IVA TPU
вычислители нейронных сетей

Neural Network Processing
Solutions IVA TPU

+7 (495) 134-66-77
www.iva-tech.ru
www.en.iva-tech.ru

НИР

Математика:
- Разрядность
- Прореживание
- Распределение

Ресурсы:
- Количество УУС
- Размер памяти
- Быстродействие памяти

Архитектура:
- Концепции
- Проверки

Квантирование:
- Типизация данных
- Контроль качества

Эффективность:
- Архитектуры
- Компилятора

Модели:
- Функциональные
- Побитовые
- Потактовые

Our research

Math:
- Precision
- Pruning
- Sparsity

HW Constraints:
- Number of MACs
- Memory Size
- Memory Bandwidth

Architecture search:
- Concept
- Modeling

Quantization:
- Prepare data types
- Validate quality

Performance estimator:
- Architecture trade-offs
- Compiler support

Model:
- Functional
- Bit accurate
- Time accurate

ОКР

СФБ IVA TPU

УПРАВЛЕНИЕ

Матрицы	Векторы	ПДП
Формирование	Формирование	Каналы
Умножение	Pooling+DWS	
Скаляры	Скаляры	

ВНУТРЕННЯЯ ПАМЯТЬ

ПЛИС

Приборный

Центральный

Development

IVA TPU IP

Control Unit

Matrix Engine	Vector Engine	Memory System
Tensor Shaper	Tensor Shaper	Multi-Channel DMA
Matrix Multiplier	Pooling+DWS	
Element-Wise Processor	Element-Wise Processor	

Shared Memory

Soft

Edge

Cloud

Квантование

Цель:

- Уменьшение разрядности операций
- Сохранение точности вычислений

Способ 1: Пороги зависят от разброса

- int8 даёт достаточную точность для большинства сетей

Способ 2: Надпороговое обучение

- Радикально повышает производительность
- Многие сети работают с 5-6 битными операциями

Квантование int8

Сеть	TOP1	TOP1	Diff
VGG 19	66.41	66.39	**-0.02**
ResNet-50	70.76	70.24	**-0.52** (+0.02)
GoogleNet	69.99	69.29	**-0.70**
MobileNet v1	70.99	69.64	**-1.35**
MobileNet v2	71.92	71.14	**-0.78** (-0.01)
PNASNet Large	82.49	82.45	**-0.04**
PNASNet Mobile	72.08	70.56	**-1.52** (-0.09)
DeepSpeech v1 (WER)	0.1545	0.1561	**0.0016**

Our research: quantization

Goal:

- Reduce data bit width
- Keep accuracy

Method 1: Thresholds selection based on data ranges

- Works with int8 on the most networks with suitable accuracy

Method 2: Thresholds training:

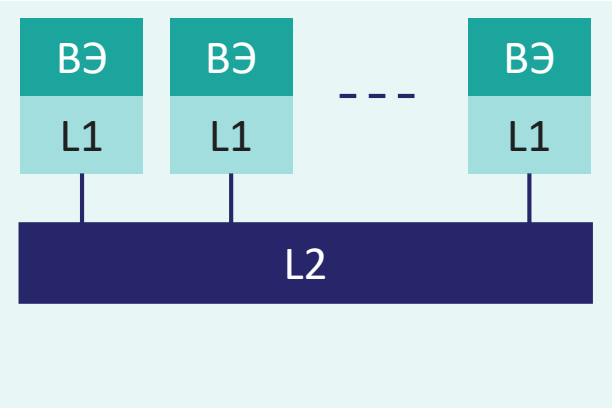
- Dramatically increase performance
- Many networks work with 5-6 bit data

Int8 quantization results:

Name	TOP1	TOP1	Diff
VGG 19	66.41	66.39	**-0.02**
ResNet-50	70.76	70.24	**-0.52 (+0.02)**
GoogleNet	69.99	69.29	**-0.70**
MobileNet v1	70.99	69.64	**-1.35**
MobileNet v2	71.92	71.14	**-0.78 (-0.01)**
PNASNet Large	82.49	82.45	**-0.04**
PNASNet Mobile	72.08	70.56	**-1.52 (-0.09)**
DeepSpeech v1 (WER)	0.1545	0.1561	**0.0016**

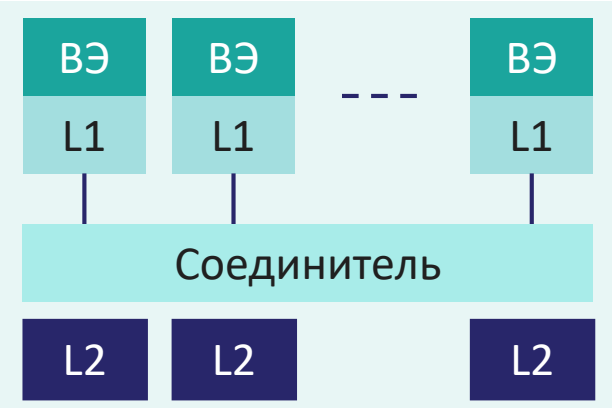
Пересылки

CPU



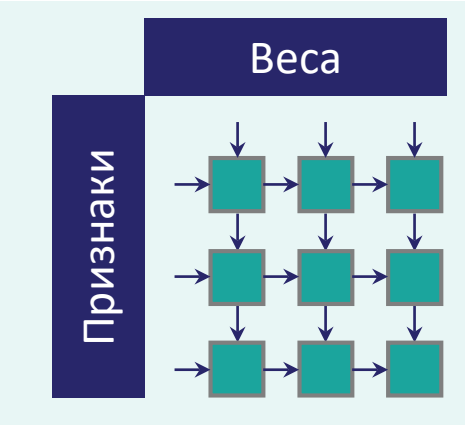
- Эффективность мала
- Задачи произвольные
- **Десятки ВЭ**

GPU



- Эффективность улучшена
- Задачи ограничены
- **Сотни ВЭ**

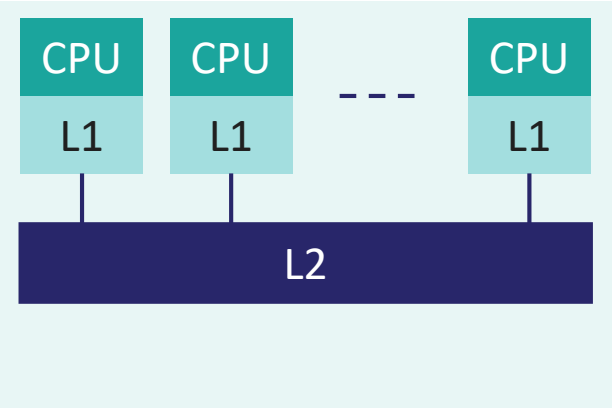
TPU



- Эффективность максимальна
- Нейросети
- **Тысячи ВЭ**

Math Engines

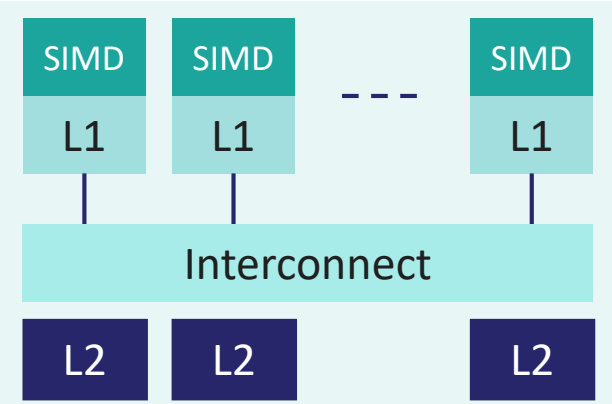
CPU



- Executes general purpose code of programs and applications
- Low computation speed

100s Cores

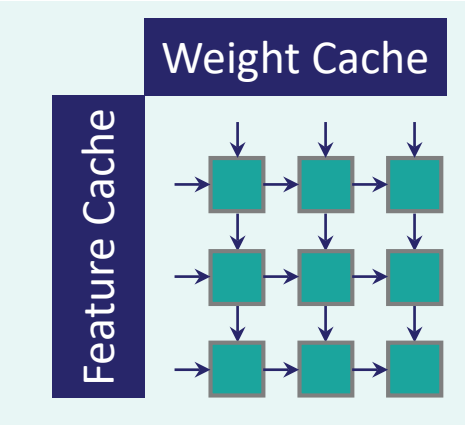
GPU



- Performs wide range of computational tasks, not optimized for work with neural networks
- Better scalability
- High interconnect load and external memory bandwidth

1000s Cores

TPU

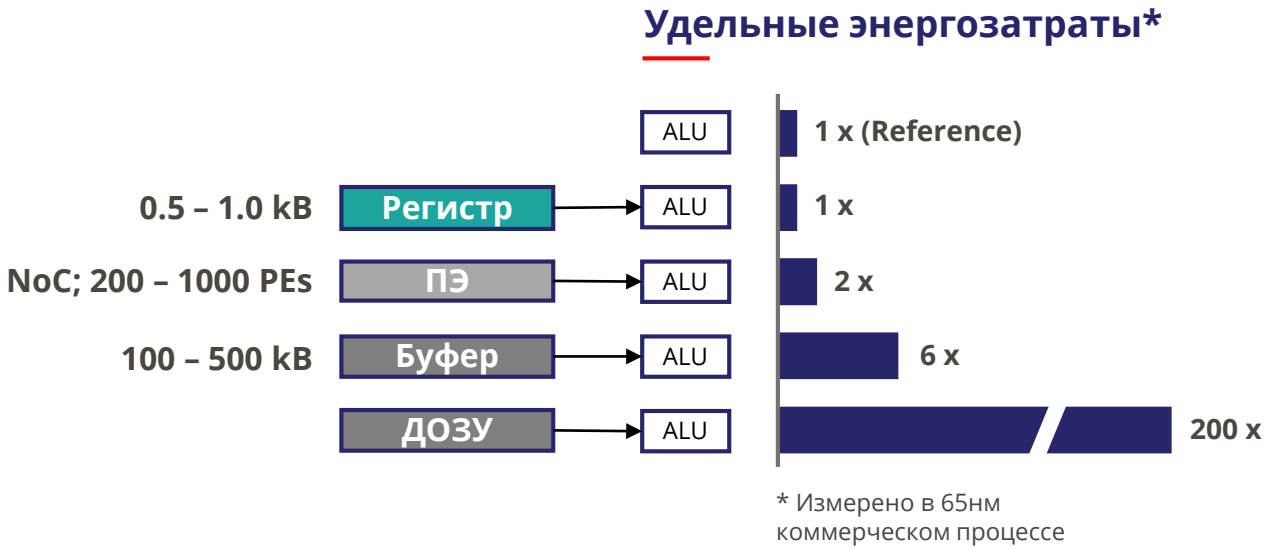


- Maximum computational performance on the neural network
- Low power consumption

10000s Cores

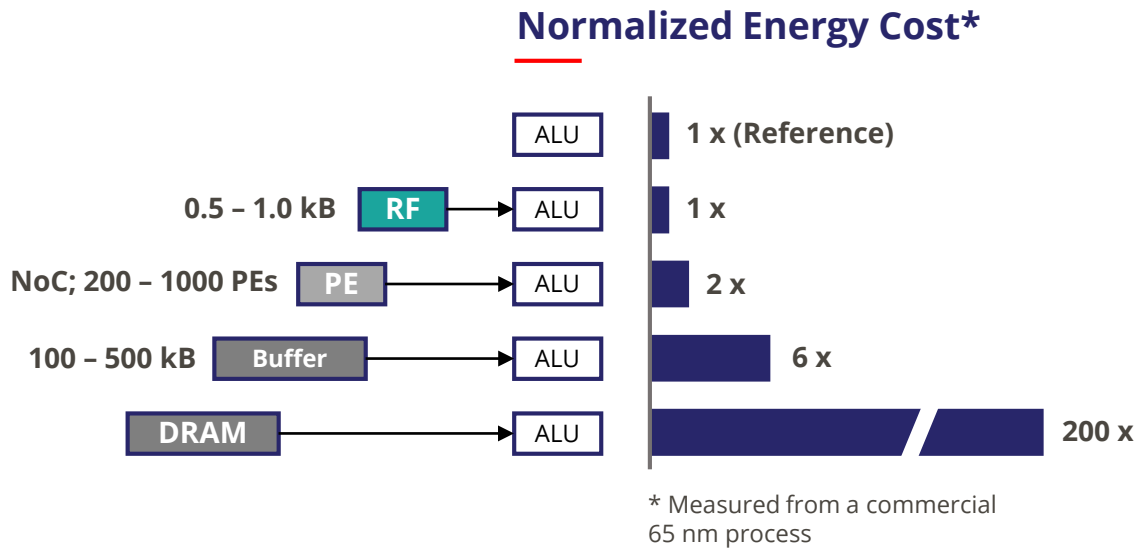
Ограничения памяти

- Вычисления быстрее пересылок
- Доступ в память наиболее энергозатратная операция
- Крайне важно оптимальное управление (как кеширование)
- Требуется баланс
  - быстродействия памяти
  - размера кеша
  - энергозатрат



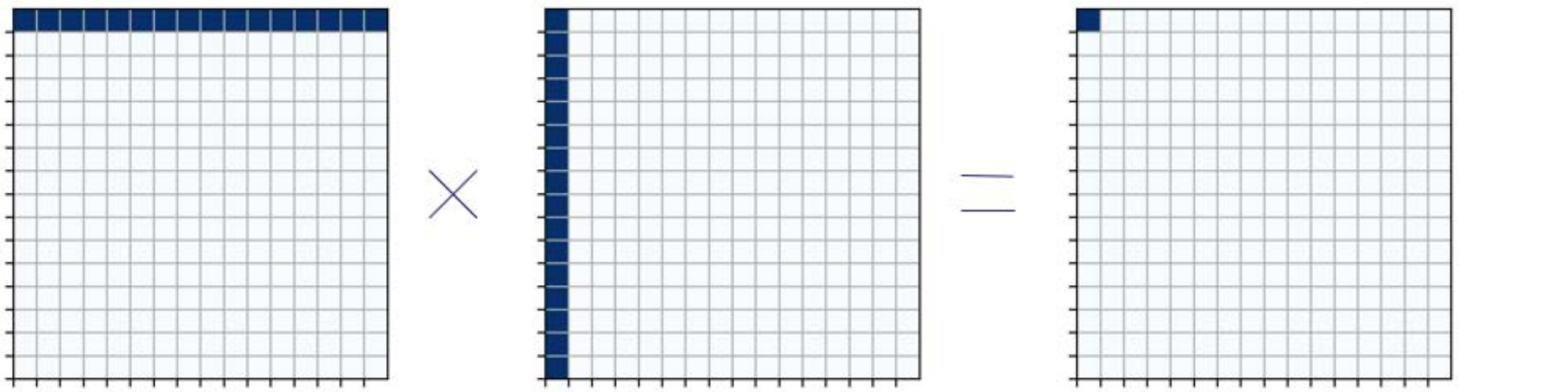
Hardware constraints highlight: Memory Bandwidth and Latency

- Computational performance is higher than memory performance
- Memory access is the main energy consumer (and performance limiter)
- Optimal scheduling (e.g. caching) is very important
- Memory bandwidth, cache size and compute power should be balanced



Пример балансировки: матричное разбиение

Задача: Перемножить матрицы размером 16x16 с кешом на 20 фрагментов
Цель: Минимизировать обращения во внешнюю память
Способ 1: Хранить строку для перемножения на столбцы



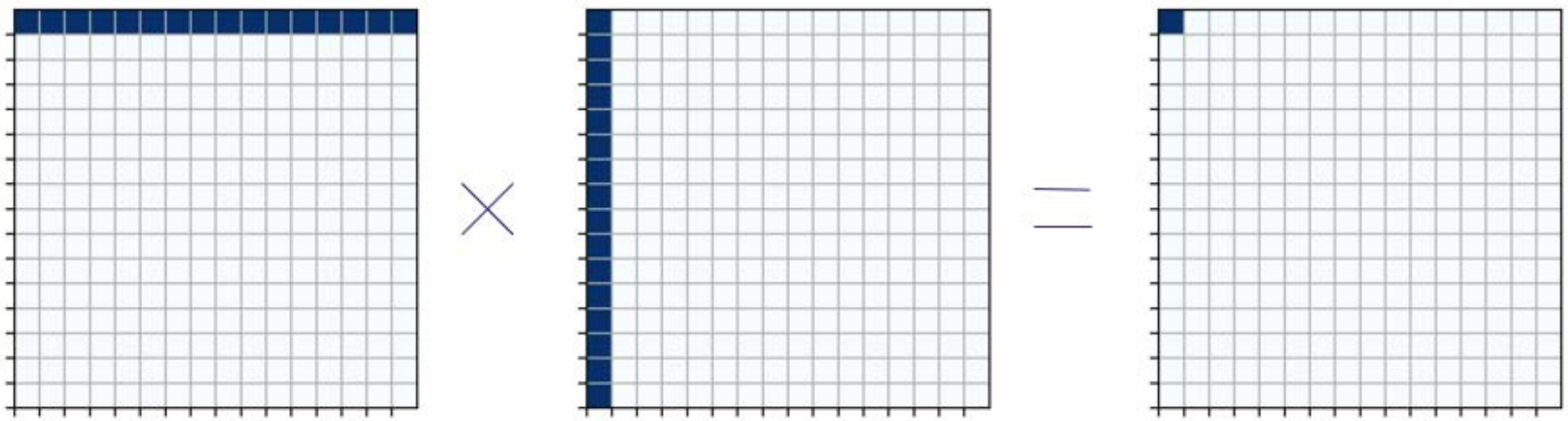
256 фрагментов

4096 фрагментов

Всего: 256 + 4096 = 4352 фрагмента

Computation strategies: Matrix Partitioning

Task: Optimize Matrix Multiplication. Size: 16x16, Cache: 20 chunks
Goal: Minimize memory access
Strategy 1: Store one string and reuse it for other matrix columns



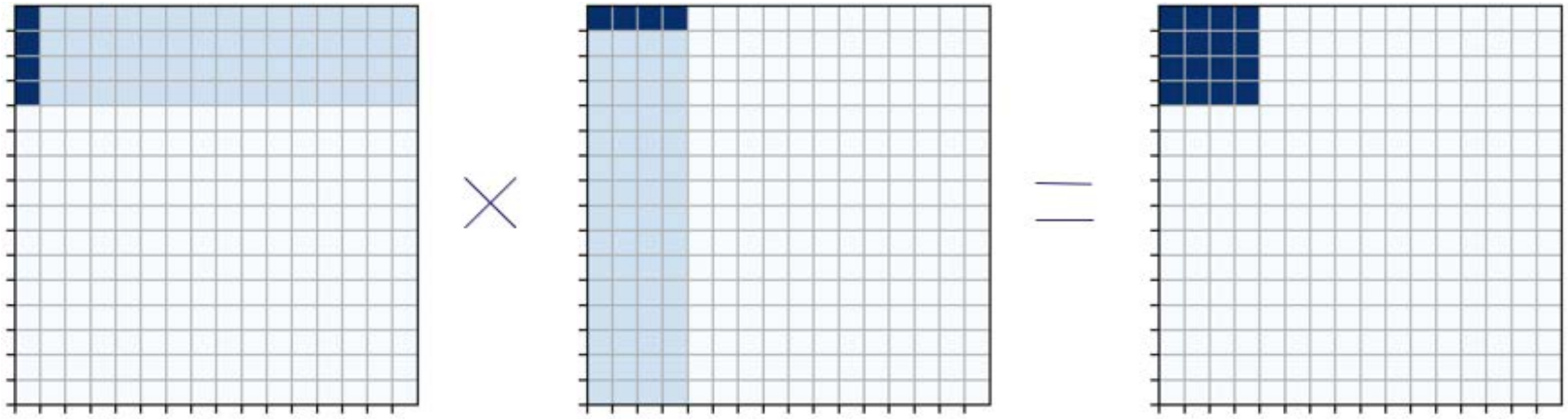
256 chunks

4096 chunks

Total cost:
256 + 4096 = 4352 chunks

Пример балансировки: матричное разбиение

Задача: Перемножить матрицы размером 16x16 с кешом на 20 фрагментов
Цель: Минимизировать обращения во внешнюю память
Способ 2: Хранить промежуточные результаты



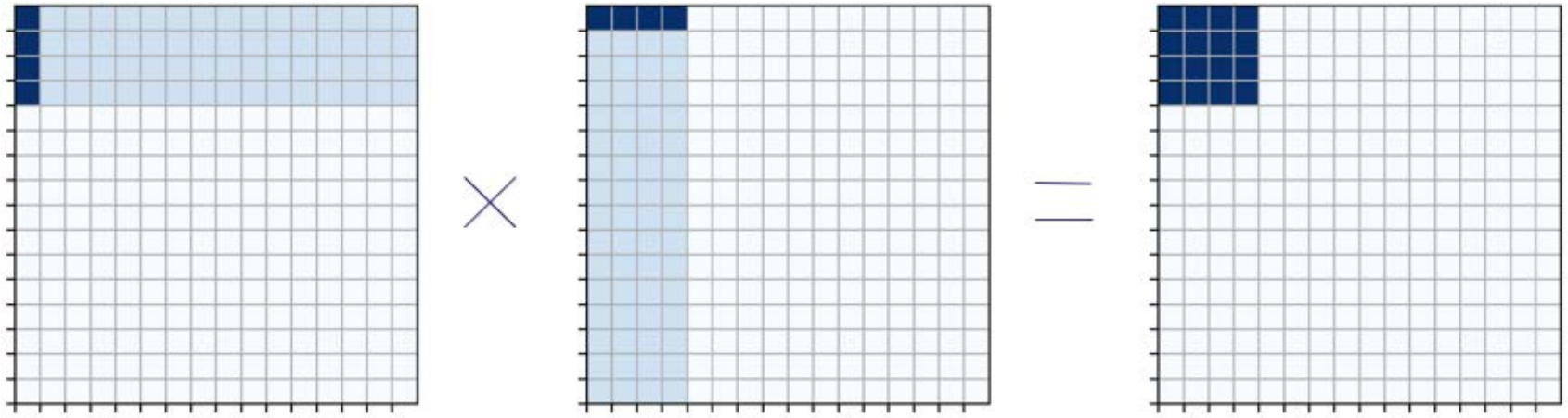
1024 фрагмента

1024 фрагмента

Всего: 1024 + 1024 = 2048 фрагментов

Computation strategies: Matrix Partitioning

Task: Optimize Matrix Multiplication. Size: 16x16, Cache: 20 chunks
Goal: Minimize memory access
Strategy 2: Cache partial results



1024 chunks

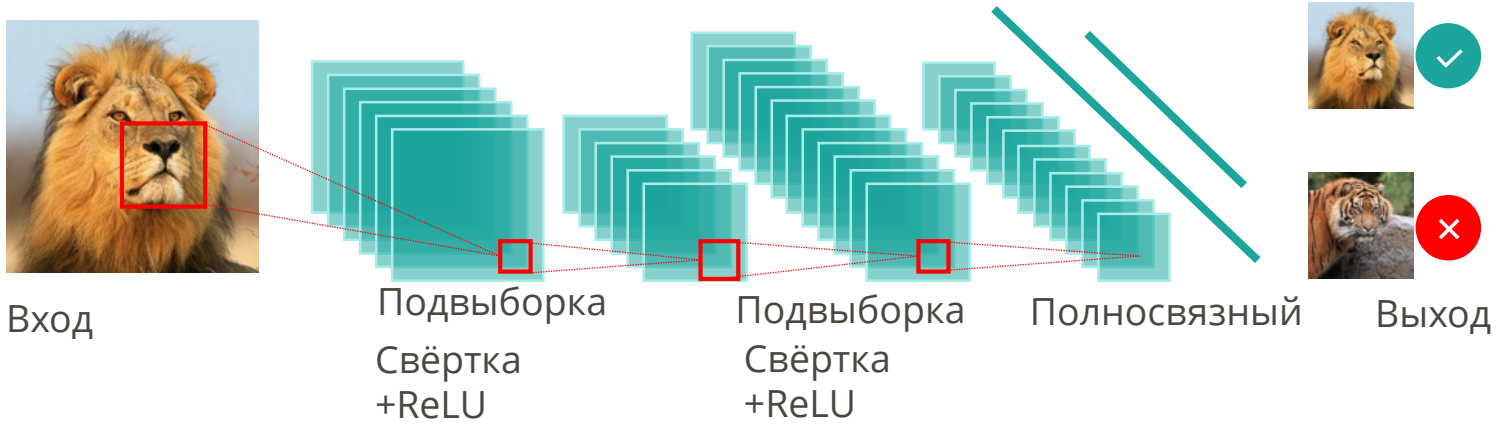
1024 chunks

Total cost:
1024 + 1024 = 2048 chunks

Вывод свёрточных сетей

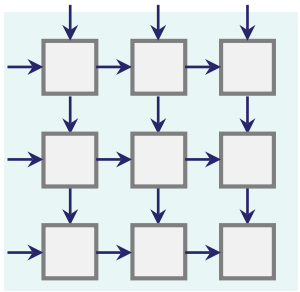
Особенности

- Устойчивость к шуму (int8)
- Основа - умножение матриц



Решение

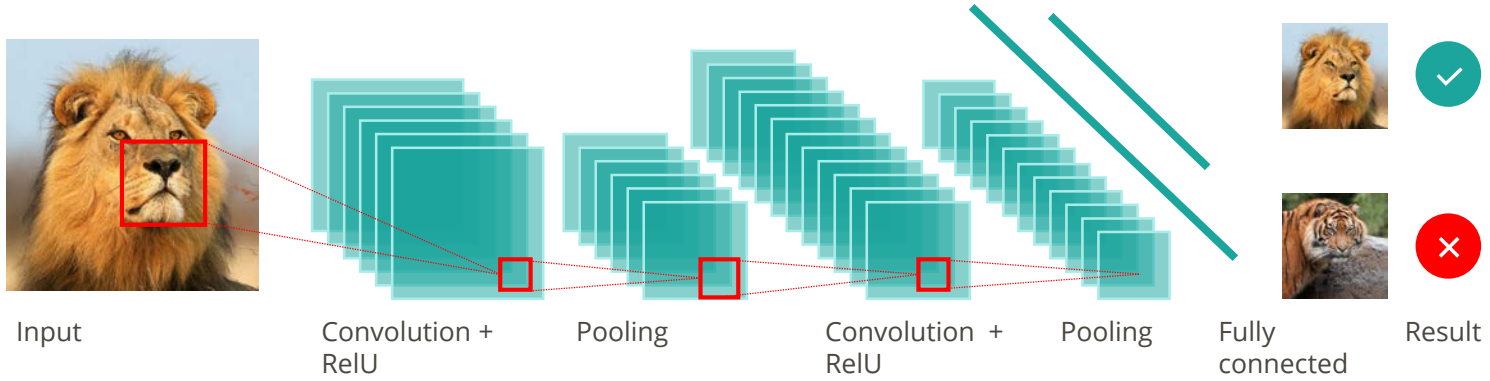
- Переменная разрядность
- Простейшие вычислители
- Только локальные связи



CNN Inference

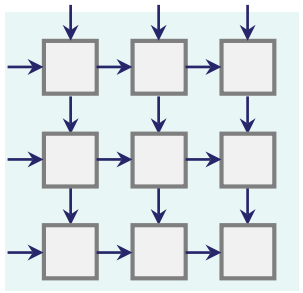
CNN inference:

- Noise tolerant -> Int8 is enough for inference
- Main operations: Matrix Multiplier

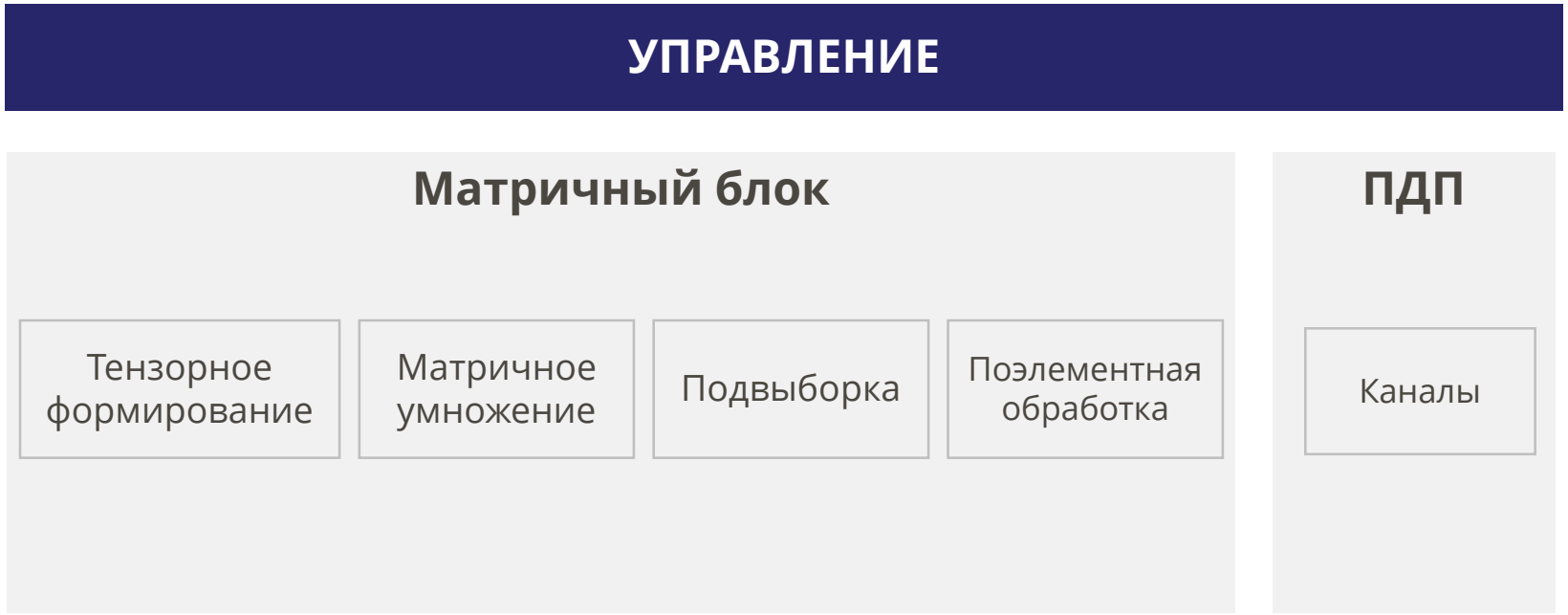


Solution:

- Systolic Architecture - tightly coupled very simple computing elements
- Variable precision
- Only local connectivity of elements
- Highly scalable



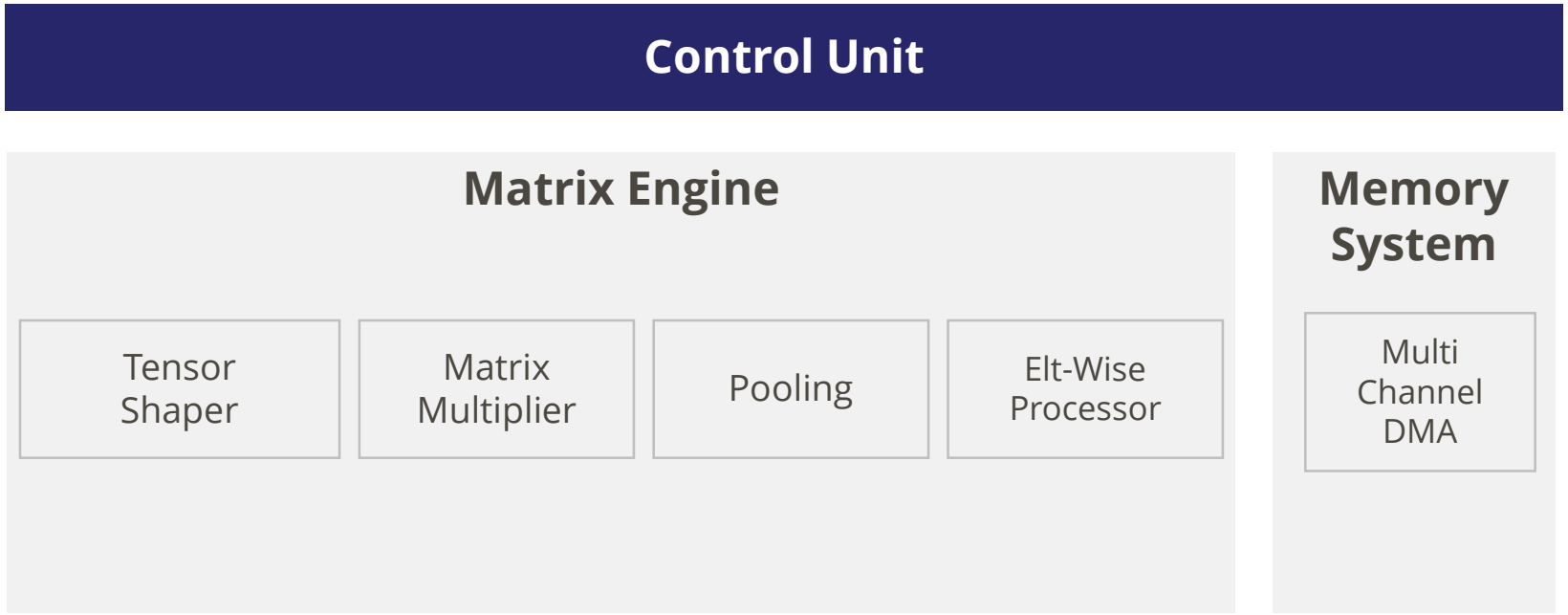
IVA TPU СФБ версия 1: свёрточные сети



Поддерживает

- VGG
- ResNet50
- GoogleNet
- Yolo2

IVA TPU IP version 1(CNNs)



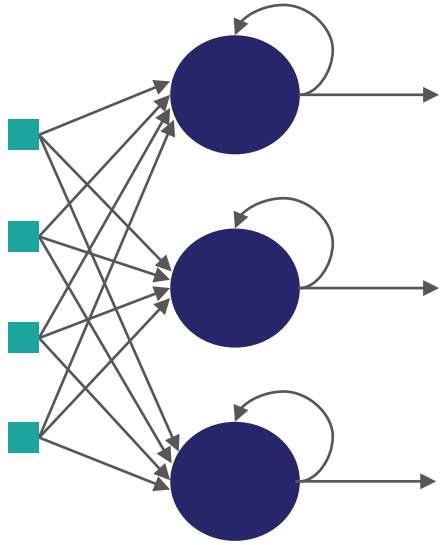
Supported CNNs

- VGG
- ResNet50
- GoogleNet
- Yolo2

Вывод рекуррентных сетей

Особенности:

- LSTM, GRU менее устойчивы к шумам
- Векторные операции (как Softmax) требуют большей точности



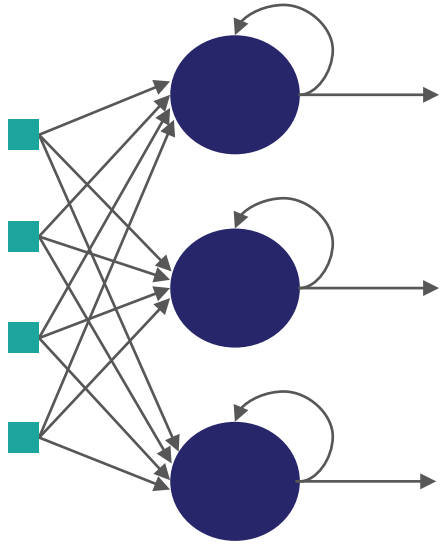
Решение:

- Вывод с различными точностями
- Отдельный векторный блок
  - fp16
  - Поддержка экспоненты, деления, Softmax
  - Память для внутренних состояний рекуррентных слоёв

RNN Inference

RNN Inference:

- LSTM and GRU cells more sensitive to reduced data types
- Other vector operations (e.g. Softmax) require higher precision



Solution:

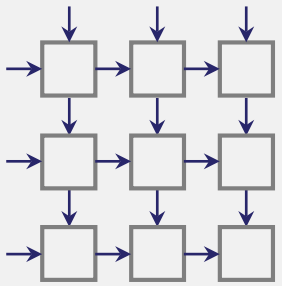
- Mixed data type inference

- Separate Vector Engine
  - FP16
  - Supports Exponent, Division, Softmax
  - Memory for RNN internal states processing

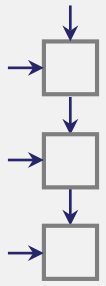
IVA TPU СФБ версия 2 : свёрточные и рекуррентные сети

УПРАВЛЕНИЕ

Матричный блок



Векторный блок



ПДП

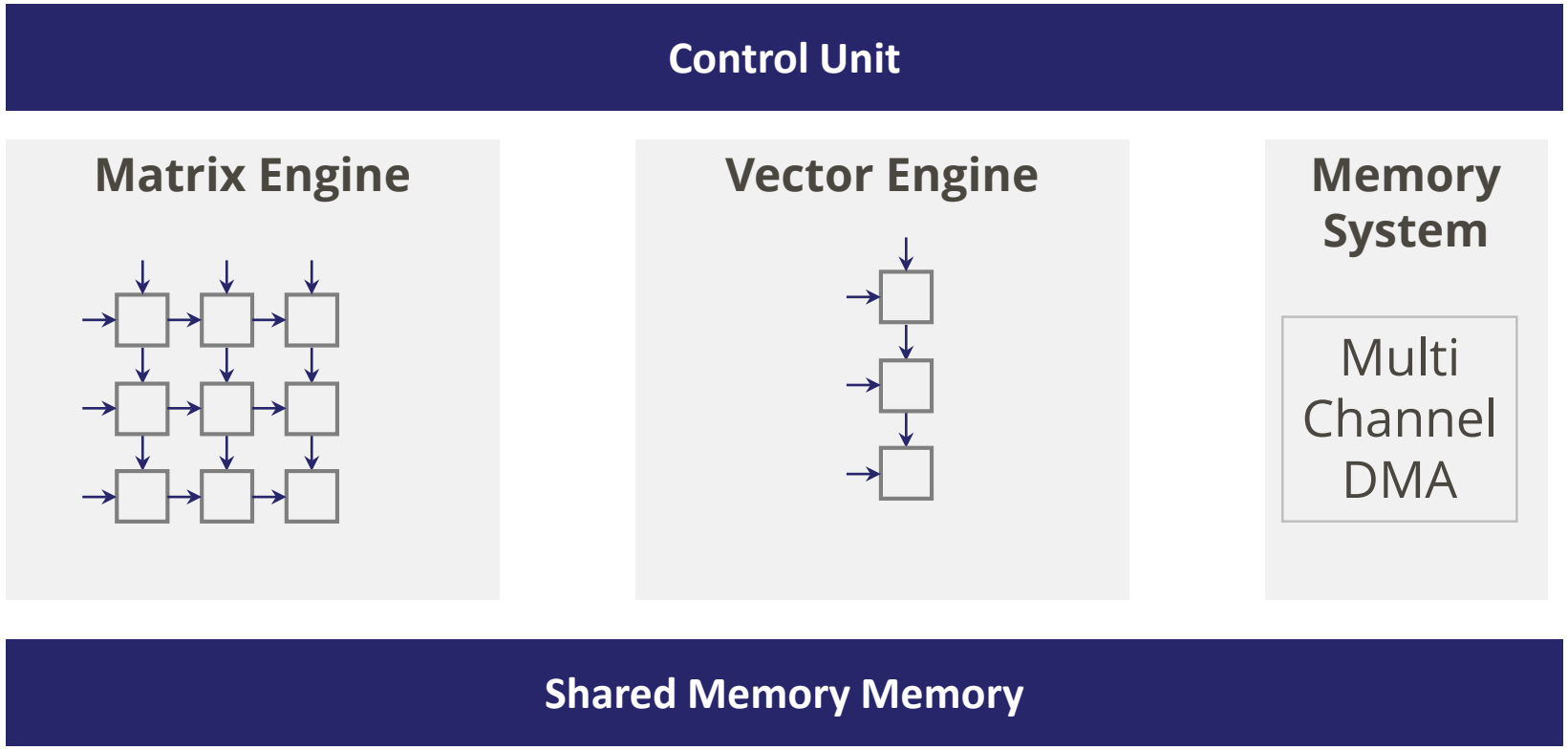
Каналы

ВНУТРЕННЯЯ ПАМЯТЬ

Дополнения к версии 1:

- LSTM, GRU
- Вывод с разными точностями
- Depth-Wise Separable
- Softmax
- Лучшее управление памятью

IVA TPU IP version 2 (CNNs & RNNs)



In addition to ver. 1:

- RNNs including LSTM, GRU
- Mixed precision inference
- Depth-Wise Separable optimization
- Softmax
- More flexible scheduling

Поддержка слоёв и других возможностей

Вывод нейронных сетей:

- **Свёртки**: Fully-connected, 2D/3D Convolution, Depth-wise separable, Dilated, Deconv
- **Рекурренты**: LSTM, GRU
- **ReLU активация**: ReLU, Leaky ReLU (PReLU), ReLU6, HardTanh
- **Non-ReLU активация**: Sigmoid, Tanh, SoftMax
- **Режимы подвыборки**: Average, Max
- **Тензорные преобразования**: depth2space, space2depth
- **Прочее**: Residual (Skip-connections), Attention, concat

Supported Layers & Capabilities

- **Conv layers:** Fully-connected, Convolution (2d/3d, Depth-wise separable, Dilated, Deconv)
- **RNNs:** LSTM, GRU
- **ReLU-based functions:** ReLU, Leaky ReLU (PReLU), ReLU6, HardTanh
- **Non-ReLU-based functions:** Sigmoid, Tanh, SoftMax
- **Pooling modes:** Average, Max
- **Other**: Residual (Skip-connections), Attention, concat
- **Tensor Transforms:** depth2space, space2depth

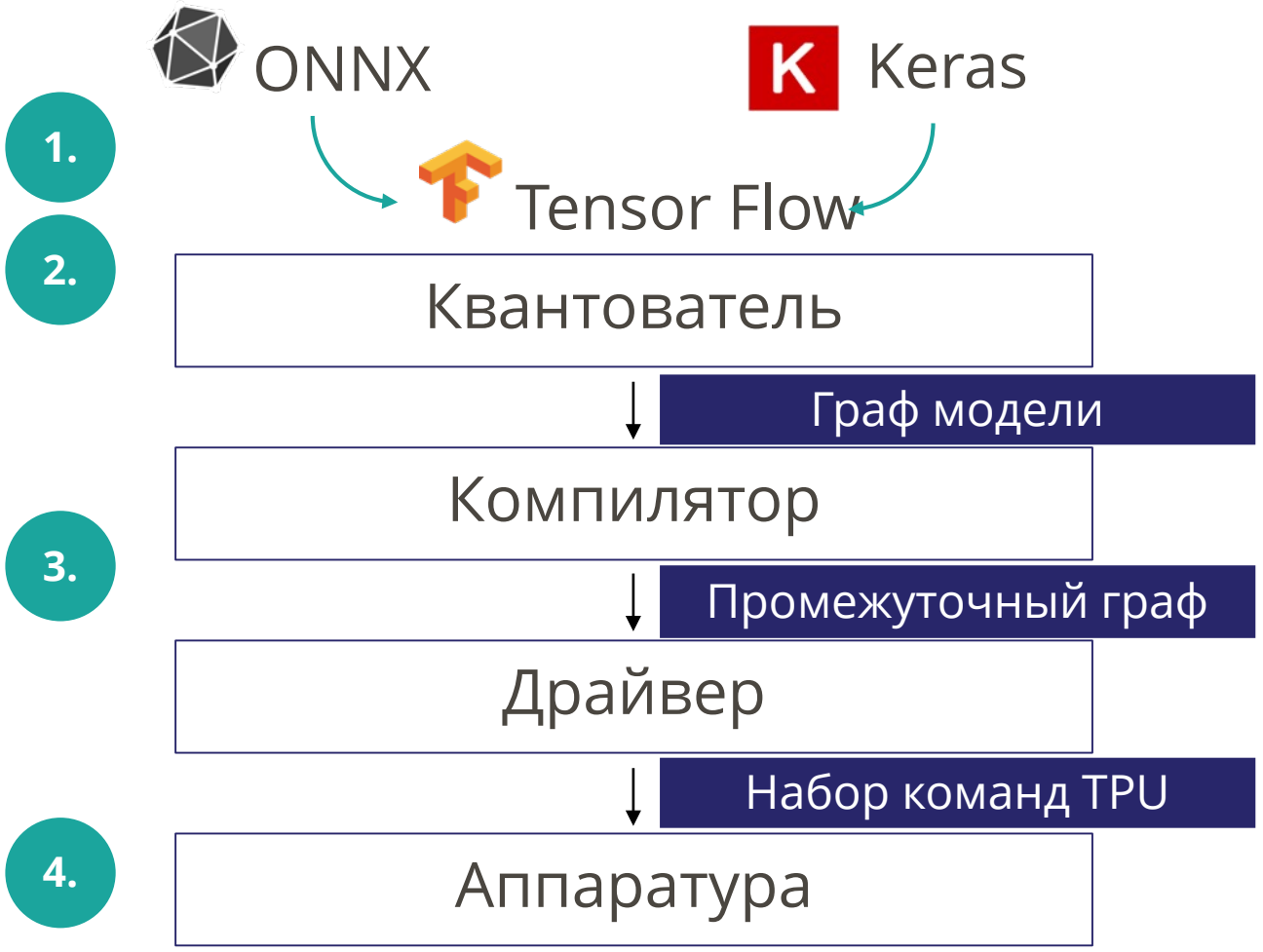
Стек ПО

Особенности

- Малые потери точности благодаря дополнительному обучения
- Оптимизация графа модели
- Автоматическая обработка «из коробки» свёрточных и большинства рекуррентных сетей

Маршрут

1. Импорт модели
2. Настройка параметров квантования
3. Компиляция графа
4. Вывод



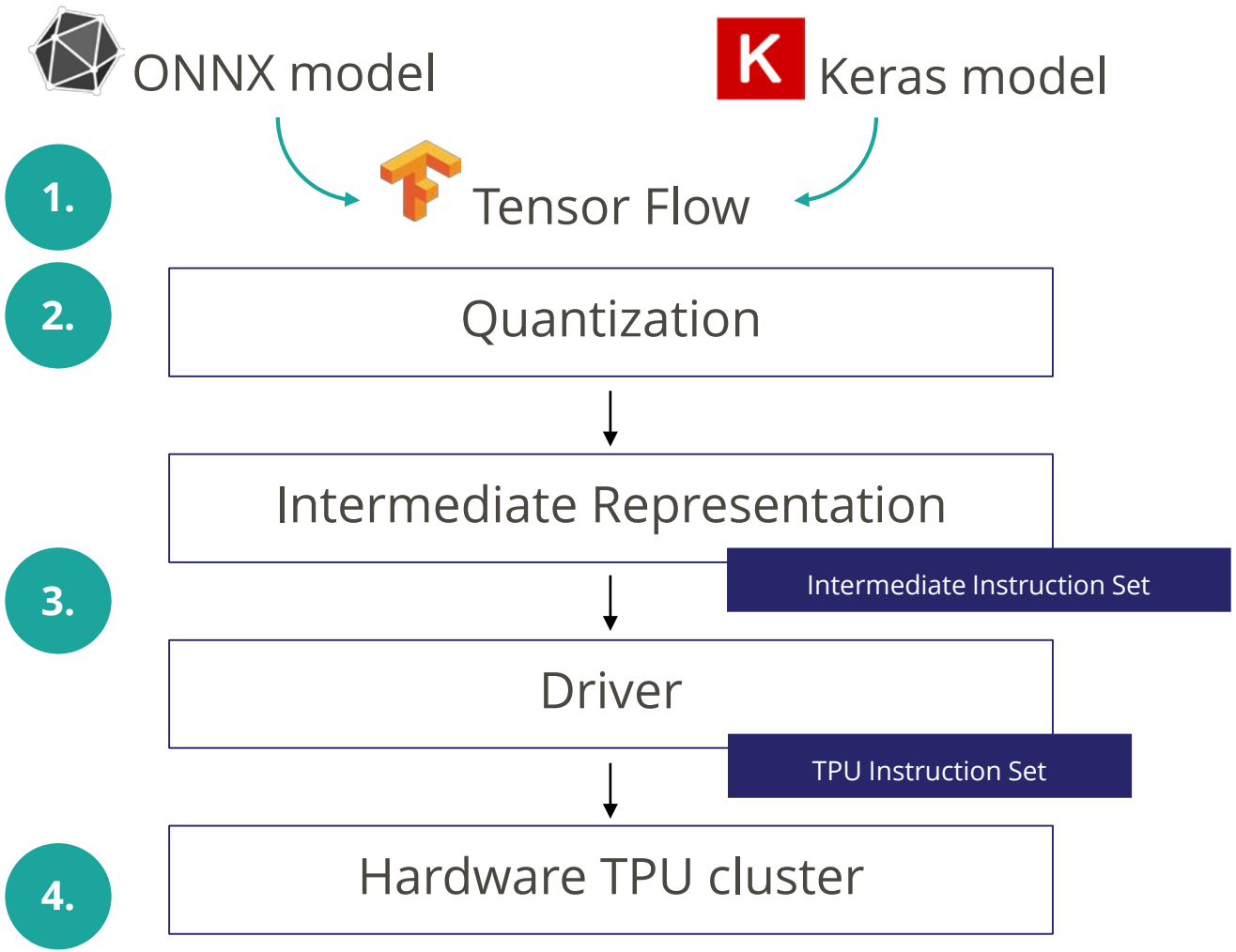
Software stack

Advantages:

- Low compromise on inference precision
- due to additional training
- Computational graph optimization
- Automated work

Workflow:

1. Import model
2. Tweak quantization parameters
3. Compile graph to Internal Representation
4. Run



Спасибо за внимание !