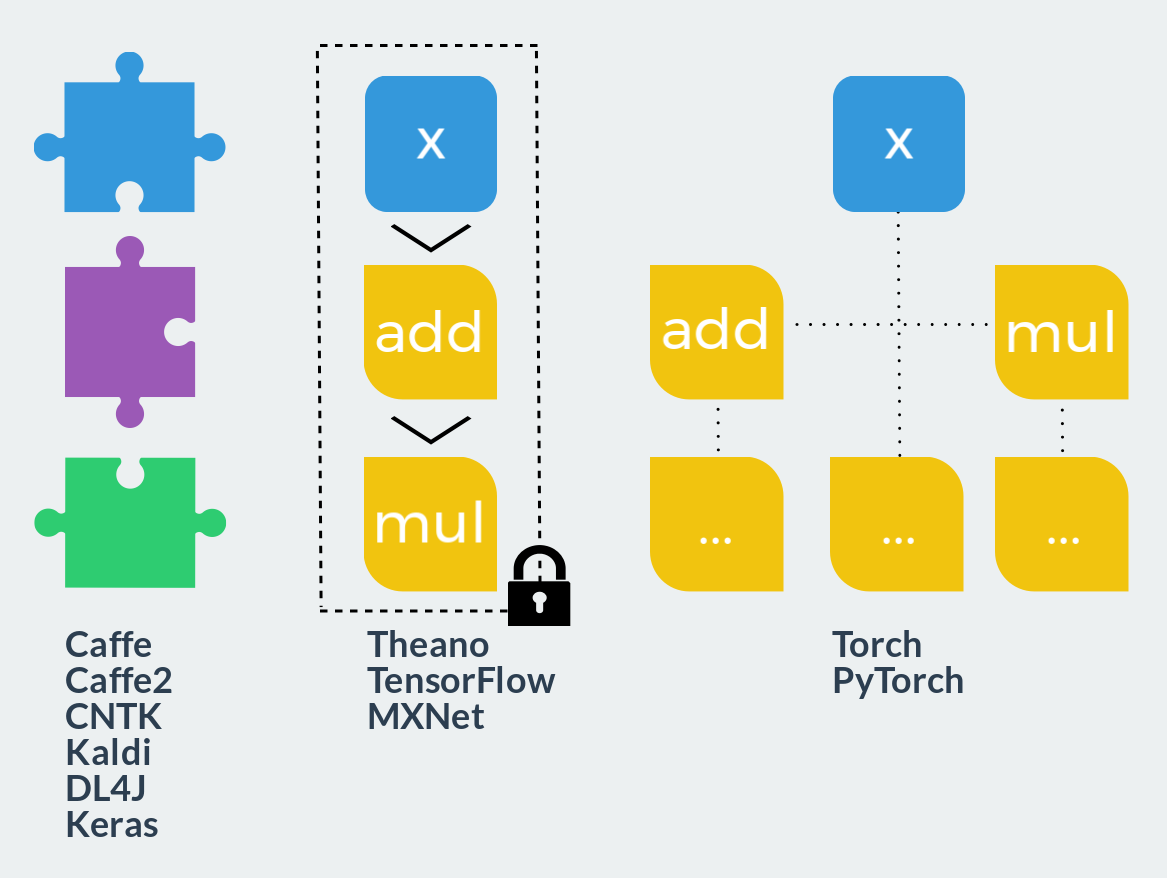
**Обзор фреймворков**

Все фреймворки здесь можно разделить на три крупные категории.

*Фиксированные модули*. Такой подход можно сравнить с конструктором Lego: пользователь комбинирует заранее определённые блоки в граф вычислений и запускает его. Прямой и обратный проходы уже зашиты в каждом таком блоке. Определение новых блоков гораздо сложнее использования готовых и требует совершенно иных знаний и умений. Расширяемость близка к нулю, однако если ваши идеи полностью реализуются в таком фреймворке, скорость разработки максимальна. Со скоростью работы, благодаря высокой оптимизированности заранее написанного кода, также не возникает проблем. Типичные представители: Caffe, Caffe2, CNTK, Kaldi, DL4J, Keras (как интерфейс).

*Статический граф вычислений*. Эти фреймворки уже можно сравнить с полимерной глиной: на этапе описания возможно создать граф вычислений произвольного размера и сложности, однако после компиляции он станет твёрдым и монолитным. Доступными останутся всего два действия: запустить граф в прямом или обратном направлениях. Все такие фреймворки используют декларативный стиль программирования и напоминают функциональный язык или математическую нотацию. С одной стороны, этот подход комбинирует гибкость на этапе разработки и скорость в момент исполнения. С другой стороны, как и в функциональных языках, отладка становится настоящей головной болью, а модели, выходящие за рамки парадигмы, требуют либо титанических усилий, либо здоровенных костылей для реализации. Представители: Theano, TensorFlow, MXNet.

*Динамический граф вычислений*. Представьте теперь, что вы можете перестраивать статический граф перед каждым его запуском. Примерно это и происходит в данном классе фреймворков. Только графа как отдельной сущности здесь нет. Граф строится динамически каждый раз при прямом проходе для того, чтобы затем иметь возможность сделать проход обратный. Подобный подход даёт максимальную гибкость и расширяемость, позволяет использовать в вычислениях все возможности используемого языка программирования и не ограничивает пользователя вообще ничем. К этому классу фреймворков как раз и относятся Torch и PyTorch.

**Библиотека Torch**

Torch — библиотека для научных вычислений с широкой поддержкой алгоритмов машинного обучения. Библиотека реализована на языке Lua с использованием C и CUDA. Быстрый скриптовый язык Lua в совокупности с технологиями SSE, OpenMP, CUDA позволяют Torch показывать неплохую скорость по сравнению с другими библиотеками. На данный момент поддерживаются операционные системы Linux, FreeBSD, Mac OS X. Основные модули также работают и на Windows. В зависимостях Torch находятся пакеты imagemagick, gnuplot, nodejs, npm и другие.

Библиотека состоит из набора модулей, каждый из которых отвечает за различные стадии работы с нейросетями. Так, например, модуль nn обеспечивает конфигурирование нейросети (определению слоев, и их параметров), модуль optim содержит реализации различных методов оптимизации, применяемых для обучения, а gnuplot предоставляет возможность визуализации данных (построение графиков, показ изображений и т.д.). Установка дополнительных модулей позволяет расширить функционал библиотеки.

Torch позволяет создавать сложные нейросети с помощью механизма контейнеров. Контейнер — это класс, объединяющий объявленные компоненты нейросети в одну общую конфигурацию, которая в дальнейшем может быть передана в процедуру обучения. Компонентом нейросети могут быть не только полносвязные или сверточные слои, но и функции активации или ошибки, а также готовые контейнеры.

**Библиотека PyTorch**

PyTorch - это пакет машинного обучения Python на основе Torch, представляющий собой пакет машинного обучения с открытым исходным кодом, основанный на языке программирования Lua. PyTorch имеет две основные функции:

- Тензорные вычисления (например, NumPy) с сильным ускорением GPU

- Автоматическое дифференцирование для построения и обучения нейронных сетей

В отличие от других библиотек, таких как TensorFlow, где вам нужно сначала определить весь вычислительный граф, прежде чем вы сможете запустить свою модель, PyTorch позволяет вам определять свой граф динамически.

Тензоры PyTorch очень похожи на массивы NumPy с тем дополнением, что они могут работать на GPU. Это важно, потому что это помогает ускорить численные вычисления, которые могут увеличить скорость нейронных сетей в 50 и более раз .PyTorch глубоко интегрирован с кодом C++, таким образом, пользователи могут программировать на C/C ++ с помощью API расширения на основе c FFI для Python. Функциональность PyTorch построена как классы Python, поэтому весь его код может легко интегрироваться с пакетами и модулями Python. Он позволяет связывать высокоуровневые нейронные сетевые модули, потому что он поддерживает Keras-like API в своем пакете torch.nn.

PyTorch в основном используется для быстрого и эффективного обучения моделей глубокого обучения, поэтому он является основой выбора для большого числа исследователей.

У этого есть некоторые существенные преимущества:

(+) Процесс моделирования прост и прозрачен благодаря архитектурному стилю фреймворка;

(+) Режим определения по умолчанию больше похож на традиционное программирование, и вы можете использовать общие средства отладки, такие как отладчик pdb, ipdb или PyCharm;

(+) Имеет декларативный параллелизм данных;

(+) Легко писать свои собственные типы слоев и работать на GPU

(+) Имеет много предварительно обученных моделей и модульных частей, которые легко объединяются;

Но некоторые недостатки, конечно, неизбежны:

(-) Не хватает модельной порции

(-) В нем отсутствуют интерфейсы для мониторинга и визуализации, такие как Tensorboard, хотя вы можете подключиться к Tensorboard извне.

(-) Нет коммерческой поддержки

(-) Неясная документация

И важно помнить: PyTorch все еще относительно нов, и он лучше всего подходит для создания прототипов или небольших проектов.

**Библиотека Theano**

Theano — это библиотека Python и оптимизирующий компилятор, которые позволяют определять, оптимизировать и вычислять математические выражения эффективно используя многомерные массивы  
Возможности библиотеки:

* тесная интеграция с NumPy;
* прозрачное использование GPU;
* эффективное дифференцирование переменных;
* быстрая и стабильная оптимизация;
* динамическая генерация кода на C;
* расширенные возможности юнит-тестирования и самопроверок;

Плюсы и минусы

(+) Python + Numpy

(+) Вычислительный граф - хорошая абстракция

(+) RNNs хорошо вписываются в вычислительный граф

(-) Сырой Theano несколько низкоуровневый

(+) Обертки высокого уровня (Keras, Lasagne) облегчают разработку

(-) Сообщения об ошибках могут быть бесполезны

(-) Большие модели могут иметь длительное время компиляции

(-) гораздо «толще», чем Torch

(-) Пятнистая поддержка для предварительно обученных моделей

(-) Багги на AWS

(-) Одиночный графический процессор

**Библиотека Keras**

Это минималистичная библиотека на основе Python, которую можно запускать поверх TensorFlow, Theano или CNTK. Он поддерживает широкий спектр слоев нейронных сетей, таких как сверточные слои, рекуррентные слои или плотные слои.

Преимущества:

(+) Прототипирование действительно быстро и просто;

(+) легкий с точки зрения построения моделей DL с большим количеством слоев;

(+) имеет полностью настраиваемые модули;

(+) упрощенный и интуитивный интерфейс

(+) имеет встроенную поддержку для обучения на нескольких графических процессорах;

(+) Его можно превратить в оценщики Tensorflow и обучить на кластерах графических процессоров в Google Cloud;

(+) может быть запущен на Spark;

(+) Он поддерживает графические процессоры NVIDIA, Google TPU и графические процессоры с поддержкой Open-CL, такие как AMD.

Недостатки:

(-) не всегда легко настроить;

(-) ограничен бэкэндами Tensorflow, CNTK и Theano, Deeplearning4j.

(-) не обеспечивает столько функций, как TensorFlow, и обеспечивает меньший контроль над сетью, поэтому это может быть серьезным ограничением, если вы планируете построить специальный тип модели DL.

**Библиотека Deeplearning4j**

Это коммерческая среда с открытым исходным кодом, написанная в основном для Java и Scala, предлагающая мощную поддержку для различных типов нейронных сетей (таких как CNN, RNN, RNTN или LTSM).

Преимущества:

Это надежный, гибкий и эффективный;

Он может обрабатывать огромные объемы данных, не жертвуя скоростью;

Он работает с Apache Hadoop и Spark, поверх распределенных процессоров или графических процессоров;

Документация ясная;

Он имеет версию для сообщества и корпоративную версию.

Поскольку Java не очень популярна среди проектов машинного обучения, сама структура не может полагаться на растущие кодовые базы.