Повышение производительности ИНС на основе применения СУБД PostgreSQL

Ф. А. Мухин[1], О. В. Прокофьев[2], Н. В. Воинов[3], С. А. Молодяков[4]

Санкт-Петербургский политехнический университет Петра Великого

[1]muhin.fa@edu.spbstu.ru, [2]prokofiev_ov@spbstu.ru, [3]voinov@ics2.ecd.spbstu.ru, [4]molodyakov_sa@spbstu.ru

Аннотация. **Работа посвящена исследованию методов взаимодействия искусственных нейронных сетей (ИНС) и баз данных с целью ускорения этапов создания, сохранения и обучения ИНС. На основе расширения PL/Python, фреймворка TensorFlow и СУБД PostrgreSQL разработан программный функционал, позволяющий повысить производительность ИНС при ее применении в различных задачах машинного обучения. Тестирование полученных результатов работы на примере ИНС для выявления болезней пшеницы продемонстрировало их эффективность по сравнению со стандартным подходом с применением языка Python без использования базы данных.**

Ключевые слова: искусственная нейронная сеть (ИНС); база данных; PostgreSQL; PL/Python; TensorFlow

I. Введение

В настоящее время обучение искусственных нейронных сетей (ИНС) достигло многообещающих результатов в широком спектре прикладных областей искусственного интеллекта, начиная от компьютерного зрения (классификация изображений и обнаружение объектов), обработки естественного языка (языковое моделирование и машинный перевод) до информационного поиска (система рекомендаций) и многих других [1–4]. Большие наборы данных для обучения неизменно дают более высокую производительность и точность, так как предотвращают переобучение модели и повышают способность сети обобщать эти данные под новые ситуации [5, 6]. Однако при значительных размерах применяемых моделей весьма актуальна проблема длительного времени их обучения. Это обуславливается необходимостью перебирать различные конфигурации и тестировать модели нейронных сетей зачастую при отсутствии быстрых каналов передачи большого потока информации [7, 8].

Одним из способов решения этой проблемы является применение системы управления базами данных (СУБД) на различных этапах работы ИНС. В работе описан подход, основанный на СУБД PostgreSQL с использованием расширения PL/Python и фреймворка TensorFlow и позволяющий создавать, обучать, сохранять и тестировать нейронные сети с возможностью настройки необходимых параметров на каждом этапе.

Цель работы состоит в повышении эффективности работы с ИНС с точки зрения ускорения создания и сохранения моделей и весов в базу данных, а также уменьшении времени доступа к обучающим и тестовым данным. Достижение поставленной цели позволит ускорить разработку и тренировку ИНС и адаптировать ее работу к большому объему данных и моделей.

II. Предлагаемый подход

Для решения проблемы большого количества конфигураций при исследовании архитектур ИНС, организации и автоматизации процесса их разработки, а также хранения наборов данных не в файловой системе, а в базе данных, предлагается создание набора функций и методов для эффективного и быстрого взаимодействия ИНС и СУБД.

При работе с ИНС без использования базы данных процесс создания, сохранения и обучения сети занимает значительное время, а также требует перезапуска настройки весов после каждого изменения в архитектуре модели. Использование базы данных для сохранения весов, моделей и данных обучения позволяет уменьшить время развертывания датасета для обучения, в любое время сохранить и загрузить текущую или другую конфигурацию, быстро запустить уже обученную ИНС.

Структура базы данных, на которой основан предлагаемый подход, представлена на рис. 1. В базу данных предлагается включить таблицы «train_table», «val_table» и «test_table» для обучающих, проверочных и тестовых данных соответственно. Каждая строка таблицы состоит из первичного ключа для определения отношения каждого отдельного изображения, идентификатора набора данных для общей связи всех трех таблиц, а также входных данных X и выходных размеченных данных Y (меток, результатов), на основе которых нейронная сеть будет обучаться.

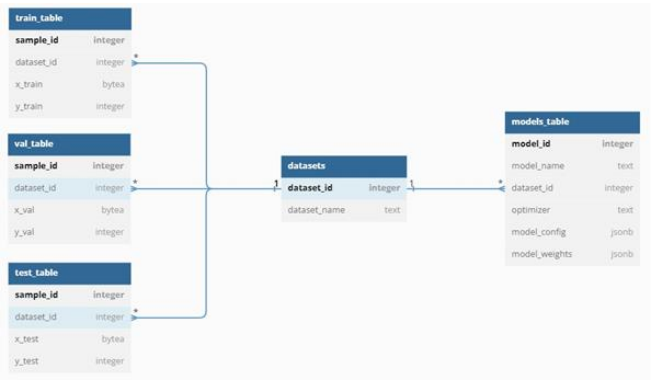

Рис. 1. Схема базы данных

Одна из основных целей предлагаемого подхода – ускорение процесса разработки ИНС. Для этого входные

данные сохраняются не просто в числовом массиве, как они изначально представлены, а конвертируются в двоичный «байтовый» формат – «bytea». Такое преобразование позволяет без потерь хранить большие данные, при этом ускоряя их загрузку и выгрузку более компактным и эффективным способом. Postgres позволяет работать с объектами и форматами такого двоичного формата намного быстрее при меньшей памяти.

Все три таблицы данных связаны отношением «many-to-one» с общей для них таблицей «datasets», которая имеет первичный ключ в виде идентификатора набора данных, а также его названия. Каждая таблица данных имеет опцию каскадного удаления, что позволяет в случае удаления набора данных из таблицы «datasets» каскадно удалить все обучающие и проверочные данные из всех трех таблиц.

Для сохранения моделей после обучения, а также дальнейшего использования, автоматизации и организации всех конфигураций, присутствует таблица «models_table», также связанная с «datasets» отношением «many-to-one». В этом случае каскадное удаление запрещено, так как в случае удаления набора данных необходимо убедиться, что никакая другая уже обученная модель не присутствует в базе данных. В таком случае каскадное удаление будет запрещено до тех пор, пока все модели, использующие этот датасет, не будут убраны. Такой механизм защищает систему от случайного удаления набора данных, модель нейронной сети которого все еще используется и не сможет быть протестирована с его отсутствием.

Помимо этого, в «models_table» присутствует идентификатор модели, её название, используемый оптимизатор весов, конфигурация архитектуры модели по слоям, а также веса «model_weights» в формате JSON («jsonb» в Postgres). PostgreSQL – объектно-реляционная СУБД, что позволяет ей создавать, хранить и извлекать сложные структуры данных, в частности, используемый JSON. Postgres эффективно и быстро работает с таким типом объектов и структур, что уменьшает время сохранения, извлечения, развертывания и использования сохраненных весов моделей нейронных сетей.

III. Реализация

Программная реализация предлагаемого подхода заключается в разработке набора функций для взаимодействия ИНС и СУБД, среди которых можно выделить функции загрузки и редактирования датасета, создания цифровых образов, внесения шума, обучения ИНС, сохранения JSON-архитектуры модели и весов по названию, что позволяет быстро развернуть любую из обученных моделей, а также иметь целые группы и массивы моделей ИНС с возможностью их сравнения, просмотра настроек и процесса обучения.

Соответствующее программное решение было реализовано на специальном расширении языка Python для СУБД PostgreSQL – PL/Python версии Python 3. В качестве СУБД использовалась Postgres Pro Standard 14. Для создания и обучения нейронных сетей применялись фреймворки Tensorflow и Keras. Для Postgres и реализации PL/Python-функций была выбрана IDE DataGrip. Разработка скриптов расширения велась на языке Python версии 3.9. Для сохранения версий и конфигураций разрабатываемого кода использовалась система контроля версий Git на платформе GitHub [9].

Ниже более подробно рассмотрены основные реализованные функции.

A. Функция загрузки и сохранения наборов данных

Функция используется для загрузки набора данных из директории файловой системы в таблицы «train», «validation» и «test». Набору данных присваивается имя после прохождения всех необходимых проверок, например, что датасета с таким именем нет в базе данных.

B. Функция создания цифровых образов

Функция используется для создания цифровых образов на основе уже имеющихся в базе данных исходных изображений по матрице смежности уровней яркости (GLCM) [10], считая для каждого образа 4 текстурных параметра Харалика [11], разделенных по 6 цветовым компонентам. Таким образом, получаются цифровые представления размером 4 на 6 пикселей, где значения характеристик соответствуют определенному черно-белому значению. Функция принимает название набора данных, для которого необходимо вычислить цифровые образы. После создания все образы помещаются в таблицы «train», «validation» и «test».

C. Функция внесения шума в цифровые образы

Как бы компактно и точно не создавались образы для нейронной сети, для повышения точности всегда полезно расширить набор данных для предотвращения переобучения. Внесение шума в цифровые образы реализуется с помощью среднеквадратического отклонения или аугментации данных, например, путем сдвига или поворота исходного изображения. Подобные адаптивные методы создания новых «похожих» данных позволяют нейронной сети улавливать предельные отклонения между классами, что повышает её точность и степень обобщения данных.

Данная функция выполняет генерацию шума с помощью среднеквадратического отклонения, которое можно указать в параметрах функции как «standard_deviation» в формате числа с плавающей точкой удвоенной точности. Настроенный шум смещает каждый пиксель цифрового образа на указанную величину отклонения, создавая тем самым новый образ, похожий на оригинал.

D. Функция создания и обучения нейронной сети

Функция используется для создания модели нейронной сети, настройки параметров слоев и процесса обучения. Входными параметрами являются название набора данных, на основе которого производится обучение новой сети, и логические параметры, указывающие на наличие валидационных и зашумленных данных. Также присутствует определение наименования модели и путь к конфигурационному файлу для настройки архитектуры.

Важным аспектом является возможность настройки структуры нейронной сети с помощью конфигурационного файла, который помимо прочего содержит также и настройки процесса обучения, такие как количество классов в обучающей выборке, размер пакета данных для каждой эпохи, количество эпох, оптимизатор и т. д. Для конфигурирования был выбран YAML-формат, как более компактный по сравнению с TOML и более гибкий по сравнению с JSON и INI форматами. Таким образом, настройка процесса обучения становится проще, эффективнее и превращает описание архитектуры нейронной сети в динамический элемент системы, которая позволяет конструировать всевозможные модели без необходимости вносить изменения в основной код.

После конструирования модели нейронной сети по конфигурации производится её обучение на обучающих данных с валидацией на каждом шаге. При этом результаты метрик точности и потерь выводятся в консоль и в виде графика.

Далее веса и архитектура модели ИНС преобразовываются в jsonb-формат и сохраняются в таблицу со всеми параметрами.

E. Функция выгрузки и тестирования модели ИНС

Функция используется для выгрузки и инициализации модели «model_name» из таблицы, а также проведения тестирования на тестовых данных, ранее не использовавшихся в процессе обучения. Такое разделение всего набора данных на train, validation и test позволяет предотвратить переобучение нейронной сети, когда модель начинает строго и точно обобщать обучающие данные и непредсказуемо вести себя на валидационных и тестовых. Функция призвана проверить, что сеть правильно предсказывает данные, которые она до этого «не видела» в процессе обучения. Дополнительной характеристикой оценки качества искусственной нейронной сети помимо точности и потерь, может служить матрица ошибок, составляемая по каждому классу. Она отражает отношения правильно и неправильно предсказанных классов. Полученная матрица может служить основой для построения различных метрик качества каждого класса: точности класса, полноты класса и F-меры.

IV. Результаты

Для тестирования разработанного программного решения, реализующего предложенный подход, была использована ИНС для распознавания болезней пшеницы с возможностью выбора и применения различных методов расширения датасета и уменьшения переобучения.

Одна из особенностей данной работы – тестирование и адаптация разработанного подхода для обучения и хранения нейронной сети прямого распространения на основе текстурных параметров Харалика. Преимуществом такого подхода является использование компактных данных в виде цифровых образов, что позволяет осуществлять самостоятельную свертку и организовывать более простую нейронную сеть. В итоге передача и обработка данных могут быть оптимизированы и ускорены, что приводит к более эффективным и предсказуемым результатам. Именно поэтому для дополнительной свертки изображений с помощью текстурных параметров Харалика были реализованы отдельные функции выбора данных из базы данных, преобразования и сохранения цифровых образов в отдельную таблицу для датасетов, а в дальнейшем для обучения ИНС. С этими цифровыми образами можно комбинировать другие методы расширения датасета и уменьшения переобучения, например, среднеквадратическое отклонение, внесение шума, регуляризация, дообучение.

Так как набор данных для обучения небольшого размера, а его физическое расширение новыми изображениями болезней растений затруднено, то для классификации такого датасета была выбрана нейронная сеть прямого распространения, которая не требует большого количества данных и может быть эффективно и точно обучена на цифровых образах.

Архитектура нейронной сети прямого распространения для распознавания болезней пшеницы (рис. 2) состоит из двух собственных сверточных слоев (Conv2D) по цифровым образам из 32 и 64 фильтров в каждом, соединенных между собой выделением признаков по максимальному параметру (MaxPooling2D). Первый слой проходит по входному образу размером 4 на 6 пикселей ядром с размером окна 2 на 2, а второй уточняет его, дополнительно проходя тем же ядром, но с уже большим количеством фильтров. Далее следует слой развертки данных в одномерный массив (Flatten), который переходит в несколько последовательных полносвязных слоев нейронов (Dense) с применением прореживания (Dropout) 25% связей. Завершает архитектуру Dense-слой из 8 нейронов (по числу классов) с функцией активации «softmax», которая преобразует выходной вектор значений в распределение вероятностей, необходимое для интерпретации распознавания конкретного класса.

Для анализа эффективности реализованного подхода предлагается сравнение используемой технологии СУБД PostgreSQL с расширением PL/Python со стандартной разработкой на Python без базы данных в различных задачах машинного обучения.

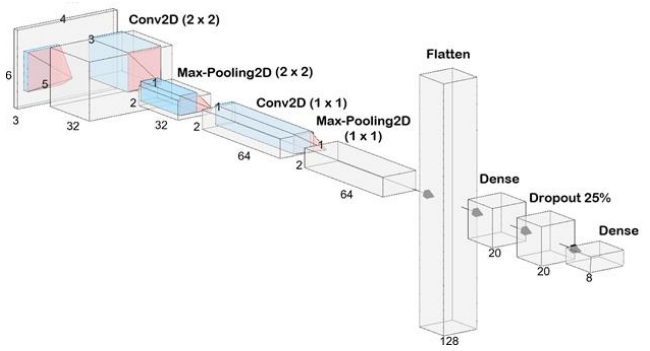

Рис. 2. Архитектура нейронной сети распознавания болезней пшеницы

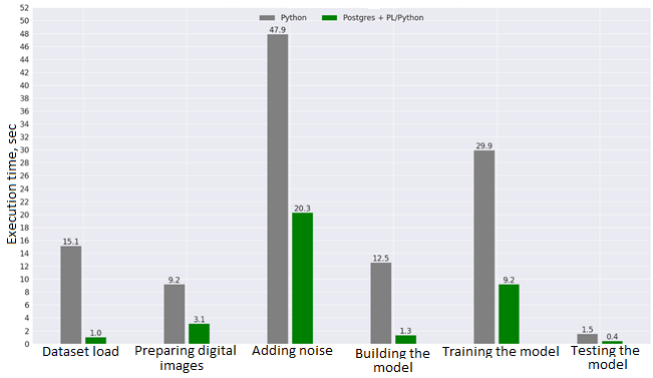

Рис. 3. Сравнение результатов применения предложенного подхода со стандартным подходом на языке Python

Все алгоритмы были использованы одинаково, измерения проводились несколько раз для получения среднего значения. Замеры в Python производились с помощью библиотек «time» и «timeit», а для PL/Python использовалась встроенная функция «EXPLANE ANALYSE» вместе с получаемым временем выполнения операций.

На основании полученных результатов можно сделать вывод, что применение реализованного подхода позволило ускорить разработку и обучение ИНС (правые столбцы на рис. 3) по сравнению со стандартной разработкой на Python (левые столбцы на рис. 3), превосходя ее в разы, а иногда и в десятки раз. Одним из заметных факторов, способствующих такой эффективности, является ускоренная загрузка наборов данных, которая происходит в 15 раз быстрее. Это достигается благодаря хранению всех обучающих данных непосредственно в таблицах базы данных, использованию специализированных типов «bytea» и методов преобразования для ускорения процессов извлечения. Благодаря оптимизации запросов для больших наборов данных Postgres способствует быстрой загрузке и выборке, выполняя эти задачи за считанные секунды. Напротив, Python занимает много процессорного времени для обработки массивов числовых данных в файловой системе. Создание цифровых образов и внесение шума также оказывается быстрее в Postgres за счет эффективного манипулирования большими объемами данных в таблицах. Для всех остальных задач использование расширения PL/Python в базе данных дает значительный прирост скорости. Следует отметить, что стандартная разработка нейронных сетей в Python не имеет возможности предложить автоматизированное и надежное параллельное хранение, использование нескольких сетей и их конфигураций, а также всех связанных с ними данных. В этом отношении предложенный подход предоставляет разработчикам существенное преимущество.

V. Заключение

В работе описан подход, реализующий взаимодействие ИНС с базой данных для ускорения основных этапов работы ИНС. Разработанный подход основан на СУБД PostgreSQL с применением расширения PL/Python и фреймворка TensorFlow и позволяет создавать, обучать, сохранять и тестировать нейронные сети, настраивая на каждом этапе необходимые параметры. Результаты работы были протестированы на примере ИНС для выявления болезней пшеницы и продемонстрировали свою эффективность с точки зрения сокращения времени работы ИНС по сравнению со стандартным подходом с применением языка Python без использования базы данных.

Список литературы

[1] Aznan A., Gonzalez Viejo C., Pang A., Fuentes S. Computer vision and machine learning analysis of commercial rice grains: A potential digital approach for consumer perception studies // Sensors. 2021, vol. 21, no. 19, p. 6354. DOI: https://doi.org/10.3390/s21196354

[2] Shankar V., Parsana S. An overview and empirical comparison of natural language processing (NLP) models and an introduction to and empirical application of autoencoder models in marketing // Journal of the Academy of Marketing Science. 2022, vol. 50, no. 6, pp. 1324-1350. DOI: https://doi.org/10.1007/s11747-022-00840-3

[3] Afoudi Y, Lazaar M, Al Achhab M. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network // Simulation Modelling Practice and Theory. 2021, vol. 113, p. 102375.

[4] Kobyshev K., Voinov N., Nikiforov I. Hybrid image recommendation algorithm combining content and collaborative filtering approaches // Procedia Computer Science. 2021, vol. 193, pp. 200-209. DOI: https://doi.org/10.1016/j.procs.2021.10.020

[5] Santos C.F., Papa J.P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks // ACM Computing Surveys (CSUR), 2022, vol. 54, no. 10s pp. 1-25.

[6] Bejani M.M., Ghatee M. A systematic review on overfitting control in shallow and deep neural networks // Artificial Intelligence Review. 2021, vol. 54, no. 8, pp. 6391-6438. DOI: https://doi.org/10.1007/s10462-021-09975-1

[7] LeCun Y. 1.1 Deep Learning Hardware: Past, Present, and Future // IEEE International Solid-State Circuits Conference - (ISSCC). 2019, pp. 12-19. DOI: https://doi.org/10.1109/ISSCC.2019.8662396

[8] Ghimire D., Kil D., Kim S.H. A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration // Electronics. 2022, vol. 11, no. 6, p. 945. DOI: https://doi.org/10.3390/electronics11060945

[9] Voinov N., Rodriguez Garzon K., Nikiforov I., Drobintsev P. Big Data Processing System for Analysis of GitHub Events // XXII International Conference on Soft Computing and Measurements (SCM). 2019, pp. 187-190. DOI: https://doi.org/10.1109/SCM.2019.8903782

[10] Madhu R.K. Detection and Classification of Tumor using SVM and ANN with GLCM features in CBIR // Journal of Algebraic Statistics. 2022, vol. 13, no. 2, pp. 1790-1804.

[11] Rehman A., Tariq Z., ul din Memon S., Zaib A., Khan M.U., Aziz S. Cucumber leaf disease classification using local tri-directional patterns and haralick features // 2021 International Conference on Artificial Intelligence (ICAI). 2021, pp. 258-263.