УДК 004.048
ББК 32.813
В17

Ванг К., Сзето Д.
В17 Конструирование систем глубокого обучения / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2023. – 462 с.: ил.

ISBN 978-5-93700-181-8

Это всеобъемлющая книга о разработке систем, которые поддерживают наведение мостов в области глубокого обучения, от научных изысканий и прототипирования до производства. Почему наша система глубокого обучения сконструирована именно так, а не иначе? Подойдет ли ее конструкция и для других конкретных случаев использования? Каждый компонент системы обсуждается очень подробно, приводится мотивация и дается представление о плюсах и минусах различных конструкционных вариантов. Для того чтобы помочь инженерам воплотить все идеи на практике, создан образец системы глубокого обучения с полностью работоспособным исходным кодом.

Издание предназначено для инженеров практически всех уровней знаний, заинтересованных в разработке эффективных систем глубокого обучения.

УДК 004.048
ББК 32.813

ISBN 978-1-6334-3986-3 (англ.)
ISBN 978-5-93700-181-8 (рус.)

Оглавление

Содержание

Вводное слово

Считается, что система глубокого обучения является эффективной, если она способна наводить мосты между двумя разными мирами – научными изысканиями / прототипированием и производственными операциями. Разрабатывающие такие системы коллективы должны уметь общаться с практиками из этих двух миров и работать с разными наборами технических требований и ограничений, которые исходят от каждого из них. В силу этого требуется безупречное понимание конструкционных особенностей компонентов в системах глубокого обучения и принципов их работы в тандеме. Этому аспекту инженерии глубокого обучения посвящено очень мало существующей литературы. Данный информационный пробел становится проблемой, когда технология глубокого обучения внедряется среди младших инженеров программного обеспечения и от них ожидается, что они станут эффективными инженерами в данной сфере.

На протяжении многих лет инженерно-конструкторские коллективы заполняли этот пробел, используя свой приобретенный опыт и извлекая то, что им нужно знать, из литературы. Их работа помогала традиционным инженерам программного обеспечения разрабатывать, конструировать и расширять системы глубокого обучения за относительно короткий промежуток времени. Поэтому я с большим волнением узнал, что Кай и Дональд, оба из которых возглавляли коллективы инженеров глубокого обучения, выступили с очень важной инициативой консолидировать эти знания и поделиться ими в форме книги.

Нам давно пора выпустить всеобъемлющую книгу о разработке систем, которые поддерживают наведение мостов в области глубокого обучения, от научных изысканий и прототипирования до производства. Книга «Разработка систем глубокого обучения», наконец, охватывает эту потребность.

Данная книга начинается с высокоуровневого введения, описывающего суть системы глубокого обучения и ее функциональности.

В последующих главах каждый компонент системы обсуждается подробно, приводится мотивация и дается представление о плюсах и минусах различных конструкционных вариантов.

Каждая глава заканчивается анализом, который помогает читателям оценить варианты, наиболее подходящие для их собственных вариантов использования. В заключение авторы, опираясь на все предыдущие главы, подробно рассказывают о сложном пути перехода от научных изысканий и прототипирования к производству. И для того чтобы помочь инженерам воплотить все эти идеи на практике, они создали образец системы глубокого обучения с полностью работоспособным исходным кодом, дабы проиллюстрировать ключевые концепции и предложить попробовать ее тем, кто только начинает работать в этой области.

В целом читатели найдут, что эту книгу легко читать и по ней легко перемещаться, а их понимание способов организации, конструирования и реализации систем глубокого обучения поднимется на совершенно новый уровень. Практики всех уровней знаний, заинтересованные в разработке эффективных систем глубокого обучения, по достоинству оценят эту книгу как бесценный ресурс и справочную информацию. Они прочтут ее один раз, чтобы получить общую картину, а затем будут возвращаться к ней снова и снова при разработке своих собственных систем, конструировании компонентов и принятии важных конструкционных решений, удовлетворяющих все коллективы, которые используют эти системы.

– *Сильвио Саварезе*, вице-президент,
главный научный сотрудник Salesforce

– *Каймин Сюн*, вице-президент Salesforce

Предисловие

Чуть более десяти лет назад нам посчастливилось разработать несколько первых продуктовых функциональностей, ориентированных на конечного пользователя, которые были основаны на искусственном интеллекте. Это было грандиозное предприятие. Сбор и организация данных, которые были пригодны для тренировки моделей, в то время не были обычной практикой. Несколько алгоритмов машинного обучения были упакованы в виде готовых к использованию библиотек. Проведение экспериментов требовало ручного управления и разработки конкретно-прикладных рабочих процессов и визуализаций. Для раздачи каждого типа моделей были созданы конкретно-прикладные серверы. За исключением ресурсоемких технологических компаний, почти каждая новая продуктовая функциональность на базе искусственного интеллекта создавалась с нуля. Это была далеко идущая мечта о том, что интеллектуальные приложения однажды станут товаром.

Поработав с несколькими приложениями ИИ, мы поняли, что всякий раз повторяли один и тот же ритуал, и нам показалось, что имеет смысл разработать системный подход с прототипированием для внедрения продуктовых функциональностей на базе ИИ в производство. Результатом этих усилий стал комплект фреймворкового программного обеспечения с открытым исходным кодом под названием PredictionIO, который соединяет в себе самые передовые программные компоненты для сбора и извлечения данных, тренировки и раздачи моделей. Полностью адаптируемый под конкретно-прикладные задачи с помощью API-интерфейсов и пригодный для развертывания в виде служб всего несколькими командами, он помог сократить время, необходимое на каждом этапе – от проведения экспериментов по обработке данных до тренировки и развертывания готовых к производству моделей. Мы были очень рады узнать, что разработчики по всему миру смогли использовать продукт PredictionIO для создания своих собственных приложений на базе искусственного интеллекта,

что привело к поразительному росту их бизнеса. Позже стартап PredictionIO был приобретен компанией Salesforce для решения аналогичной задачи в еще большем масштабе.

К тому времени, когда мы решили написать эту книгу, индустрия уже процветала благодаря здоровой экосистеме программного обеспечения для искусственного интеллекта. Стало доступно множество алгоритмов и инструментов для решения разных задач. Некоторые поставщики облачных технологий, такие как Amazon, Google и Microsoft, даже предоставляют законченные развернутые системы, которые позволяют коллективам сотрудничать в проведении экспериментов, прототипировании и развертывании производственных приложений в одном централизованном месте. Независимо от вашей цели теперь у вас есть множество вариантов и множество способов их достижения.

Тем не менее, поскольку мы работаем с коллективами над внедрением продуктовых функциональностей глубокого обучения, возникают повторяющиеся вопросы. Почему наша система глубокого обучения сконструирована именно так, а не иначе? Подойдет ли ее конструкция и для других конкретных случаев использования? Мы заметили, что эти вопросы чаще всего задают младшие инженеры программного обеспечения, и мы проинтервьюировали некоторых из них, чтобы выяснить причину. В результате обнаружилось, что их традиционная ученая программа в области разработки программного обеспечения не подготовила их к эффективной работе с системами глубокого обучения. И когда они искали учебные ресурсы, они находили лишь скудную и разрозненную информацию о конкретных системных компонентах, и почти ни в одном ресурсе не обсуждались основы программных компонентов, причины, по которым они были собраны таким образом, и характер их работы вместе, образуя целостную систему.

В целях решения этой проблемы мы начали разрабатывать базу знаний, которая в конечном итоге превратилась в похожий на руководство учебный материал, объясняющий принципы конструирования каждого компонента системы, плюсы и минусы конструкционных решений, а также обоснование как с технической, так и с продуктовой точки зрения. Мы узнали, что наш материал помогал быстро набирать новых сотрудников в коллективы разработчиков и позволял традиционным инженерам программного обеспечения, не имеющим предшествующего опыта в разработке систем глубокого обучения, входить в курс дела. И решили поделиться этим учебным материалом с гораздо большей аудиторией в формате книги. Мы связались с издательством Manning, и остальное стало историей.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не по-

нравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Manning Publications очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

О книге

Данная книга призвана оснастить инженеров знаниями о том, как конструировать, строить или организовывать эффективные системы машинного обучения и адаптировать эти системы под любые потребности и ситуации, с которыми они могут столкнуться. Разрабатываемые ими системы будут облегчать, автоматизировать и ускорять разработку проектов машинного обучения (в частности, проектов глубокого обучения) в различных областях.

В области глубокого обучения именно модели привлекают все внимание. Возможно, это справедливо, если учесть, что на рынок регулярно поступают новые приложения, разработанные на основе этих моделей, – приложения, приводящие потребителей в восторг, такие как камеры слежения, распознающие человека, виртуальные персонажи в интернет-видеоиграх, которые ведут себя как настоящие люди, программа, которая может писать исходный код решения поставленных перед ней произвольных задач, а также передовые системы помощи водителю, которые в один прекрасный день приведут к полностью автономным и самоуправляемым автомобилям. За очень короткий промежуток времени область глубокого обучения наполнилась огромным волнением и многообещающим потенциалом, ожидающим полного воплощения.

Но модель действует не в одиночку. Материализация продукта или службы предусматривает, что модель будет располагаться в системе или на платформе (мы используем эти термины взаимозаменяемо), которая поддерживает модель различными службами и хранилищами. Для этого, среди прочего, нужны, например, API, менеджер наборов данных и склад артифактов[1] и метаданных. Таким образом, за каждым коллективом разработчиков моделей глубокого обучения

[1] Арт*и*факт – это продукт деятельности вообще, в данном случае результат работы системы или модели, тогда как арт*е*факт – это произведение искусства или изделие художественного творчества. – *Прим. перев.*

стоит коллектив разработчиков, в чьи обязанности входит не само глубокое обучение, а создание инфраструктуры, которая содержит модель и все остальные компоненты.

Мы заметили в отрасли проблему, которая заключается в том, что нередко разработчики, которым поручено разрабатывать систему и компоненты глубокого обучения, обладают лишь поверхностными знаниями о глубоком обучении. Они не понимают конкретных требований, предъявляемых к инженерно-конструкторской работе над системой в глубоком обучении, поэтому при разработке системы они склонны следовать обобщенным подходам. Например, они могут принять решение передать всю работу, связанную с разработкой модели глубокого обучения, исследователю данных и сосредоточиться только на автоматизации. И поэтому разрабатываемая ими система будет опираться на традиционную систему планирования списков заданий или бизнес-аналитическую систему анализа данных, которая не оптимизирована ни под характер выполнения заданий глубокого обучения, ни под специфичные для глубокого обучения шаблоны доступа к данным. В результате систему будет сложно использовать для разработки моделей, а скорость доставки моделей останется низкой. По сути, инженеров, которым не хватает глубокого понимания глубокого обучения, просят разрабатывать системы с поддержкой моделей глубокого обучения. Как следствие эти инженерные системы неэффективны и не подходят для систем глубокого обучения.

О разработке моделей глубокого обучения было написано немало с точки зрения исследователя данных, охватывая сбор данных и насыщение наборов данных, написание алгоритмов тренировки и т. п. Но очень немногие книги или даже блоги были посвящены системе и службам, которые поддерживают все эти мероприятия по глубокому обучению.

В данной книге мы обсуждаем строительство и конструирование систем глубокого обучения с точки зрения разработчика программного обеспечения. Наш подход заключается в том, чтобы сначала описать типичную систему глубокого обучения в целом, включая ее главнейшие компоненты и их взаимосвязь; затем в отдельной главе мы углубляемся в каждый главнейший компонент. Мы начинаем каждую главу о компонентах с изложения технических требований. Затем знакомим с принципами конструирования, а также образцами служб / исходного кода и, наконец, оцениваем технологические решения с открытым исходным кодом.

Поскольку мы не можем охватить в книге все существующие системы глубокого обучения (от производителя или с открытым исходным кодом), то сосредоточиваемся на изложении технических требований и принципов конструирования (с примерами). Изучив эти принципы, попробовав примеры служб из книги и прочитав наше обсуждение вариантов с открытым исходным кодом, мы надеемся, что читатели смогут провести собственное исследование и найти то, которое подходит им лучше всего.

Кому следует прочитать эту книгу?

Первостепенная аудитория этой книги – инженеры программного обеспечения (включая недавно окончивших учебу студентов со специализацией в области вычислительных наук), которые хотят быстро перейти к разработке систем глубокого обучения, например те, кто хочет работать на платформах глубокого обучения или интегрировать какую-либо функциональность искусственного интеллекта – например, раздачу моделей – в свои продукты.

Исследователи данных, инженеры-изыскатели, менеджеры и все остальные, кто использует машинное обучение для решения реальных задач, также найдут эту книгу полезной. Разобравшись в опорной инфраструктуре (или системе), они будут оснащены всем необходимым для предоставления точной обратной связи коллективу инженеров в том, что касается повышения эффективности процесса разработки моделей.

Это инженерная книга, и вам не нужны знания в области машинного обучения, но вы должны быть знакомы с базовыми концепциями вычислительных наук и инструментами программирования, такими как микросервисы, gRPC и Docker, чтобы работать с лабораторией исходного кода и понимать технический материал. Независимо от вашего образования, вы все равно сможете извлечь пользу из нетехнических материалов книги, которые помогут вам лучше понять принципы работы машинного обучения и систем глубокого обучения, для того чтобы переносить продукты и службы из области идей в производство.

Прочитав эту книгу, вы сможете понять механизмы работы систем глубокого обучения и способы разработки каждого компонента. Вы также поймете ситуации, когда следует собирать технические требования пользователей, транслировать их в конструкционные варианты системных компонентов и интегрировать компоненты с целью создания целостной системы, которая поможет вашим пользователям быстро разрабатывать и предоставлять функциональные возможности глубокого обучения.

Как эта книга организована: дорожная карта

В этой книге 10 глав и три дополнения (включая одно лабораторное дополнение). В первой главе объясняется, что такое цикл освоения глубокого обучения и как выглядит базовая система глубокого обучения. В следующих главах мы подробно рассмотрим каждый функциональный компонент эталонной системы глубокого обучения. Наконец, в последней главе обсуждается вопрос о том, как модели отправляются в производство. Дополнение к книге содержит лабораторию исходного кода, позволяющую читателям опробовать образец системы глубокого обучения.

В главе 1 описывается, что такое система глубокого обучения, разные участвующие в разработке системы интересанты и как они

взаимодействуют с ней в целях ее оснащения функциональными возможностями глубокого обучения. Это взаимодействие называется циклом освоения глубокого обучения. Вдобавок вы сформируете для себя концепцию системы глубокого обучения, именуемую эталонной системной архитектурой, которая содержит все необходимые элементы и может быть адаптирована в соответствии с вашими требованиями.

Главы со 2 по 9 охватывают каждый стержневой компонент эталонной архитектуры системы глубокого обучения, такой как служба управления наборами данных, служба тренировки моделей, служба автоматической гиперпараметрической оптимизации и служба оркестровки рабочих процессов.

В главе 10 обсуждается вопрос о том, как подготовить конечный продукт в фазе научного изыскания или прототипирования к выпуску в публичное пространство. В дополнении A представлен образец системы глубокого обучения и демонстрируется лабораторное упражнение, в дополнении B приводится обзор существующих технологических решений, а в дополнении C обсуждается система Kubeflow Katib.

Об исходном коде

Мы считаем, что самый лучший способ учиться – это делать, практиковаться и экспериментировать. В целях демонстрации описанных в этой книге принципов конструирования и получения практического опыта мы создали образец системы глубокого обучения и лабораторию исходного кода. Весь исходный код, инструкции по настройке и лабораторные скрипты образца системы глубокого обучения доступны на GitHub (https://github.com/orca3/MiniAutoML). Вы также можете получить исполняемые фрагменты исходного кода из онлайновой версии этой книги в liveBook по адресу https://livebook.manning.com/book/software-engineers-guide-to-deep-learning-system-design и с веб-сайта издательства Manning (www.manning.com).

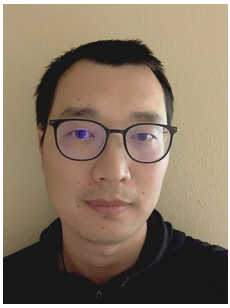
Лаборатория исходного кода «hello world» (в дополнении A) содержит полную, хотя и упрощенную, мини-систему глубокого обучения с наиболее важными компонентами (управлением наборами данных, тренировкой и раздачей моделей). Мы предлагаем вам опробовать указанную там систему после прочтения первой главы книги либо сделать это до того, как вы испытаете наши образцы служб, описанные в этой книге. Данная лаборатория также предоставляет скрипты командной оболочки и ссылки на все ресурсы, необходимые для того, чтобы приступить к работе.

Помимо лаборатории исходного кода, эта книга содержит множество примеров исходного кода в виде отдельных нумерованных листингов и внутри обычного текста. В обоих случаях исходный код отформатирован `шрифтом фиксированной ширины, подобным этому`, чтобы отделять его от обычного текста. Иногда исходный код также выделяется **жирным шрифтом**, чтобы выделять тот исходный код, который

изменился по сравнению с предыдущими шагами в данной главе, например когда новая функциональная возможность добавляется к существующей строке исходного кода.

Во многих случаях изначальный исходный код был переформатирован; мы добавили переносы строк и переработали отступы, чтобы уместиться в доступное пространство страницы книги. В редких случаях даже этого было недостаточно, и листинги включали маркеры продолжения строки (➥). Вдобавок нередко комментарии в исходном коде из листингов удалялись, когда исходный код описывался в тексте. Многие листинги сопровождаются аннотациями к исходному коду, выделяющими важные концепции.

Об авторах



Кай Ванг является главным разработчиком программного обеспечения в Salesforce Einstein group, где он трудится над платформой глубокого обучения, используемой миллионами пользователей Salesforce. Ранее он работал в Microsoft Bing и Azure, создавая крупномасштабные распределенные системы. Кай подал шесть патентов, в основном связанных с системами глубокого обучения, и недавно окончил учебу по программе сертификации выпускников Стэнфорда по искусственному интеллекту.

Дональд Сзето был соучредителем и техническим директором стартапа PredictionIO, целью которого было помочь демократизировать и ускорить внедрение машинного обучения. Стартап PredictionIO был приобретен компанией Salesforce, где он продолжил свою работу над машинным обучением и системами глубокого обучения. Дональд – основатель Aftermint, целью которого является объединение Web2 и Web3. Он также инвестирует в технологические стартапы, предоставляет по ним консультации и наставнические услуги.

1 Введение в системы глубокого обучения

Данная глава охватывает следующие ниже темы:

- определение системы глубокого обучения;
- цикл освоения продукта и поддержка цикла системой глубокого обучения;
- обзор базовой системы глубокого обучения и ее компонентов;
- различия между разработкой системы глубокого обучения и разработкой модели.

В этой главе вы познакомитесь с ментальным образом системы глубокого обучения в целом. Мы проведем обзор нескольких определений, предоставим конструкцию эталонной системной архитектуры и познакомим с образцом полноценной реализации данной конструкции. Мы надеемся, что этот мысленный образ поможет вам понять, как остальные главы, в которых каждый компонент системы рассматривается подробно, вписываются в общую картину.

В начале этой главы мы обсудим еще более широкую картину, выходящую за рамки системы глубокого обучения: то, что называется *циклом освоения глубокого обучения*. Данный цикл определяет различные технические роли и фазы, связанные с выведением на рынок продуктов, основанных на глубоком обучении. Модель и платформа существуют не в вакууме; они влияют на управление продуктом,

маркетинговые исследования, производство и другие этапы. Мы считаем, что инженеры разрабатывают более эффективные системы, когда они понимают этот цикл и знают, чем каждый коллектив занимается и что ему нужно для выполнения своей работы.

В разделе 1.2 мы начнем наше изложение конструкции системы глубокого обучения с образца архитектуры типичной системы, которая может быть адаптирована под вашу собственную систему глубокого обучения. Описанные в этом разделе компоненты будут рассмотрены подробнее в соответствующих главах. Наконец, мы подчеркнем различия между разработкой модели и разработкой системы глубокого обучения. Это различие часто приводит к путанице, поэтому мы хотим сразу же его прояснить.

Прочитав эту вводную главу, вы получите четкое представление о ландшафте глубокого обучения. Вы сможете начать создавать свою собственную конструкцию системы глубокого обучения, а также разбираться в существующих конструкциях и в том, как их использовать и расширять, благодаря чему вам не придется разрабатывать все с нуля. Продолжая читать эту книгу, вы увидите, как все элементы взаимосвязаны между собой и работают вместе как система глубокого обучения.

Терминология

Прежде чем перейти к остальной части главы (и книги в целом), давайте определим и проясним несколько терминов, которые мы используем на протяжении всей книги.

Глубокое обучение в сопоставлении с машинным обучением

Глубокое обучение – это машинное обучение, но оно считается эволюционным развитием машинного обучения. Машинное обучение, по определению, – это применение искусственного интеллекта, включающее алгоритмы, которые выполняют синтаксический разбор данных, усваивают знания из этих данных, а затем применяют полученные знания для принятия обоснованных решений. Глубокое обучение – это особая форма машинного обучения, в которой в качестве алгоритма усвоения знаний из данных и принятия точных решений используется программируемая нейронная сеть.

Хотя эта книга в первую очередь посвящена обучению приемам разработки системы или инфраструктуры, которая способствует разработке систем глубокого обучения (все примеры основаны на нейросетевых алгоритмах), обсуждаемые нами концепции разработки конструкций и проектов применимы и к машинному обучению. Таким образом, в этой книге мы используем термины *глубокое обучение* и *машинное обучение* в некотором роде взаимозаменяемо. Например, представленный в данной главе цикл освоения глубокого обучения и представленная в главе 2 служба управления данными также работают и в контексте машинного обучения.

Варианты использования глубокого обучения

Вариант использования глубокого обучения относится к сценарию, в котором задействуется технология глубокого обучения, – другими словами, задаче, которую вы хотите решить с помощью глубокого обучения. Примеры включают:

- *чат-бот* – пользователь может инициировать текстовый разговор с виртуальным агентом на веб-сайте службы поддержки потребителей. Виртуальный агент использует модель глубокого обучения, чтобы понимать вводимые пользователем предложения и вести беседу с пользователем как реальный человек;
- *самоуправляемый автомобиль* – водитель может переводить автомобиль в режим вспомогательного вождения, который автоматически управляет автомобилем в соответствии с дорожной разметкой. Разметка улавливается несколькими камерами на борту автомобиля, чтобы формировать представление о дороге с использованием технологии компьютерного зрения, основанной на глубоком обучении.

Модель, предсказание и модельный вывод, а также раздача моделей

Эти термины описываются следующим образом:

- *модель* – модель глубокого обучения можно рассматривать как исполняемую программу, которая содержит алгоритм (модельную архитектуру) и данные, необходимые для генерирования предсказания;
- *предсказание и модельный вывод* – и предсказание, и модельный вывод относятся к исполнению модели с заданными данными с целью получения набора выходных данных. Поскольку предсказание и модельный вывод широко используются в контексте раздачи моделей, в этой книге они используются взаимозаменяемо;
- *раздача моделей* (служба предсказания) – в этой книге раздача моделей описывается как размещение моделей машинного обучения в веб-приложении (в облаке либо локально) и предоставление приложениям глубокого обучения возможности интегрировать функциональность модели в свои системы через API. Веб-программа раздачи моделей обычно называется службой предсказания, или службой раздачи моделей.

Приложение глубокого обучения

Приложение глубокого обучения – это часть программного обеспечения, в которой для решения задач используют технологию глубокого обучения. Обычно оно не выполняет никаких задач, требующих больших вычислительных затрат, таких как перемалывание данных, тренировка модели глубокого обучения и раздача моделей (за исключением размещения моделей на периферии, такой как автономное транспортное средство). Примеры включают:

- *приложение чат-бот*, которое предоставляет пользовательский интерфейс или API-интерфейсы для получения от пользователя предложений на естественном языке, интерпретирует их, выполняет действия и предоставляет пользователю осмысленный ответ. Осно-

вываясь на выходных модельных данных, рассчитанных в системе глубокого обучения (из службы раздачи моделей), чат-бот отвечает и предпринимает соответствующие действия;

- *программное обеспечение для самоуправляемых автомобилей*, которое принимает данные от множества датчиков, таких как видеокамеры, датчики приближения и лазерный радар[1], с целью формирования восприятия окружающей обстановки автомобиля с помощью моделей глубокого обучения, и соответствующим образом управляет автомобилем.

Платформа, система и инфраструктура

В этой книге термины *платформа глубокого обучения*, *система глубокого обучения* и *инфраструктура глубокого обучения* имеют одно и то же значение: опорная система, которая обеспечивает всю необходимую поддержку для эффективной разработки приложений глубокого обучения и их масштабирования. Чаще всего мы используем термин «*система*», но в контексте этой книги все три термина имеют одинаковое значение.

Теперь, когда мы все согласны с условиями, давайте начнем!

1.1 Цикл освоения глубокого обучения

Как мы уже говорили, системы глубокого обучения – это инфраструктура, необходимая для эффективной разработки *проекта на базе глубокого обучения*. И поэтому, прежде чем углубиться в структуру системы глубокого обучения, разумно взглянуть на парадигму разработки, которую система глубокого обучения обеспечивает. Эта парадигма называется *циклом освоения глубокого обучения*.

Возможно, вы задаетесь вопросом, почему в технической книге мы хотим сделать акцент на столь нетехнической теме, как освоение продукта. Дело в том, что целью большинства работ по глубокому обучению в конечном счете является выведение продукта или услуги на рынок. Тем не менее многие инженеры незнакомы с другими этапами освоения продукта точно так же, как многие разработчики продуктов не знают об инженерном деле или моделировании. Из нашего опыта строительства систем глубокого обучения мы узнали, что уговоры людей, выполняющих в компании различные технические роли, внедрять систему во многом зависят от того, действительно ли система будет решать их конкретные задачи. Мы считаем, что описание различных этапов и технических ролей в цикле освоения глубокого обучения помогает формулировать, учитывать, доносить до каждого и в конечном итоге решать болевые точки.

---

[1] Англ. Light Detection and Ranging (LiDAR) – обнаружение и определение дальности с помощью света. – *Прим. перев.*

Понимание этого цикла также решает несколько других проблем. За последнее десятилетие было разработано множество новых пакетов программного обеспечения для глубокого обучения, предназначенных для решения разных задач. Некоторые из них посвящены тренировке и раздаче моделей, тогда как другие посвящены отслеживанию результативности моделей и экспериментированию. Всякий раз, когда исследователям и инженерам данных требовалось решать задачу конкретного приложения или варианта использования, они собирали эти инструменты воедино; такая работа называется операционализацией машинного обучения (MLOps[1]). По мере роста числа таких приложений сборка этих инструментов каждый раз с нуля для нового приложения все время повторяется и отнимает много времени. В то же время, по мере того как растет важность этих приложений, растут и ожидания в отношении их качества. Обе эти проблемы требуют выверенного подхода к быстрой и надежной разработке и доставке функциональностей глубокого обучения. Этот выверенный подход начинается с того, что все работают в рамках одной и той же парадигмы или цикла освоения глубокого обучения.

Как система глубокого обучения вписывается в *цикл* освоения глубокого обучения? Хорошо сложенная система глубокого обучения будет поддерживать цикл освоения продукта и делать его выполнение простым, быстрым и надежным. В идеале исследователи данных смогут использовать систему глубокого обучения в качестве инфраструктуры для выполнения всего цикла освоения проекта глубокого обучения без изучения всех технических деталей опорных сложных систем.

Поскольку каждый продукт и организация уникальны, разработчикам систем крайне важно понимать уникальные требования различных технических ролей, необходимые для разработки успешной системы. Под «успешной» системой мы подразумеваем ту, которая помогает заинтересованным сторонам продуктивно сотрудничать в деле быстрого внедрения функциональностей глубокого обучения. На протяжении всей этой книги, по мере того как мы будем знакомиться с принципами конструирования систем глубокого обучения и детально рассматривать работу каждого компонента, ваше понимание требований со стороны интересантов поможет вам адаптировать эти знания, чтобы формировать свои собственные конструкции систем. Когда мы будем обсуждать технические детали, мы будем указывать на ситуации, когда вам необходимо будет обращать внимание на определенные типы заинтересованных сторон во время конструирования системы. Цикл освоения глубокого обучения будет служить ориентиром при рассмотрении требований к конструкции каждого компонента системы глубокого обучения.

Давайте начнем с картинки. На рис. 1.1 показано, как выглядит типичный цикл. Он показывает пофазное освоение машинного

[1] Англ. machine learning operations. – *Прим. перев.*

обучения (в особенности глубокого обучения) в развитии. Как видно по рисунку, межфункциональное сотрудничество происходит практически на каждом шаге. В следующих двух разделах мы обсудим задействованную в этой диаграмме каждую фазу и техническую роль.

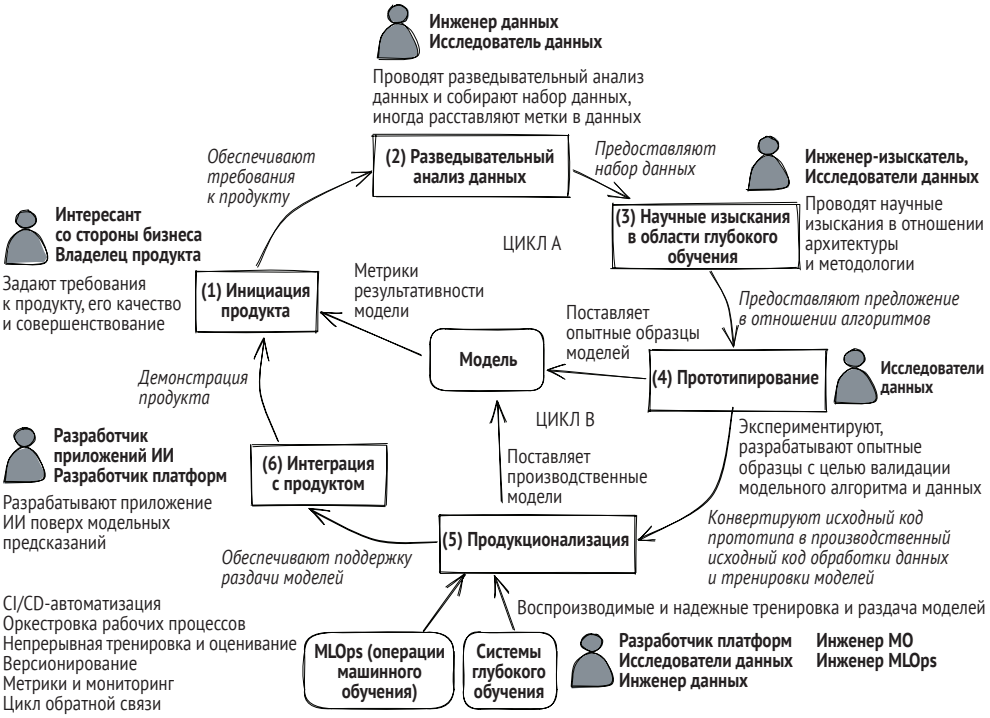


Рис. 1.1 Типичный сценарий перенесения глубокого обучения из области научных изысканий в продукт. Этот цикл называется циклом освоения глубокого обучения

1.1.1 Фазы цикла освоения продукта на базе глубокого обучения

Цикл освоения глубокого обучения обычно начинается с бизнес-возможности и обусловливается продуктовым планом и его управлением. После этого указанный цикл обычно проходит через четыре фазы: разведывательный анализ данных, прототипирование, продукционализация (поставка в производство) и интеграция с приложением. Давайте рассмотрим эти фазы по очереди. Затем мы разберем все задействованные технические роли (обозначенные на рис. 1.1 значком человека).

> **ПРИМЕЧАНИЕ** Число в круглых скобках рядом с каждым последующим подразделом соответствует тому же числу на рис. 1.1.

Инициация продукта (1)

Сначала интересант со стороны бизнеса (владелец продукта или руководитель проекта) анализирует бизнес и определяет потенциальную бизнес-возможность или задачу, которые могут быть решены с помощью машинного обучения.

Разведывательный анализ данных (2)

Когда исследователи данных получают четкое представление о требованиях со стороны бизнеса, они начинают работать с инженерами данных, чтобы собрать как можно больше данных, расставить в них метки и сформировать наборы данных. Сбор данных может включать поиск публично доступных данных и обследование внутренних источников. Также может происходить очистка данных. Расстановка меток в данных может быть передана на аутсорсинг либо выполнена собственными силами.

По сравнению со следующими фазами, эта ранняя фаза разведки данных не структурирована и часто выполняется мимоходом. Это может быть скрипт на Python, или скрипт командной оболочки, или даже ручная копия данных. Для анализа данных исследователи данных нередко используют специально предназначенные для этой цели веб-приложения, такие как Jupyter Notebook (с открытым исходным кодом; https://jupyter.org), Amazon SageMaker Data Wrangler (https://aws.amazon.com/sagemaker/data-wrangler) и Databricks (www.databricks.com). Формального конвейера сбора данных, который необходимо было бы построить, не существует.

Разведывательный анализ данных не только важен, но и имеет критически важное значение для успеха проекта глубокого обучения. Чем больше релевантных данных имеется, тем выше вероятность разработки эффективных и результативных моделей глубокого обучения.

Научные изыскания и прототипирование (3, 4)

Прототипирование призвано находить наиболее осуществимый алгоритм/подход, который удовлетворял бы требованиям бизнеса (владельца продукта) с использованием предоставленных данных. В этой фазе исследователи данных могут сотрудничать с инженерами-изыскателями, работающими в области искусственного интеллекта, чтобы рекомендовать и оценивать различные алгоритмы тренировки с использованием наборов данных, сформированных в предыдущей фазе разведывательного анализа данных. В этой фазе исследователи данных обычно тестируют несколько идей и строят опытный образец[1], которые затем оценивают.

---

[1] Англ. proof-of-concept (POC); подтверждение концепции. – *Прим. перев.*

Хотя нередко рассматриваются недавно опубликованные алгоритмы, большинство из них не будут приняты. Точность алгоритма – не единственный фактор, который следует учитывать; при оценивании алгоритма также необходимо учитывать требования к вычислительным ресурсам, объем данных и стоимость реализации алгоритма. Обычно победителем оказывается наиболее практичный подход.

Обратите внимание, что из-за ограниченности ресурсов инженеры-изыскатели не всегда участвуют в фазе прототипирования. Нередко научно-изыскательскую работу, а также опытные испытания концепций осуществляют исследователи данных.

Вы также, возможно, заметили, что на рис. 1.1 в большом цикле освоения показан внутренний цикл (цикл А: Инициация продукта > Разведывательный анализ данных > Научные изыскания в области глубокого обучения > Прототипирование > Модель > Инициация продукта). Цель этого цикла – получить обратную связь о продукте в ранней фазе путем построения опытного образца модели. Мы можем повторять этот цикл несколько раз до тех пор, пока все интересанты (исследователи данных, владельцы продуктов) не придут к консенсусу относительно алгоритмов и данных, которые будут использоваться для удовлетворения требований бизнеса.

Многочисленные тяжелые уроки в конце концов научили нас тому, что прежде чем приступать к дорогостоящему процессу разработки – формированию производственных данных, разработке конвейеров тренировки и размещению моделей на платформе, – мы должны согласовывать решение с коллективом разработчиков или потребителем (что еще лучше). Цель проекта глубокого обучения ничем не отличается от любого другого проекта по разработке программного обеспечения: решить бизнес-задачу. Согласование подхода с коллективом разработчиков на ранней стадии будет препятствовать дорогостоящему и деморализующему процессу его доработки на более поздних стадиях.

Продукционализация, или MLOps (5)

Продукционализация, также именуемая «поставкой в производство», – это процесс изготовления продукта, пригодного для производства и готового к его употреблению пользователями. Производственная пригодность обычно определяется как способность обслуживать запросы потребителей, выдерживать определенный уровень вызванной запросами нагрузки и корректно справляться с неблагоприятными ситуациями, такими как неправильный ввод данных и вызванная запросами перегрузка. Производственная пригодность также включает в себя постпроизводственные усилия, в частности непрерывный мониторинг модельных метрик и их оценивание, сбор отзывов потребителей и перетренировка моделей.

Продукционализация – это наиболее трудоемкая с инженерной точки зрения часть цикла освоения, поскольку мы будем конверти-

ровать эксперименты по прототипированию в серьезные производственные процессы. Неисчерпывающий список дел по продукционализации может включать:

- разработку конвейера данных для многократного извлечения данных из разных источников данных, поддержания версий и обновления набора данных;
- разработку конвейера данных для предобработки набора данных, например с целью улучшения или обогащения данных и интеграции с внешними инструментами расстановки меток;
- переработку и докеризацию прототипного исходного кода в исходный код тренировки моделей производственного качества;
- обеспечение воспроизводимости результатов работы исходных кодов тренировки и раздачи моделей путем версионирования и отслеживания входных и выходных данных. Например, мы могли бы включить в исходный код тренировки отчетность о метаданных тренировки (дату и время тренировки, продолжительность, гиперпараметры) и о модельных метаданных (используемые метрики результативности, данные и исходный код), чтобы обеспечить полную отслеживаемость каждого сеанса тренировки моделей;
- настройку непрерывной интеграции (Jenkins, GitLab CI) и непрерывного развертывания с целью автоматизации сборки, валидации и развертывания исходного кода;
- разработку непрерывного конвейера тренировки и оценивания моделей, чтобы при тренировке моделей иметь возможность автоматически использовать новейший набор данных и генерировать модели воспроизводимым, проверяемым и надежным образом;
- разработку конвейера развертывания моделей, который автоматически выпускает модели, прошедшие проверку качества, чтобы предоставлять компоненту раздачи моделей возможность к ним обращаться; в зависимости от требований бизнеса модельное предсказание может генерироваться асинхронно либо в реальном времени. Компонент раздачи моделей служит пристанищем для модели и выставляет ее в публичное пространство через веб-API;
- разработку конвейеров непрерывного мониторинга, которые периодически оценивают набор данных, модель и результативность раздачи моделей, чтобы обнаруживать потенциальный дрейф признаков (изменение в распределении данных) в наборе данных или деградацию результативности модели (дрейф концепции) и предупреждать разработчиков либо перетренировать модель.

В наши дни шаг продукционализации получил новый термин с оттенком хайпа: MLOps (операции машинного обучения), который отличается расплывчатостью, а его определение неоднозначно для

инженеров-изыскателей и профессионалов. Мы интерпретируем MLOps как преодоление разрыва между разработкой моделей (экспериментированием) и операциями в производственных средах (Ops) с целью облегчения продукционализации проектов машинного обучения. Примером может служить оптимизация процесса переноса моделей машинного обучения в производство, а затем их мониторинг и техническое сопровождение.

MLOps – это парадигма, основанная на применении принципов, аналогичных принципам DevOps, которые применяются при разработке программного обеспечения вообще. В указанной выше парадигме используются три дисциплины: машинное обучение, инженерия программного обеспечения (в особенности оперирование) и инженерия данных. Рисунок 1.2 позволяет взглянуть на глубокое обучение через призму MLOps.

Рис. 1.2 В инженерии машинного обучения (MLOps) к глубокому обучению применяются подходы DevOps в фазе продукционализации, когда модели подставляются в производство (источник: Инженерия машинного обучения в действии, Бен Уилсон, Мэннинг, 2022, рис. 2.7[1])

Поскольку эта книга посвящена разработке систем машинного обучения, поддерживающих операции машинного обучения, мы не будем вдаваться в подробности о методах, показанных на рис. 1.2. Но, как вы видите, инженерные усилия, которые поддерживают раз-

[1] Machine Learning Engineering in Action, Ben Wilson, Manning, 2022.

работку моделей машинного обучения на производстве, огромны. По сравнению с тем, что исследователи данных использовали в предыдущей фазе разведывательного анализа данных и прототипирования моделей инструментарий (программное обеспечение), инженерные стандарты и процессы кардинально изменились и стали намного сложнее.

Почему поставка моделей в производство сопряжена с трудностями?

Двумя самыми большими препятствиями для поставки моделей в производство являются огромная опорная инфраструктура (инструменты, службы, серверы) и интенсивное взаимодействие между коллективами. В этом посвященном продукционализации (то есть MLOps-операциям) разделе констатируется, что выведение алгоритма/модели из стадии прототипа в производство предусматривает, что исследователи данных будут работать с инженерами данных, разработчиками платформ, инженерами DevOps и инженерами машинного обучения, а также изучать обширную инфраструктуру (систему глубокого обучения). Неудивительно, что продукционализация модели занимает так много времени.

В целях преодоления этих препятствий нам нужно абстрагировать исследователей данных от сложности при конструировании и разработке системы глубокого обучения. Как и при конструировании автомобиля, мы хотим посадить исследователей данных за руль, но не требуем от них каких-то особых знаний о самом автомобиле.

Теперь, возвращаясь к циклу освоения, вы, возможно, заметили, что на рис. 1.1 есть еще один внутренний цикл (цикл B), который проходит от продукционализации (Блок 5) и модели к инициации продукта (Блок 1). Это второе согласование с коллективом разработчиков, которое выполняется перед тем, как интегрировать модельный вывод с приложением искусственного интеллекта.

Наше второе согласование (цикл B) призвано сравнивать модель и данные между фазой прототипирования и производством. Мы хотим обеспечить соответствие производительности и масштабируемости модели (например, пропускной способности модели) техническим требованиям со стороны бизнеса.

> **ПРИМЕЧАНИЕ** Если вы хотите узнать об MLOps больше, то рекомендуем следующие две статьи – они станут отличной отправной точкой: «Операционализация машинного обучения: исследование на основе интервью» (arXiv: 2209.09125) и «Операции машинного обучения (MLOps): обзор, определение и архитектура» (arXiv:2205.02302)[1].

[1] Operationalizing Machine Learning: An Interview Study и Machine Learning Operations (MLOps): Overview, Definition, and Architecture.

Интеграция с приложением (6)

Последним шагом цикла освоения продукта является интеграция модельного предсказания с приложением искусственного интеллекта. Распространенный шаблон заключается в размещении моделей в службе раздачи моделей (которая будет обсуждаться в разделе 1.2.2) системы глубокого обучения и интеграции логики бизнес-приложения с моделью путем отправки запросов на модельное предсказание через интернет.

В качестве образца пользовательского сценария пользователь чат-бота взаимодействует с пользовательским интерфейсом чат-бота, вводя или озвучивая вопросы. Когда приложение чат-бот получает входные данные от потребителя, оно вызывает дистанционную службу раздачи моделей, чтобы та сгенерировала модельное предсказание, а затем предпринимает действие или отвечает потребителю на основе результата модельного предсказания.

Наряду с интеграцией раздачи моделей с логикой приложения эта фаза также предусматривает оценивание важных для продукта метрик, таких как процент переходов по ссылкам и уровень оттока пользователей. Хорошие специфичные для машинного обучения метрики (хорошая кривая прецизионности-полноты) не всегда гарантируют удовлетворение требований бизнеса. Поэтому на данном этапе интересанты со стороны бизнеса часто проводят собеседования с потребителями и дают оценку метрикам продукта.

1.1.2 Технические роли в цикле освоения

Поскольку теперь у вас есть четкое представление о каждом шаге типичного цикла освоения, давайте рассмотрим ключевые технические роли, которые взаимодействуют в этом цикле. В разных организациях определения, названия должностей и обязанности каждой роли могут различаться. Поэтому следует уточнить, кто чем занимается в вашей организации, и соответствующим образом скорректировать конструкцию своей системы.

Интересанты со стороны бизнеса (владелец продукта)

Многие организации назначают заинтересованных лиц на несколько должностей, таких как продуктовые менеджеры[1], менеджеры инженерно-конструкторских разработок[2] и старшие разработчики. Интересанты со стороны бизнеса определяют бизнес-цель продукта и несут ответственность за связь и исполнение цикла освоения продукта. В их обязанности входит следующее:

- получение вдохновения от научных изысканий в области глубокого обучения, обсуждение потенциального применения функ-

[1] Англ. product manager; син. продакт-менеджер. – *Прим. перев.*

[2] Англ. engineering manager. – *Прим. перев.*

циональных возможностей глубокого обучения в продуктах и определение бизнес-требований к продукту, которые, в свою очередь, становятся движущей силой разработки модели;
- владение продуктом; общение с потребителями и обеспечение соответствия инженерного решения бизнес-требованиям и доставке нужных результатов;
- координация межфункционального сотрудничества между разными техническими ролями и коллективами;
- управление исполнительными действиями по освоению проекта; предоставление рекомендаций или обратной связи в течение всего цикла освоения в целях обеспечения гарантии, что функциональные возможности глубокого обучения представляют реальную ценность для потребителей продукта;
- оценивание метрик продукта (таких как уровень оттока пользователей и используемость функциональных возможностей), то есть немодельных метрик (прецизионности или точности), и стимулирование улучшений в разработку модели, продукционализацию или интеграцию с продуктом.

Инженеры-изыскатели

Инженеры-изыскатели[1] в области машинного обучения занимаются научными изысканиями и разрабатывают новые архитектуры нейронных сетей. Они также разрабатывают методики повышения точности моделей и эффективности их тренировки. Эти архитектуры и методики могут использоваться при разработке модели.

> **ПРИМЕЧАНИЕ** Роль инженера-изыскателя в области машинного обучения часто ассоциируется с крупными технологическими компаниями, такими как Google, Microsoft и Salesforce. Во многих других компаниях ту же роль выполняют исследователи данных.

Исследователи данных

Исследователи данных[2] могли бы носить научно-изыскательскую шляпу, однако в большинстве случаев они транслируют бизнес-задачу в задачу машинного обучения и реализуют ее с использованием методов машинного обучения. Исследователи данных руководствуются потребностями продукта и применяют исследовательские методики не к стандартным эталонным наборам данных, а к производственным данным. Помимо исследования модельных алгоритмов, в обязанности исследователя данных могут также входить:

- комбинирование нескольких нейросетевых архитектур глубокого обучения и/или технологий из разных исследований в еди-

---

[1] Англ. researcher. – *Прим. перев.*

[2] Англ. data scientist. – *Прим. перев.*

ное решение. Иногда помимо чистого глубокого обучения они применяют дополнительные методики машинного обучения;
- разведывательный анализ имеющихся данных, определение полезности тех или иных данных и принятие решения о способах их предобработки, перед тем как передавать их для тренировки моделей;
- прототипирование разных подходов (написание экспериментального исходного кода) в целях решения бизнес-задачи;
- конвертирование исходного кода прототипирования моделей в производственный исходный код с автоматизацией рабочего процесса;
- слежение за инженерным процессом поставки моделей в производство благодаря использованию системы глубокого обучения;
- итеративная проверка необходимости получения любых дополнительных данных, которые помогают в разработке модели;
- постоянный мониторинг и оценивание результативности данных и моделей в производстве;
- устранение связанных с моделями неполадок, таких как деградация качества модели.

Инженеры данных

Инженеры данных[1] помогают собирать данные и настраивать конвейеры данных, организовывать непрерывный ввод и обработку данных, включая преобразование и обогащение данных, а также расстановку в них меток.

Инженер MLOps / инженер машинного обучения

Инженер MLOps[2] выполняет ряд технических ролей в нескольких областях, включая инженера данных, инженера DevOps, исследователя данных и инженера платформ. Помимо настройки и эксплуатации инфраструктуры машинного обучения (как систем, так и аппаратного обеспечения), они обычно управляют конвейерами автоматизации в целях создания наборов данных, тренировки и развертывания моделей. Инфраструктуры машинного обучения и действия пользователей, в том числе тренировка и раздача моделей, также контролируются инженерами MLOps.

Как вы видите, операционализация машинного обучения (MLOps) отличается сложностью, поскольку требует от людей владения набором практических навыков в области разработки программного обеспечения, эксплуатации, технического сопровождения и машинного обучения. Инженеры MLOps призваны обеспечивать эффективность и надежность разработки, развертывания, мониторинга и технического сопровождения моделей машинного обучения.

---

[1] Англ. data engineer. – *Прим. перев.*

[2] Англ. MLOps engineer. – *Прим. перев.*

Инженер систем / платформ глубокого обучения

Инженеры систем глубокого обучения[1] разрабатывают и сопровождают технически общие компоненты инфраструктуры машинного обучения – первостепенный предмет этой книги – с целью поддержания всех мероприятий по освоению машинного обучения для исследователей данных, инженеров данных, инженеров MLOps и приложений искусственного интеллекта. Среди компонентов системы машинного обучения находятся хранилища данных, вычислительные платформы, службы оркестровки рабочих процессов, склады модельных метаданных и артифактов, службы тренировки моделей, службы раздачи моделей и многое другое.

Инженер прикладного программного обеспечения

Инженеры прикладного программного обеспечения[2] разрабатывают ориентированные на потребителя приложения (как фронтендовые, так и бэкендовые[3]) с целью удовлетворения заданных бизнес-требований. Логика приложения будет предпринимать решения или действия, основываясь на модельном предсказании в ответ на конкретный запрос потребителя.

> **ПРИМЕЧАНИЕ** В будущем, по мере развития систем (инфраструктур) машинного обучения, количество технических ролей, задействованных в цикле освоения глубокого обучения, будет становиться все меньше и меньше. В конечном счете исследователи данных смогут выполнять весь цикл от начала до конца самостоятельно.

1.1.3 Пошаговый обход цикла освоения глубокого обучения

Приведя пример, мы можем продемонстрировать технические роли и процесс более конкретно. Предположим, вам было поручено задание разработать систему поддержки потребителей, которая автоматически отвечала бы на вопросы о продуктовой линейке компа-

---

[1] Англ. deep learning system engineer. – *Прим. перев.*

[2] Англ. application engineer. – *Прим. перев.*

[3] В переводе намеренно используется транслитерация терминов frontend и backend по причине отсутствия приемлемых русских аналогов. Переводы «интерфейсный/серверный» и «внешний/внутренний» по разным причинам не подходят. Так, например, в качестве бэкенда может выступать резидентное хранилище данных, и нередко встречается термин «backend server», что исключает перевод «серверный». В других языках указанные термины используются в английском написании, но есть и аналоги. Например, в немецком используются термины «верхняя часть / нижняя часть». Наиболее подходящими русскими аналогами могли бы быть «лицевая сторона» и «тыльная сторона» приложения. – *Прим. перев.*

нии. Следующие ниже шаги помогут вам в процессе выведения этого продукта на рынок.

1. Требование к продукту заключается в создании приложения поддержки потребителей, которое представляет меню, помогающее потребителям ориентироваться и находить ответы на часто задаваемые вопросы. По мере роста числа вопросов меню становится больше, со множеством уровней навигации. Аналитика показала, что навигационная система приводит многих потребителей в замешательство, от чего они перестают ориентироваться в меню, пытаясь найти ответы.
2. Продуктовый менеджер, которому принадлежит продукт, мотивирован на повышение уровня удержания пользователей и улучшение их опыта (быстрый поиск ответов). Проведя небольшое расследование с привлечением потребителей, продуктовый менеджер пришел к выводу, что большинство потребителей хотели бы получать ответы без сложной системы меню, предпочтительно так же просто, как задавать вопросы на их естественном языке.
3. Продуктовый менеджер обращается к инженерам-изыскателям в области машинного обучения за потенциальным решением. Оказывается, глубокое обучение способно помочь. Эксперты считают, что для такого варианта использования данная технология достигла достаточной зрелости, и предлагают несколько подходов, основанных на моделях глубокого обучения.
4. Продуктовый менеджер пишет спецификацию продукта, в которой указывается, что приложение должно принимать по одному вопросу от потребителя за раз, распознавать намерение по вопросу и сопоставлять его с соответствующими ответами.
5. Исследователи данных получают требования к продукту и приступают к прототипированию моделей глубокого обучения, соответствующих указанным потребностям. Сначала они проводят разведывательный анализ данных, чтобы собрать имеющиеся тренировочные данные, и консультируются с инженерами-изыскателями по поводу выбора алгоритмов. А затем исследователи данных начинают работать над исходным кодом фазы прототипирования, чтобы произвести экспериментальные модели. В конце концов, они приходят к нескольким наборам данных, нескольким алгоритмам тренировки и нескольким моделям. После тщательного оценивания из нескольких экспериментов выбирается одна модель обработки естественного языка.
6. Затем продуктовый менеджер собирает коллектив инженеров платформ, инженеров MLOps и инженеров данных для работы с исследователем данных над внедрением созданного на шаге 5 прототипного исходного кода. Указанная работа включает в себя разработку конвейера непрерывной обработки данных и конвейера непрерывной тренировки, развертывания и оценивания модели, а также настройку функциональности раздачи

модели. Продуктовый менеджер также задает число предсказаний в секунду и требуемую задержку.

7 По завершении настройки производства инженеры прикладного программного обеспечения интегрируют бэкенд службы поддержки потребителей со службой раздачи моделей (созданной на шаге 6), в результате чего, когда пользователь вводит вопрос, служба возвращает ответы, основанные на модельном предсказании. Продуктовый менеджер также определяет метрики продукта, в частности среднее время, затрачиваемое на поиск ответа, чтобы оценить конечный результат и использовать его для следующего раунда улучшения.

1.1.4 Масштабируемость при разработке проекта

Как вы увидели в разделе 1.1.2, доведение проекта глубокого обучения до завершения предусматривает, что будет заполнено семь разных технических ролей. Межфункциональное сотрудничество между этими ролями происходит практически на каждом шаге. Например, над продукционализацией проекта работают сообща инженеры данных, разработчики платформ и исследователи данных. Любой, кто был вовлечен в проект, который требует участия многих заинтересованных сторон, знает, сколько общения и координации требуется, чтобы подобный проект бесперебойно продвигался вперед.

Эти препятствия затрудняют обеспечение масштабируемости при освоении глубокого обучения, поскольку либо нет ресурсов для заполнения всех требуемых ролей, либо невозможно уложиться в сроки выпуска продукта из-за затрат на связь и замедлений. В целях сокращения огромного объема операционной работы, затрат на связь и координацию между коллективами компании инвестируют в инфраструктуру машинного обучения и сокращают количество людей и объем знаний, необходимых для разработки проекта машинного обучения. Стек инфраструктуры глубокого обучения призван не только автоматизировать разработку моделей и обработку данных, но и сделать возможным объединение технических ролей таким образом, чтобы исследователь данных мог выполнять все эти функции в рамках проекта автономно.

Ключевым индикатором успешности системы глубокого обучения является плавность процесса продукционализации модели. При наличии хорошей инфраструктуры исследователь данных, от которого не ожидается, что он внезапно станет экспертом в DevOps или инженером данных, должен уметь реализовывать модели масштабируемым образом, настраивать конвейеры данных, а также самостоятельно развертывать и отслеживать модели в производстве.

Используя эффективную систему глубокого обучения, исследователи данных смогут выполнять цикл освоения с минимальными дополнительными непроизводительными издержками – потребуется меньше связи и меньше времени, расходуемого на ожидание дру-

гими пользователями, – и сосредоточиваться на наиболее важных задачах в области науки данных, таких как понимание данных и экспериментирование с алгоритмами. Истинная ценность системы глубокого обучения состоит именно в способности масштабировать разработку проектов глубокого обучения.

1.2 Обзор конструкции системы глубокого обучения

Учитывая контекст раздела 1.1, давайте углубимся в центральную тему этой книги: саму систему глубокого обучения. Конструирование системы – любой системы – это искусство достижения целей в условиях набора ограничений, уникальных для вашей ситуации. Это верно и для систем глубокого обучения. Например, допустим, у вас есть несколько моделей глубокого обучения, которые необходимо раздавать одновременно, но ваш бюджет не позволяет вам использовать машину с достаточным объемом памяти, чтобы применять их все одновременно. Возможно, вам потребуется сконструировать механизм кеширования, чтобы обмениваться моделями между памятью и диском. Однако такой взаимообмен будет увеличивать задержку модельного вывода. Осуществимость этого решения будет зависеть от требований к задержке. Еще одна возможность заключается в использовании для каждой модели нескольких машин меньшего размера, если размеры ваших моделей и бюджет это позволяют.

Либо, в качестве еще одного примера, представьте, что продукт вашей компании должен соответствовать определенным стандартам сертификации. Эти стандарты могут предписывать политику доступа к данным, которая накладывает существенные ограничения на любого, кто хочет получить доступ к данным, собранным с помощью продукта компании. Возможно, вам потребуется сконструировать фреймворк, обеспечивающий доступ к данным в совместимом со стандартами формате, чтобы инженеры-изыскатели, исследователи данных и инженеры данных могли устранять неполадки и разрабатывать новые модели, требующие такого доступа к данным в вашей системе глубокого обучения.

Как вы видите, тут есть много кнопок, на которые можно нажимать. Безусловно, это будет итеративный процесс, ориентированный на формирование конструкции, которая будет удовлетворять как можно большему числу требований. Но для того чтобы сократить итеративный процесс, желательно начинать с конструкции, максимально приближенной к конечному состоянию.

В данном разделе мы сначала предложим конструкцию системы глубокого обучения, включающую в себя только существенные компоненты, а затем объясним сферу ответственности каждого из этих компонентов и рабочие процессы пользователей. Исходя из нашего

опыта конструирования и адаптации систем глубокого обучения, несколько ключевых компонентов являются общими для разных конструкций. Мы считаем, что они могут быть использованы в качестве разумной отправной точки для вашей конструкции. Она называется *эталонной системной архитектурой*.

Вы можете сделать копию этого эталона для своего проекта конструкции, просмотреть свой список целей и ограничений и начать с определения кнопок в каждом компоненте, которые вы можете настроить в соответствии со своими потребностями. Поскольку это не авторитетная архитектура, вам также следует принять решение о целесообразности включения всех компонентов целиком и добавлении либо удалении компонентов по своему усмотрению.

1.2.1 Эталонная системная архитектура

На рис. 1.3 показан высокоуровневый вид эталонной архитектуры системы глубокого обучения. Система глубокого обучения состоит

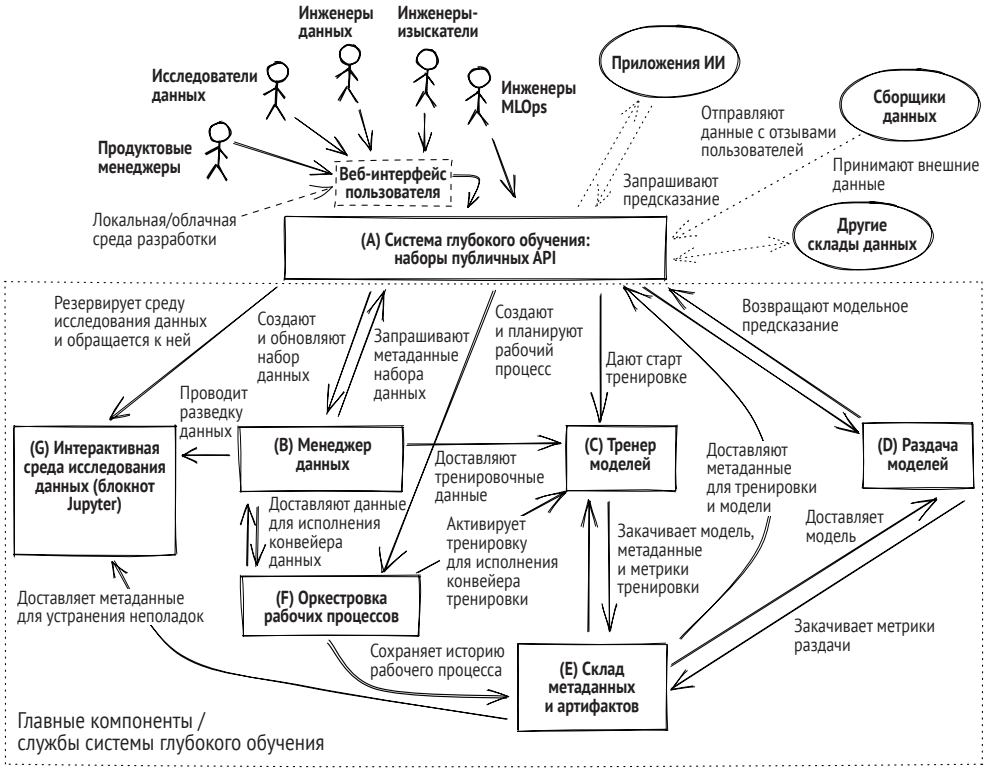


Рис. 1.3 Общий вид типичной системы глубокого обучения, которая включает в себя базовые компоненты, поддерживающие цикл освоения глубокого обучения. Приведенная выше эталонная системная архитектура может использоваться в качестве отправной точки и адаптирована дальше. В последующих главах мы подробно обсудим каждый компонент и объясним то, как он вписывается в эту общую картину

из двух ведущих частей. Первая – это интерфейс прикладного программирования (API; блок A) системы, расположенный в середине диаграммы. Вторая – это набор компонентов системы глубокого обучения, который представлен всеми прямоугольными блоками, расположенными внутри большого прямоугольника, обведенного пунктирной линией и занимающего нижнюю половину диаграммы. Каждый блок представляет собой системный компонент:

- API (блок A);
- менеджер наборов данных (блок B);
- тренер моделей (блок C);
- раздача моделей (блок D);
- склад метаданных и артифактов (блок E);
- оркестровка рабочих процессов (блок F);
- интерактивная среда исследования данных (блок G).

В этой книге мы исходим из допущения, что приведенные выше системные компоненты являются микросервисами.

> **ОПРЕДЕЛЕНИЕ** Единого определения термина *микросервис* не существует. Здесь этот термин будет использоваться нечасто, обозначая процессы, которые взаимодействуют по сети с помощью протокола HTTP либо gRPC.

Указанное допущение означает следующее: мы можем ожидать, что эти компоненты будут разумно и безопасно поддерживать нескольких пользователей с разными техническими ролями и будут легко доступны по сети или интернету. (Однако в данной книге все инженерные аспекты конструирования или разработки микросервисов рассматриваться не будут. Мы сосредоточим наше изложение на особенностях, которые имеют отношение к системам глубокого обучения.)

> **ПРИМЕЧАНИЕ** Возможно, вы задаетесь вопросом, нужно ли вам самостоятельно конструировать, разрабатывать и размещать у себя все компоненты системы глубокого обучения. И действительно, существуют альтернативы с открытым исходным кодом (Kubeflow) и с возможностью размещения на платформах (Amazon SageMaker). Мы надеемся, что после того, как вы изучите основы каждого компонента и то, как они вписываются в общую картину и как используются разными техническими ролями, вы примете решение, которое подойдет для вашего варианта использования наилучшим образом.

1.2.2 Ключевые компоненты

Теперь давайте рассмотрим ключевые компоненты, которые, по нашему мнению, необходимы для базовой системы глубокого обуче-

ния, как показано на рис. 1.3. Вы можете добавить дополнительные компоненты или упростить их еще больше, если сочтете нужным в соответствии с вашими требованиями.

Интерфейс прикладного программирования

Точкой входа (блок A на рис. 1.3) нашей системы глубокого обучения является доступный по сети API. Мы выбрали API, потому что система должна поддерживать не только графические пользовательские интерфейсы, которые будут использоваться инженерами-изыскателями, исследователями данных, инженерами данных и т. п., но и приложениями и, возможно, другими системами, например складами данных из партнерской организации.

Хотя концептуально API является единой точкой входа в систему, вполне возможно, что API определяется как сумма всех API, предоставляемых каждым компонентом, без дополнительного слоя, который агрегирует все в конечной точке единой службы. На протяжении всей этой книги мы будем использовать сумму всех API, предоставляемых непосредственно каждым компонентом, и ради простоты пропустим агрегирование.

> **ПРИМЕЧАНИЕ** Какой API системы глубокого обучения следует выбрать: централизованный или же распределенный? В эталонной системной архитектуре (рис. 1.3) API системы глубокого обучения показан в виде отдельного блока. Его следует интерпретировать как логический контейнер полного набора API вашей системы глубокого обучения, независимо от того, реализован ли он на одном (например, API-шлюзе, который служит посредником для всех компонентов) или же на нескольких конечных точках службы (прямое взаимодействие с каждым компонентом). Каждая реализация имеет свои достоинства и недостатки, и вам следует поработать со своим коллективом, чтобы выяснить, какая из них подойдет лучше всего. Прямое взаимодействие с каждым компонентом может быть проще, если вы начнете с небольшого варианта использования и малого коллектива.

Менеджер наборов данных

Глубокое обучение основано на данных. Нет никаких сомнений в том, что компонент управления данными является центральной частью системы глубокого обучения. Каждая система обучения – это система, работающая по принципу «мусор на входе, мусор на выходе», поэтому обеспечение хорошего качества тренировочных данных для машинного обучения имеет первостепенное значение. Хороший компонент управления данными должен обеспечивать решение этой задачи. Он позволяет собирать, организовывать, описывать и хранить данные, что, в свою очередь, дает возможность

проводить разведывательный анализ данных, размечать и использовать их для тренировки моделей.

На рис. 1.3 мы видим по крайней мере четыре взаимосвязи менеджера наборов данных (блок B) с другими сторонами:

- сборщики данных отправляют сырые данные менеджеру наборов данных, чтобы тот создавал или обновлял наборы данных;
- служба оркестровки рабочих процессов (блок F) исполняет конвейер данных, который берет данные у менеджера наборов данных, чтобы обогащать тренировочный набор данных либо преобразовывать формат данных, и возвращает результат обратно;
- исследователи данных, инженеры-изыскатели и инженеры данных используют блокнот Jupyter (блок G), чтобы брать данные у менеджера данных и проводить их разведывательный анализ и обследование;
- служба тренировки моделей (блок C) берет тренировочные данные у менеджера данных, чтобы тренировать модели.

В главе 2 мы подробно обсудим управление наборами данных. На протяжении всей книги мы используем термин *набор данных* как совокупность собранных данных, которые могут быть связаны между собой.

Тренер моделей

Тренер моделей (или служба тренировки моделей; блок C) отвечает за предоставление базовых вычислительных ресурсов, таких как центральные процессоры, оперативная память и графические процессоры, а также логики управления заданиями, необходимых для исполнения исходного кода тренировки моделей и генерирования модельных файлов. На рис. 1.3 мы видим, что служба оркестровки рабочих процессов (блок F) сообщает тренеру моделей, что нужно исполнить исходный код тренировки модели. Тренер берет входные тренировочные данные у менеджера наборов данных (блок B) и продуцирует модель. Затем он закачивает модель с метриками тренировки и метаданными на склад метаданных и артифактов (блок E).

Обычно создание высококачественных моделей глубокого обучения, которые могут генерировать точные предсказания, предусматривает выполнение интенсивных вычислений на крупном наборе данных. Внедрение новых алгоритмов и библиотек/фреймворков тренировки также является критически важным требованием. Эти требования порождают проблемы на нескольких уровнях:

- *возможность сокращения времени тренировки моделей* – несмотря на растущий объем тренировочных данных и сложность модельной архитектуры, системы тренировки должны поддерживать разумное время тренировки;
- *горизонтальная масштабируемость* – эффективная производственная система тренировки должна быть способна поддер-

живать несколько запросов на тренировку от разных приложений и пользователей одновременно;

- *стоимость внедрения новых технологий* – сообщество глубокого обучения активно работает и постоянно вносит обновления и улучшения в алгоритмы и инструментарии (SDK, фреймворки). Система тренировки должна быть достаточно гибкой, чтобы легко приспосабливаться к инновациям, не влияя на существующую рабочую нагрузку.

В главе 3 мы рассмотрим разные подходы к решению вышеупомянутых проблем. В этой книге мы не будем углубляться в теоретический аспект алгоритмов тренировки, поскольку они не влияют на то, как мы конструируем систему. В главе 4 мы рассмотрим способы распределения тренировки с целью ускорения этого процесса. В главе 5 разберем несколько разных подходов к оптимизации гиперпараметров тренировки.

Раздача моделей

Модели могут использоваться в различных обстановках, таких как онлайновый модельный вывод для предсказания в реальном времени или офлайновый модельный вывод для пакетных предсказаний с использованием крупных объемов входных данных. Именно здесь проявляется раздача моделей – когда система размещает модель у себя, принимает входные запросы на предсказание, генерирует модельное предсказание и возвращает предсказание пользователям. При этом необходимо ответить на несколько ключевых вопросов:

- Поступают ли ваши запросы на модельный вывод из сети? Или же они поступают от датчиков, которые должны обслуживаться локально?
- Какова допустимая задержка? Являются ли запросы на модельный вывод разовыми или потоковыми?
- Сколько моделей раздается? Каждая ли модель в отдельности обслуживает определенный тип запроса на модельный вывод или это делает ансамбль моделей?
- Насколько велики размеры моделей? Какой объем памяти нужен для того, чтобы предусмотреть это в бюджете?
- Какие модельные архитектуры необходимо поддерживать? Требуется ли для них графический процессор? Сколько вычислительных ресурсов нужно для генерирования модельных выводов? Существуют ли другие вспомогательные компоненты раздачи – например, векторные вложения, нормализация, агрегирование и т. д.?
- Достаточно ли ресурсов для поддержания моделей в режиме онлайн? Или же необходима стратегия взаимообмена (например, перемещения моделей между памятью и диском)?

По рис. 1.3 видно, что главными входными и выходными данными раздачи моделей (блок D) являются соответственно запросы на

модельный вывод и возвращаемое предсказание. Для генерирования модельных выводов модели извлекаются со склада метаданных и артифактов (блок E). Некоторые запросы и ответы на них могут регистрироваться в журналах и отправляться в службу мониторинга и оценивания моделей (на рис. 1.3 это не показано и в данной книге не описано), которая обнаруживает аномалии в этих данных и выдает предупреждения. В главах 6 и 7 мы углубимся в архитектуру раздачи моделей, проведем обследование этих ключевых аспектов и обсудим их решения.

Склад метаданных и артифактов

Представьте, что вы работаете над простым приложением глубокого обучения не в коллективе, а в одиночку, в котором вам нужно работать всего с несколькими наборами данных, тренировать и развертывать только один тип модели. Вероятно, вы можете отслеживать характер взаимосвязей наборов данных, исходных кодов тренировки, моделей, исходных кодов модельного вывода и модельных выводов друг с другом. Эти взаимосвязи необходимы для разработки модели и устранения неполадок, поскольку вам необходимо иметь возможность прослеживать те или иные наблюдения в обратном направлении вплоть до первопричины.

Теперь представьте, что вы добавляете больше приложений, больше типов моделей, и вы уже работаете в коллективе из нескольких человек. Число этих взаимосвязей будет расти в геометрической прогрессии. В системе глубокого обучения, сконструированной поддерживать несколько типов пользователей, работающих с несколькими наборами данных, исходным кодом и моделями на различных этапах, существует потребность в компоненте, который отслеживает паутину взаимосвязей. Склад метаданных и артифактов в системе глубокого обучения делает именно это. Артифакты содержат исходный код, который тренирует модели и продуцирует модельные выводы, а также любые сгенерированные данные, такие как натренированные модели, модельные выводы и метрики. Метаданные – это любые данные, которые описывают артифакт или взаимосвязь между артифактами. Вот несколько конкретных примеров:

- автор и версия фрагмента исходного кода тренировки;
- ссылка на входной тренировочный набор данных и среду тренировки натренированной модели;
- метрики тренировки натренированной модели, такие как дата и время тренировки, продолжительность тренировки и владелец задания на тренировку;
- специфичные для модели метрики, такие как версия модели, происхождение модели (данные и исходный код, использовавшиеся во время тренировки) и метрики результативности;
- модель, запрос и исходный код модельного вывода, которые генерируют тот или иной модельный вывод;

- история рабочего процесса, отслеживающая каждый шаг тренировки модели и все исполнения конвейера обработки данных.

Это всего лишь несколько примеров того, что базовый склад метаданных и артифактов помогает отслеживать. Вы должны адаптировать этот компонент под потребности вашего коллектива или организации.

Любой другой компонент, который генерирует метаданные и артифакты на рис. 1.3, будет поступать на склад метаданных и артифактов (блок E). Склад играет важную роль в раздаче моделей, поскольку он предоставляет модельные файлы и их метаданные службе раздачи моделей (блок D). Хотя на рисунке это не показано, конкретно-прикладные инструменты трассировки и устранения неполадок обычно разрабатываются в слое пользовательского интерфейса, работающего на базе склада метаданных и артифактов.

По мере прохождения главы 8 мы рассмотрим базовый склад метаданных и артифактов. Этот склад обычно является центральным компонентом пользовательского интерфейса системы глубокого обучения.

Оркестровка рабочих процессов

Оркестровка рабочих процессов (рис. 1.3, блок F) нашла широкое распространение во многих системах. Она помогает автоматически запускать вычислительные задания, инициированные программными условиями. В контексте системы машинного обучения оркестровка рабочих процессов является движущей силой для всех видов автоматизации, выполняемых в системе глубокого обучения. Она позволяет людям определять рабочие процессы или конвейеры – ориентированные ациклические графы[1] – для склеивания отдельных заданий, выстраивая их в порядке исполнения. Компонент оркестровки рабочих процессов управляет исполнением заданий в рамках этих рабочих процессов. Вот несколько типичных примеров:

- запуск тренировки модели всякий раз, когда формируется новый набор данных;
- мониторинг вышестоящих источников данных, обогащение за счет новых данных, перенос их формата, уведомление внешних расстановщиков меток и слияние новых данных с существующими наборами данных;
- развертывание натренированной модели на сервере моделей, если она проходит по каким-либо принятым критериям;
- постоянный мониторинг метрик результативности модели и оповещение разработчиков при обнаружении деградации.

В главе 9 вы узнаете, как разрабатывать и настраивать систему оркестровки рабочих процессов.

---

[1] Англ. directed acyclic graph (DAG). – *Прим. перев.*

Интерактивная среда исследования данных

Данные о потребителях и моделях невозможно скачивать на локальную рабочую станцию из производственной среды по соображениям соответствия стандартам и безопасности. Для того чтобы исследователи данных могли интерактивно проводить разведывательный анализ данных, устранять неполадки при исполнении конвейера в рамках оркестровки рабочих процессов и отлаживать модели, требуется дистанционная интерактивная среда исследования данных, также именуемая средой науки о данных (рис. 1.3, блок G), расположенная внутри системы глубокого обучения.

В компаниях общепринято настраивать свою собственную заслуживающую доверия среду исследования данных, используя блокноты Jupyter с открытым исходным кодом (https://jupyter.org/) или задействуя облачные вычислительные решения поставщиков на базе JupyterLab, такие как Amazon SageMaker Studio (https://aws.amazon.com/sagemaker/studio/).

Типичная интерактивная среда исследования данных должна обеспечивать следующие ниже функциональные возможности:

- *разведывательный анализ данных* – предоставляет исследователям данных легкий доступ к данным потребителей, но обеспечивает их безопасность и соответствие стандартам; утечек данных не происходит, и любой несанкционированный доступ к данным будет отклоняться;
- *прототипирование моделей* – предоставляет исследователям данных инструментарий, столь необходимый для быстрой разработки опытных образцов моделей внутри системы глубокого обучения;
- *устранение неполадок* – позволяет инженерам отлаживать любые действия, происходящие внутри системы глубокого обучения, такие как скачивание модели и ее воспроизведение с целью анализа ее поведения или проверка всех артифактов входных/выходных данных (промежуточных наборов данных или конфигураций) из сбойного конвейера.

1.2.3 Ключевые пользовательские сценарии

В целях более глубокого понимания того, как системы глубокого обучения могут использоваться в течение цикла освоения (рис. 1.1), мы подготовили образцы сценариев, которые иллюстрируют способы их возможного использования. Давайте начнем с программных потребителей, показанных на рис. 1.4. Сборщики данных, которые отправляют данные в систему, обычно в итоге посредством API оказываются в службе управления данными, которая собирает и организовывает сырые данные для тренировки модели.

Рис. 1.4 Данные передаются из источников или от сборщиков через API в службу управления данными, в которой данные далее организовываются и хранятся в форматах, более удобных для тренировки модели

Приложения глубокого обучения обычно обращаются к службе генерирования модельных выводов для получения модельных выводов из натренированной модели, которая используется для приведения в действие функциональностей глубокого обучения, которые будут потребляться конечными пользователями. На рис. 1.5 показана последовательность этого взаимодействия. Программными потребителями также могут быть скрипты или даже полноценные службы управления. Поскольку их наличие является опциональным, ради простоты на рисунке мы их опустили.

Рис. 1.5 Приложения глубокого обучения запрашивают модельные выводы посредством API. Служба генерирования модельных выводов принимает и обрабатывает эти запросы при участии натренированных моделей и выдает модельные выводы, которые возвращаются обратно в приложения

Между потребителями-людьми и API обычно лежит дополнительный слой – пользовательский интерфейс. Указанный интерфейс может быть организован на базе веба либо командной строки. Некоторые опытные пользователи даже могут этот интерфейс пропускать и использовать API напрямую. Давайте пройдемся по каждому персонажу по очереди.

Типичный сценарий использования системы инженерами-изыскателями проиллюстрирован на рис. 1.6. Инженеры-изыскатели просматривают имеющиеся данные, чтобы опробировать свою новую методику моделирования. Они обращаются к пользовательскому интерфейсу и посещают раздел разведывательного анализа и визуализации данных, который извлекает данные из службы управления данными. Для преобразования данных в формы, которые могут подаваться в новые методики тренировки, может требоваться значительная ручная обработка. После того как инженеры-изыскатели определятся с методикой, они упаковывают ее в виде библиотеки, чтобы ее могли потреблять другие.

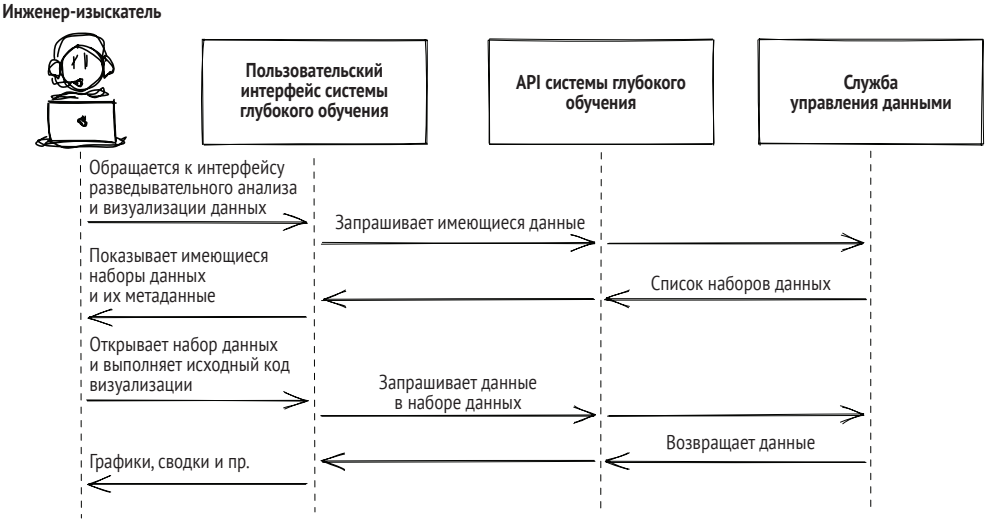


Рис. 1.6 Последовательность используемости инженером-изыскателем, который заинтересован быть в курсе о данных, имеющихся для научных изысканий и разработки новой методики моделирования. Инженер-изыскатель взаимодействует с пользовательским интерфейсом, который за кулисами поддерживается API и службой управления данными

Исследователи данных и инженеры данных могут работать над вариантами использования, сначала просматривая имеющиеся данные, аналогично тому, что изначально делали инженеры-изыскатели в предыдущем абзаце. Это будет поддерживаться службой управления данными. Они выдвигают гипотезы и объединяют методики обработки данных и тренировки в виде исходного кода. Указанные шаги можно комбинировать, формируя рабочий процесс с помощью службы управления рабочим процессом.

Когда служба управления рабочим процессом исполняет рабочий процесс, она взаимодействует со службой управления данными и службой тренировки моделей, выполняя фактические обязанности и отслеживая их продвижение. Гиперпараметры, версии исходного кода, метрики тренировки моделей и результаты тестирования – все это сохраняется каждой службой и исходным кодом тренировки на складе метаданных и артифактов.

С помощью пользовательского интерфейса исследователи данных и инженеры данных могут сравнивать прогоны экспериментов и определять наилучший способ тренировки моделей. Этот вышеупомянутый сценарий показан на рис. 1.7.

Пользовательский интерфейс также может использоваться продуктовыми менеджерами, которые будут просматривать и запрашивать все виды метрик по всей системе. Метрические данные могут поставляться со склада метаданных и артифактов.

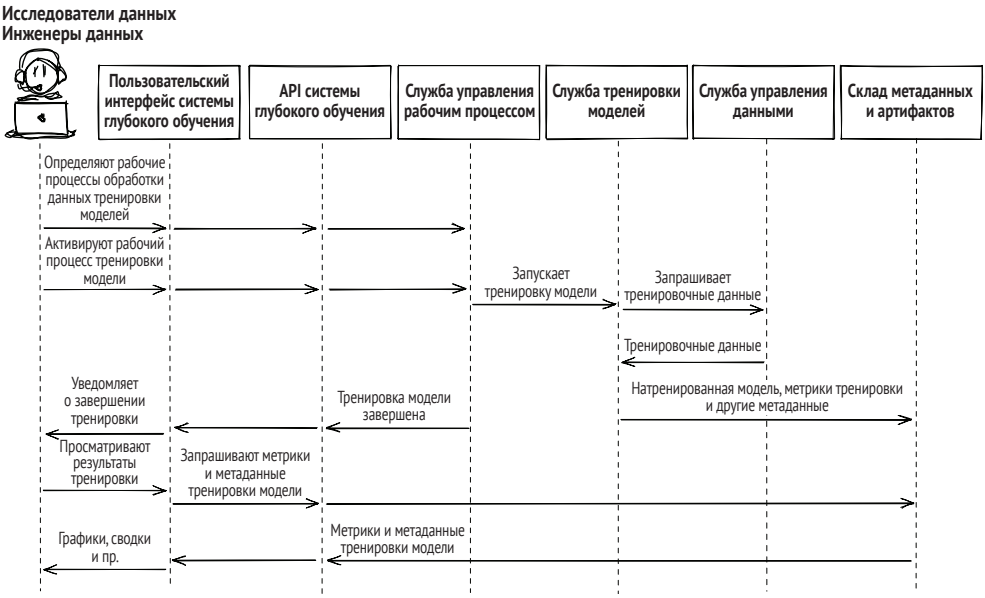


Рис. 1.7 Последовательность использования исследователем данных, который задает рабочий процесс тренировки моделей, выполняет его и просматривает результаты

1.2.4 Выведение своей собственной конструкции

Теперь, когда мы прошлись по всем аспектам эталонной системной архитектуры, давайте рассмотрим несколько рекомендаций по адаптации вашей собственной версии.

Сбор целей и требований

Первым шагом в конструировании любой успешной системной конструкции является наличие набора четких целей и требований, с которыми предстоит работать. В идеале они должны исходить от пользователей вашей системы, прямо или косвенно через коллектив продуктового менеджмента или инженерно-конструкторского менеджмента. Этот краткий список целей и требований поможет вам сформировать видение того, как будет выглядеть ваша система. Это видение, в свою очередь, должно быть ориентиром, которым вы будете руководствоваться во всех фазах конструирования и разработки вашей системы.

> **ПРИМЕЧАНИЕ** Иногда инженеров просят разработать систему, могущую поддерживать уже существующие одно или несколько приложений глубокого обучения. В этом случае вы, напротив, можете начать с определения набора требований,

общих для этих приложений, и с того, как ваша система может быть сконструирована таким образом, чтобы помогать быстро внедрять инновации в эти приложения.

В целях сбора системных целей и требований вам необходимо выявить разные типы пользователей и интересантов, или персон, системы. (Эта концепция является общей и может применяться к большинству задач системного конструирования.) В конце концов, именно пользователи помогут вам сформулировать цели системы и требования к ней.

Если вы не уверены в хорошей отправной точке, то мы рекомендуем начать с вариантов использования или требований к приложению. Вот несколько образцов вопросов, которые вы можете задать своим пользователям:

- *инженерам и продуктовым менеджерам*: позволяет ли система собирать приложениям данные для тренировки? Нужно ли системе обрабатывать потоковые входные данные? Какой объем данных собирается?
- *исследователям данных и инженерам данных*: как мы обрабатываем и расставляем метки в данных? Должна ли система предоставлять инструменты расстановки меток внешним поставщикам? Как мы оцениваем модель? Как мы оперируем тестовым набором данных? Нужен ли для работы с данными интерактивный пользовательский интерфейс блокнотирования?
- *инженерам-изыскателям и исследователям данных*: насколько большой объем данных необходим для тренировки моделей? Каково среднее время тренировки модели? Какой объем вычислительных мощностей и данных необходим для научных изысканий и исследования данных? Какие эксперименты должна поддерживать система? Какие метаданные и метрики необходимо собирать для оценивания разных экспериментов?
- *продуктовым менеджерам и инженерам прикладного программного обеспечения*: раздача моделей осуществляется на дистанционном сервере или же на клиенте? Генерируется ли модельный вывод в реальном времени или же это офлайновое пакетное предсказание? Существует ли требование к задержке?
- *продуктовым менеджерам*: какие задачи мы пытаемся решить в нашей организации? Какова наша бизнес-модель? Как мы собираемся оценивать эффективность наших реализаций?
- *коллективам обеспечения безопасности*: какой уровень безопасности необходим в вашей системе? Является ли доступ к данным широко открытым или строго ограниченным/изолированным? Существует ли требование к аудиту? Должна ли система достигать определенного уровня соответствия стандартам или сертификации (например, Общим нормативам защиты дан-

ных, Правилам системного и организационного контроля версии 2[1] и т. д.)?

Адаптация эталонной системной архитектуры под конкретно-прикладные требования

После того как станут ясны требования к конструкции и область применения, мы можем приступить к адаптации эталонной системной архитектуры, показанной на рис. 1.3. Сначала можно решить, нужно добавлять или удалять какие-либо компоненты. Например, если требование заключается исключительно в управлении тренировкой моделей в дистанционной серверной ферме, то мы могли бы удалить компонент управления рабочими процессами. Если исследователи данных хотят эффективно оценивать результативность моделей с помощью производственных данных, то они могли бы также добавить компонент управления экспериментами. Этот компонент позволит им проводить тренировку и валидацию с использованием полномасштабных данных, которые уже существуют в системе, и проводить онлайновое A/B-тестирование на основе производственного трафика с ранее не встречавшимися данными.

Вторым шагом является конструирование и реализация каждого набора ключевых компонентов в соответствии с вашими конкретными потребностями. В зависимости от требований из службы управления наборами данных можно исключить API потоковой передачи данных и добавить поддержку распределенной тренировки, если вас беспокоит скорость тренировки. Вы можете либо разработать каждый ключевой компонент с нуля, либо использовать программное обеспечение с открытым исходным кодом. В остальной части книги мы рассмотрим оба варианта для каждого ключевого компонента, чтобы вы знали, что делать.

> **СОВЕТ** Конструкция системы должна быть простой, а ее использование – удобным. Создание такой большой системы глубокого обучения призвано повышать продуктивность разработки проектов глубокого обучения, поэтому, пожалуйста, имейте этот аспект в виду при ее конструировании. Мы хотим облегчить исследователям данных строительство высококачественных моделей без необходимости изучать механизмы, происходящие в опорной системе.

1.2.5 Разработка компонентов поверх Kubernetes

Мы представили список ключевых компонентов, которые реализуются в виде служб. При таком количестве служб вы, возможно, за-

---

[1] General Data Protection Regulation, System and Organization Controls 2. – *Прим. перев.*

хотите управлять ими с помощью высокотехнологичной системы на инфраструктурном уровне, таком как Kubernetes.

Kubernetes – это система с открытым исходным кодом, служащая для автоматизации развертывания, масштабирования и управления контейнеризированными приложениями, которые выполняются в изолированных средах выполнения, например Docker-контейнерах. Мы встречали целый ряд систем глубокого обучения, построенных поверх Kubernetes. Некоторые люди учатся использовать Kubernetes, даже не зная, почему он используется для выполнения служб глубокого обучения, поэтому мы хотим объяснить, что за этим стоит. Если вы знакомы с Kubernetes, то можете спокойно этот раздел пропустить.

> **ПРИМЕЧАНИЕ** Kubernetes – сложная платформа, для ознакомления с которой потребовался бы материал объемом в целую книгу, поэтому мы обсуждаем ее достоинства только для системы глубокого обучения. Если вы хотите изучить Kubernetes, то мы настоятельно рекомендуем вам книгу «Kubernetes в действии» Марко Лукши (Manning, 2018)[1].

Трудности в управлении вычислительными ресурсами

Исполнение одного Docker-контейнера на дистанционном сервере кажется простой задачей, но исполнение 200 контейнеров на 30 разных серверах – это совсем другая история. Существует целый ряд трудностей, таких как мониторинг всех дистанционных серверов в целях определения того, на каком из них выполнять контейнер, необходимость переключения контейнера на исправный сервер, перезапуск контейнера при его зависании, отслеживание каждого исполнения контейнера и получение уведомлений о его завершении и т. д. Способность справляться с этими трудностями предусматривает, что мы умеем сами контролировать аппаратное обеспечение, процессы операционной системы и сетевое взаимодействие. Это не только технически сложно, но и требует огромного объема работы.

В чем помощь Kubernetes

Kubernetes – это платформа с открытым исходным кодом, предназначенная для оркестровки контейнеров, служащая для планирования и автоматизации развертывания, управления и масштабирования контейнеризированных приложений. После настройки кластера Kubernetes операции ваших серверных групп (развертывание, внесение исправлений, обновления) и ресурсы станут управляемыми. Вот пример развертывания: с помощью команды вы можете сообщить Kubernetes, чтобы он запустил Docker-образ с 16 Гб памяти и 1 гра-

[1] Marko Luksa, Kubernetes in Action, Manning, 2018.

фическим процессором; Kubernetes выделит ресурс и запустит этот Docker-образ за вас.

Это имеет огромное преимущество для разработчиков программного обеспечения, поскольку не каждый из них имеет большой опыт работы с оборудованием и развертыванием. С платформой Kubernetes нам нужно только объявлять конечное состояние нашего кластера, и Kubernetes будет выполнять фактическую работу для достижения наших целей.

Помимо преимуществ развертывания контейнеров, ниже приведено несколько других ключевых функциональных возможностей Kubernetes, которые имеют решающее значение для управления нашими контейнерами тренировки:

- *функциональности автомасштабирования* – Kubernetes автоматически изменяет число узлов в кластере в зависимости от рабочей нагрузки. Это означает, что если произойдет внезапный всплеск пользовательских запросов, то Kubernetes автоматически добавит емкость; это называется *эластичным управлением вычислениями*;
- *способности самовосстанавливаться* – Kubernetes перезапускает, заменяет или перепланирует поды, когда они выходят из строя или когда узлы умирают. Он также уничтожает поды, которые не реагируют на пользовательские проверки работоспособности;
- *задействованность и изоляция ресурсов* – Kubernetes заботится о насыщении вычислительных ресурсов; он обеспечивает, чтобы каждый сервер задействовался полностью. Внутри платформа Kubernetes запускает контейнеры приложений в виде подов. Каждый под представляет собой изолированную среду с гарантией обеспеченности ресурсами, и в нем работает функциональный блок. В Kubernetes несколько подов могут находиться на одном узле (сервере) до тех пор, пока их совокупные требования к ресурсам (процессору, памяти, диску) не превысят ограничения узла, поэтому серверы могут легко использоваться разными функциональными блоками с гарантированной изоляцией;
- *пространства имен* – Kubernetes поддерживает разделение физического кластера на разные виртуальные кластеры. Эти виртуальные кластеры называются *пространствами имен*. Вы можете определить квоту ресурсов для каждого пространства имен, что позволяет изолировать ресурсы для разных коллективов путем их назначения разным пространствам имен.

С другой стороны, за эти преимущества приходится платить – они также потребляют ресурсы. Когда вы выполняете под Kubernetes, сам под занимает некоторый объем системных ресурсов (процессор, память). Эти ресурсы потребляются в дополнение к тем, которые необходимы для выполнения контейнеров внутри подов. Во многих

ситуациях непроизводительные издержки платформы Kubernetes выглядят разумными; например, из эксперимента, опубликованного в статье «Как мы минимизировали непроизводительные издержки Kubernetes в нашей системе заданий» (http://mng.bz/DZBV) Лалли Сингх и Ашвин Венкатесан (февраль 2021 г.)[1], непроизводительные издержки у центрального процессора в расчете на под составили около 10 мс в секунду.

> **ПРИМЕЧАНИЕ** Мы рекомендуем вам ознакомиться с дополнением B к книге, чтобы увидеть, как существующие системы глубокого обучения соотносятся с концепциями, представленными в этой главе. В указанном дополнении мы сравниваем описанную в разделе 1.2.1 эталонную системную архитектуру с Amazon SageMaker, Google Vertex AI, Microsoft Azure Machine Learning и Kubeflow.

1.3 Разработка системы глубокого обучения в сравнении с разработкой модели

Перейдем к заключительной части черновой работы, прежде чем мы начнем: мы считаем крайне важным указать на различия между разработкой системы глубокого обучения и разработкой модели глубокого обучения. В этой книге мы определяем практику разработки модели глубокого обучения для решения задачи как процесс:

- разведывательного анализа имеющихся данных и способов их преобразования в целях тренировки;
- определения эффективного алгоритма(ов) тренировки, который будет использоваться для решения задачи;
- тренировки модели и разработки исходного кода модельного вывода в целях тестирования на основе ранее не встречавшихся данных.

Напомним, что система глубокого обучения должна поддерживать не только все задания, требуемые разработкой модели, но и те, которые должны выполняться другими техническими ролями, и обеспечивать беспрепятственное взаимодействие между этими ролями. Разрабатывая систему глубокого обучения, вы не разрабатываете модели глубокого обучения; вы разрабатываете систему, которая поддерживает разработку моделей глубокого обучения, делая этот процесс более эффективным и масштабируемым.

Мы нашли массу опубликованных материалов о разработке моделей. Но в написанном мы встретили очень мало о конструировании

[1] *Lally Singh, Ashwin Venkatesan*, How We Minimized the Overhead of Kubernetes in Our Job System, February 2021.

и строительстве платформ либо систем, поддерживающих эти модели. И именно поэтому мы написали данную книгу.

Резюме

- Типичная разработка проекта машинного обучения проходит следующий цикл: инициация продукта, разведывательный анализ данных, прототипирование модели, продукционализация и интеграция в производство.
- В разработке проекта глубокого обучения задействовано семь разных технических ролей: продуктовый менеджер, инженеры-изыскатели, исследователи данных, инженеры данных, инженеры MLOps, инженеры систем машинного обучения и инженеры прикладного программного обеспечения.
- Система глубокого обучения должна снижать сложность внутри цикла освоения глубокого обучения.
- С помощью систем глубокого обучения исследователь данных, от которого не ожидается, что он внезапно станет экспертом в DevOps или инженером данных, должен уметь масштабируемо реализовывать модели, настраивать конвейеры данных, а также самостоятельно развертывать и осуществлять мониторинг моделей в производственной среде.
- Эффективная система глубокого обучения должна позволять исследователям данных сосредоточиваться на интересных и важных задачах в области науки о данных.
- Высокоуровневая эталонная системная архитектура, подобная той, которую мы представляем на рис. 1.3, поможет вам быстро приступить к работе над новой конструкцией. Сначала сделайте свою собственную копию, а затем соберите цели и требования. Наконец, добавляйте, изменяйте или удаляйте компоненты и их взаимосвязи по своему усмотрению.
- Базовая система глубокого обучения состоит из следующих ключевых компонентов: менеджера наборов данных, тренера моделей, раздачи моделей, склада метаданных и артифактов, оркестровки рабочих процессов и среды исследования данных.
- Компонент управления данными помогает собирать, организовывать, описывать и хранить данные в виде наборов данных, которые можно использовать в качестве входных тренировочных данных. Он также поддерживает операции по разведывательному анализу данных и отслеживает информацию о происхождении (эволюционных изменениях с течением времени) между наборами данных. В главе 2 управление данными будет рассмотрено подробно.
- Компонент тренировки моделей отвечает за обработку нескольких запросов на тренировку и их эффективное выполнение при

заданном ограниченном наборе вычислительных ресурсов. В главах 3 и 4 будет рассмотрен компонент тренировки моделей.

- Компонент раздачи моделей обрабатывает входящие запросы на модельный вывод, генерирует выводы с использованием моделей и возвращает их отправителям запросов. Об этом будет рассказано в главах 6 и 7.
- Компонент складирования на складе метаданных и артифактов регистрирует метаданные и сохраняет артифакты из остальной части системы. Любые получаемые системой данные могут рассматриваться как артифакты. Большинство из них будут моделями, сопровождаемыми метаданными, которые будут складироваться в том же компоненте. Он обеспечивает полную информацию о происхождении данных с целью поддержки экспериментов и устранения неполадок. Мы поговорим об этом компоненте в главе 8.
- Компонент управления рабочими процессами хранит определения рабочего процесса, которые связывают воедино разные шаги обработки данных и тренировки моделей. Он отвечает за инициирование периодических прогонов рабочего процесса и отслеживание хода выполнения каждого шага прогона, который выполняется в других компонентах, например шага тренировки модели, выполняемого в службе тренировки моделей. В главе 9 мы представим пошаговое описание этого компонента.
- Система глубокого обучения должна поддерживать цикл освоения глубокого обучения и облегчать взаимодействие между несколькими техническими ролями.
- Разработка системы глубокого обучения отличается от разработки модели глубокого обучения. Система представляет собой инфраструктуру, служащую для поддержки разработки моделей глубокого обучения.