Создание инструментов для машинного обучения на MPL

Докладчик:
Минкашев Талгат Наильевич, 19.Б09-мм

Научный руководитель:
доцент кафедры СП, к.т.н. Т.А. Брыксин

Введение I

- популярность машинного обучения: много данных, доступность вычислительных мощностей
- нейронные сети: быстрое, дешевое и эффективное решение

Введение II

- Lei — интеллектуальная система для имитации поведения человека в интернете
- MPL — язык программирования общего назначения
- после переписывания кода Lei с C++ на MPL скорость работы увеличилась многократно
- для Lei понадобился классификатор изображений

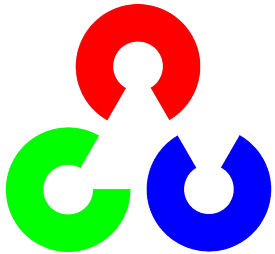
Постановка задач

1) изучить MPL и испытать его на новом классе задач
2) проанализировать существующие решения
3) реализовать структуру для хранения данных
4) реализовать загрузку изображений
5) спроектировать модуль автодифференцирования и реализовать построение графа функции

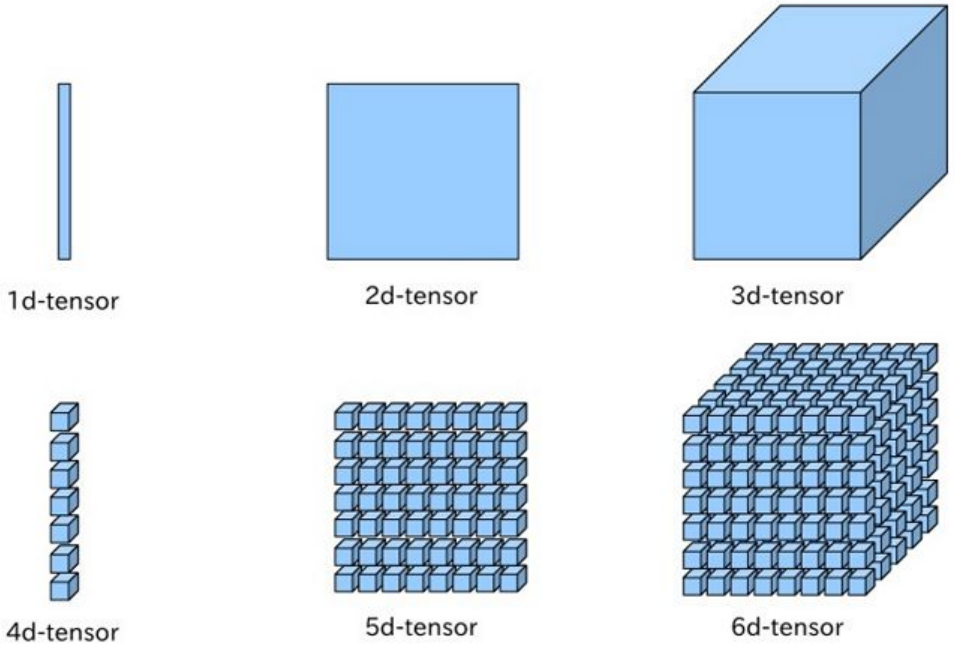
Аналоги

- TensorFlow, PyTorch — большие многофункциональные фреймворки
- OpenCV + DNN — компьютерное зрение с использованием нейросетей
- OpenNN — готовые типы для создания нейросетей

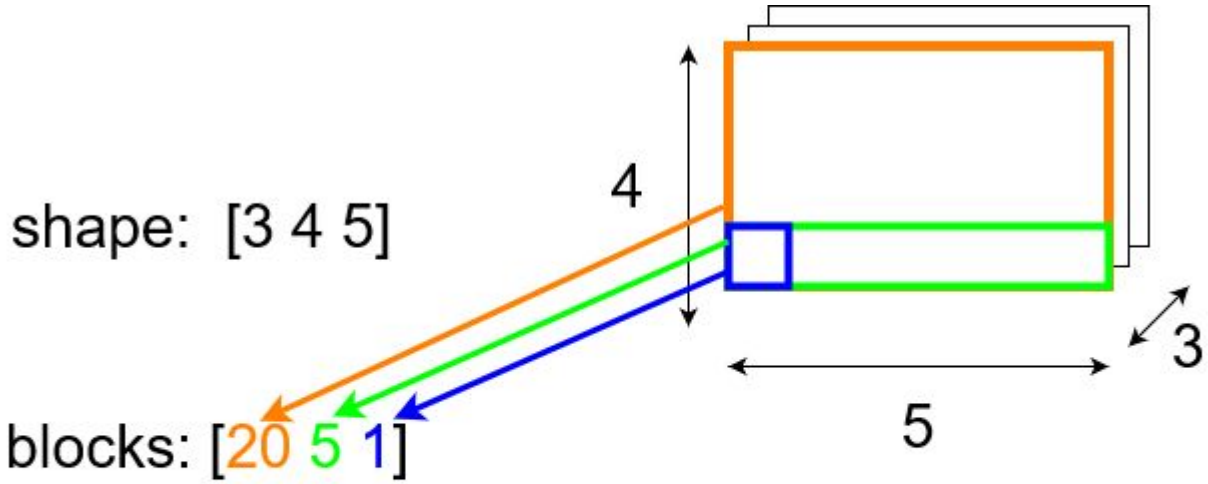





Устройство тензора



Реализация Tensor: хранение данных, индексация



Мультииндекс

(2, 3, 4)

индекс во внутреннем массиве:

20 × 2 + 5 × 3 + 1 × 4 = 59

Реализация Tensor: операции

- операции транспонирования, сечения и изменения формы со сложностью O(rank) ≈ O(1)
- преобразование в строку
- поэлементные бинарные операции
- получение информации о тензоре (ранг, кол-во элементов и т.д.)
- потенциал для модификации и добавления операций

Загрузка изображений

- модуль imageLoader
- использование stb_image для загрузки изображений
- структура Image
- функция преобразования Image в Tensor

Автоматическое дифференцирование

- необходимо для обучения модели
- должно быть быстрым, точным и масштабируемым
- для реализации строится граф вычислений функции

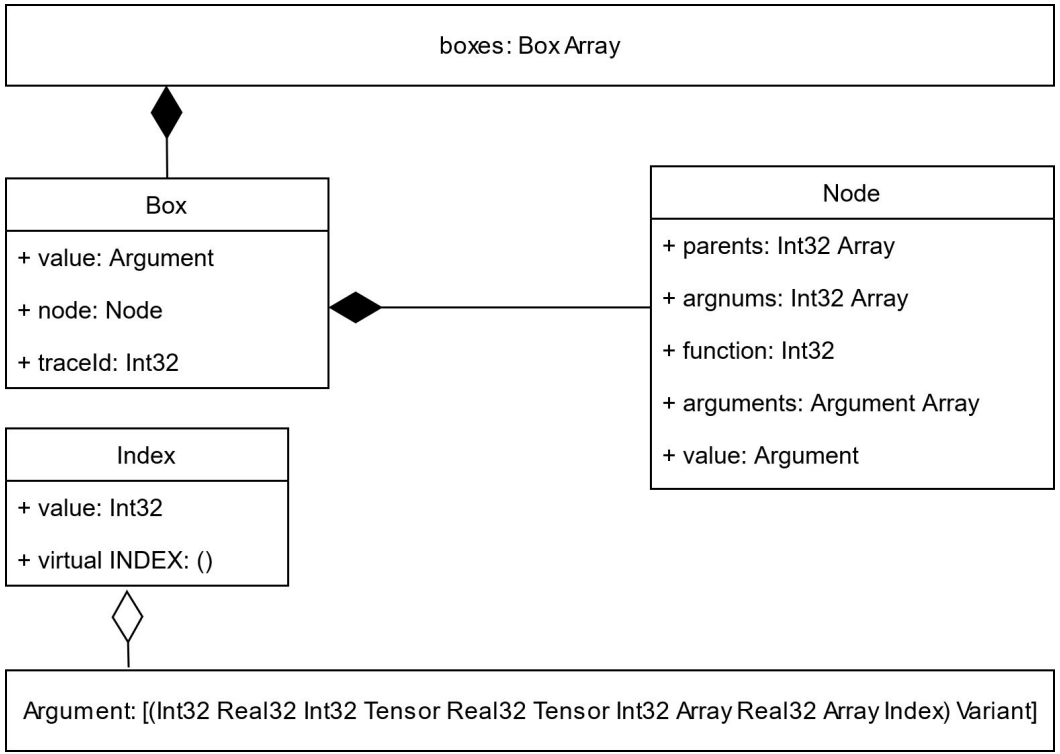
Подход к реализации автоматического дифференцирования

- выбран метод перегрузки операторов (проще в реализации и использовании)
- выбран метод обратного распространения ошибки (быстрее для классификации изображений)

Построение графа вычислений

- модуль tracer
- для нескольких примитивных функций созданы перегруженные варианты
- для хранения данных в графе был создан Box
- для представления вершин был создан Node
- для хранения данных разного типа использован Variant

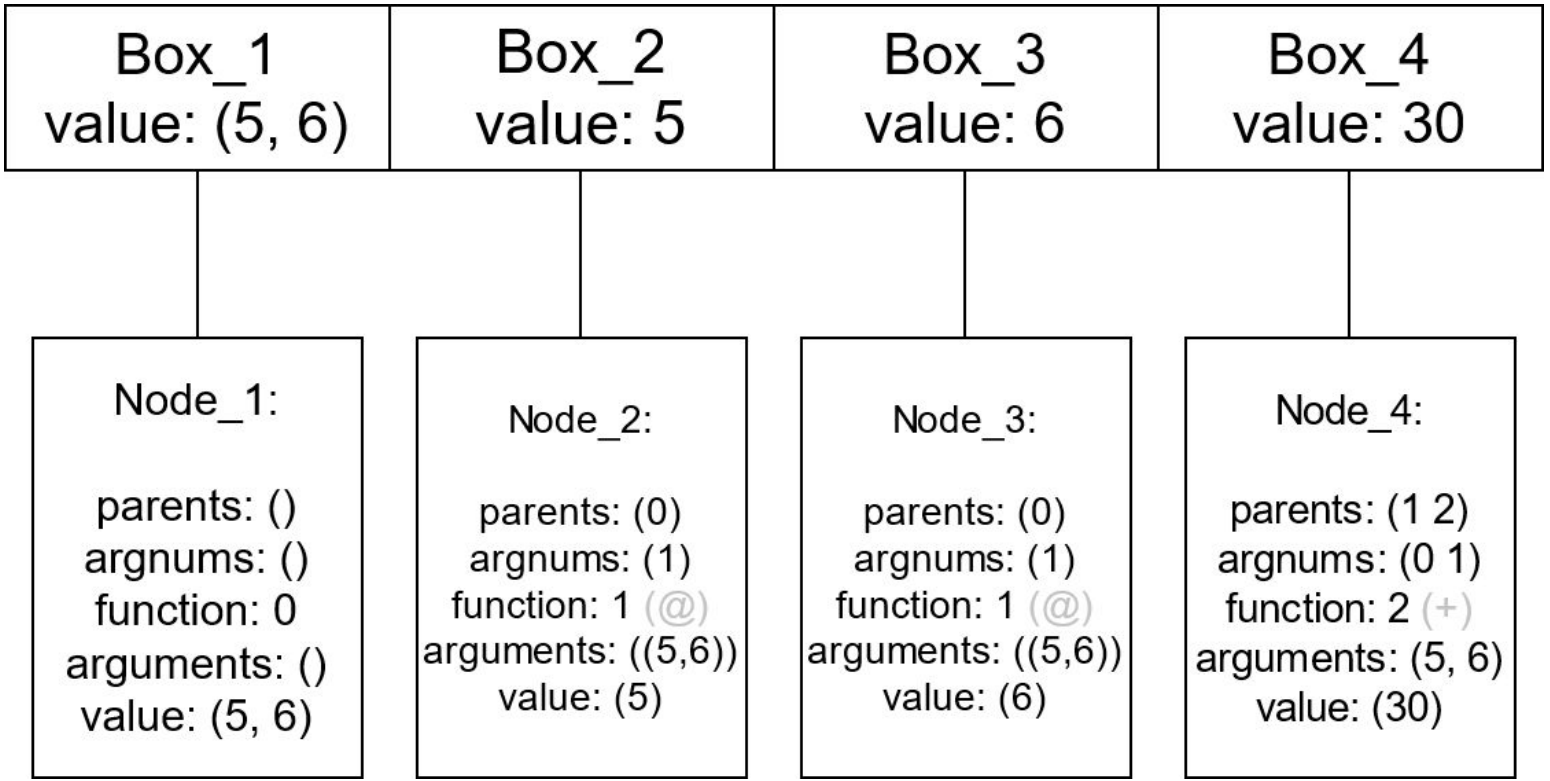
Устройство графа вычислений



Определяем функцию для дифференцирования

```
func: [                 # задаем функцию
    x:;                 # считываем аргумент
    a: 0 @x @   # присваиваем a значение 0-го элемента x
    b: 1 @x @   # присваиваем b значение 1-го элемента x
    @a @b *     # возвращаем результат умножения a на b
];
```

@func (5 6) toArray trace # передаем в trace func и (5 6) => Box_4



Тестирование

- юнит-тесты для Tensor
- протестированы основные сценарии использования реализованных возможностей tracer
- imageLoader протестирован вручную

Заключение

1. проанализированы аналоги
2. реализован класс Tensor
3. реализован модуль imageLoader
4. выбраны методики реализации автоматического дифференцирования
5. реализован модуль tracer
6. протестирована функциональность