

Применения нейронных сетей в системах реального времени:

- техническое зрение (детекция, сегментация);
- робототехника и автономные системы;
- интеллектуальное видеонаблюдение;
- человеко-машинные интерфейсы;
- промышленная диагностика и контроль качества.

Ключевые требования реального времени:

- высокая точность распознавания/классификации;
- гарантированная задержка (latency) и предсказуемость выполнения;
- достаточная пропускная способность.

Средства развертывания и ускорения вывода:

- ONNX;
- TensorRT (GPU NVIDIA);
- PyTorch-TensorRT
(интеграция PyTorch с TensorRT);
- TriPy (высокоуровневый фронтенд NVIDIA).

Цель сравнения:

- оценить влияние оптимизаций на точность и скорость;
- сформировать практические рекомендации для реального времени.

Цель и задачи

Изучение технологий развертывания свёрточных нейронных сетей для приложений реального времени с использованием PyTorch и TriPy, а также сравнение точности и быстродействия моделей после экспорта в формат TensorRT.

1. Рассмотреть требования систем реального времени к выполнению вывода нейронных сетей;
2. Изучить архитектуру LeNet и адаптировать её для набора данных CIFAR-10;
3. Реализовать модель в PyTorch и TriPy и обеспечить эквивалентность вычислений;
4. Обучить модель на CIFAR-10;
5. Экспортировать реализацию в TensorRT (через PyTorch и через TriPy);
6. Сравнить точность и быстродействие.

Курсовая работа				
Изм.	Лист	№ докум.	Подп.	Дата
Разраб.	Ефремов С.А.			
Пров.	Лычков И.И.			
Т.контр.				
Н.контр.				
Утв.				

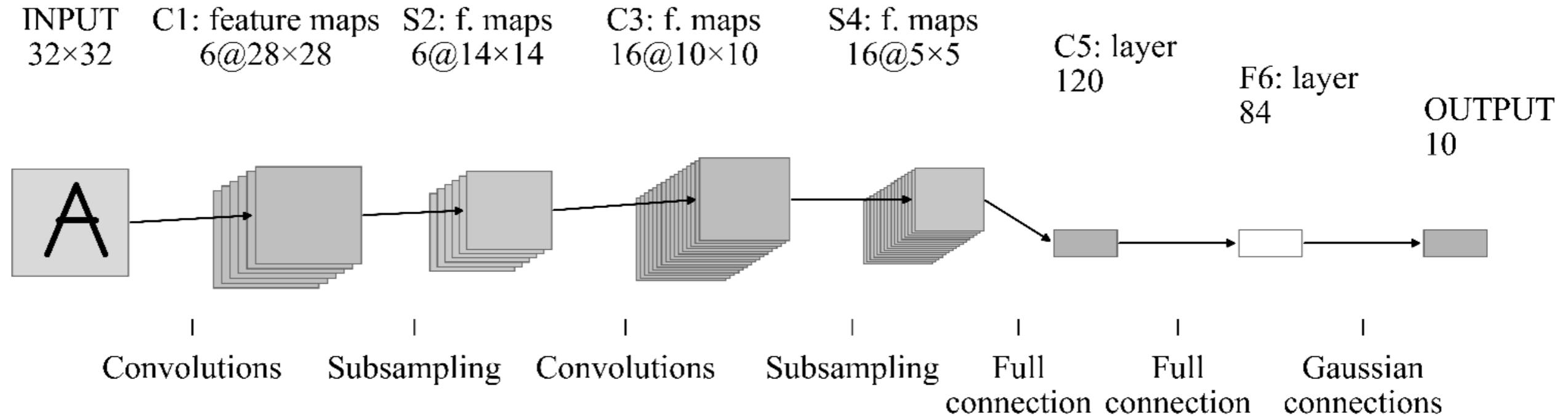
Цель и задачи

Лист 1 Листов 5

МГТУ им. Н.Э. Баумана
группа ИУЗ-11М

Архитектура выбранной нейронной сети

Стандартная структура сети LeNet [3]



Характеристики архитектуры:

Структура сети LeNet для CIFAR-10

Вход: $3 \times 32 \times 32$ (RGB).

- Conv($3 \rightarrow 6$, $k=5 \times 5$) \rightarrow ReLU \rightarrow MaxPool(2×2 , $s=2$)
- Conv($6 \rightarrow 16$, $k=5 \times 5$) \rightarrow ReLU \rightarrow MaxPool(2×2 , $s=2$)
- Flatten($16 \times 5 \times 5 = 400$)
- FC($400 \rightarrow 120$) \rightarrow ReLU
- FC($120 \rightarrow 84$) \rightarrow ReLU
- FC($84 \rightarrow 10$) (логиты классов)

Изм.	Лист	№ докум.	Подп.	Дата	Курсовая работа					
Разраб.	Ефремов С.А.				Архитектура					
Пров.	Лычков И.И.				LeNet					
Т.контр.										
Н.контр.										
Утв.										
				Лист 2 / Листов 5						
				МГТУ им. Н.Э. Баумана группа ИУЗ-11М						

Подготовка данных

Обучающая выборка (train):

1. аугментация:
 - случайное кадрирование 32×32 с padding=4;
 - случайное горизонтальное отражение;
2. преобразования:
 - преобразование изображения в тензор;
 - нормализация по каналам RGB.

Тестовая/валидационная выборка (test/val):

1. без аугментации;
2. преобразования:
 - преобразование изображения в тензор;
 - нормализация по каналам RGB.



Рисунок 1 — График изменения средних потерь

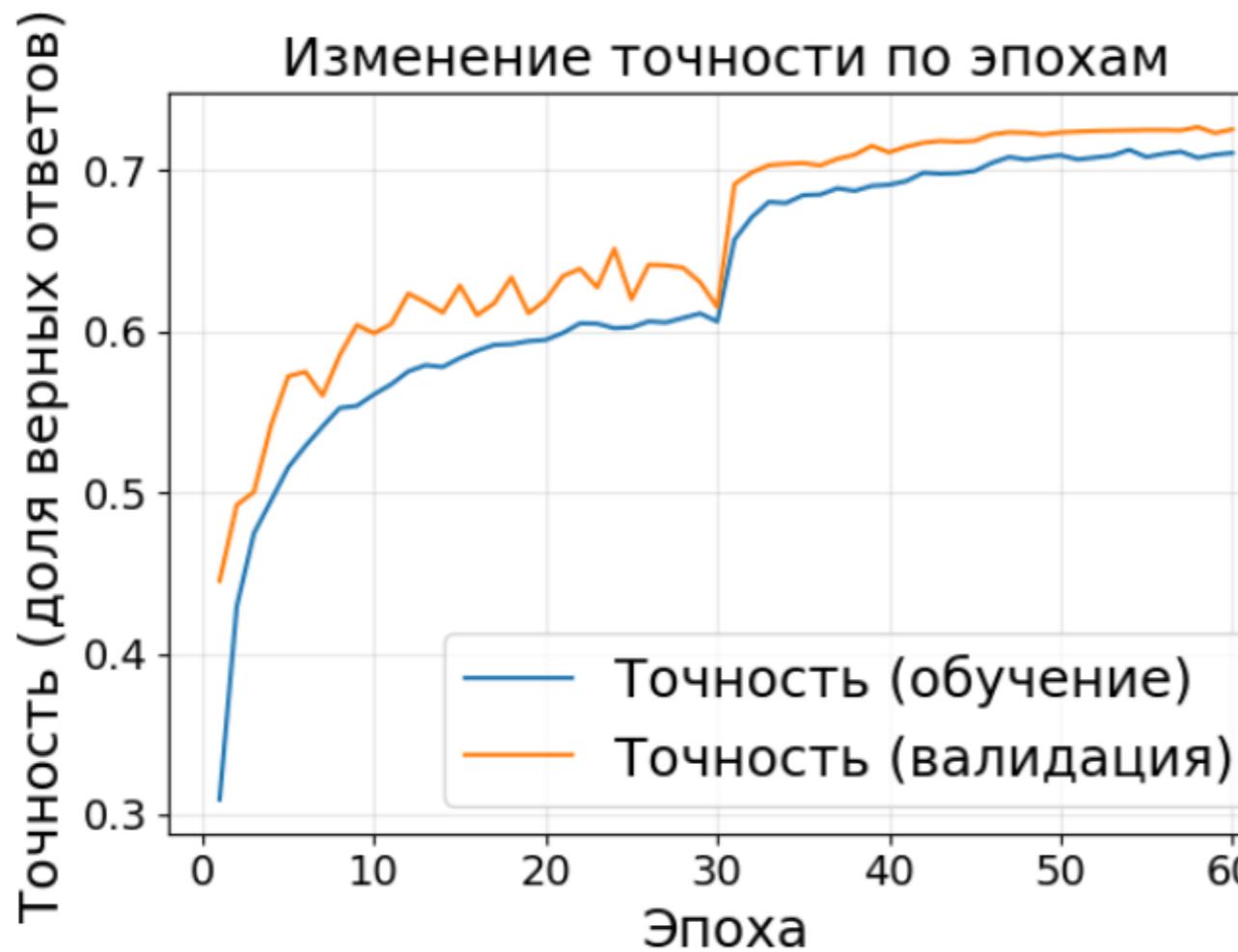


Рисунок 2 — График изменения точности

Гиперпараметры обучения

Размер батча: 200.

Оптимизатор: SGD (learning rate 0.05, momentum 0.9, weight decay 5e-4).

План изменения learning rate: уменьшение в 10 раз после 30-й и 45-й эпох.

Число эпох: 60.

Курсовая работа				
Реализация				Лит.
PyTorch				Масса
Изм.	Лист	№ докум.	Подп.	Дата
Разраб.	Ефремов С.А.			
Пров.	Лычков И.И.			
Т.контр.				
Н.контр.				
Утв.				
				Лист 3 Листов 5
				МГТУ им. Н.Э. Баумана группа ИУЗ-11М

Реализация в TriPy

TriPy используется как высокоуровневый фронтенд, ориентированный на производительный вывод и тесно интегрированный с TensorRT. Архитектура TriPy-модели структурно повторяет PyTorch-вариант LeNet: два свёрточно-пулинговых блока и полно связная часть классификатора.

Перенос параметров из PyTorch

```
def convert_state_torch_to_tripy(torch_state):
    tripy_state = {}
    for name, param in torch_state.items():
        np_value = param.detach().cpu().numpy().astype("float32")
        tripy_state[name] = tp.Tensor(np_value)
    return tripy_state
```

Для обеспечения сопоставимости результатов используются одни и те же обученные веса. Параметры PyTorch-модели конвертируются в формат TriPy (приведение к float32 и загрузка в tp.Tensor), затем загружаются в TriPy-модель. Такой перенос позволяет получить функционально эквивалентную реализацию для дальнейшего сравнения точности и скорости.

Сравнение результатов

PyTorch: loss = 0.7994318, accuracy = 0.7270;
TriPy: loss = 0.7994541, accuracy = 0.7269.

Незначительное расхождение носит ожидаемый характер и обусловлено особенностями численной реализации вычислений и преобразований данных при переходе между средами выполнения (округления, порядок операций с плавающей запятой). В целом результаты подтверждают, что TriPy-реализация воспроизводит качество PyTorch-модели с практически полным совпадением метрик.

Курсовая работа					
Реализация			Лист.	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата	
Разраб.	Ефремов С.А.				
Пров.	Лычков И.И.				
Т.контр.					
Н.контр.					
Утв.					
Лист 4 Листов 5			МГТУ им. Н.Э. Баумана группа ИУЗ-11М		

Hyperbolic geometry

Результаты сравнения быстродействия

Среда выполнения	forward, мс/пакет	forward, мс/изобр.	e2e, мс/пакет	e2e, мс/изобр.
PyTorch	0,814	0,0008	1,702	0,0017
TensorRT	0,652	0,0007	1,620	0,0016
TriPy → TensorRT	0,548	0,0005	1,476	0,0015

Методика измерений производительности

Измеряются два режима

1. forward-only: входные данные находятся в GPU, учитывается только прямой проход модели;
 2. end-to-end: включает копирование CPU→GPU, вычисления и синхронизацию.

Анализ результатов

Оптимизация вычислительной части (forward-only):
TensorRT уменьшает время на $\approx 19.9\%$ относительно
PyTorch, а TriPy → TensorRT даёт уменьшение $\approx 32.7\%$
относительно PyTorch и $\approx 16.0\%$ относительно TensorRT

В режиме end-to-end эффект слабее из-за заметной доли времени на передачу данных и синхронизацию: TensorRT уменьшает e2e на $\approx 4.8\%$ относительно PyTorch, TriPy \rightarrow TensorRT на $\approx 13.3\%$ относительно PyTorch.

Условия эксперимента

Тестирование проводилось на CIFAR-10 (10 000 изображений) при одинаковой предобработке. Использовались warm-up 5 батчей и серия из 100 прогонов с усреднением. Оборудование: Ubuntu 24.04, CPU AMD Ryzen 5 7500F, RAM 32 ГБ DDR5, GPU NVIDIA GeForce RTX 4060 Ti 8 ГБ.