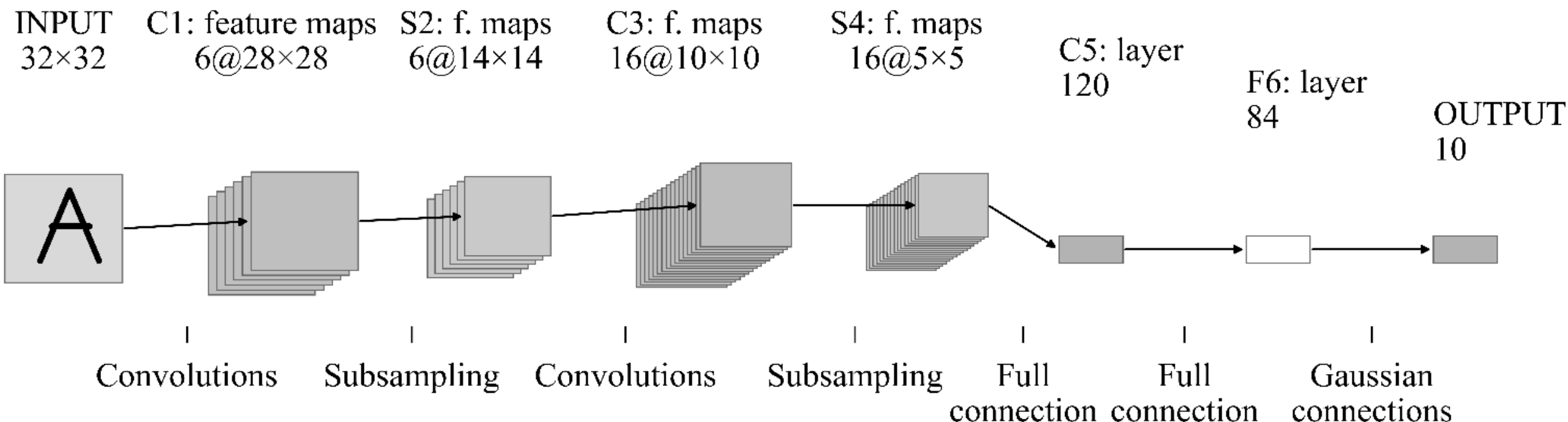


Архитектура выбранной нейронной сети

Стандартная структура сети LeNet [3]



Характеристики архитектуры:

Структура сети LeNet для CIFAR-10

Вход: 3×32×32 (RGB).

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

- формат входных изображений: RGB, 3×32×32;
- набор данных / задача: CIFAR-10;
- количество классов: 10;
- тип модели: свёрточная нейронная сеть (CNN);
- число параметров: $\approx 6.2 \times 10^4$;
- вычислительная сложность: умеренная;
- требования к памяти: невысокие.

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

Обучающая выборка (train):

- аугментация:
 - случайное кадрирование 32×32 с $\text{padding}=4$;
 - случайное горизонтальное отражение;
- преобразования:
 - преобразование изображения в тензор;
 - нормализация по каналам RGB.

Тестовая/валидационная выборка (test/val):

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

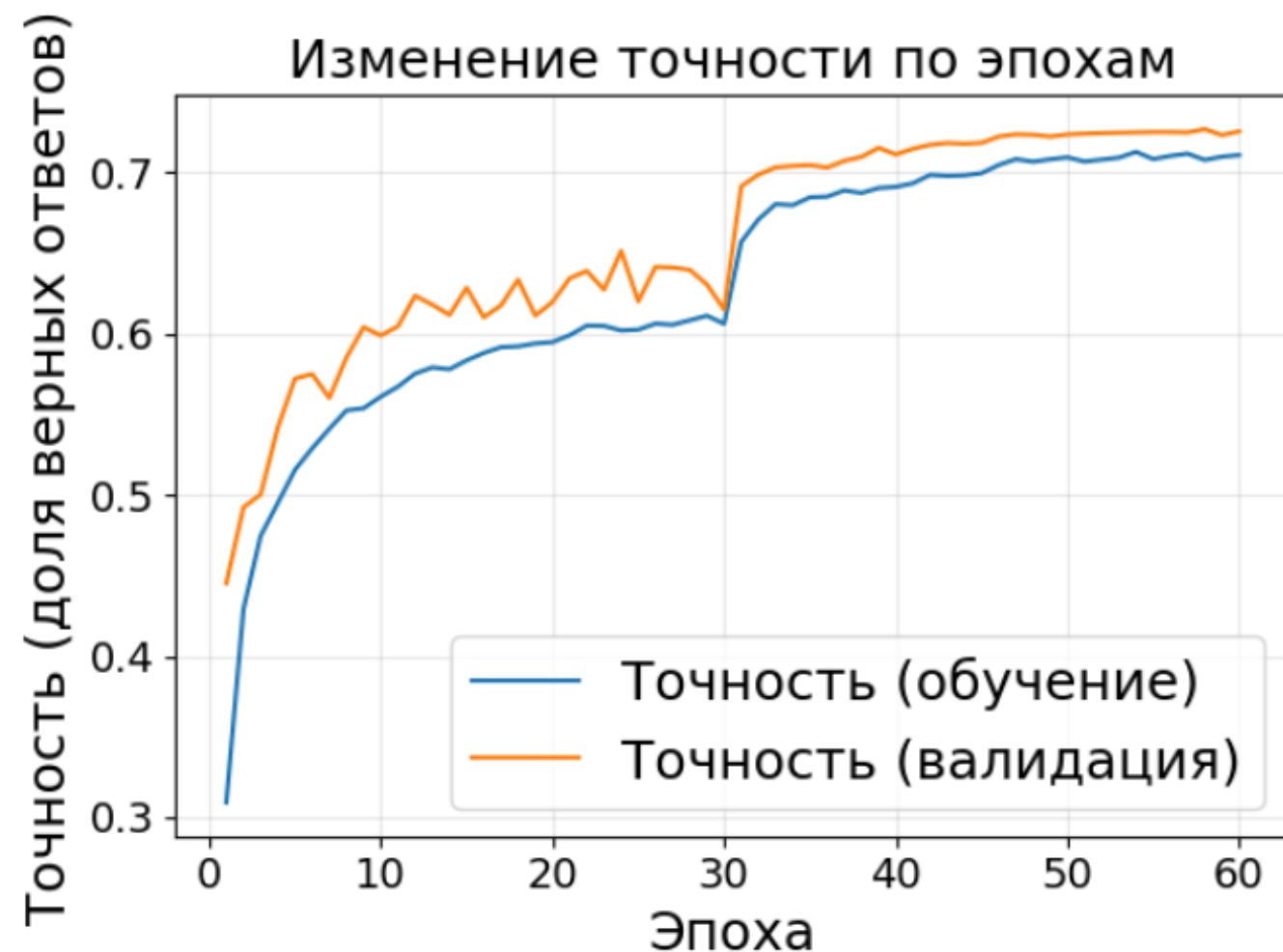


Рисунок 2 — График изменения точности



Рисунок 1 — График изменения средних потерь

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

Юридический отдел
Инв. № подл.
Взам. инв. №
Подп. и дата
Инв. № докл.
Подп. и дата

Перв. примен.
Справ. №

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

КУРСОВАЯ РАБОТА
по дисциплине «Информационные технологии в машиностроении»
на тему «Оптимизация производительности нейронных сетей»

Перв. примен.
Справ. №

Подп. и дата
Инв. № докум.
Взам. инв. №
Подп. и дата
Инв. № подл.

курсовая работа

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

Условия эксперимента

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.

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.

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→TensorRT на $\approx 13.3\%$ относительно PyTorch.

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

Копировал

Формат А3

Итого листов