DOI 10.37882/2223–2966.2021.08.28

КЛАССИФИКАЦИЯ ТЕКСТА С ИСПОЛЬЗОВАНИЕМ TENSORFLOW.JS

TEXT CLASSIFICATION USING TENSORFLOW.JS

A. Prokhorov

Summary. The topic of this paper is the use of machine learning on the example of the TensorFlow software package in its implementation for the JavaScript programming language for the task of natural language recognition. The relevance of the research and subsequent development of the software system is due to the rapid development of artificial intelligence and the increasingly frequent need for classification and categorization of large volumes of data (Big Data) in information systems. Objective: to develop an optimal natural language classification system using artificial intelligence tools, such as machine learning. Methods: when working on the study, the analysis method was used to study the principles of neural networks, as well as the features of TensorFlow implementation. When testing the system, methods of comparison and empirical measurement of performance were used. Results: the result of the research is a software system that classifies data sets of the subject area containing words and sentences of the natural language. Also, system performance tests were conducted on central processors and video accelerators to identify ways to increase the speed of operation and improve scalability. Conclusions: in the course of the study, the most suitable type of neural network was determined, conclusions were obtained about the optimal software settings of the model for the tested data, as well as optimal hardware configurations. The study of this material will allow beginners in the field of machine learning to quickly move on to developing models and solving real-world applied problems.

Keywords: TensorFlow, JavaScript, machine learning, ML, neural network, web applications, search, NVIDIA, GPU, CPU.

Прохоров Андрей Валерьевич
Программист, АО «Семейный Доктор», Москва
mobilelookfree@gmail.com

Аннотация. Тема настоящей работы — использование машинного обучения на примере программного комплекса TensorFlow в его реализации для языка программирования JavaScript для задачи распознавания естественного языка. Актуальность исследования и последующей разработки программной системы обусловлена бурным развитием искусственного интеллекта и все более частой необходимостью в классификации и категоризации больших объёмов, данных (Big Data) в информационных системах. Цель: разработка оптимальной системы классификации естественного языка средствами инструментов искусственного интеллекта, таких как машинное обучение. Методы: при работе над исследованием использовался метод анализа для изучения принципов работы нейронных сетей, а также особенностей реализации TensorFlow. При тестировании системы применялись методы сравнения и эмпирического измерения производительности. Результаты: результатом исследования является программная система, осуществляющая классификацию наборов данных предметной области, содержащих слова и предложения естественного языка. Также были проведены тесты производительности системы на центральных процессорах и видеоускорителях для выявления способов повышения скорости работы и улучшения масштабируемости. Выводы: в ходе исследования определен наиболее подходящий тип нейронной сети, получены выводы об оптимальных программных настройках модели для тестируемых данных, а также оптимальных аппаратных конфигурациях. Изучение данного материала позволит новичкам в сфере машинного обучения быстро перейти к разработке моделей и решению реальных прикладных задач.

Ключевые слова: TensorFlow, JavaScript, машинное обучение, ML, нейросеть, веб-приложения, поиск, NVIDIA, GPU, CPU.

Введение

Современные компьютерные системы в большинстве случаев работают с большими объемами неструктурированных или же мало структурированных данных — так называемым большими данными (big data). Обработку таких данных обычно отводят программным инструментам с хорошей горизонтальной масштабируемостью: эти системы представляют собой связанную сеть вычислительных узлов, которую можно расширять при увеличении нагрузки. Коммуникация между узлами осуществляется при помощи сетевых протоколов и программных интерфейсов обработки параллельных вычислений, таких как MPI.

Машинное обучение (Machine Learning — ML) с момента своего появления в ЭВМ решало задачи классификации паттернов и наборов данных. Основная цель и идея машинного обучения — позволить компьютерам обучаться самим, автоматически и без вмешательства человека. Однако на первичном этапе обучение им все-таки требуется.

Обучать модель обработки данных — нейронную сеть — можно разными способами: с учителем или же без него. Обучение с учителем (supervised learning) предполагает наличие полного набора размеченных данных для тренировки модели на всех этапах ее построения. Наличие подготовленного набора данных

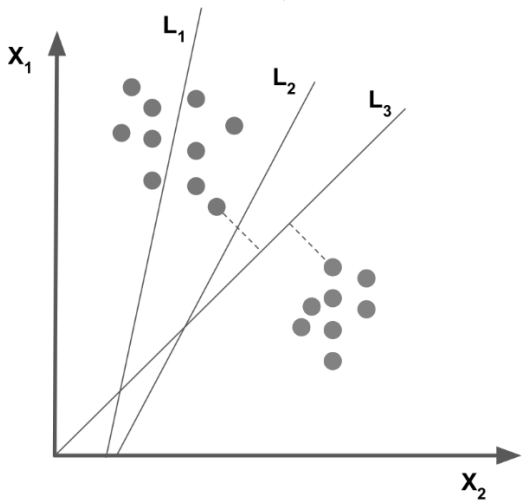

Рис. 1. Классифицируемые гиперплоскостью объекты в SVM

означает, что каждому тестовому случаю в обучающем наборе соответствует ответ, который модель и должна выдать. Обучение с учителем эффективно для решения задач классификации — именно этот метод используется в рамках предметной задачи доклада.

Обучение без учителя (unsupervised learning) предполагает наличие набора данных без соответствующих тестовым случаям ответам. Обучение модели происходит естественным путем. Основная причина использования обучения без учителя — сложность и дороговизна получения подготовленных данных, в более редких случаях — неочевидность классификации и аномалий данных для экспертов.

Сейчас машинное обучение — это один из основных способов применения искусственного интеллекта в компьютерных системах при работе с данными различного объема и структурной сложности. ML применяется в огромном числе программных продуктов: существуют как библиотеки с открытым исходным кодом, так и фреймворки от ИТ-гигантов, таких как Apple (CoreML), Google (TensorFlow), Microsoft (Azure ML), IBM (Watson ML). Также аппаратное ускорение операций машинного обучения получают актуальные модели процессоров Intel, AMD, а также решения на архитектуре ARM и графические ускорители (GPU).

Литературный обзор

Для обучение нейронных сетей, осуществляющих классификацию текста, наиболее часто используют метод свертки, метод опорных векторов — SVM (Support Vector Machine), метод k-ближайших соседей — KNN (K-Nearest Neighbors) и наивный байесовский классификатор (Naive Bayes Classifiers — NBC).

Метод опорных векторов [1] — один из наиболее популярных методов обучения с учителем, который применяется для решения задач классификации и регрессии. Принцип метода заключается в построении гиперплоскости (Рис. 1), разделяющей объекты выборки оптимальным способом.

Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора. Это эквивалентно тому, что сумма расстояний до гиперплоскости от двух ближайших к ней точек, лежащих по разные стороны от неё, максимальна. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором.

Метод k-ближайших соседей [2] относится к классу непараметрических, т.е. не требует предположений о том, из какого статистического распределения был сформирован обучающий набор данных. Следовательно, классификационные модели, построенные с помощью метода KNN, также будут непараметрическими. Это означает, что структура модели не задаётся изначально, а определяется на основе данных. Так как признаки, на основе которых осуществляется классификация, могут иметь различную физическую природу и диапазоны значений, для улучшения результатов классификации будет полезно выполнить нормализацию обучающих данных.

Алгоритм KNN использует в своей работе две фазы: обучение и классификация. При обучении алгоритм запоминает векторы признаков наблюдений и их метки классов. Также задаётся параметр алгоритма k, который задаёт число соседей, которые будут использоваться при классификации. На фазе классификации для объекта без метки класса определяются k ближайших предварительно классифицированных наблюдений. Затем выбирается класс, которому принадлежит большинство из k ближайших примеров-соседей, и к этому же классу относится классифицируемый объект.

Самый простой способ классификации текста основан на использовании теоремы Байеса (1), которая позволяет определить вероятность какого-либо события (A) при условии, что произошло другое статистически взаимозависимое с ним событие (B).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Наивный байесовский классификатор [3] используется для распознавания и классификации, которые подпадают под различные вариации в классификаторах образов по базовой вероятности и правдоподобию. Наивные байесовские техники основаны на самоопределяющихся предположениях.

Однако, не только выбор алгоритма обучения определяет эффективность работы модели нейронной сети. Как показывают современные исследования, для классификации текста крайне эффективны **сети сверточного тип**а — CNN. В данные сетях каждый слой использует собственное преобразование данных путем свертки.

Исследование [4], проведенное в 2018 году группой сотрудников отдела биомедицинской статистики и информатики Департамента медицинских исследований клиники Мэйо штата Миннесота, показало, что CNN-сети лучше других нейронных сетей и алгоритмов справляются с задачей классификации. Сравнение проводилось с: многослойным персептроном (MLP), методом опорных векторов и методом «случайного леса» (RF). Превосходство CNN было выявлено в тестах как при обучении с учителем, так и без него. Данными и задачей классификации служило распознавание статуса курения и переломов тазобедренного сустава. CNN также может захватывать дополнительные шаблоны даже при обучении без учителя, что выгодно отличает ее от других моделей.

В ходе исследования выявлены и недостатки CNN: это высокая чувствительность к объему обучающих данных в сравнении с другими типами сетей и алгоритмов, а также низкая эффективность в задачах сложной много классовой классификации: например, классификация курения.

Сотрудники кафедр математики Университета Патр и Технологического образовательного института западной Греции в своем исследовании [5] классификаторов электронных документов пришли к двум выводам:

1. Эффективность классификаторов связана с их обучающим набором данных (corpus);
2. Качественные данные для обучения и тренировочные данные для повторного обучения могут с гарантировать классификацию высокой точности.

Авторы отмечают, что больших объемах обучающих данных стоит снижать число переменных путем получения главных переменных, когда при маленьких размерах датасетов это менее критично. При работе с малыми объемами обучающих данных важны настройки модели, а именно число эпох и величина допустимого отклонения: при многократном обучении на малых датасетах происходит «переобучение» (overfitting) модели — шаблонизация ее прогнозов.

Говоря о «переобучение» модели, нельзя не затронуть вопрос оптимального количества эпох обучения. Согласно ряду испытаний [6], для большинства задач достаточно ограничиться 10–12 итерациями обучения. Далее наступает «переобучение», и точность результатов падает.

Резюмируя, наиболее оптимальным типом нейронной сети для классификации данных, особенно текстовых, является CNN, особенно если объем обучающих данных значителен. Но, к сожалению, литература о машинном обучении мало уделяет внимания обучающим данным и особенностям их подготовки, а преимущественно лишь алгоритмическим составляющим моделей. Однако при низком качестве данных даже самый эффективный алгоритм не даст должного эффекта при применении. Исследования энтузиастов машинного обучения постепенно приносят некоторые практики подготовки данных, принципы работы с ними и способы решения проблем обучения моделей, однако еще не на все вопросы даны однозначные ответы.

Материалы и методы

Для обучения нейронной сети использовались данные комплексной оценки ОПК, содержащие порядка 10 тысяч записей об организациях на различные темы: специфика предметной области деятельности организации; уровень технического оснащения производства; нормативные документы; данные хозяйственного учета; плановая информация; данные финансовой и иной отчётности и т.п. При помощи подготовленных скриптов на языке JavaScript данные были приведены к формату JSON, а схема их полей сохранена для после-

дующего формирования наборов данных для обучения и тестирования. Для тестирования модели также были подготовлен скрипт, осуществляющий подготовку проверочного набора данных.

Для создания модели распознавания, наиболее подходящей для задачи классификации, был проанализирован математический персептрон и полносвязные нейронные сети. Путем декомпозиции на уровни абстракции изучены основные принципы работы библиотеки TensorFlow. Проводились беседы с преподавателями различных учебных и научных организаций по теме исследования, актуальных направлениях применения машинного обучения для работы с естественным языком. Экспериментальным путем были определены оптимальные параметры модели для используемого объема учебных и тестируемых данных, дабы повысить скорость и точность классификации. Скорость генерации и загрузки модели в оперативную память измерялась при помощи программных средств.

TensorFlow

TensorFlow [7] — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов. Применяется как для исследований, так и для разработки собственных продуктов Google. Изначально создавалась как проприетарный продукт с закрытым исходным кодом, однако в 2015 году была переведена в раздел открытых по лицензии Apache 2.0.

У TensorFlow есть две основных версии: 1.0 и 2.0. Вторая итерация привносит значительное расширение возможностей написания нейросетей благодаря наличию библиотеки глубокого машинного обучения Keras. Keras реализует поддержку полносвязных (dense), сверточных (CNN) и рекуррентных (RNN, LSTM) слоев, а также различных функций активации. Также TensorFlow 2.0 упрощает разработку, повышая уровень абстракции и выполняя ряд вычислений на лету.

Ключевые преимущества TensorFlow перед аналогичными библиотеками (PyTorch, Keras, DL4J):

- Широкая поддержка глубокого обучения;
- Большой выбор паттернов слоев моделей и самих моделей;
- Простая масштабируемость благодаря OPS API, составляющей ядро библиотеки;
- Автоматическое вычисление градиентов (для поиска локальных минимумов и максимумов);
- Встроенные инструменты графического представления TensorBoard (по умолчанию используется графовое представление);
- Поддержка различных аппаратных ускорителей вычислений;
- Встроенные инструменты профилирования приложений.

TensorFlow.js

Сообщество разработчиков приложении на языке JavaScript — самое многочисленное в мире. Согласно статистике использования языков программирования сервиса GitHub [8], JavaScript занимает первое место с долей около 19%. Популярность JavaScript превышает популярность Python — самого богатого и удобного языка для разработки систем с поддержкой средств искусственного интеллекта и машинного обучения. Поэтому приход в JavaScript машинного обучения и появление инструментов удобного создания моделей был лишь вопросом времени.

Поэтому TensorFlow.js — реализация библиотеки TensorFlow для языка JavaScript — быстро завоевал свою долю в области инструментов разработки интеллектуального ПО. Она создавалась как понятный новичкам инструмент с реактивным вектором развития, учитывающий ключевые аспекты языка JavaScript и его применение в программных продуктах:

- Особенности JavaScript окружения;
- Понятность;
- Производительность;
- Безопасность.

Рассмотрим их подробнее:

JavaScript окружение. Одна из особенностей JavaScript — возможность выполнения в различных средах: на клиенте средствами веб-браузера, на сервере с использование интерпретатора JavaScript V8 в составе пакета Node.js, а также в окружении операционных систем в виде приложений, использующих фреймворк Electron, также включающий в себя Node.js.

Понятность. TensorFlow.js создавался как понятный новичкам инструмент с реактивным вектором развития. Layers API реализует высокоуровневый доступ ко всем возможностям библиотеки, что является преимуществом для новичков. Для более опытных разработчиков приложений с функциями искусственного интеллекта схожесть Layers API с Keras API — программный интерфейс глубокого обучения на Python. В основе Layers — концепция модели и слоев, единая для большинства продуктов с машинным обучением.

Производительность. Основное ограничение JavaScript и его окружения — скорость интерпретации кода. Данная проблема была решена путем связывания

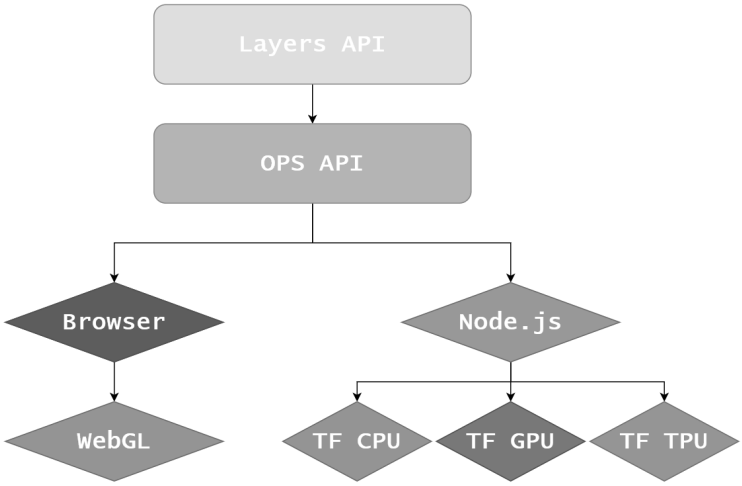

Рис. 2. Архитектура TensorFlow.js

объектов языка JavaScript внутри библиотеки с внешними файлами на C, чья производительность значительно выше. Также благодаря этому осуществляется транзитный доступ к аппаратному обеспечению компьютера. Данная тема будет расширена в разделе «Архитектура TensorFlow.js».

Безопасность. При использовании нейро-модели в браузере теряется необходимость в передаче пользовательских данных на сервер, благодаря чему они полностью сохранны и анонимны.

Архитектура TensorFlow.js

Как и большинство актуальных программных продуктов, TensorFlow.js обладает иерархичной структурой (Рис. 2), определенной уровнем абстракции того или иного процесса.

Рассмотрим снизу вверх уровни подробнее:

- **WebGL** — API для обработки трёхмерной графики в браузере. Разрабатывается консорциумом Khronos Group. Благодаря тому, что элементы WebGL являются частью DOM, API поддерживается всеми современными браузерами. За счёт использования низкоуровневых средств поддержки OpenGL [9], часть кода на WebGL может выполняться непосредственно на видеокартах, значительно повышая скорость вычислений;
- **Browser** в данной модели — это набор инструментов для доступа к памяти устройства, датчикам и системам безопасности и аутентификации пользователя. Браузер содержит в себе API для доступа к вышеупомянутому WebGL. В рамках TensorFlow.js, используемого на клиентской стороне — это необходимый и незаменимый компонент.

Серверная часть состоит из следующих элементов:

- **Node.js** — программная платформа, представляющая собой интерпретатор языка JavaScript, построенная на V8 из браузера Google Chrome и расширенная рядом библиотек, необходимых для серверной разработки (доступ к файловой системе, веб-соединению и т.п.). В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом. Так как JavaScript — интерпретируемый, а не компилируемый язык, то возникает вопрос о скорости выполнения операций и общей производительности, которая для интерпретируемых языков ниже, чем для компилируемых. Именно поэтому существуют 3 модуля, о которых речь пойдет ниже. Все они написаны на C++ и связаны с библиотекой TensorFlow.js, что позволяет вызывать их из JavaScript и выполнять со скоростью скомпилированного C++;
- **TF CPU** (TensorFlow CPU) — корневая библиотека, содержащая математическую и сущностную логику работы с моделями, ее слоями и внешними модулями. Содержит в себе файл TensorFlow C, исполняемый на ЦПУ с использованием аппаратного ускорения;
- **TF GPU** — как и TF CPU, содержит в себе файл TensorFlow C, однако исполнение тензорных операций осуществляется не на ЦПУ, а на графическом процессоре с поддержкой CUDA, то есть на видеоускорителях производства NVIDIA. CUDA позволяет реализовывать на специальных упрощённых диалектах языков программирова-

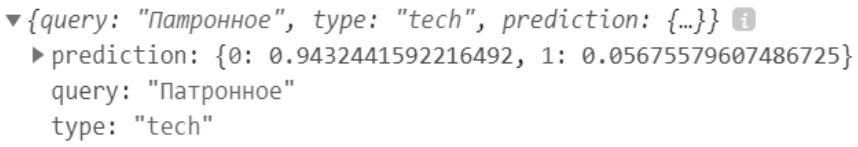

Рис. 3. Поисковый запрос для типа «tech».

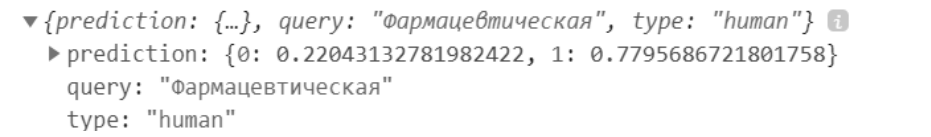

Рис. 4. Поисковый запрос для типа «human».

ния Си, C++ и Фортран алгоритмы, выполнимые на графических и тензорных процессорах NVIDIA;
- **TF TPU** — исполнение тензорных операций осуществляется на TPU (Tensor Processing Unit). TPU — специальный тензорный процессор, относящийся к классу нейронных процессоров, являющийся специализированной интегральной схемой. По сравнению с графическими процессорами, рассчитан на более высокий объём вычислений с пониженной точностью (например, всего 8-разрядную точность) при более высокой производительности на ватт.

Общие для клиента и сервера уровни:
- **OPS API** — высокоуровневый API C++ для параллельного выполнения математических операций в приложениях [10];
- **Layers API** — API для разработчиков конечных приложений с использованием TensorFlow.js. Позволяет создавать и взаимодействовать с существующими моделями высокой сложности. Поддерживаются различные виды слоев. Имеется совместимость с моделями TensorFlow, написанными на Python-версии библиотеки.

Результаты

Основная задача модели: классификация поступающего на вход текстового сообщения по типу на основе нейронных связей, полученных ан этапе обучения. В модели использовались следующие параметры:
- **Тип сети**: CNN или же сверточная нейронная сеть — эффективна в задачах классификации;
- **Тип слоя**: conv2d или же сверточный слой;
- **Функция активации**: softmax. Это обобщённая логистической функции для мульти классовой (не бинарной) выборки;
- **Максимальное отклонение**: 0.1. Значения превышаю данное негативно сказывались на точности прогнозирования;
- **Количество эпох обучения**: 10.

Обучение модели осуществлялось по заранее подготовленному набору данных (обучение с учителем), состоящему из 1000 элементов: 500 предложений на русском языке со значением вывода «tech» (производства технических изделий) и 500 со значением «human» (организации, работающие с кадрами, персоналом и в сферах легкой промышленности.

Так как нейросети наиболее эффективны при работе с числами, естественную речь в виде текста нужно перекодировать в понятный алгоритму формат. Для кодирования использовалась библиотека, входящая в пакет TensorFlow — Universal Sentence Encoder [11]. Данный инструмент кодирует массив текста в массив чисел, которые легко обрабатываются тензорами.

На вход модель получает строку естественного языка. Она передается в модель, где кодируется в числовой вид и классифицируется на основе сформированных при обучении связей. Результатом классификации является объект с тремя полями:
1. prediction — объект с коэффициентами совпадения: ключ 0 — вариант «tech», а ключ 1 — «human». Сами коэффициенты лежат в диапазоне от 0.0 до 1.0;
2. query — исходная строка;
3. type — результат распознавания в текстовом виде.

Ниже приведены результаты тестирования корректности работы модели для двух тестовых строк с ожидаемо различными ответами: тип «tech» для рисунка 3 и тип «human» для рисунка 4. Результаты работы модели верны и попадают в требуемую точность 0.7.

Так как разработка модуля предиктивного поиска велась на двух компьютерах, один из которых оснащен

Таблица 1. Результаты тестов производительности

	CPU, мс	GPU, мс	Прирост
Генерация	3135	2757	13%
Загрузка	2066	978	211%

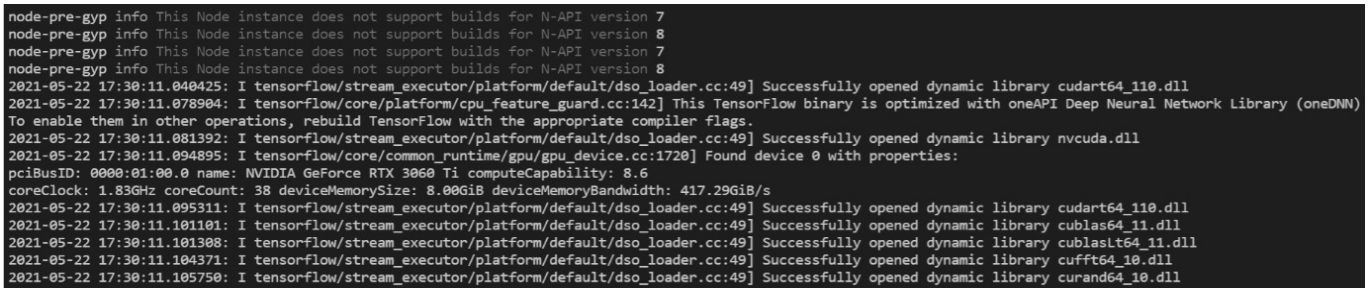

Рис. 5. Лог запуска модели на GPU с использованием CUDA

видеоускорителем NVIDIA, а другой нет, то при тестировании использовались две корневых библиотеки:

1. tensorflow/tfjs-node — TensorFlow.js для вычислений на CPU;
2. tensorflow/tfjs-node-gpu — TensorFlow.js для вычислений на NVIDIA GPU.

При помощи параметра «platform» системного процесса Node.js «process» было реализовано переключение между ними в зависимости от операционной системы, будь то macOS или Windows.

Также было проведено тестирование производительности нейросети при исполнении только на центральном процессоре, а также с использованием CUDA-ядер видеоускорителя NVIDIA. Конфигурация тестового стенда:

- ОС: Windows 10 версии 20H2;
- Процессор: Intel Core i7 10700K 4.8 ГГц 8-ядерный, 16-поточный;
- ОЗУ: 32 ГБ DDR4 3600 МГц;
- Видеоадаптер: NVIDIA RTX 3060Ti 8 Гб.

При запуске модели на видеоускорителе NVIDIA используются тензорные ядра [12] — специальные аппаратные блоки, ускоряющие обработку операций машинного обучения. Судить об использовании тензорных ядер можно по рейтингу вычислительных возможностей (compute capability) GPU. Модели с индексом 7.0 и выше построены на архитектурах NVIDIA Turing и NVIDIA Ampere, содержащих тензорные блоки и RTX блоки. Использованный в работе видеоускоритель основывается на NVIDIA Ampere и обладает индексом 8.6, а значит использует тензорные ядра (Рис. 5):

Результаты тестирования производительности приведены в Таблице 1:

Из результатов тестов производительности очевидно, что использование вычислительных блоков NVIDIA CUDA значительно ускоряет выполнение операций машинного обучения. В больших моделях это дает еще большее преимущество благодаря эффективному распараллеливанию операций. NVIDIA является одним из мировых лидеров в области разработки аппаратного обеспечения для машинного обучения.

Заключение

В ходе настоящей работы описана актуальность использования машинного обучения в программных комплексах, приведена основная информация о фреймворке машинного обучения TensorFlow и его реализации для языка программирования JavaScript. Рассмотрена архитектура пакета TensorFlow.js, его преимущества и ограничения.

Для решаемой задачи классификации текста описан выбор программной платформы, аппаратная конфигурация оборудования, а также тип и параметры модели обработки данных.

Результатом работы является серверное приложение на платформе Node.js, реализующее классифика-

цию текста с использованием библиотеки машинного обучения TensorFlow. Созданная CNN-нейросеть была обучена при помощи подготовленного «датасета», составленного на основе реальных данных комплексной оценки ОПК. В ходе тестирования были получены корректные результаты работы модели классификации запросов, проведено сравнение производительности при генерации и загрузке модели средствами CPU и GPU, сделаны выводы об оптимальных аппаратных составляющих систем, использующих нейросети в своей работе: применение графических и тензорных ускорителей значительно повышает скорость обучения и загрузки готовой модели.

Благодарности

Автор выражает признательность старшему преподавателю ФГУП «ВНИИ «Центр» Дрягиной Анастасии Дмитриевне за помощь при написании данной статьи. Автор выражает благодарность к.т.н., доценту РТУ МИРЭА Сорокину Алексею Борисовичу за теоретическую подготовку в области искусственного интеллекта и машинного обучения.

ЛИТЕРАТУРА

1. Scikit-learn — Machine Learning in Python. Support Vector Machines [Электронный ресурс]: URL: https://scikit-learn.org/stable/modules/svm.html (дата обращения: 18.06.2021)
2. Scikit-learn — Machine Learning in Python. Nearest Neighbors [Электронный ресурс]: URL: https://scikit-learn.org/stable/modules/neighbors.html (дата обращения: 18.06.2021)
3. Wikipedia — The Free Encyclopedia. Naive Bayes classifier [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier (дата обращения: 18.06.2021)
4. Wang, Y., Sohn, S., Liu, S. et al. A clinical text classification paradigm using weak supervision and deep representation. BMC Med Inform Decis Mak [Электронный ресурс]: URL: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911–018–0723–6 (дата обращения: 19.06.2021)
5. Ikonomakis E., Kotsiantis S., Tampakas V. Text Classification Using Machine Learning Techniques. WSEAS Transactions on Computers [Электронный ресурс]: URL: https://www.researchgate.net/publication/228084521_Text_Classification_Using_Machine_Learning_Techniques (дата обращения: 19.06.2021)
6. GeeksforGeeks — A computer science portal for geeks. Choose optimal number of epochs to train a neural network in Keras. [Электронный ресурс]: URL: https://www.geeksforgeeks.org/choose-optimal-number-of-epochs-to-train-a-neural-network-in-keras/ (дата обращения: 27.06.2021)
7. TensorFlow.js — Официальная документация (API). [Электронный ресурс]: URL: https://js.tensorflow.org/api/latest/ (дата обращения: 01.05.2021)
8. GitHub Languages Stats. [Электронный ресурс]: URL: https://madnight.github.io/githut/#/pull_requests/2021/1 (дата обращения: 01.05.2021)
9. OpenGL ES for the Web — Kronos Group. [Электронный ресурс]: URL: https://www.khronos.org/webgl/ (дата обращения: 08.05.2021)
10. OPS C++ User's Manual. [Электронный ресурс]: URL: https://op-dsl.github.io/docs/OPS/user.pdf (дата обращения: 14.05.2021)
11. TensorFlow — Universal Sentence Encoder. [Электронный ресурс]: URL: https://tfhub.dev/google/universal-sentence-encoder/1 (дата обращения: 15.05.2021)
12. TensorFlow — Mixed precision supported hardware. [Электронный ресурс]: URL: https://www.tensorflow.org/guide/mixed_precision?hl=en#supported_hardware (дата обращения: 22.05.2021)
13. Сорокин А.Б. Интеллектуальные САиСИС. — лекции, М.: РТУ МИРЭА, 2019/2020 уч. год.

Журнал «Современная наука: актуальные проблемы теории и практики»