СОЗДАНИЕ КЛАСТЕРА ДЛЯ ОБУЧЕНИЯ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ В УЧЕБНОЙ АУДИТОРИИ

Кривошеев Н.А. [1], *Кузнецов А.В.* [2], *Спицын В.Г.* [3]
[1] *Томский политехнический университет, гр. А0-39, аспирант, e-mail: nikola0212@mail.ru*
[2] *Томский политехнический университет, ИШИТР, ассистент, e-mail: akuznetsov@tpu.ru*
[3] *Томский политехнический университет, ИШИТР, профессор, spvg@tpu.ru*

Аннотация

В данной работе рассматривается реализация кластера для обучения алгоритмов искусственного интеллекта на основе персональных компьютеров в учебной аудитории. Рассмотрен процесс реализации и подходы к решению проблемы низкой скорости Ethernet соединения между персональными компьютерами, которая связанна с особенностями архитектуры кластера. Проведены программная реализация, анализ и сравнение реализованных подходов. Анализ реализованных алгоритмов производится на основе обучения нейронной сети на задаче семантической сегментации изображений.

Ключевые слова: машинное обучение, кластер, сжатие градиентов.

Введение

Одной из основных проблем, с которой сталкиваются специалисты по машинному обучению является нехватка вычислительных ресурсов при обучении нейросетевых алгоритмов и предварительной подготовке выборок данных. Данная проблема возникает при коммерческой разработке, у научно-исследовательских групп, в учебном процессе при обучении студентов.

Использование нескольких GPU является распространённым подходом. В своей работе Алекс Крижевский [1] использует 8 GPU NVIDIA K20 для тестирования предложенного алгоритма ускорения обучения нейронных сетей. В работе [2] использовалось два варианта с 8 GPU NVidia Tesla K20Xm и 24 серверами по 2 GPU K20Xm подключенными через Infiniband.

По представленным примерам видно, что кластера из нескольких GPU используются для решения различных задач машинного обучения. В представленных работах разработаны различные подходы, позволяющие увеличить скорость обучения нейронных сетей на основе нескольких GPU.

Зачастую кластера с несколькими видеокартами являются малодоступными. В данной работе рассматривается реализация кластера для разработки алгоритмов машинного обучения на основе персональных компьютеров в учебной аудитории.

Аналогичные работы

Был проведен анализ ранее опубликованных работ, в которых были рассмотрены подходы позволяющие увеличить скорость обучения нейронных сетей с использованием нескольких GPU. Данные подходы могут быть использованы для реализации распределенных вычислений на нескольких персональных компьютерах.

В работе Алекса Крижевски [1] рассматривается подход, основанный на смешивании параллелизма данных и параллелизма моделей. Для обучения сверточных слоев используется подход на основе параллелизма данных, а для обучения полносвязных слоев используется подход на основе параллелизма моделей. Данный подход протестирован на задаче классификации изображений ImageNet и позволил ускорить обучение в 6.25 раз на 8 GPU.

В работе [2] рассматривается подход на основе параллелизма данных, в процессе передачи градиентов между устройствами, производится квантование градиентов до 1 бита. Данный подход является одним из наиболее популярных, модификация данного подхода используется в следующей рабо-те [3].

Идея передачи градиентов нейронной сети в типе данных, который использует меньшее количество бит, применяется в данной работе при передаче градиентов между персональными компьютерами.

В результате проведенного анализа были найдены подходы позволившие реализовать обучение нейронных сетей с использованием нескольких персональных компьютеров.

Характеристики оборудования

Для реализации данной работы используются персональные компьютеры в 402 учебной аудитории 10 корпуса Томского политехнического университета. В реализации данной работы используется 6 из 12 персональных компьютеров, т.к. остальные не были настроены к моменту написания данной работы.

Характеристики персональных компьютеров представлены ниже:

- CPU: AMD Ryzen 5 3600X 6-Core.
- GPU: NVIDIA GeForce GTX 1650 SUPER.
- RAM: 16 Гб.
- OS: Windows 10.

Передача данных между персональными компьютерами производится на основе Gigabit Ethernet. Скорость передачи данных составляет 1 Гбит/с (125 Мегабайт в секунду).

Параллельное распределение данных

В данной работе рассматривается подход, основанный на параллельном распределении данных. Данный подход заключается в распределении нескольких обучающих примеров (относящихся к одному мини-пакету) между вычислительными устройствами, на каждом вычислительном устройстве находится копия нейронной сети. В процессе обучения градиенты или весовые параметры, вычисленные на основе переданных примеров, синхронизируется между вычислительными устройствами.

Например, при решении задачи семантической сегментации спутниковых снимков на основе 4 вычислительных устройств, зададим размер мини-пакета равный 32 изображения. Далее каждое вычислительное устройство получает 8 изображений из мини-пакета и вычисляет градиент для весовых параметров нейронной сети. Затем происходит передача полученных градиентов на основное вычислительное устройство (сервер) для последующей синхронизации. Полученные на сервере градиенты синхронизируются и отправляются на остальные вычислительные устройства. На основе полученных градинтов происходит изменение весовых параметров. Приведенный алгоритм повторяется до окончания обучения нейронной сети.

Программная реализация

Для реализации кластера на основе нескольких персональных компьютеров была предпринята попытка реализации с использованием библиотек TensorFlow и PyTorch. Данные библиотеки позволили провести обучение нейронных сетей на одном персональном компьютере, но не позволили реализовать вычислительный кластер на основе нескольких персональных компьютеров с операционной системой Windows.

В результате оценки объема требуемых исследований для устранения некорректной работы библиотеки было принято решение изучить возможность реализации другим методом. При обзоре возможных подходов был найден хорошо зарекомендовавший себя и документированный интерфейс сокетов Беркли.

Для тестирования вычислительного кластера была разработана собственная программная реализация, представленная на сайте [4]. Данная программа реализована на основе библиотеки PyTorch и предназначена для синхронизации нейросетевых моделей и градиентов библиотеки PyTorch между вычислительными устройствами.

Алгоритм распределенного обучения нейросетевых моделей

Алгоритм обучения нейронных сетей на основе распределения данных состоит из следующих шагов:

1. Предварительное распределение данных для обучения между вычислительными узлами.
2. Инициализация нейронной сети на сервере.
3. Распределение копии нейронной сети между вычислительными узлами.
4. Вычисление градиента ошибки для весовых параметров нейронной сети на вычислительных узлах и сервере. В данной реализации сервер участвует в процессе вычислений.
5. Синхронизация полученных градиентов на сервере.

6. Распределение полученных градиентов между вычислительными узлами.
7. Повторяем шаги 4-6 до окончания обучения нейронной сети.

Критериями остановки обучения нейронной сети могут выступать: заданное количество итераций, достижение необходимой точности работы нейронной сети, увеличение ошибки на тестовой выборке.

Выбор задачи для тестирования алгоритмов

Для тестирования реализованных алгоритмов была выбрана задача, связанная с семантической сегментацией снимков дистанционного зондирования земли.

Задача семантической сегментации и сверточная нейронная сеть были выбраны в связи с малыми размерами сверточных нейронных сетей (меньшее количество весовых параметров) и относительно большими объемами вычислений, что позволяет снизить объемы передаваемых данных по Ethernet между вычислительными устройствами.

Целью данной работы является оценка возможности реализации кластера на основе персональных компьютеров, в данной работе не преследуется цель повышения качества сегментации объектов на изображении. В связи с поставленной целью, для реализации и тестирования была выбрана нейронная сеть U-Net. U-Net является наиболее простой и распространённой архитектурой нейронной сети для семантической сегментации изображений.

Выборка данных

В данной работе используется выборка данных Vaihingen. Данный набор содержит снимки дистанционного зондирования города Файхинген в Германии, которые были получены Немецкой ассоциацией фотограмметрии и дистанционного зондирования.

Набор данных содержит 157 изображений размером 512x512 пикселей. Набор данных был разделен на обучающую и тестовую выборки, обучающая выборка содержит 127 изображений, тестовая выборка содержит 30 изображений. Сегментация проводится на основе следующих 6 классов: здания, низкая растительность, деревья, автомобили, непроницаемые поверхности, водные объекты.

Примеры изображений из выборки данных Vaihingen представлены ниже:

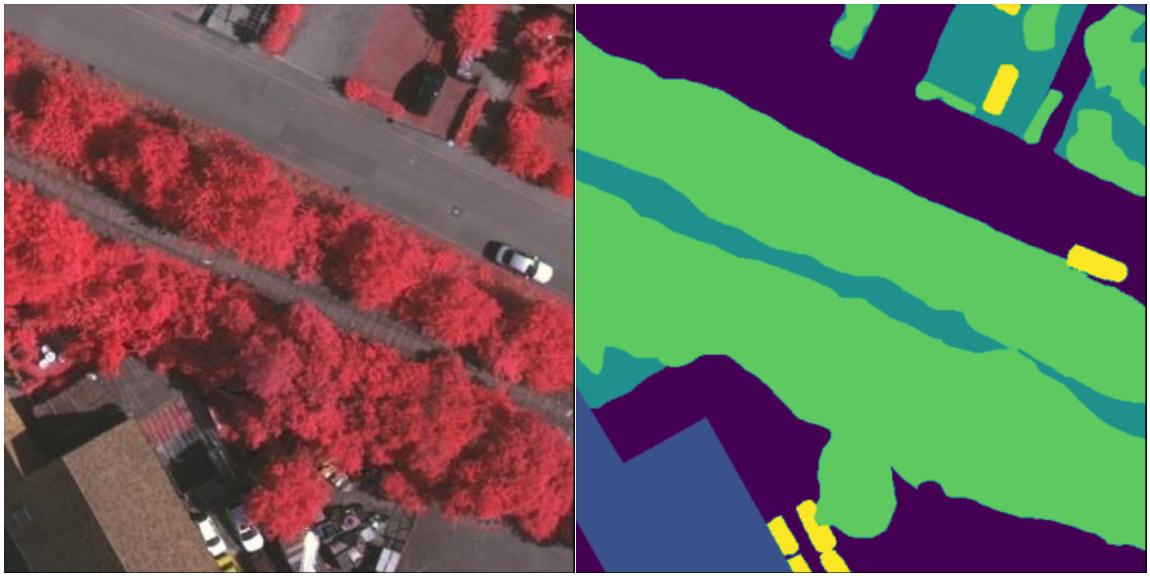

Рис. 1. Исходное изображение – слева. Сегментированное изображение – справа

Топологии тестируемых нейронных сетей

Для тестирования алгоритмов была выбрана топология нейронной сети небольшого размера (8,74 Мегабайт). Архитектура данной нейронной сети представлена ниже:

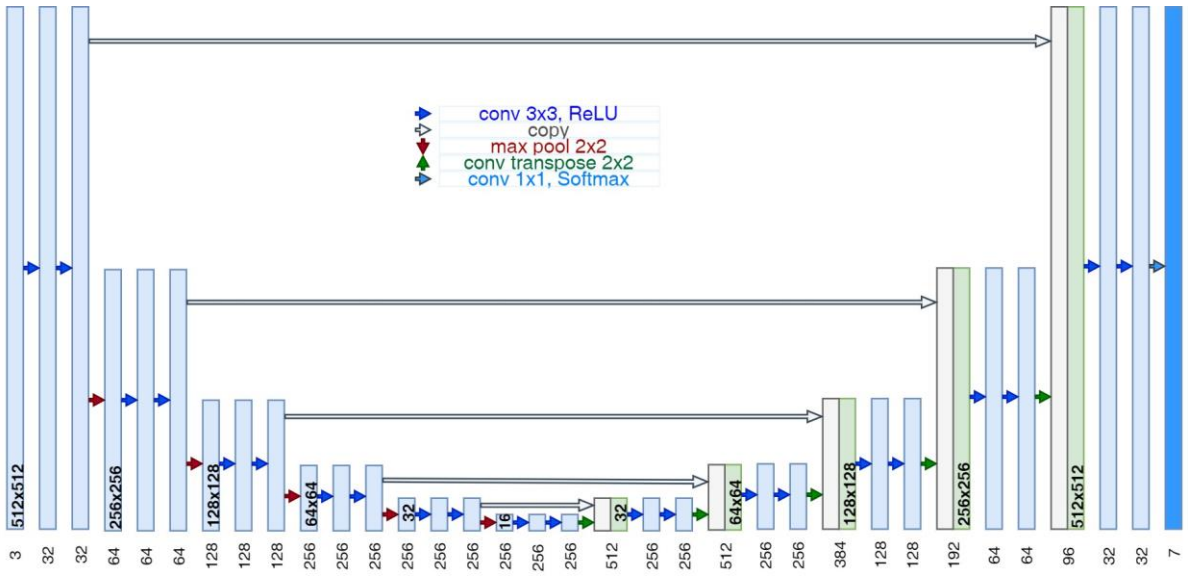

Рис. 2. Топология нейронной сети U-Net

Было проведено тестирование различных архитектур нейронных сетей. Увеличение количества нейронов в два раза не привело к увеличению точности U-Net на данной выборке данных. Изменение топологии нейронной сети и тестирование являются целью дальнейшей работы, но не рассматриваются в данной работе.

Влияние размера мини-пакета в выборке данных на качество обучения

Одним из подходов, позволяющим ускорить обучение нейронных сетей является обучение нейронных сетей с использованием мини-пакета. Увеличение размера мини-пакета позволяет уменьшить количество передач данных между устройствами или использовать большее количество вычислительных устройств, что позволяет ускорить обучение нейронных сетей.

Было проведено тестирование данного подхода на 1 ПК. Данное тестирование позволяет выбрать оптимальный размер мини-пакета для обучения на заданном количестве персональных компьютеров и проверить корректность работы реализованной программы.

Результаты тестирования представлены на изображении ниже.

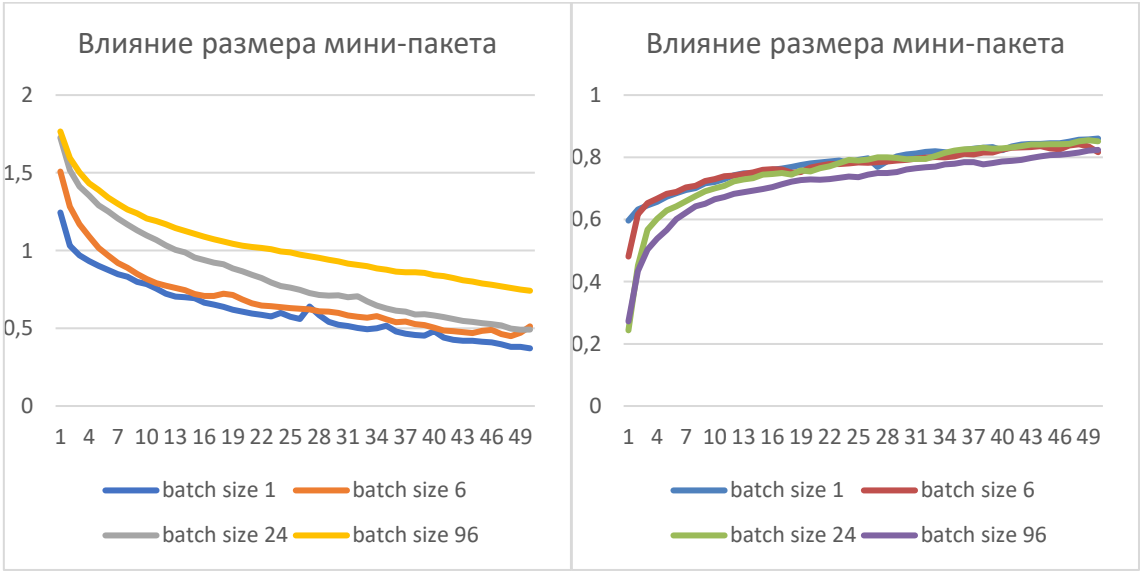

Рис. 3. Влияние размера мини-пакета на качество обучения

По представленным графикам видно, что увеличение размера мини-пакета приводит к значительному замедлению обучения нейронной сети на первых эпохах обучения, но постепенно разница в качестве обучения сокращается. Решение представленной проблемы рассматривал Алекс Крижевский [1], но данная модификация не использовалась в реализации данной работы.

Для достижения точности 0.5 с размером мини-пакета равным 1, потребовалось 32 эпохи. Для достижения точности 0.5 с размером мини-пакета равным 24, потребовалось 48 эпох. Таким образом, можно сделать вывод, что при использовании мини-пакета равного 24, можно значительно ускорить обучение нейронной сети.

По представленным результатам видно, что увеличение размера мини-пакета до 96 обучающих примеров приводит к значительному снижению скорости обучения. Данный размер мини-пакета может быть использован при обучении нейронной сети на большем количестве компьютеров, чем использовалось в данной работе.

На основе проведенного анализа, в дальнейшей работе применяется размер мини-пакета равный 24 обучающих примера. Данный размер пакета не требует частых передач градиентов, как при использовании мини-пакета равного 6 обучающих примеров.

При использовании размера мини-пакета равного 24, нейронная сеть достигает требуемой точности в 80% на тестовой выборке за 36 эпох. Данный результат является одним из лучших в проведенных тестах, так лучшим результатом является 31 эпоха с использованием размера мини-пакета равного 6 обучающих примеров.

Подходы к сжатию градиентов при передаче данных

Одной из основных проблем при обучении нейронной сети на нескольких вычислительных устройствах является низкая скорость передачи данных между вычислительными узлами. Так при объединении нескольких персональных компьютеров в кластер, скорость передачи данных между двумя компьютерами будет составлять 125 мегабайт в секунду.

Данная проблема рассматривается во многих работах [2, 3] при обучении нейронных сетей на кластерах. В результате проведенного анализа были рассмотрены различные подходы к решению данной проблемы:

1. Увеличение размера мини-пакета в процессе обучения нейронной сети.
2. Передачи градиентов в типе данных, использующем меньшее количество бит.
3. Передача градиентов по порогу, если размер накопленного градиента превышает заданный порог.
4. Сжатие градиентов на основе архивации байтов.

В данной работе используется подход на основе сжатия градиентов на основе архиватора байтов, но данный подход не был найден в процессе анализа аналогичных работ. Планируется дальнейший аналитический обзор данных подходов.

Было проведено тестирование подходов, основанных на увеличении размера мини-пакета, передача градиентов в типе данных с меньшим количеством бит и сжатием градиентов на основе архиватора.

Преобразование градиентов в тип данных int8

В данной работе рассмотрен подход на основе преобразования градиентов из типа данных float32 в тип данных int8. Данный подход позволяет сжать градиенты в 4 раза, что позволяет значительно ускорить скорость передачи данных между устройствами по сети Ethernet.

Преобразование градиентов в типе данных float32 в тип данных int8 приводит к потере десятичной части градиента. Для решения данной проблемы проводится нормализация градиентов.

В данной работе нормализация заключается в поиске максимального градиента по модулю. Далее все градиенты делятся на найденное максимальное значение градиента, что позволяет привести значения градиентов к интервалу от -1 до 1. Чтобы преобразовать градиенты к целочисленному интервалу все градиенты умножаются на 127, таким образом значения градиентов находятся в интервале от -127 до 127. Далее применяется преобразование типа данных градиентов из float32 в тип данных int8.

Обратное преобразование сжатых градиентов производится на основе применения противоположных операций в обратном порядке. Применяется деление градиентов на 127, далее градиенты делятся на максимальное значение градиента.

Данная нормализация позволяет преобразовать градиенты в тип данных int8, но приводит к потере точности градиента. Влияние, данное подхода на качество обучения нейронной сети, было протестировано и описано ниже.

Сжатие градиентов в типе данных int8

В данной работе рассматривается подход, основанный на сжатии градиентов на основе сжатия байтов без потерь. Данная модификация заключается в преобразовании градиентов в байтовый вид. Далее полученный набор данных сжимается архиватором в оперативной памяти.

Для реализации сжатия байтов была использована библиотека Python mgzip, данная библиотека поддерживает сжатие и распаковку с использованием нескольких ядер центрального процессора. Среднее время сжатия градиентов нейронной сети в типе данных int8 составляет 0.0067 секунды.

Данная модификация позволила сжать градиенты нейронной сети в 52 раза, что позволяет ускорить передачу данных между персональными компьютерами.

Результаты тестирования на основе передачи градиентов в типе данных float32

Было проведено обучение и тестирование нейронной сети U-Net на выборке данных Vaihingen. Обучение проводилось на 1 и 6 персональных компьютерах. Градиенты нейронной сети передавались в типе данных float32 без изменений. Обучение нейронной сети проводилось в течение 50 эпох на каждой машине (суммарно 300 эпох на 6 компьютерах).

Результаты точности обучения нейронной сети представлены на графике ниже.

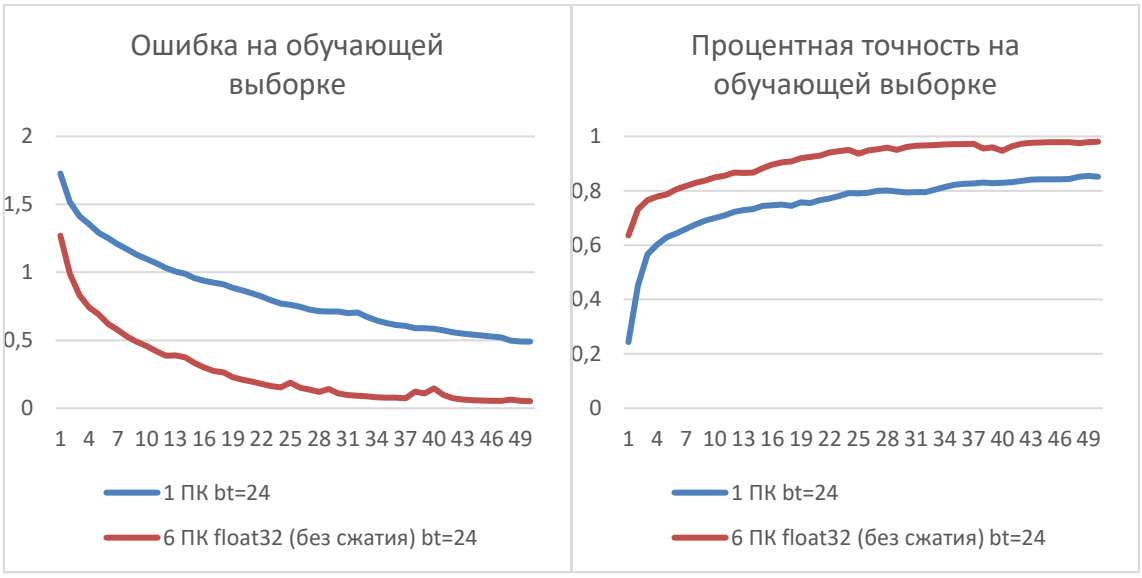

Рис. 4. Точность работы нейронной сети, обученной на 1 ПК и на 6 ПК

Результаты времени обучения представлены в таблице ниже.

Таблица 1

Время обучения на 1 ПК и 6 ПК на основе передачи градиентов в типе данных float32

	1 ПК	6 ПК
Среднее время на 1 эпоху	34.63 (сек.)	96.996 (сек.)
Время синхронизации	-	68.7 (сек.)

По полученным результатам на рисунке 1 видно, что для обучения нейронной сети на 6 ПК потребовалось меньшее количество эпох. Так ошибка равная 0,5 на 1 ПК была получена на 48 эпохе, а соответствующая ошибка на 6 ПК была получена на 9 эпохе. Таким образом, на обучение нейронной сети на 6 ПК потребовалось в 5 раз меньшее количество эпох.

На обучающей выборке точность 80 % на 1 ПК была достигнута на 28 эпохе, а соответствующая точность на 6 ПК была достигнута на 6 эпохе. Таким образом, для обучения нейронной сети на 6 ПК потребовалось в 4,5 раза меньшее количество эпох.

Для обучения нейронной сети на 1 ПК для достижения точности 80 % потребовалось около 970 секунд, а для достижения соответствующей точности на 6 ПК потребовалось 582 секунды. Таким образом, фактическое ускорение обучения составляет 1,7 раза.

На основе результатов, представленных в таблице 1, можно сделать вывод, что для обучения нейронной сети на 1 эпохе на 6 ПК требуется в 2,8 раза большее количество времени (1 эпоха на 6 ПК равноценна 6 эпохам на 1 ПК). Для обучения на 6 эпохах на 1 ПК требуется 208 секунд. Таким образом, вычисления были ускорены в 2.14 раза.

Результаты тестирования на основе передачи градиентов в типе данных float16 и int8

На основе результатов, представленных в таблице 1, можно сделать вывод, что одной из основных проблем обучения нейронной сети на 6 ПК является объем передаваемых данных по сети Ethernet. Уменьшение объема передаваемых данных градиентов по сети Ethernet является ключевым фактором, влияющим на ускорение обучения нейронной сети. Было проведено обучение нейронной сети на 6 ПК с разными вариантами сжатия градиентов:

- Передача градиентов в типе данных float16;
- Передача градиентов в типе данных int8;
- Передача градиентов в типе данных int8 со сжатием байтов.

Результаты точности обучения нейронной сети представлены на графике ниже.

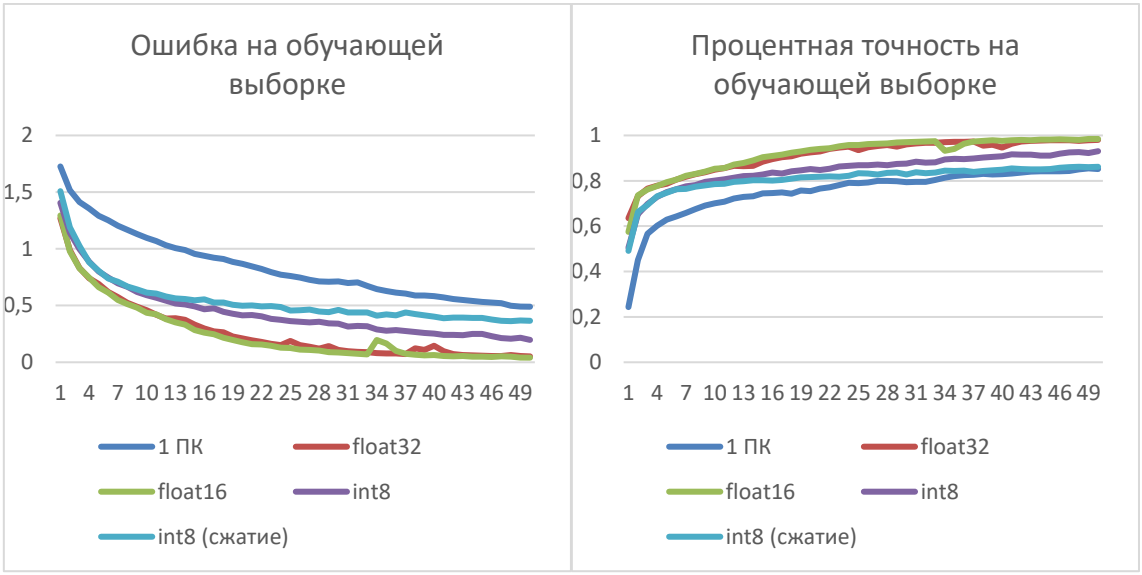

Рис. 5. Тестирование вариантов сжатия градиентов

Результаты времени обучения представлены в таблице ниже.

Таблица 2

Время обучения на 6 ПК на сжатия градиентов при передаче данных

	float32	float16	int8	int8 (сжатый)
Среднее время на 1 эпоху	96.996 (сек.)	67.51 (сек.)	52.57 (сек.)	43.62 (сек.)
Сжатие градиентов НС	x1	x2	x4	x208
Время синхронизации	68.7 (сек.)	40.6 (сек.)	21.5 (сек.)	17 (сек.)
Размер нейронной сети	8.33 Мбайт	4.16 Мбайт	2.08 Мбайт	0.04 Мбайт

По представленным данным на рис. 2 видно, что при изменении типа данных с float32 на float16, при передаче градиентов точность обучения нейронной сети не изменилась. Однако по результатам, представленным в таблице 2 видно, что время необходимое на одну эпоху сократилось на 30 %.

По результатам, представленным на рис. 2 и в таблице 2 видно, что передача градиентов в типе данных int8 с использованием сжатия байтов позволяет ускорить обучение в 2 раза. Однако при данном подходе int8 происходит значительная потеря качества обучения нейронной сети. Для решения данной проблемы существует множество подходов, один из которых представлен в следующей ста- тье [2].

Нейронная сеть на 6 ПК обучается значительно быстрее чем на 1 ПК. Так в типе данных int8 с использованием сжатия байтов на 1 эпоху на 6 ПК требуется 43.6 секунд. Для обучения на 6 эпохах на 1 ПК требуется 208 секунд. Таким образом, вычисления были ускорены в 4.76 раза.

По полученным результатам видно, что проанализированные подходы позволили сжать нейронную сеть в 208 раз. Увеличение точности нейронной сети, обученной на основе сжатия градиентов, является целью дальнейшей работы.

Заключение

В данной работе был проведен анализ подходов, позволяющих увеличить скорость обучения нейронной сети с использованием нескольких GPU. Проведены реализация и тестирование подходов, основанных на увеличении размера мини-пакета, сжатии градиентов на основе типа данных и сжатие градиентов в байтовом виде.

По полученным результатам можно сделать вывод, что использование нескольких персональных компьютеров позволяет значительно ускорить обучение нейронных сетей на задаче семантической сегментации изображений, на основе применения нескольких персональных компьютеров. Так объединение подходов на основе сжатия градиентов в тип данных int8 и сжатие градиентов в байтовом виде позволило сжать нейронную сеть в 208 раз.

На основе проведенного тестирования видно, что сжатие градиентов нейронной сети приводит к потере качества обучения. В дальнейшем планируется реализация и тестирование подходов, решающих проблему потери качества обучения нейронной сети.

Список использованных источников

1. Alex Krizhevsky «One weird trick for parallelizing convolutional neural networks». – 2014.

2. Seide F., Fu H., Droppo J., Li G. and Yu D. «1- bit stochastic gradient descent and its application to dataparallel distributed training of speech dnns» // Fifteenth Annual Conference of the International Speech Communication Association. – 2014.

3. Yujun Lin Deep «Gradient Compression: Reducing the Communication Bandwidth for Distributed Training», 2017.

4. «Программная реализация». – URL: https://github.com/NikolayKrivosheev/Distributed-deep-learning-on-personal-computers