SkillFactory

Обзор фреймворков глубокого обучения

Шпилевский Яромир

Ведущий разработчик First Line Software

Agenda

- Keras — верхнеуровневый API для Tensorflow.
- Tensorflow — фреймворк вычисления тензоров.
- Параллельное выполнение и аппаратное ускорение в Tensorflow.
- Распределённое выполнение в Tensorflow.
- PyTorch — и верхнеуровневый API, и фреймворк вычисления моделей.
- Глубокое обучение в sklearn.
- Фреймворки градиентного бустинга: CatBoost, XGBoost, LightGBM.

Keras. API

- Более верхнеуровневый API для Tensorflow.
  - Позволяет быстрее экспериментировать с архитектурой нейронной сети.
- До версии 2.4 поддерживались другие бекенды.
- Начиная с версии 2.4 все усилия направляются на тесную интеграцию с Tensorflow.
- Позволяет оперировать непосредственно слоями:
  - `tf.keras.layers.Dense`
  - `tf.keras.layers.Dropout`
  - `tf.keras.layers.Flatten`

Keras

- Разработчик: François Chollet (Франсуа Шоле).
- OpenSource
  - Лицензия MIT: модификации библиотеки и работы, включающие библиотеку, при необходимости могут распространяться
    - под другими условиями,
    - без публикации исходного кода,
    - в коммерческих целях.
  - Требуется только сохранение упоминание авторства и авторских прав.

Tensorflow

- Фреймворк от Google.
- Разработчик: Google Brain Team.
- OpenSource
  - Лицензия Apache 2.0: модификации библиотеки и работы, включающие библиотеку, при необходимости могут распространяться
    - под другими условиями,
    - без публикации исходного кода,
    - в коммерческих целях.
  - Требуется только сохранение упоминание авторства и авторских прав.
  - Контрибьютор так же передает право на использование своих патентов.

Tensorflow. Поддержка GPU и TPU

- Тесно интегрированный программно-аппаратный стек.
- Заранее скомпилированный бинарный код для каждой операции. С использованием аппаратного ускорения.
- Google часто называет его платформой.
- Поддержка GPU и TPU.
- В Colab есть всё необходимое окружение.
- Можно также работать в локальном окружении:
  - Поддержка вычислений на GPU NVIDIA через CUDA присутствует «из коробки»:

```
pip install tensorflow
```

  - Облегчённый пакет, если нужны только вычисления на CPU:

```
pip install tensorflow-cpu
```

Может использоваться для IoT решений.

Tensorflow. XLA

- Accelerated Linear Algebra.
- Альтернативный способ трансляции.
- По принципу Just-in-Time (JIT) компиляции.
- Транслирует граф вычислений данной конкретной модели.
- Может проводить оптимизации, специфичные для модели.
- Документация. [1]

Tensorflow. Параллельность

- Продумана архитектура параллельности.
- Решает 3 больших вопроса:
  - Физическая конфигурация.
  - Логическая конфигурация.
  - Синхронное vs асинхронное обучение.
- Есть понятие «стратегия» распределённого вычисления.
- Доступно несколько стратегий.
- Каждая стратегия – определённые принятые решения по конфигурации.

Tensorflow. Параллельность. Физическая конфигурация

- Основные понятия:
  - Вычислительный кластер (Cluster) – несколько хостов.
  - Хост (Host) – физический сервер.
  - Устройство (Device) – устройство на хосте (cpu:3, gpu:1, tpu:0 и т.п.)

Tensorflow. Параллельность. Логическая конфигурация

- Основные понятия:
  - Parameter Server – сервер параметров обучения.
    - Используется для асинхронного обучения.
  - Worker – производит вычисления для обучения.
- Объект класса `tf.train.ClusterSpec` – хранит логическую конфигурацию.

Tensorflow. Синхронное vs асинхронное обучение

- Синхронное обучение – каждый worker обучается на своей части обучающей выборки, агрегируя градиенты.
- Асинхронное обучение – worker может пересекаться с другим worker по данным обучающей выборки. Обновляют параметры обучения независимо.
    - Для этого нужен сервер параметров обучения.

Tensorflow. Параллельность. Паттерны проектирования

- Object-oriented patterns:
  - Interface – implementation pattern.
    - Отделяем интерфейс от реализации.
    - Интерфейс определяет сигнатуры методов и пр. (и, в более общем виде, «контракт» взаимодействия).
    - Детали реализации определяются конкретной реализацией интерфейса.
  - Strategy pattern.
    - Используется, когда нужно передать объект с какой-либо логикой в другой компонент.
    - Частный случай паттерна «интерфейс – реализация».
    - Интерфейс определяет контракт взаимодействия компонентов, а разные реализации стратегий могут по-разному реализовывать этот контракт.

Tensorflow. Параллельность. Архитектура стратегий

- `tf.distribute.Strategy` – интерфейс стратегии распараллеливания.
- Есть несколько реализаций стратегии распараллеливания.

Tensorflow. Параллельность. Стратегии

- `ParameterServerStrategy`
  - Асинхронная.
  - Используем сервер параметров.
  - Поддерживает гетерогенность (какие-то worker работают на CPU, какие-то на GPU).
  - Неизвестно, поддерживает ли так же TPU.
  - Экспериментальная стратегия. Активно разрабатывается.

Tensorflow. Параллельность. Стратегии

- `MirroredStrategy`
  - Синхронная.
  - Использовать несколько GPU одного хоста.
  - Каждый параметр обучения зеркалируется на каждый GPU (точнее, в его GRAM).
  - Обучение на конкретном GPU настраивает своё подмножество параметров, которое зависит от обрабатываемой части обучающей выборки.
- `MultiWorkerMirroredStrategy`
  - Синхронная.
  - Похожа на `MirroredStrategy`, размножена на несколько хостов.
  - Зеркалирование происходит так же и между хостами.

Tensorflow. Параллельность. Стратегии

- `CentralStorageStrategy`
  - Синхронная.
  - Параметры обучения в RAM, т.е. под управлением CPU. Альтернатива их зеркалированию в GRAM каждого GPU.
  - Вычислительные задачи отправляются на GPU.

Tensorflow. Параллельность. Стратегии

- `TPUStrategy`
  - Синхронная
  - Похожа на `MirroredStrategy`, только вместо GPU используются TPU.
  - Неизвестно, только ли в рамках одного хоста или поддерживает так же зеркалирование между хостами.
  - Доступна только в облачных окружениях:
    - Colab
    - TensorFlow Research Cloud
    - Cloud TPU

PyTorch

- numpy – «низкоуровневая» реализация численных методов.
- PyTorch – более высокоуровневые понятия, построенные на численных методах.
- Может использоваться для создания ещё более высокоуровневых вещей.
    - Например, FAIR (Facebook AI Research) Detectron 2 построен на PyTorch.

sklearn

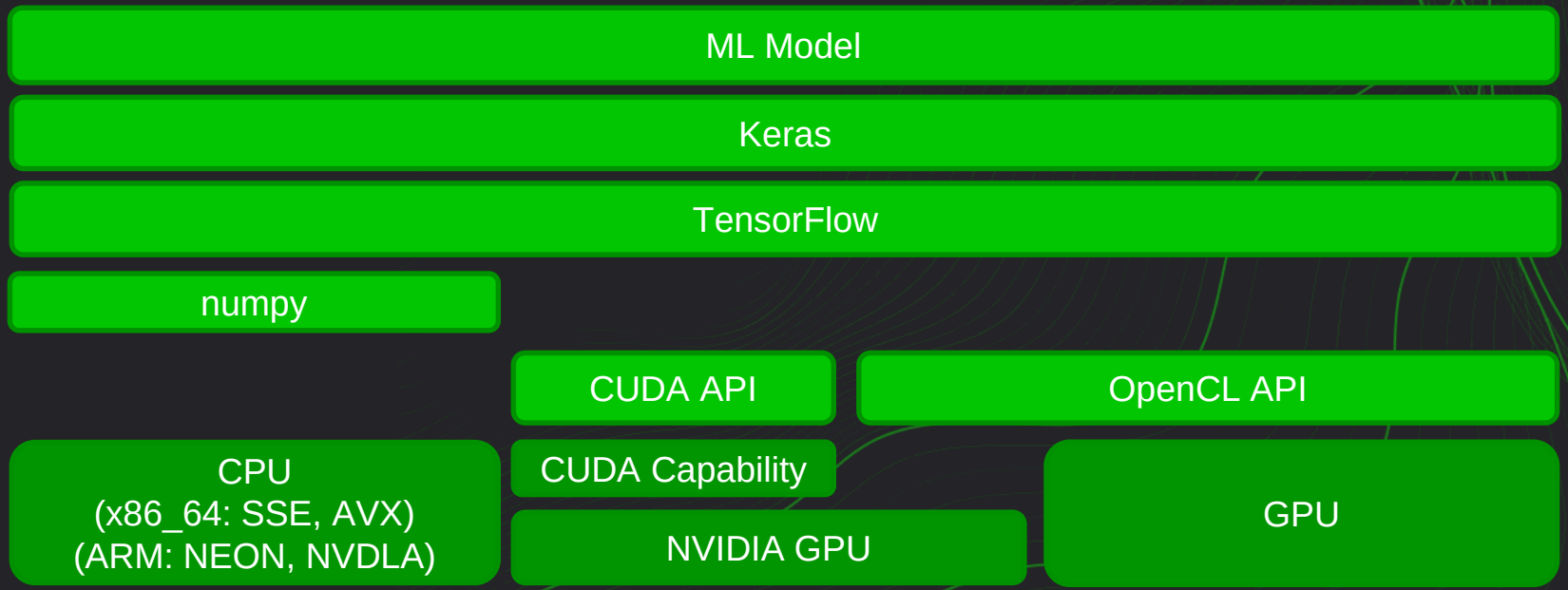
- Базовые возможности для глубокого обучения.
- Тем не менее, что-то есть.
  - Например, MLPClassifier.
- К сожалению, нет поддержки аппаратного ускорения на GPU (и на TPU).
- Хорошие вспомогательные функции для оценки моделей других фреймворков.
  - Например, classification_report.

Немного про градиентный бустинг

- Непосредственно градиентный бустинг не относится к глубокому обучению.
- Тем не менее, часто является составной частью сложных моделей.
- Используется ансамбль предсказателей.
- Bagging (bootstrap aggregating) – агрегация результатов ансамблей каким-либо методом (среднее, взвешенное среднее, голосование (для дискретных значений)).
- Boosting – предсказатели из ансамбля построены не независимо, а один на основе другого.
  - CatBoost (Yandex)
  - XGBoost
  - LightGBM (Microsoft)

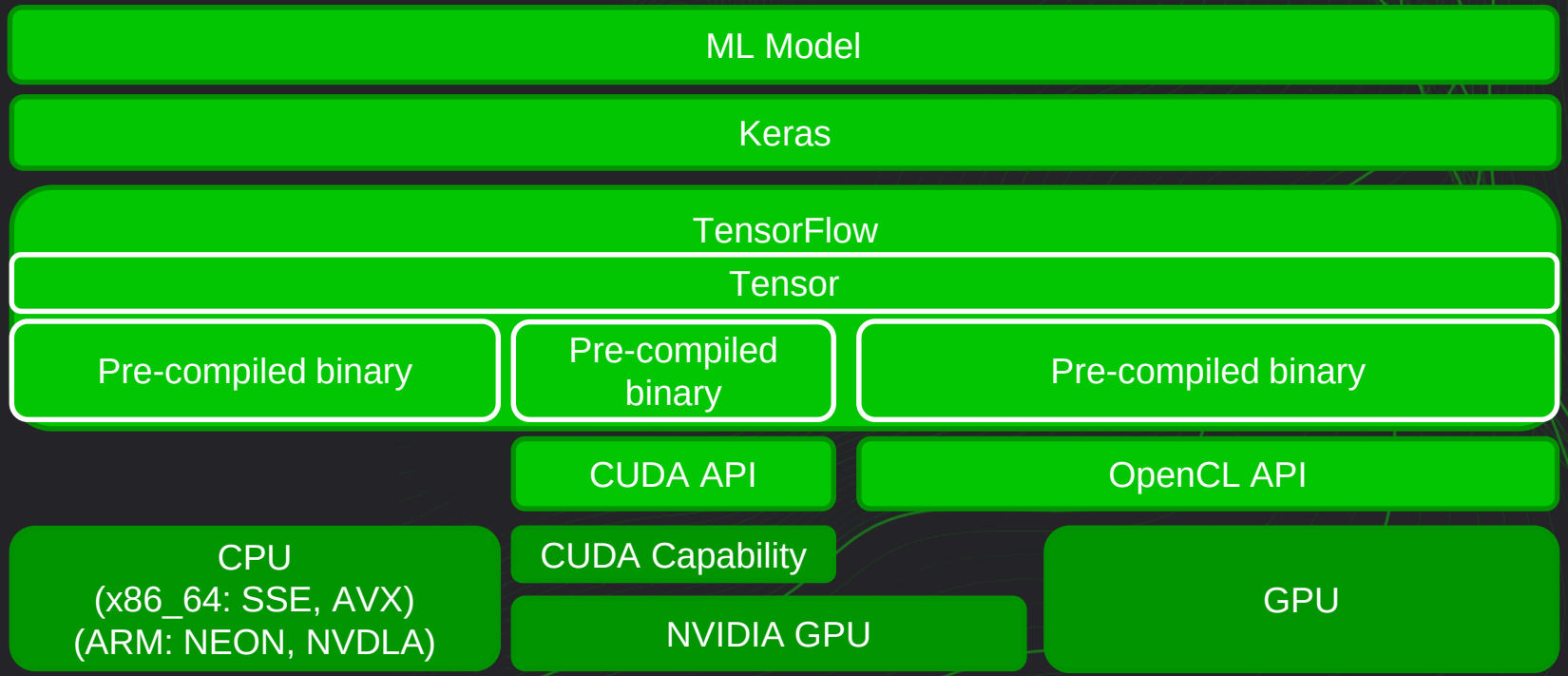
Программно-аппаратный стек

- На протяжении курса будем периодически возвращаться к этой картине.
- Исходная реализация.



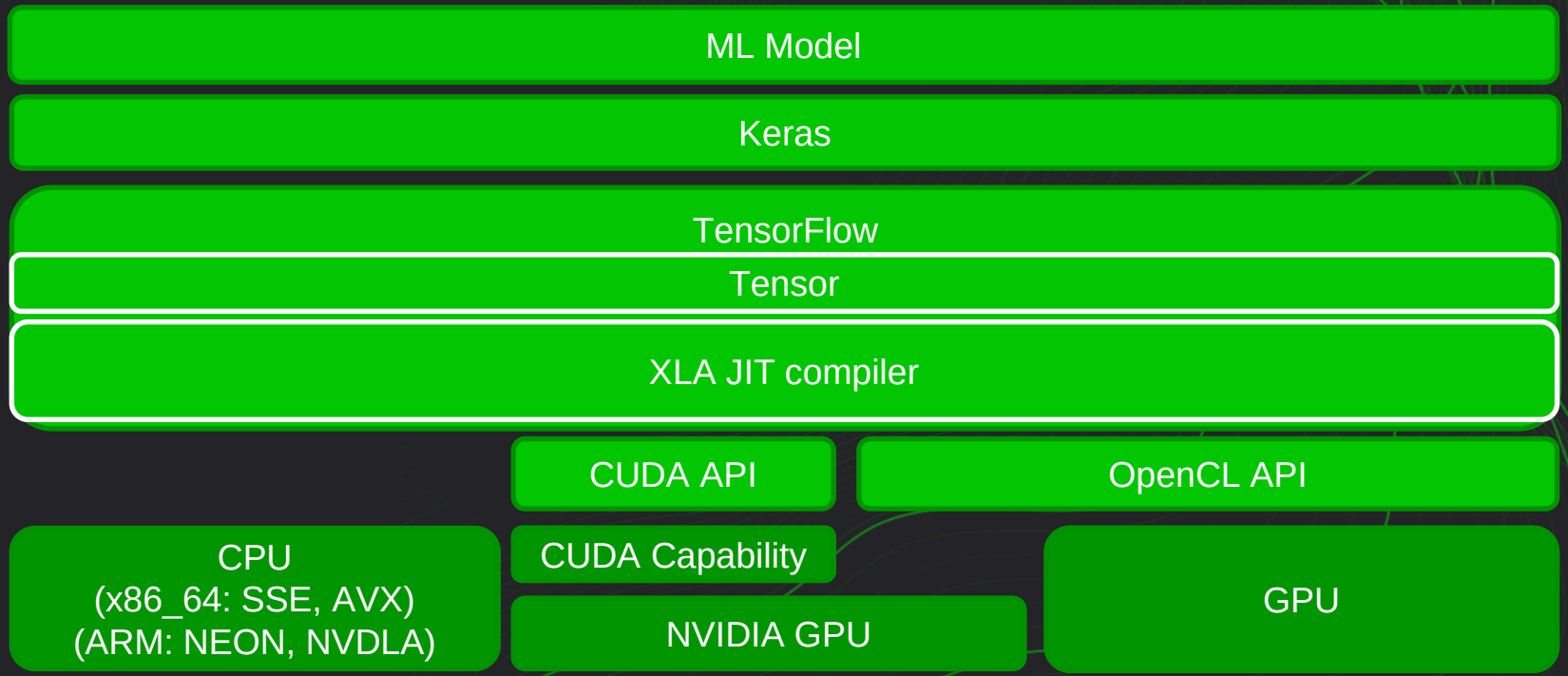
Программно-аппаратный стек

- Реализация с заранее скомпилированным бинарным кодом, использующим аппаратное ускорение.



Программно-аппаратный стек

- Реализация с XLA.



Ссылки

1) Документация XLA:
   - https://www.tensorflow.org/xla

Резюме

- Рассмотрены различные фреймворки:
  - Tensorflow + Keras.
  - PyTorch.
- Рассмотрены возможности глубокого обучения в sklearn.
- Рассмотрены фреймворки градиентного бустинга:
  - CatBoost.
  - XGBoost.
  - LightGBM.

Вопросы для самоконтроля

- Какие функциональные слои можно выделить во фреймворках глубокого обучения?
- Что реализует каждый такой слой:
  - В связке Tensorflow + Keras?
  - В PyTorch?
  - В sklearn?
- Какие фреймворки для градиентного бустинга вы можете назвать?

Спасибо!