3.2. ЧТО ТАКОЕ KERAS

Keras — это библиотека глубокого обучения для Python, основанная на TensorFlow, которая обеспечивает удобный способ определения и тренировки моделей глубокого обучения. Первоначально Keras создавалась для исследований с целью упростить эксперименты с глубоким обучением.

Благодаря TensorFlow библиотека Keras может работать на различных типах оборудования (рис. 3.1) — графическом, тензорном или обычном процессоре — и поддерживает простую возможность распределения вычислений между тысячами компьютеров.

Особое внимание в Keras уделяется опыту разработчиков. Данная библиотека предназначена для людей, а не для машин. Она следует передовым методам снижения когнитивной нагрузки, предлагая простые и логичные рабочие процессы, минимизируя количество действий, необходимых для типичных случаев использования, и давая четкую и действенную обратную связь на ошибки пользователя. Это упрощает освоение Keras начинающими исследователями и увеличивает продуктивность экспертов.

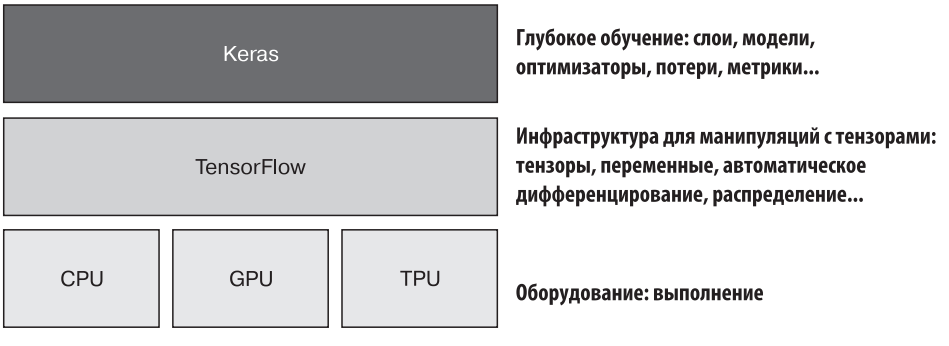

Рис. 3.1. Keras и TensorFlow: TensorFlow — это низкоуровневая платформа тензорных вычислений, а Keras — высокоуровневая библиотека глубокого обучения

По состоянию на конец 2021 года насчитывалось более одного миллиона пользователей Keras: от исследователей, инженеров и специалистов по обработке данных в начинающих и крупных компаниях до аспирантов и любителей. Keras используется в Google, Netflix, Uber, CERN, NASA, Yelp, Instacart, Square и сотнях стартапов, работающих над решением широкого круга задач во всех отраслях. Рекомендации на YouTube для вас подбираются моделями Keras. Беспилотные автомобили Waymo также управляются ими. Keras популярна на

Kaggle — веб-сайте, проводящем соревнования по машинному обучению, большинство из которых было выиграно с использованием данного фреймворка.

Такое число пользователей библиотека Keras приобрела потому, что не вынуждает следовать одному «истинному» способу конструирования и обучения моделей. Вместо этого она дает возможность применять широкий спектр подходов, соответствующих уровню подготовки пользователей, от очень высокого до очень низкого. Например, в вашем распоряжении множество способов конструирования моделей и множество способов их обучения, каждый из которых предлагает определенные компромиссы между удобством и гибкостью. В главе 5 мы подробно рассмотрим значительную их часть. Вы можете использовать Keras так же, как использовали бы Scikit-learn — просто вызывая `fit()` и позволяя фреймворку выполнить свою работу, — или как NumPy, определяя и управляя всеми самыми мелкими деталями.

А значит, все знания, приобретаемые вами в самом начале пути, сохранят актуальность, когда вы станете экспертом. Keras поможет быстро начать работу, а затем постепенно погружаться в рабочие процессы и писать с нуля все больше и больше логики. Вам не придется переключаться на совершенно другой фреймворк, когда вы вырастете от студента до исследователя или от специалиста по данным до инженера глубокого обучения.

Эта философия мало чем отличается от философии самого Python! Некоторые языки (например, объектно-ориентированные или функциональные) дают только один способ написания программ. Python — многопарадигменный язык: он предлагает множество возможных подходов к программированию, прекрасно уживающихся вместе. Это делает Python пригодным для использования в самых разных случаях: для системного администрирования, анализа данных, машинного обучения, веб-разработки... или просто для обучения программированию. Точно так же Keras можно рассматривать как диалект Python для глубокого обучения: удобный язык глубокого обучения, предлагающий множество подходов для пользователей с разным уровнем подготовки.

3.3. KERAS И TENSORFLOW: КРАТКАЯ ИСТОРИЯ

Среда Keras старше TensorFlow на восемь месяцев. Она была выпущена в марте 2015 года, а TensorFlow — в ноябре. Вы можете спросить: если Keras основана на TensorFlow, как она могла появиться раньше? Дело в том, что первоначально Keras основывалась на Theano — еще одной библиотеке для работы с тензорами, обеспечивавшей автоматическое дифференцирование и поддержку вычислений на графических процессорах, самой первой в своем классе. Theano была разработана в Монреальском институте алгоритмов обучения (Montréal Institute

for Learning Algorithms, MILA) при Монреальском университете и во многих отношениях может считаться предшественницей TensorFlow. В ней впервые была реализована идея использования статических графов вычислений для автоматического дифференцирования и компиляции кода для выполнения на CPU и GPU.

В конце 2015 года, после выпуска TensorFlow, архитектура Keras была преобразована для поддержки нескольких базовых библиотек: появилась возможность выбора между Theano и TensorFlow, при этом переключение было таким же простым, как изменение переменной окружения. К сентябрю 2016 года TensorFlow достигла достаточно высокого уровня технической зрелости, чтобы использовать ее в качестве опции по умолчанию. В 2017 году в Keras была добавлена поддержка еще двух библиотек тензорных операций: CNTK (разработана в Microsoft) и MXNet (в Amazon). В настоящее время разработка Theano и CNTK прекратилась, а MXNet не получила широкого распространения за пределами Amazon. Keras снова стала библиотекой, основанной на одном тензорном фреймворке — TensorFlow.

Keras и TensorFlow уже много лет успешно сосуществуют вместе. В течение 2016 и 2017 годов Keras приобрела широкую известность как удобное средство для разработки приложений TensorFlow, привлекающее новых пользователей в экосистему TensorFlow. К концу 2017 года большинство пользователей фреймворка TensorFlow использовали его через Keras или в сочетании с Keras. В 2018 году руководство TensorFlow выбрало Keras в качестве официального высокоуровневого интерфейса TensorFlow. В результате библиотека Keras заняла центральное место в версии TensorFlow 2.0, выпущенной в сентябре 2019 года, — кардинально переделанного комплекса TensorFlow и Keras, учитывающего отзывы пользователей и технический прогресс за предыдущие четыре года.

Теперь вы готовы начать использовать код для Keras и TensorFlow на практике. Приступим.

3.4. НАСТРОЙКА ОКРУЖЕНИЯ ДЛЯ ГЛУБОКОГО ОБУЧЕНИЯ

Прежде чем приступать к разработке приложений глубокого обучения, нужно настроить рабочее окружение. Для выполнения кода, реализующего глубокое обучение, рекомендуется (но это не обязательно) использовать современный графический процессор NVIDIA. Некоторые приложения — в частности, для обработки изображений с применением сверточных сетей — показывают крайне низкую производительность даже на очень быстрых многоядерных CPU. И даже

приложениям, которые вполне могут выполняться на CPU, выполнение на современном GPU часто дает прирост скорости примерно в 5–10 раз.

Есть три варианта настройки окружения для глубокого обучения на графическом процессоре:

- купить и установить на рабочую станцию физический графический процессор NVIDIA;
- использовать экземпляры GPU в Google Cloud или AWS EC2;
- использовать бесплатную среду выполнения на графическом процессоре от Colaboratory — службы для блокнотов Jupiter, поддерживаемой компанией Google (мы рассмотрим эти блокноты подробнее в следующем разделе).

Служба Colaboratory предлагает самый простой способ начать работу: она не требует покупки оборудования и установки программного обеспечения — просто откройте вкладку в браузере и приступайте к программированию. Именно этот вариант я рекомендую для выполнения примеров этой книги. Однако бесплатная версия Colaboratory подходит только для небольших рабочих нагрузок. Для масштабных проектов вам придется использовать первый или второй вариант.

Если у вас еще нет GPU (последней, высокопроизводительной модели NVIDIA GPU), который можно было бы использовать для нужд глубокого обучения, эксперименты с глубоким обучением в облаке — это простой и недорогой способ, не требующий покупки дополнительного оборудования. При использовании Jupyter Notebook работа в облаке ничем не будет отличаться от работы на локальном компьютере.

Однако тем, кто планирует заниматься глубоким обучением всерьез, такой подход не годится — он не подойдет даже новичкам, собирающимся фокусироваться на теме дольше нескольких месяцев. Облачные экземпляры недешевы: один час работы графического процессора V100 в Google Cloud стоил 2,48 доллара. Между тем хороший графический процессор потребительского класса обойдется вам от 1500 до 2500 долларов. Эта цена остается стабильной, она не растет со временем даже при улучшении характеристик GPU. Если вы намерены всерьез заняться глубоким обучением, подумайте об оснащении рабочей станции одним или несколькими GPU.

Кроме того, независимо от окружения, локального или облачного, лучше взять рабочую станцию Unix. Технически библиотеку Keras можно использовать непосредственно в Windows, но я не рекомендую этого. Если у вас Windows и вы хотите заниматься глубоким обучением на собственной рабочей станции, самое простое решение — установить Ubuntu второй операционной системой или использовать подсистему Windows Subsystem for Linux (WSL) — слой совместимости, позволяющий запускать приложения для Linux в Windows.

Может показаться, что это слишком хлопотно, однако подобный подход поможет сэкономить вам массу времени и избавит от многих проблем в будущем.

3.4.1. Jupyter Notebook: предпочтительный способ проведения экспериментов с глубоким обучением

Блокноты Jupyter Notebook — отличный способ проведения экспериментов по глубокому обучению и, в частности, апробации примеров этой книги. Они широко применяются в сообществах машинного обучения и науки о данных. *Блокнот* (notebook) — это файл, сгенерированный приложением Jupyter Notebook (https://jupyter.org), который можно редактировать в браузере. В блокнот можно вставлять код на Python и сопровождать результаты его выполнения примечаниями с богатым оформлением. Блокноты позволяют разбить объемный эксперимент на несколько коротких шагов, выполняемых независимо, что добавляет интерактивности в разработку и избавляет от необходимости повторно запускать предыдущий код, если что-то пошло не так на следующем шаге в эксперименте.

Я настоятельно рекомендую использовать блокноты Jupyter Notebook на первых порах работы с Keras, хотя это и не является обязательным требованием: вы можете также запускать автономные сценарии на Python или выполнять код в интегрированной среде, такой как PyCharm. Все примеры данной книги доступны в виде блокнотов Jupiter с открытым исходным кодом на сайте GitHub: github.com/fchollet/deep-learning-with-python-notebooks.

3.4.2. Использование Colaboratory

Colaboratory (или просто Colab) — это бесплатная облачная служба для блокнотов Jupyter, не требующая установки дополнительного программного обеспечения. По сути, это веб-страница, позволяющая сразу же писать и выполнять сценарии, использующие Keras. Она дает доступ к бесплатной (но ограниченной) среде выполнения на графическом процессоре и даже к среде выполнения на тензорном процессоре (TPU), благодаря чему вам не придется покупать свой GPU. Рекомендую использовать Colaboratory для выполнения примеров данной книги.

Первые шаги с Colaboratory

Для начала работы с Colab откройте страницу https://colab.research.google.com и нажмите кнопку New Notebook (Создать блокнот). Вы увидите стандартный интерфейс блокнота, показанный на рис. 3.2.