучение происходит в течение большого числа эпох (одна эпоха - полное прохождение всей обучающей выборки через нейронную сеть – в ходе прямого и обратного распространения). Так как даже одно вычисление прямого и обратного прохода одного объекта может иметь большое число математических вычислений – то все обучение, содержащее большое число эпох, может происходить в течение продолжительного времени. На мощнейших CPU (central processing unit) это может занимать недели и даже месяцы. С использованием GPU время обучения может значительно сокращаться благодаря большему количеству блоков арифметико-логических устройств, которые эффективно и быстро производят вычисления матричных операций. Однако даже с использованием нескольких GPU ускорителей на одной вычислительной машине сложно достичь хорошей скорости вычислений.

В настоящее время эта проблема решается специализированными и промышленными GPU-ускорителями, а также с помощью кластерных вычислений. [Аль-хулайди Абдулмаджид Ахмед Галеб (2011) – «Распределенные вычисления (кластерные вычисления) с использованием пакета параллельного программирования»;]

Многие современные фреймворки для машинного и глубокого обучения также имеют встроенную поддержку распределенных кластерных вычислений (pytorch.distributed) или же существуют программные библиотеки, используя которые возможно обучать нейронные сети и классические алгоритмы машинного обучения (такие как градиентный бустинг) распределенно (horovod, dask).

## **1.3 Фреймворки для глубокого обучения и способы организаций распределенных вычислений.**

На текущий момент наибольшей популярностью пользуются следующие фреймворки глубокого обучения:

* TensorFlow (1 и 2 версии);
* PyTorch;
* Sonnet;
* Keras;
* MXnet;
* Gluon;
* CNTK;
* Chainer;
* DL4J;
* ONNX;

Рассмотрим особенности каждого из них.

**TensorFlow** – низкоуровневый инструмент для построения и реализации нейронных сетей. Требует строгого определения размерности, а также объемы возможных входных и выходных данных. В основе фреймворка статический вычислительный граф.

**PyTorch –** в основе фреймворка, в отличие от TensorFlow – динамический вычислительный граф, что дает ему возможность изменять и обновлять архитектуру нейронной сети в процессе работы. Фреймворк поддерживает модели параллелизма данных, а также распределенного обучения (что является огромным преимуществом при использовании кластерных вычислительных систем).

**Sonnet** – является надстройкой и ответвлением фреймворка TensorFlow. Фреймворк позволяет конструировать объекты определенных частей нейронных сетей. Затем подключать их к графу вычислений TensorFlow.

**Keras** – наиболее простой в освоении, но вместе с тем и наименее гибкий инструмент для построения нейронных сетей.

**MXnet** – еще один фреймворк в списке, позволяющий распределенно обучать модели глубокого обучения на GPU.

**Gluon** – также простой в освоении, но по совместительству и базирующийся на динамическом графе вычислений фреймворк. Вместе с тем обладает высокой производительностью.

**CNTK** – фреймворк от компании Windows, имеющий большое количество готовых реализаций для распознавания изображений и речи.

**Chainer** – фреймворк, реализованный на примитивах NumPy и CuPy. Также базирующийся на динамическом графе вычислений. Наиболее часто занимает первые места по производительности среди других фреймворках в ряде задач.

**DL4J** – Фреймворк для языка Java. Обучение нейронных сетей реализовано посредством итераций над кластерами Hadoop и Spark.

**ONNX** – фреймворк для облегчения переноса моделей между другими различными фреймворками.

В ходе анализа был установлен ряд способов организации распределенного обучения нейронных сетей:

**Horovod** – фреймворк для распределенного обучения нейронных сетей реализованных на базе TensorFlow, PyTorch, MXnet и Keras. Для распределенного обучения моделей хоровод использует базовые концепты MPI.

**Pytorch distributed** – механизм, встроенный в фреймворк pytorch. Также использует примитивы MPI для организации распределенного обучения нейронных сетей (но исключительно базирующихся на фреймворке PyTorch).

Проанализируем MPI примитивы, использующиеся в обоих подходах:

Communicator – общее пространство группы процессов. Каждый процесс внутри коммуникатора имеет свой “rank”, который является уникальным идентификатором для MPI и используется для идентификации процесса.

* Point-to-point communicator – обозначает межпроцессорное взаимодействие посредством коммуникатора.
* Collective operation – MPI операции для агрегации вычислений в коммуникаторе.

Разберем некоторые MPI операции:

* Broadcast – копирование тех же данных из корневого процесса в другие процессы.
* Gather – сбор данных из других процессов и помещение их в корневой процесс.
* Scatter – разделение набора данных в корневом процессе на несколько частей и распределение одной части по каждому процессу.
* Reduce – выполнение какого-либо действия (сумма, максимум, минимум и т. д.) и помещение результата в корневой процесс.
* Allreduce – операция reduce и копирование результата в другие процессы.

В качестве фреймворка нейронных сетей в данной работе был выбран TensorFlow (за счет мультиплатформенности и довольно высокой производительности). В качестве фреймворка для организации распределенных вычислений на задаче обучения нейронных сетей был выбран Horovod.

Для распределенного обучения с использованием GPU-ускорителей требуется NCCL-backend. Он реализует операции MPI на базе CUDA-тензоров (которые используются в обучениях нейронных сетей на GPU-ускорителях).

[A Hitchhiker’s Guide On Distributed Training of Deep Neural Networks Karanbir Chahal1 , Manraj Singh Grover2 and Kuntal Dey3]

**1.4 Архитектуры нейронных сетей**

В рассматриваемой задаче повышение производительности обеспечивается за счет повышения скорости вычисления в задаче обучения нейронной сети.

Нейронная сеть – это система объединенных друг с другом нейронов, которые принимают и передают сигналы, получаемые от других нейронов. В нейронах реализована функция активации, которая при тех или иных входных данных активируется, либо не активируется. Базовая структура нейронной сети изображена ниже (*Рисунок 2*).

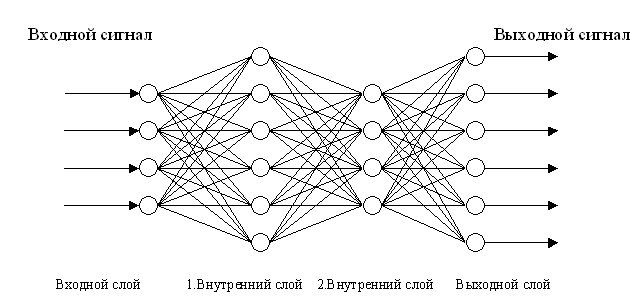


Рисунок 2 Базовая структура нейронной сети

Информация попадает в нейронную сеть через входной слой. Затем в ходе промежуточных слоев совершаются некоторые преобразования, заложенные в ту или иную архитектуру нейронной сети. Затем результат преобразований и вычислений попадает на выходной слой, где уже может каким-либо образом быть интерпретирован.

Схема модели нейрона представляет из себя следующее: сбор некоторой суммирующей функцией сигналы от отправителей. Веса в таком случае представляют из себя некоторые синаптические мощности, а результат – взвешенная сумма. (рисунок 3)

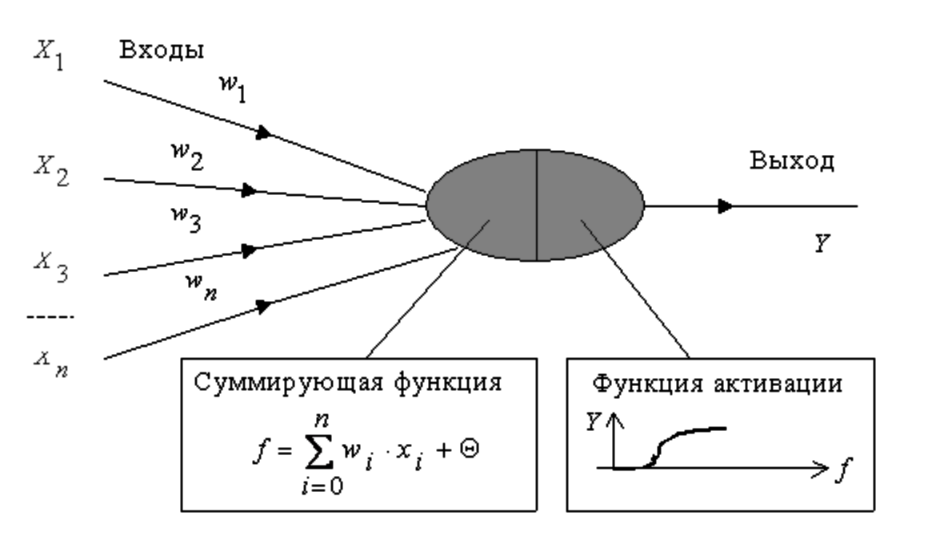


Рисунок 3 Простая математическая модель нейрона

Существует большое количество разновидностей нейронных сетей [22]. Постоянно появляются новые виды, некоторые из них похожи по архитектуре на предыдущие, но различаются методами обучения, или же некоторые схожи методами обучения, но имеют разительные отличия в архитектуре. Есть и уникальные новые сети.

Рассмотрим наиболее базовую архитектуру нейронной сети – сеть прямого распространения. Такая нейронная сеть имеет фиксированные входы и выходы. (рисунок х)

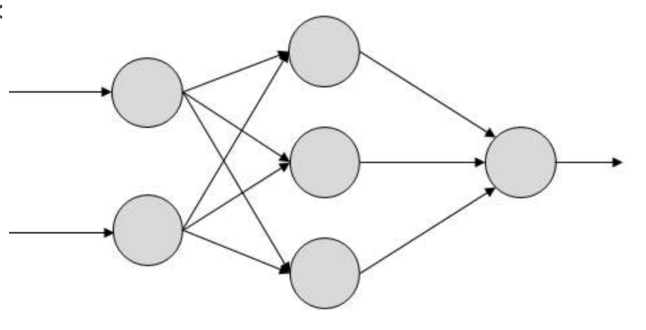


Рисунок 4 - Схематичное изображение нейросети прямого распространения

Другая архитектура – нейронная сеть с обратной связью. В такой нейронной сети результат вычислений одного батча передается в следующий батч и может повлиять на него. (рисунок х)

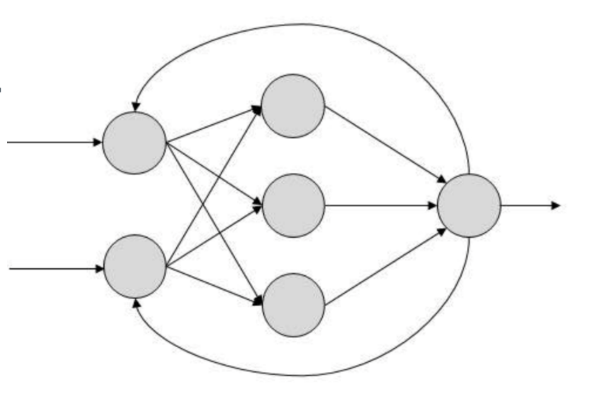


Рисунок 4 - Схематичное изображение нейросети прямого распространения

Примером таких архитектур могут являться LSTM или GRU – рекуррентные нейронные сети. Такие архитектуры часто находят применение в задачах Natural Language Processing или анализе временных рядов. (рисунок х)

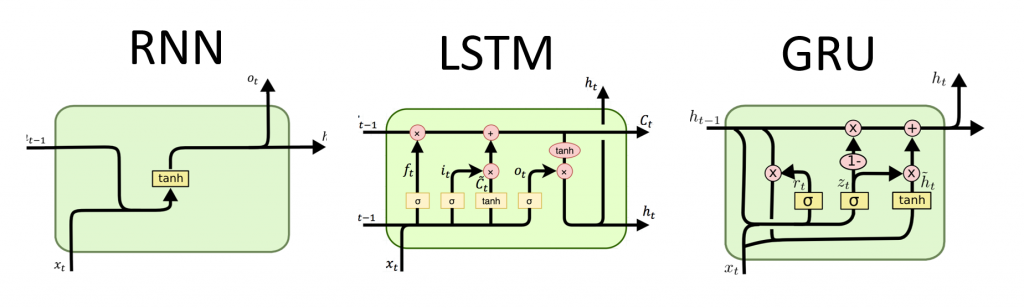


Рисунок 4 - Схематичное изображение нейросети прямого распространения

Для распознавания образов на изображениях наиболее часто применяются сверточные архитектуры нейронных сетей. Отличительной особенностью таких нейронных сетей является ядро свертки, которое проходит по областям батча (это могут быть изображения, звук, табличные данные), извлекает информацию о признаках и характеристиках в рамках этого окна, а затем передается на следующий слой для дальнейших преобразований. (*Рисунок 10*).

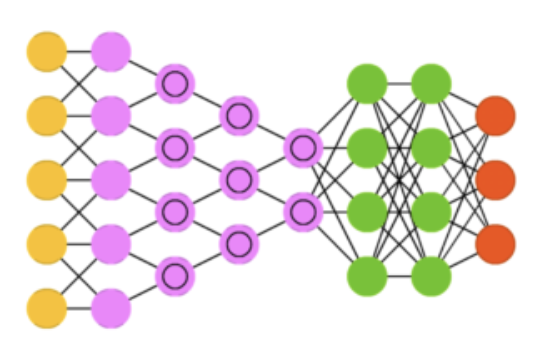


Рисунок 10 Схема сверточной нейросети

Еще одна разновидность архитектур нейронных сетей – сети Хопфилда. Это сети с симметричной матрицей связей. Каждый узел в таких сетях является входом, промежуточным слоем и выходом. Такие сети часто называются сетями ассоциативной памяти. (рисунок х)

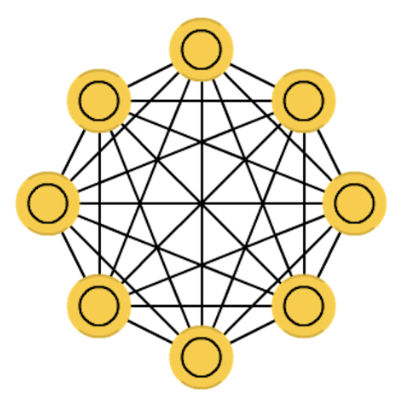


Рисунок 6 - Схематичное изображение нейронной сети Хопфилда

Схожей архитектурой обладают и сети Больцмана. Основная разница в том, что некоторые нейроны помечены исключительно как входные, а некоторые исключительно как выходные. В такой нейронной сети нейроны одного типа не связаны между собой. (*Рисунок 8*).

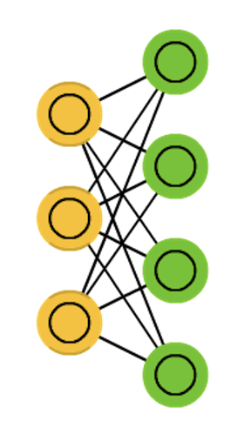


Рисунок 8 - Схематичное изображение сети ограниченной машины Больцмана

Следующая возможная архитектура нейронной сети – автокодировщик (в настоящее время появились архитектуры, называемые «трансформерами», которые являются усовершенствованным видом автокодировщиков). Идея этих нейронных сетей заключается в автоматическом сжатии информации, кодирования, а затем декодирования.

## **1.5 Выводы**

ТУТ НАДО НАПИСАТЬ

# **2 КОНСТРУКТОРСКИЙ РАЗДЕЛ**

## **2.1 Архитектура системы для ускоренного вычисления задач при помощи кластера.**

Рассмотрим в следующую ситуацию: на некотором сервере развернута нейронная сеть. Эта нейронная сеть подвержена “concept drift” и её необходимо обучать заново с некоторой периодичностью. Так как ресурсы сервера загружены вычислениями результата прогноза для объектов, подающихся на вход посредством API сервера в нейронную сеть, то обучение нейронной сети следует провести на удаленной вычислительной машине. По причине того, что сама операция полного обучения нейронной сети достаточно ресурсозатратная, необходимо использовать высокопроизводительные вычислительные машины. Примером таких вычислительных машин может являться кластер. Обучение будет производиться на вычислительном кластере с использованием распределенных и параллельных вычислений для повышения скорости решения задачи обучения.

Для эффективности использования вычислительного кластера целесообразно использовать систему контроля заданий. В обязанности такой системы должно входить следующее:

* Контроль дублей задач (удаление лишних);
* Контроль, логирование и учёт списка задач;
* Управление запуском задач и запуск с параметрами, соответствующим текущему статусу кластера;
* Отслеживание загрузки/выгрузки файлов в хранилища и запуск событий по окончанию;
* Перезапуск задач в случае необходимости;
* Мониторинг статусов и смены статусов выполняемых задач;
* Управление статусами задач при необходимости;
* Контроль обновления и замены обученной нейронной сети на сервере;
* ???

Все задания с сервера, которые необходимо будет обработать на кластере, сначала посылаются на эту систему контроля заданий.

Рассмотрим детально все обязанности этой системы.

**Контроль дублей задач** — эта обязанность вызвана необходимостью контролировать ситуацию, при которой в ходе какой-либо ошибки (прикладного ПО сервера, сетевой и т.д.) поступает более одного одинакового задания с необходимостью вычисления на кластере. Если задания абсолютно идентичны – необходимо обрабатывать на кластере лишь одно из них.

**Контроль и логирование списка задач** — данная обязанность является следствием необходимости учета всех задач за некоторый промежуток времени для дальнейшего возможного анализа, изучения истории и непосредственно учёта задач, отправляемых на вычисление на кластер.

**Управление запуском задач и запуск с параметрами, соответствующим текущему статусу кластера** — одна из ключевых функциональностей системы управления заданиями – это корректировка отправляемых заданий в соответствии с текущей нагрузкой и свободными ресурсами кластера. Данный функционал должен обеспечивать отправку задания на кластер, принимая в учет другие задания в списке, в очереди, свободные ресурсы кластера, оценочное время выполнения.

**Отслеживание загрузки/выгрузки файлов в хранилища и запуск событий по окончанию** — эта функция системы управления заданиями должна контролировать старт, окончание отправки файлов с кластера на объектное хранилище, с объектного хранилища на кластер, с объектного хранилища на сервер или с сервера на объектное хранилище и вызывать необходимые процедуры по окончанию загрузки.

**Перезапуск задач в случае необходимости** — данный функционал должен обеспечивать гарантированную выполнимость нужного задания вне зависимости от каких-либо факторов (в частности, в случае аварии на некоторых кластерных узлах, компонент должен отследить это событие и запустить задание на активных узлах повторно, возможно, внеся корректировки в параметры задания.

**Мониторинг статусов и смены статусов выполняемых задач** — наличие данного функционала обусловлено тем, что далеко не все кластерные вычислительные системы обладают функционалом обратного вызова по наступлению каких-либо событий, поэтому отслеживать задания или процессы необходимо самостоятельно.

**Управление статусами задач при необходимости** — эта функция при появлении более актуального задания останавливать старое (при соблюдении некоторых условий, например, если задание отрабатывало менее некоторого временного периода) и заменять на более актуальное. Затем необходимо изменять условие, блокирующее запуск нового более актуального задания (например, уменьшать время, по истечению которого, задание считается незаменяемым), с целью предотвращения бесконечного чередования актуальных заданий без фактического окончания какого-либо из них.

**Контроль обновления и замены обученной нейронной сети на сервере** — данная функциональность не является обязательной, но в рамках этой работы было целесообразно реализовать контроль обновления нейронной сети как на самом сервере, так и на системе контроля заданий (чтобы внести соответствующую информацию в поле в списке заданий в базе данных).

Рассмотрим ключевые интерфейсы всех подсистем, участвующих во взаимодействии.

Внешние REST API интерфейсы сервера, на котором развернута нейронная сеть (вычислительная станция, производительность которой повышается путем отправки некоторых вычислительных заданий на вычислительный кластер):

* Predict – возвращает результат вычисления на основе собранного батча данных от пользователя по метрикам, характеризующим клавиатурный почерк пользователя, является ли он в действительности тем, от чьего имени была проведена аутентификация.
* Также сервер должен отслеживать “concept drifting”. В рамках данной работы он отслеживается путем сопоставления статистической информации из обучающей выборки, характеризующей клавиатурный почерк пользователя и данных, который поступают на вход в нейронную сеть. В рамках задачи опытно-экспериментальным путём было выбраны следующие интервалы:
  + Нейронная сеть возвращает вероятность корректного распознавания пользователя в диапазоне [0; 0.65] – пользователь считается непрошедшим аутентификацию по клавиатурному почерку.
  + Нейронная сеть возвращает вероятность успешной аутентификации в диапазоне [0.65; 0.8] – пользователь считается прошедшим аутентификацию, но также считается обнаруженным “concept drift”. Создается задача на дообучение нейронной сети на основе базовых данных вместе с батчем, полученным от пользователя. Задача формируется и отправляется либо непосредственно на кластер (в случае неиспользования системы управления заданиями), либо на систему управления заданиями с дальнейшей отправкой задания на вычислительный кластер.
  + Нейронная сеть возвращает вероятность корректного распознавания в диапазоне [0.8; 1] – пользователь считается успешно прошедшим аутентификацию по клавиатурному почерку.

Ниже приведен интерфейс, который необходим в системе управления заданиями для взаимодействия сервера с ним:

* Система управления заданиями должна принимать батч данных для обучения нейронной сети и создавать соответствующее задание в базе данных, а затем управлять его состоянием, передачей данных между компонентами системы, а также возвращать сериализованную обученную нейронную сеть в систему хранения сервера для дальнейшего развертывания на сервере и замены устаревшей нейронной сети.

Далее перечислены методы внешнего программного интерфейса, которые имеет выбранный кластер (для примера реализации метода был выбран кластер Кристофари – мощнейший вычислительный кластер в России на данный момент) и с которыми должна взаимодействовать система управления заданиями:

* **Аутентификация.** Принимает пару email, password, возвращает токен доступа и токен обновления, которые нужные для последующего использования других методов внешнего интерфейса кластера от лица, чьи логин-пароль были отправлены.
* **Запуск задачи на кластере**. Принимает на вход адрес скрипта в локальном хранилище, образ, на котором в каждом поде-воркере должен быть исполнен скрипт, количество узлов исполнителей и количество GPU-ускорителей на каждом узле, необходим ли прогрев кэша (предварительная загрузка данных), а также ряд других опциональных параметров. В результате ставит задачу в очередь к исполнению на кластере.
* **Получение статуса задачи.** Используя уникальный идентификатор задачи, который генерируется в момент создания задачи, позволяет узнать статус задачи на данный момент.
* **Остановка и удаление задачи.** Используя уникальный идентификатор задания, позволяет удалить задачу из очереди и прекратить выполнение на кластере.
* **Получение логов задачи.** Используя идентификатор задачи, позволяет получить весь вывод, который был получен на момент. Является Streaming API.
* **Получение списка задач.** Метод API возвращает список задач пользователя на кластере вместе со статусами и другой метаинформацией.
* **Сохранение настроек доступа к объектному хранилищу для передачи файлов и данных.** Метод позволяет установить соединение на кластере с аккаунта пользователя к объектному облачному хранлищу.
* **Перемещение файлов между объектным хранилищем и NFS-дисками кластера.** Метод позволяет создавать задачу переноса данных из локального хранилища кластера на объектное облачное хранилище. Также и из объектного облачного хранилища в локальное хранилище кластера.
* **Завершение сессии.** Прерывает сессию пользователя и удаляет активированные токены доступа на стороне вычислительного кластера.

Диаграмма потоков данных, отражающая данный подход и метод, а также саму реализацию, между компонентами системы отслеживания заданий, внешнего хранилища и вычислительного кластера изображена ниже:

# 

Рисунок X. Диаграмма потоков данных

Архитектура взаимодействия кластера и системы отслеживания заданий приведена ниже:

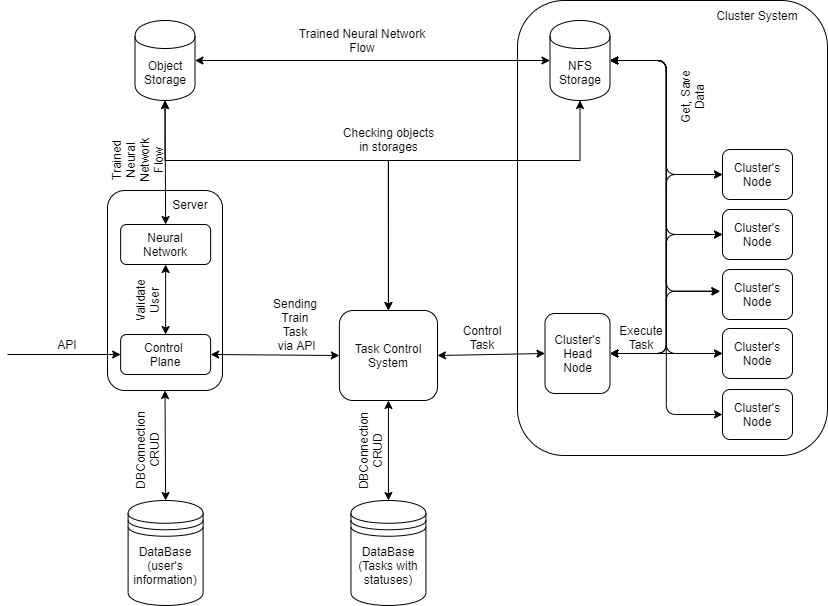


Рисунок X. Архитектура взаимодействия кластера и системы отслеживания заданий

На ней отображены все необходимые базы данных и все хранилища, которые взаимодействуют с теми или иными компонентами. Также на ней отображено ключевое взаимодействие компонентов сервера с развернутой нейронной сетью и вычислительного кластера с компонентом Task Control System.

Архитектура всей системы, включая компоненты на клиентской части (а именно на пользовательском десктопе) отображены ниже. На ней также отображено взаимодействие любого открытого на пользовательском компьютере приложения и приложения, собирающего метрики клавиатурного почерка и отправляющего на вход нейронной сети на сервер для проведения фоновой аутентификации по самому клавиатурному почерку.

Взаимодействие с системой контроля заданий реализовано на основе REST API. Сама система контроля заданий взаимодействует с кластером (для примера был выбран крупнейший, на момент написания работы, в России кластер Кристофари) также по REST API. Обмен сообщениями происходит в асинхронном режиме. Система контроля заданий имеет свою собственную базу данных, в которой хранится и обновляется информация о новых заданиях, поступивших на кластер, обработанных на нем, а также о других обновлениях статуса заданий в системе.

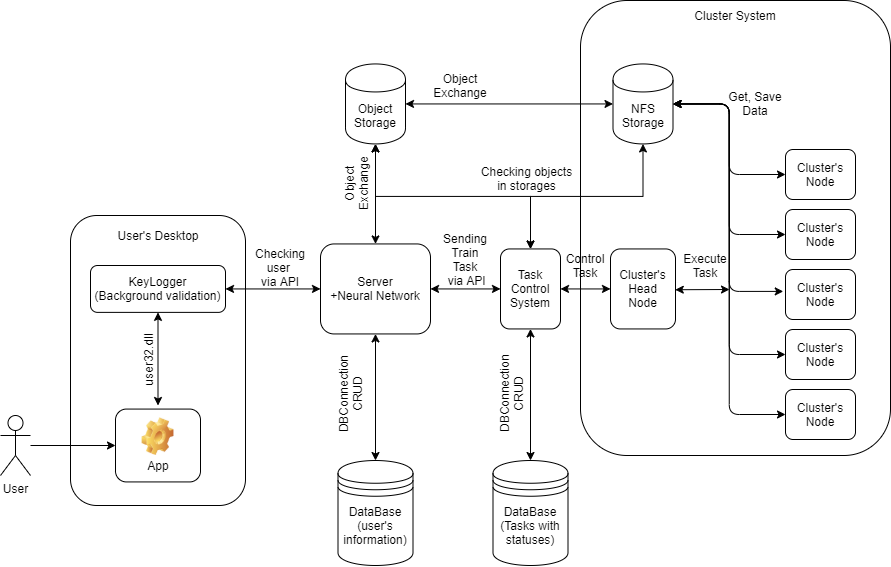


Рисунок X. Архитектура всей системы, представленной в работе

В результате успешной отправки задания на кластер в сервисе, реализующем взаимодействие с кластером, можно отследить задания и их статус (рисунок х):

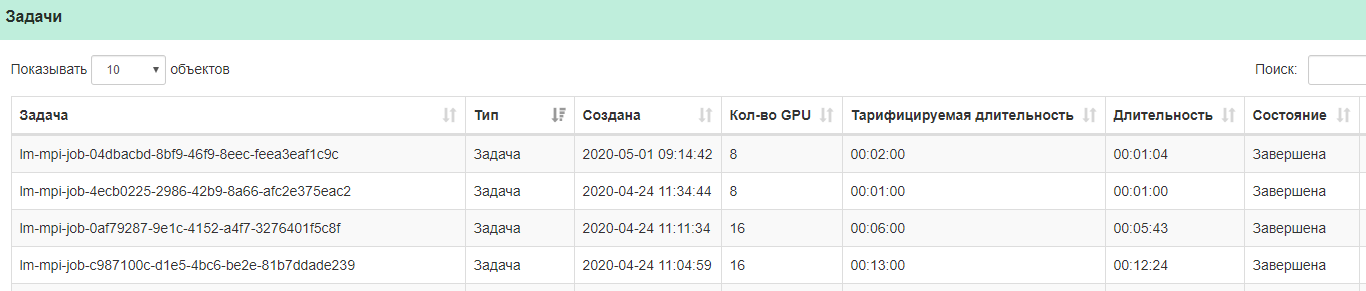


Рисунок X. Архитектура всей системы, представленной в работе

## **2.2 Пользовательское приложение для сбора метрик и взаимодействие его с системой контроля заданий.**

В качестве системы, взаимодействующей с системой контроля сообщений, будет взаимодействовать пользовательское приложение. В данной работе в рамках апробации и реализации метода ускорения вычислений на сервере для полноценной связанной системы было разработано приложение для сбора пользовательских метрик во время работы, характеризующие его «клавиатурный почерк» для дальнейшего подтверждения аутентификации и обладания доступа к каким-либо конфиденциальным данным.

В состав такого приложения входят модули для следующих операций:

* Сбор метрик по началу нажатия на каждую клавишу клавиатуры (время до миллисекунд);
* Сбор метрик по окончанию нажатия (отпусканию) на каждую клавишу клавиатуры (время до миллисекунд);
* Составление файла с минимально необходимым набором по данным метрикам;
* Отправка файла на валидацию на сервер по API;
* Получение результата от API сервера и соответствующие действия по разрешению/блокированию доступа;

Данное программное обеспечение было реализовано на основе пакета user32.dll.

Данное приложение имеет следующую диаграмму классов:

Итоговый формат собранных метрик, который приложение будет отправлять на сервер выглядит следующим образом (рисунок х):

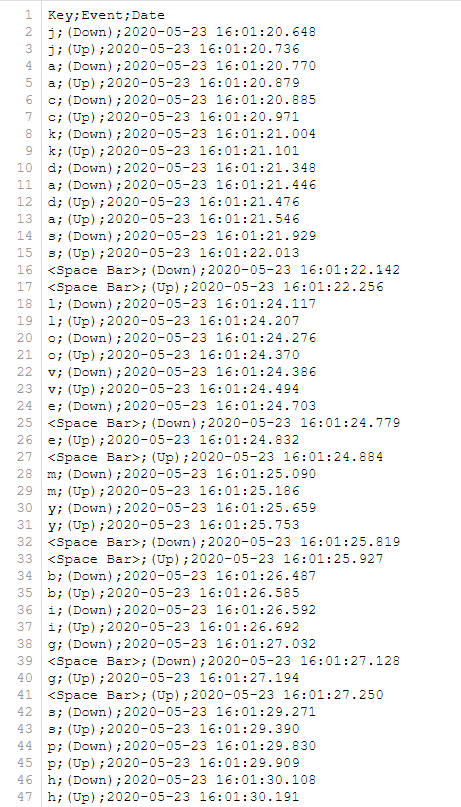


Рисунок X. Формат собранных пользовательским приложением данных

Реализованное приложение занимается сбором:

* Событий нажатий и отпускания клавиш (соответствует колонке ‘Event’),
* клавиш клавиатуры (параметр ‘Key’ отвечает за то, событие какой клавиша нажатия или отпускания было зафиксировано)
* меток времени, когда событие произошло (параметр ‘Date’).

## **2.3 Архитектура нейронной сети и методы распределенных вычислений на кластере, применительно к ней.**

ВОТ ТУТ ПОНАПИСАТЬ ДА

# **3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ**

## **3.1 Выбор технологического стека и реализация**

В качестве фреймворков для реализации компонентов системы был выбран следующий стек:

В качестве среды разработки были выбраны:

* PyCharm – для проектирования и реализации микросервисов (микросервис с нейронной сетью и микросервис с системой управления задачами);
* Jupyter Notebook – для препроцессинга данных и реализации архитектуры нейронной сети, а также для ее первоначального обучения;
* Visual Studio – для проектирования и реализации системы сбора метрик пользовательского клавиатурного почерка.

Система управления заданиями была реализована с помощью языка Python и фреймворка веб сервисов – Django Framework.

…

База данных, интегрированная с микросервисом была реализована на основе ?????????????????????????

В качестве фреймворка глубокого обучения для построения нейронной сети выступал фреймворк TensorFlow 2, базирующийся на языке программирования Python. В качестве фреймворка распределенных вычислений (и распределенного обучения модели) выступал Horovod, который оптимизирован в том числе и под TensorFlow. С помощью связки Tensorflow – Horovod возможно реализовать нейронные сети, решающие любые задачи классификации, кластеризации и регрессии с использованием распределенного обучения.

В качестве сервера с развернутой нейронной сетью также выступал микросервис, реализованный на Django Framework по ряду следующих причин:

* Полная совместимость сред для развертывания нейронных сетей. Нейронные сети, написанные на Python и обученные в Jupyter Notebook, при наличии подключенных библиотек на сервере и чекпоинтов/сериализованной модели, из которых будет развернута модель глубокого обучения хорошо встраиваются в микросервис, реализованный на языке Python. В дальнейшем обращение к модели происходит напрямую в рамках реализованного микросервиса.
* 123123123

Приложение для сбора пользовательских метрик было реализовано на языке C# с использованием модулей user32.dll (библиотека функций пользовательского интерфейса – Windows API) на основе Windows Forms. Выбор этого типа приложения обусловлен корпоративным стандартом операционных систем, принятого в подавляющем большинстве корпораций и компаний – Windows. Данный фреймворк позволяет успешно отслеживать события нажатий клавиш на клавиатуре в фоновом режиме. Также с помощью компонента HTTP.Client, входящего в библиотеку пакетного менеджера NuGet была реализована отправка батчей данных о пользовательском клавиатурном почерке на микросервис с развернутой нейронной сетью. Сбор событий был реализован через Windows hook посредством установления на адреса функций, вызов которых необходимо перехватывать (а именно нажатие на клавишу клавиатуры и отпускание клавиши клавиатуры). Этими адресами являются msgType = 0x0100 и msgType = 0x0101. Вся информация, собранная в течение устанавливаемого времени записывается в файл. Затем конвертируется в Json-структуру и отправляется в HTTP REST запросе на микросервис с развернутой моделью нейронной сети.

## **3.2 Детали подготовки данных для обучения и тестирования системы**

Препроцессинг тыры пыры картинки до после выбросов, почему выбрасываем, как шлём.

## **3.3 че-нить про систему обработки заданий и реализацию библиотеки про токены там и тп**

## **3.4 Выводы**

# **4 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ**

## **4.1 сравнения по времени и производительности, перспективы развития**

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Князев А. В. Распознавание слитного рукописного текста / Вестник МЭИ, 2013. С. 15.
2. Попова В. В. Распознавание рукописного текста на китайском языке на основе ключей при помощи нейронных сетей // Информационные технологии XXI века, 2013. С. 47
3. РАСПОЗНАВАНИЕ РУКОПИСНЫХ СИМВОЛОВ С ПРИМЕНЕНИЕМ НЕЙРОСЕТЕВОЙ ТЕХНОЛОГИИ, Андреев Всеволод Владимирович, Журавлев Михаил Сергеевич, // Прикладная Информатика № 2, с. 115-117
4. Исрафилов Х. С. Применение нейронных сетей в распознавании рукописного текста // Молодой ученый. - 2016. - №29. - С. 24-27 [Электронный ресурс]. – Ресурс доступа: <https://moluch.ru/archive/133/37372/> (Дата обращения: 2018-11-15)
5. Кулакова О.А., Воронова Л.И. // Московский Технический Университет Связи и Информатики. - 2017. С. 10-11.
6. Comp. nanotechnol., 2016, выпуск 3, страницы 224–241. Защита информации. Биометрические технологии идентификации в системах контроля и управления доступом. С 224-241.
7. Алгоритмы оффлайн-распознавания рукописных цифр. Козлов Владимир Дмитриевич. Выпускная квалификационная работа. Научный руководитель к. ф.–м. н, доцент С. И. Гуров. Московский государственный университет имени М. В. Ломоносова. С. 73.
8. Маркова С.В., Жигалов К.Ю. Применение нейронной сети для создания системы распознавания изображений // Фундаментальные исследования. – 2017. – № 8-1. С 60-64.
9. Geoffrey E. Hinton. Dynamic Routing Between Capsules // Google Brain, 2017. С 15-18.
10. 36 материалов о нейросетях: книги, статьи и последние исследования [Электронный ресурс]. – Ресурс доступа: [https://habrahabr.ru/company/neurodatalab/blog/336218](https://habrahabr.ru/company/neurodatalab/blog/336218/) (дата обращения: 11.02.2019)
11. Применение нейросетей в распознавании изображений [Электронный ресурс]. – Ресурс доступа: https://geektimes.ru/post/74326 (дата обращения: 19.01.2019)
12. Метрики качества [Электронный ресурс]. – Ресурс доступа: <http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html>(дата обращения: 21.01.2019)
13. Калькулятор рецептивных полей [Электронный ресурс]. – Ресурс доступа: https://fomoro.com/projects/project/receptive-field-calculator (дата обращения: 11.12.2018)
14. Pooling // объединение слоев [Электронный ресурс]. – Ресурс доступа: http://datareview.info/article/eto-nuzhno-znat-klyuchevyie-rekomendatsii-po-glubokomu-obucheniyu-chast-2 (дата обращения: 21.01.2019)
15. Набор данных для обучения и тестирования [Электронный ресурс]. – Ресурс доступа: http://www.iapr-tc11.org/mediawiki/index.php/ICDAR\_2011\_Signature\_Verification\_Competition\_(SigComp2011) (дата обращения: 15.02.2019)
16. Сверточная нейронная сеть на PyTorch[Электронный ресурс]. – Ресурс доступа: [https://neurohive.io/ru/tutorial/cnn-na-pytorch](https://neurohive.io/ru/tutorial/cnn-na-pytorch/) (дата обращения: 10.03.2019)
17. Как работает FaceID в iPhone X: алгоритм на языке Python[Электронный ресурс]. – Ресурс доступа: <http://ai-news.ru/2018/03/kak_rabotaet_faceid_v_iphone_x_algoritm_na_yazyke_python.html> (дата обращения: 14.04.2019)
18. Face ID Security Guide. – Ресурсдоступа: <https://www.apple.com/business/site/docs/FaceID_Security_Guide.pdf>
19. Зарплаты ИТ-специалистов на середину 2018 года [Электронный ресурс]. – Ресурс доступа: <https://vc.ru/flood/43849-zarplaty-it-specialistov-na-seredinu-2018-goda> (дата обращения: 01.05.2019)
20. EnterpriseAndroidvs IOS [Электронный ресурс]. – Ресурс доступа: <https://medium.com/predict/enterprise-android-vs-ios-which-is-more-secure-51a95cac53c2> (дата обращения: 11.03.2019)
21. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Электронный ресурс]. – Ресурс доступа: <https://habr.com/ru/post/348000/> (дата обращения: 23.02.2019)
22. Градиентый бустинг [Электронный ресурс]. – Ресурс доступа: <https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/> (дата обращения: 26.04.2019)
23. Разновидности нейронных сетей [Электронный ресурс]. – Ресурс доступа: <https://tproger.ru/translations/neural-network-zoo-1/> (дата обращения: 05.05.2019)
24. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес [Электронный ресурс]. – Ресурс доступа: <https://habr.com/ru/company/ods/blog/324402/#1-begging> (дата обращения: 03.05.2019)
25. Логистическая функция ошибки [Электронный ресурс] <https://dyakonov.org/2018/03/12/логистическая-функция-ошибки/> (дата обращения: 03.05.2019);
26. Майков К.А. Козлова Ю.А. Метод идентификации пользователя по рукописному кодовому слову // Новые информационные технологии в автоматизированных системах, М.- 2019, с.19-21;

# **ПРИЛОЖЕНИЕ A**