Санкт–Петербургский государственный университет

Тихонков Сергей Алексеевич

Выпускная квалификационная работа

Исследование и применение алгоритмов на базе консенсуса для распределенного обучения нейронных сетей

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа СВ.5003.2017 «Программирование и информационные технологии»
Профиль «Автоматизация научных исследований»

Научный руководитель:
профессор, кафедра математического моделирования энергетических систем, д.ф. - м.н. Крылатов Александр Юрьевич

Рецензент:
доцент, зав. каф. компьютерных технологий и систем, к.ф. - м.н. Погожев Сергей Владимирович

Санкт-Петербург
2021 г.

Содержание

Введение

С каждым годом количество накопленных человечеством данных растет. Современное аппаратное обеспечение часто не позволяет проводить вычисления с большим количеством данных на одной машине. Это часто связано во-первых, с ограниченным объемом оперативной памяти компьютера, а во-вторых, с ограниченным временем на решение конкретной задачи. Область машинного обучения испытывает обе перечисленные проблемы. Часто для достижения требуемого результата необходимо обучение нейронной сети с множеством параметров на огромном объеме данных. Это препятствует прогрессу в исследованиях и разработках. В связи с этим существует потребность в методах распределенного обучения. Они позволяют не ограничиваться мощностями одной машины при конструировании новых методов в машинном обучении и ускоряют уже существующие решения.

Помимо этого на практике часто возникает потребность обрабатывать информацию разной степени приватности. В некоторых случаях при обучении нейронной сети используются данные, распространение которых запрещено или нежелательно. Отличным примером этого является какой-либо совместный проект двух банков, в ходе которого они хотят решить общую задачу, но не могут передать, пусть даже и обезличенные, данные своих клиентов друг другу. В этом случае возникает необходимость использовать методы распределенного обучения, которые обеспечивали бы изоляцию данных.

Топология сети и методы взаимодействия узлов являются важными свойствами распределенных систем. Не всегда вычислительные кластеры имеют регулярную структуру, часто доступные вычислительные ресурсы представляют собой гетерогенную компьютерную сеть, где пропускная способность каналов связи и мощности каждого узла могут сильно отличаться. В этом случае применяемые методы распределенного обучения должны быть устойчивы к неоднородности сети.

В данной работе проводится краткий обзор основных методов распределенного обучения нейронных сетей и исследование возможности применения алгоритма на базе консенсуса и алгоритма сплетен для распределенного обучения с изоляцией данных. Также рассматривается способность алгоритма на

базе консенсуса корректно решать задачу в сети с неоднородной вычислительной мощностью узлов.

Постановка задачи

Целью данной работы является исследование возможности применения алгоритма на базе консенсуса и алгоритма сплетен для распределенного обучения нейронных сетей. Для достижения поставленной цели были сформулированны следующие задачи:

- изучение предметной области и обзор существующих методов
- изучение математической теории, описывающей исследуемый подход
- реализация алгоритма и прототипа распределенной сети для проведения экспериментов
- проведение экспериментов и анализ результатов

Глава 1. Теория

1.1 Задача машинного обучения

Вспомним стандартную постановку задачи машинного обучения с учителем: есть некоторое параметрическое семейство функций $f(\bullet, \theta)$ и набор данных $(x_i, y_i)_{i=0}^{m-1}$, требуется подобрать параметры θ так, чтобы $f(\bullet, \theta)$ как можно лучше соответствовала этим данным.

Формулировка в виде задачи оптимизации:

$$\mathcal{J}(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} g(f(x_i, \theta), y_i) \to_\theta \min$$

где $g(f(x_i, \theta), y_i)$ – функция ошибки.

Часто эту задачу решают с помощью пакетного градиентного спуска (Mini Batch Gradient Descent) [1]:

$$\theta_{k+1} = \theta_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla g(f(x_i, \theta_k), y_i) \tag{1}$$

где S_k — случайно выбранное подмножество $\{0, \ldots, m-1\}$, α_k — коэффициент скорости обучения.

1.2 Алгоритм среднего консенсуса

Алгоритм консенсуса – это процесс в теории автоматического управления, используемый для достижения согласия по единому значению данных среди распределенных процессов или систем. То есть форма стабильной системы, в которой состояния $\pi(t)_i$ распределенной системы сошлись к одному значению:

$$\pi_i(t) \to \pi^*, \ 0 \leq i \leq m-1.$$

Базовый (средний) консенсус можно использовать для децентрализованного вычисления среднего нескольких чисел, каждое из которых хранится

на отдельном вычислительном узле [2]:

$$x^0 = (x_1^0, \ldots, x_p^0)$$

$$x_i^* = \frac{1}{p} \sum_{j=1}^{p} x_j^0.$$

Рассмотрим сеть агентов, принимающих решения, с динамикой $\dot{x}_i = u_i$, заинтересованных в достижении консенсуса посредством локальной коммуникации со своими соседями на графе $G = (V, E)$. То есть стремящихся к равенствам:

$$x_1 = x_2 = \ldots = x_n.$$

Это может быть выражено как $x = \alpha \mathbb{1}$, где $\mathbb{1} = (1, \ldots, 1)^T$ и $\alpha \in R$ – коллективное решение группы агентов. Пусть $A = a_{ij}$ – матрица смежности графа G. Множество соседей агента i – это N_i и определяется формулой

$$N_i = \{j \in V : a_{ij} \neq 0\}; \qquad V = \{1, \ldots, n\}.$$

Агент i коммуницирует с агентом j, если j является соседом i. Набор всех узлов и их соседей определяет набор ребер графа

$$E = \{(i, j) \in V \times V : a_{ij} \neq 0\}.$$

Динамический граф $G(t) = (V, E(t))$ – это граф, в котором набор ребер $E(t)$ и матрица смежности $A(t)$ изменяются во времени. Ясно, что множество соседей $N_i(t)$ каждого агента в динамическом графе также является изменяющимся во времени множеством.

В [3] показано, что линейная система

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}(x_j(t) - x_i(t)) \tag{2}$$

представляет собой алгоритм распределенного консенсуса, т.е. гарантирует сходимость к коллективному решению через локальные межагентные взаи-

модействия. Предполагая, что граф неориентированный ($a_{ij} = a_{ji}, \forall i, j \in V$), следует, что сумма состояний всех узлов является инвариантной величиной, или $\sum_i \dot{x}_i = 0$. В частности, применив это условие для $t = 0$ и $t = \infty$ получаем:

$$\alpha = \frac{1}{n} \sum_i x_i.$$

Другими словами, если консенсус достигается асимптотически, то коллективное решение обязательно равно среднему значению начального состояния всех узлов.

В компактном виде изменения системы (2) можно записать как

$$\dot{x} = -Lx$$

где L – лапласиан графа G. По определению, L имеет правый собственный вектор, равный $\mathbb{1}$, связанный с нулевым собственным значением из-за тождества $L\mathbb{1} = 0$. В случае неориентированных графов лапласиан графа удовлетворяет следующему свойству суммы квадратов (ССК):

$$x^T Lx = \frac{1}{2} \sum_{(i,j) \in E} a_{ij}(x_j - x_i)^2. \tag{3}$$

Определив квадратичную функцию несогласия как

$$\varphi(x) = \frac{1}{2} x^T Lx$$

становится ясно, что алгоритм (2) такой же, как

$$\dot{x} = -\nabla \varphi(x)$$

или алгоритм градиентного спуска. Этот алгоритм асимптотически сходится при соблюдении двух условий:

1. L - положительно полуопределенная матрица;
2. единственное решение системы (2) это $\alpha\mathbb{1}$ для некоторого α.

Оба эти условия выполняются для связного графа и следуют из ССК-свойства лапласиана графа в (3). Следовательно, средний консенсус достигается асимптотически для всех начальных состояний. Этот факт резюмируется в [2] следующей леммой:

Лемма 1: Пусть G - связный неориентированный граф. Тогда алгоритм (2) асимптотически решает задачу среднего консенсуса для всех начальных состояний.

1.3 Быстрая сходимость

Выяснив, что алгоритм (2) асимптотически решает задачу среднего консенсуса, можно задаться вопросом о скорости его сходимости. В [4] говорится о том, что можно выбрать такие веса a_{ij} ребер графа G, чтобы алгоритм сходился с максимальной скоростью. Иногда некоторые оптимальные веса могут быть даже отрицательными, такой случай рассматривается далее в параграфе 4.6. Для нахождения оптимальных весов необходимо решить задачу оптимизации, в которой максимизируется второе из минимальных собственных значений лапласиана L графа G при известных ограничениях на веса ребер. Стоит отметить, что второе собственное значение лапласиана графа возникает из теории графов и называется алгебраическая связность.

1.4 Консенсус и задача МО

Применим алгоритм среднего консенсуса к задаче машинного обучения. Пусть каждый вычислительный узел идентифицирован своим индексом $t = \{1, \dots, p\}$ и хранит свою копию параметров θ^t. Происходит разделение исходных данных $S = \cup_{t=1}^{p} S^t$, причем $S_i \cup S^j = \oslash$ для $i \neq j$. Каждое S^t является множеством исходных данных соответствующего узла t. Тогда, в соответствии с [5], каждый шаг алгоритма для каждого агента t будет выглядеть как:

- Обучить текущие веса θ_k^t на подмножестве локальных данных:

$$\theta_{k+\frac{1}{2}}^t = \theta_k^t - \alpha_k \frac{1}{|S_k^t|} \sum_{i \in S_k^t} \nabla g(f(x_i, \theta_k), y_i) \tag{4}$$

- Сделать консенсусное усреднение параметров с соседями:

$$\theta^t_{k+1} = \sum_{i \in N_t} a_{ti} \theta^i_{k+\frac{1}{2}} \tag{5}$$

где N_t – соседи узла t, причем $t \in N_t$.

Шаг (4) отличается от стандартного подхода (1) только тем, что множество S^t_k зависит от t, эти множества не пересекаются для разных t, то есть происходит разделение данных между вычислительными узлами.

Шаг (5) — усреднение параметров модели между соседними узлами. Здесь происходит обмен информацией между узлами, причем этот шаг не обязан выполняться каждую итерацию.

Как следует из теории, изложенной в предыдущем параграфе, эти шаги эквивалентны обучению одиночной модели с использованием батча $S^1_k \cup S^2_k \cup \ldots \cup S^p_k$.

В случае распределенного обучения нейронных сетей под достижением консенсуса мы подразумеваем асимптотическую сходимость параметров моделей на каждом узле $\theta^*_1 = \theta^*_2 = \ldots = \theta^*_p$. Однако на практике удобно не отслеживать факт достижения консенсуса с какой-то точностью, а делать конечное количество итераций, заданное вначале.

1.5 Алгоритм сплетен

Кроме алгоритма среднего консенсуса к задаче машинного обучения можно применить распределенные алгоритмы сплетен [6]. Это класс алгоритмов для обмена информацией и вычислений в произвольно связанной сети узлов. Их основным отличием от алгоритма среднего консенсуса является то, что на конкретной итерации каждый узел может обменяться информацией со случайным соседом в сети. Данный факт означает, что матрица A связей графа динамически меняется во времени, что учитывалось в (2), а значит на алгоритм сплетен распространяется уже описанная в параграфе 1.2 теория. На практике консенсусное усреднение отличается от (5) дополнительным шагом – выбором случайных пар вершин графа для усреднения:

- Выбрать случайные пары вершин для усреднения:

$$pairs = kRandomElementsFrom(E),$$

где E – множество ребер графа распределенной сети.

- Сделать консенсусное усреднение параметров с соседями:

$$\theta^i_{k+1} = \theta^j_{k+1} = \frac{\theta^i_{k+\frac{1}{2}} + \theta^j_{k+\frac{1}{2}}}{2}, \forall (i,j) \in pairs.$$

Усреднения между узлами могут происходить асинхронно, что делает алгоритм сплетен перспективным для применения в слабосвязанных и непостоянных сетях. А количество шагов на этапе усреднения весов может быть выбрано пользователем или автоматически для сохранения высокой скорости сходимости при небольшой нагрузке на коммуникационную сеть. Например, это актуально для мобильных сетей.

Глава 2. Обзор

Подходы к параллельным вычислениям делятся на два основных направления:

- параллелизм по данным (Data Parallel)
- модельный параллелизм (Model Parallel)

В параллелизме по данным происходит вычисление искомой функции каждым узлом на своей части данных. В модельном параллелизме каждый узел вычисляет на общих данных какую-то часть функции. Выбранный для экспериментов подход является разновидностью Data Parallel.

Наиболее часто используемые распределенные системы машинного обучения делятся на синхронные / асинхронные и на централизованные / децентрализованные. На практике их применяют для распределенного вычисления суммы

$$\sum_{i \in S_k} \nabla g(f(x_i, \theta_k), y_i)$$

в 1. Кратко опишу основные существующие подходы:

При синхронном параллельном стохастическом градиентном спуске (S-PSGD) [7] каждый узел хранит локальную копию основной модели. На каждой итерации он получает минибатч данных от центрального сервера, вычисляет градиент и передает его обратно серверу. После того как центральный сервер получил данные со всех узлов, он вычисляет среднее значение градиента и дает задание каждому узлу обновить веса модели в соответствии с этим значением. Затем наступает следующая итерация.

В асинхронном параллельном стохастическом градиентном спуске (A-PSGD) [8, 9, 10, 11] тот же подход, но асинхронность достигается за счет дополнительного разрешения узлам использовать устаревшие значения весов при вычислении градиентов. Это позволяет избавиться от необходимости ждать самый медленный узел на каждом шаге синхронизации параметров.

AllReduce стохастический градиентный спуск (AllReduce-SGD) [12, 13] похож на S-PSGD, однако в нем нет сервера параметров, а узлы образуют кольцевую сеть. На очередной итерации каждый узел вычисляет градиент по

минибатчу исходных данных. После чего данные передаются по сети, используя парадигму AllReduce, пока каждый узел не получит значения градиентов со всех остальных узлов. Градиенты усредняются и каждый узел обновляет веса своей локальной модели. Из-за наличия фазы синхронизации и большого количества взаимодействий между узлами, данный метод плохо себя показывает в сетях с низкой пропускной способностью.

В децентрализованном параллельном стохастическом градиентном спуске (D-PSGD) [14, 15, 16] все узлы связаны в сеть, которую можно представить в виде связного графа. Каждый узел имеет свою локальную копию модели. На каждой итерации все узлы вычисляют градиенты по локальным данным и усредняют их, используя градиенты своих соседей в сети. Используя полученное значение градиента, узел обновляет веса своей локальной модели. В данном методе обмен данными и обучение могут происходить параллельно, но из-за синхронизации все равно существует зависимость сети от самого медленного узла.

В 2018 году исследователи из IBM предложили [17] асинхронный децентрализованный стохастический градиентный спуск (AD-PSGD), основанный на сплетнях. Он отличается от D-PSGD тем, что в каждом узле усредняются не градиенты, а веса модели, и не по всем соседям узла, а с одним случайным. В 2020 году группа исследователей [5] обобщила и дополнила существующие наработки в этой сфере. В частности, они проанализировали ситуацию, когда случайно выбирается не один, а произвольное число соседей узла. Данный новый метод и был взят за основу для исследования.

Глава 3. Алгоритм

Для проведения экспериментов был разработан однопоточный прототип сети, который позволяет обучать, в том числе используя GPU, несколько экземпляров нейронных сетей используя алгоритм среднего консенсуса или алгоритм сплетен, а также собирать, сохранять и визуализировать необходимую для последующего анализа статистику.

Для реализации использовался язык python с фреймворком для работы с нейронными сетями pytorch. Стоит отметить, что инициализация весов нейронных сетей в данном фреймворке происходит случайным образом, из-за чего приходится уравнивать их вручную. Если проигнорировать данный факт, то метод становится менее точным, первые эпохи обучения тратятся на компенсацию случайного разброса параметров, а результаты имеют большую дисперсию.

3.1 Средний консенсус

Основные этапы алгоритма обучения нейронных сетей с использованием алгоритма среднего консенсуса:

Algorithm 1 Средний консенсус

Определить топологию сети W и количество узлов p.

Инициализировать локальные модели θ_0^t на всех узлах $t \in \{1, \dots, p\}$.

Разделить исходные данные по узлам $S = \cup_{t=1}^{p} S^t$, причем $S^i \cap S^j = \oslash$ для $i \neq j$.

Определить расписание изменения коэффициента скорости обучения $\alpha(k)$.

Определить функцию ошибки $g(f(x_i,), y_i)$, где $(x_i, y_i) \in S$.

Определить количество эпох n, итераций в каждой эпохе m, общее количество итераций $K = mn$.

Определить частоту итераций консенсуса q.

for $k \in \{0 \dots K-1\}$ **do**

  for $t \in \{0 \dots p-1\}$ **do**

    $S_k^t \leftarrow RandomSubset(S^t)$. ▷ Выбор батча данных

    $\theta_{k+\frac{1}{2}}^t \leftarrow \theta_k^t - \alpha_k \frac{1}{|S_k^t|} \sum_{i \in S_k^t} \nabla g(f(x_i, \theta_k), y_i)$ ▷ Итерация обучения

if $k = 0 \pmod q$ **then** ▷ Итерация консенсуса

$\theta^t_{k+1} \leftarrow \sum_{j \in N_t} w_{tj} \theta^j_{k+\frac{1}{2}}$

else

$\theta^t_{k+1} \leftarrow \theta^t_{k+\frac{1}{2}}$.

end if

end for

end for

3.2 Несбалансированные данные

Предыдущий алгоритм был обобщен для имитации поведения агентов с разной вычислительной мощностью. В этом случае подразумевается, что данные распределяются по узлам пропорционально их вычислительной мощности:

Algorithm 2 Консенсус с несбалансированными данными

Определить топологию сети W и количество узлов p.

Инициализировать локальные модели θ^t_0 на всех узлах $t \in \{1, \ldots, p\}$.

Разделить исходные данные по узлам $S = \cup_{t=1}^{p} S^t$, причем $S^i \cap S^j = \oslash$ для $i \neq j$. Размеры множеств S^t необязательно равны.

Определить расписание изменения коэффициента скорости обучения $\alpha(k)$.

Определить функцию ошибки $g(f(x_i,), y_i)$, где $(x_i, y_i) \in S$.

Определить количество эпох n, итераций в каждой эпохе m, общее количество итераций $K = mn$.

Определить частоту обучения на одном батче для каждого агента qt^t.

Определить частоту итераций консенсуса для каждого агента qc^t.

for $k \in \{0 \ldots K-1\}$ **do**

for $t \in \{0 \ldots p-1\}$ **do**

$S^t_k \leftarrow RandomSubset(S^t)$. ▷ Выбор батча данных

if $k = 0 \pmod{qt^t}$ **then** ▷ Итерация обучения

$\theta^t_{k+\frac{1}{2}} \leftarrow \theta^t_k - \alpha_k \frac{1}{|S^t_k|} \sum_{i \in S^t_k} \nabla g(f(x_i, \theta_k), y_i)$

else

$\theta^t_{k+\frac{1}{2}} \leftarrow \theta^t_k$.

end if

if $k = 0 \pmod{qc^t}$ **then** ▷ Итерация консенсуса

$\theta^t_{k+1} \leftarrow \sum_{j \in N_t} w_{tj} \theta^j_{k+\frac{1}{2}}$

else

$\theta^t_{k+1} \leftarrow \theta^t_{k+\frac{1}{2}}$.

end if

end for

end for

3.3 Сплетни

Для тестирования работоспособности метода использовалась однопоточная реализация алгоритма. Однако при применении данного метода для обучения нейронных сетей на конечных устройствах, шаг усреднения весов моделей может быть выполнен асинхронно.

Algorithm 3 Сплетни

Определить топологию сети W и количество узлов p.

Инициализировать локальные модели θ^t_0 на всех узлах $t \in \{1, \ldots, p\}$.

Разделить исходные данные по узлам $S = \cup_{t=1}^{p} S^t$, причем $S^i \cap S^j = \oslash$ для $i \neq j$.

Определить расписание изменения коэффициента скорости обучения $\alpha(k)$.

Определить функцию ошибки $g(f(x_i,), y_i)$, где $(x_i, y_i) \in S$.

Определить количество эпох n, итераций в каждой эпохе m, общее количество итераций $K = mn$.

Определить частоту итераций консенсуса q.

Определить количество шагов усреднения b на каждой итерации консенсуса.

for $k \in \{0 \ldots K-1\}$ **do**

for $t \in \{0 \ldots p-1\}$ **do**

$S^t_k \leftarrow RandomSubset(S^t)$. ▷ Выбор батча данных

$\theta^t_{k+1} \leftarrow \theta^t_k - \alpha_k \frac{1}{|S^t_k|} \sum_{i \in S^t_k} \nabla g(f(x_i, \theta_k), y_i)$ ▷ Итерация обучения

end for

if $k = 0 \pmod{q}$ **then** ▷ Итерация консенсуса

$pairs \leftarrow RandomChoices(E, b)$ ▷ Выбор b случайных пар вершин

for (i, j) $\in$ pairs **do**

$\theta^i_{k+1} \leftarrow \theta^j_{k+1} \leftarrow \frac{1}{2}(\theta^i_{k+1} + \theta^j_{k+1})$

end for

end if

end for

Глава 4. Эксперименты

4.1 Выбор модели и датасета

В первую очередь стояла задача проводить эксперименты на известном датасете, чтобы иметь возможность отталкиваться от чужих результатов. Была выбрана область классификации изображений и набор данных «CIFAR-10». Он входит в топ популярнейших датасетов, а также не требует огромных вычислительных ресурсов. Проведя анализ топ state of the art моделей машинного обучения в этой области, пришел к выводу, что большинство решений – это модификации Residual Network (ResNet) моделей. Исходя из этого принял решение, что нужно проводить эксперименты на каком-нибудь конкретном представителе, выбрав реализацию исходя из:

- скорости обучения
- количества параметров
- наличия реализации на фреймворке с необходимой гибкостью и доступом к параметрам для их модификации

В итоге была отобрана реализация ResNet-20 [18].

После того, как исследуемые методы неплохо сработали для упомянутой пары модели и датасета, встала задача выбора следующих бенчмарков. Исходя из тех же соображений, выбор пал на датасеты «CIFAR-100» и «FashionMNIST». Для «CIFAR-100» применялась та же модель, что и для «CIFAR-10». Для «FashionMNIST» была написана собственная реализация Convolutional Neural Network (CNN). Общую информацию о датасетах и моделях можно наблюдать в Таблице 1:

Таблица 1: Датасеты и модели.

Название датасета	CIFAR-10	CIFAR-100	FashionMNIST
Размер обучающей выборки	50000	50000	60000

Размер валидационной выборки	10000	10000	10000
Количество классов	10	100	10
Название модели	ResNet20	ResNet20	CNN
Количество оптимизируемых параметров	0.27M	0.27M	1.48M

4.2 Преобразование данных

Для воспроизводимости результатов считаю необходимым зафиксировать преобразования, которые осуществлялись над данными перед началом обучения. Использовались только распространенные в области классификации изображений преобразования:

- CIFAR-10 и CIFAR-100
  - Поворот изображения по горизонтали с вероятностью 0,5
  - Случайное обрезание до размера 32 на 32 пикселей с отступом 4 пикселя от края
  - Нормирование по всем каналам (R, G, B), с параметрами:
    * среднее: (0.485, 0.456, 0.406)
    * среднеквадратическое отклонение: (0.229, 0.224, 0.225)
- FashionMNIST
  - Нормирование с параметрами:
    * среднее: 0.28604
    * среднеквадратическое отклонение: 0.35302

4.3 Разминка модели

Исследователи из Facebook, применяя [19] схожий метод к другой модели и датасету, получили, что предварительная разминка весов моделей перед

основным обучением помогает достигнуть лучшего качества и делает процесс обучения более стабильным. Разминка часто успешно применяется при обучении одиночных моделей. Идея заключается в том, чтобы уменьшить скорость обучения на первых эпохах. Действительно, эксперименты показали, что данный подход имеет место быть и при распределенном обучении нескольких моделей с использованием алгоритма среднего консенсуса. Применив на датасете «CIFAR-10» постепенную разминку, а именно линейное увеличение скорости обучения до значения стартовой каждую итерацию на протяжении первых пяти эпох, удалось получить прирост точности в среднем на 0.4+-0.05%.

4.4 Увеличение скорости обучения

Исторически в машинном обучении выработалось [20] общее правило: при увеличении размера батча следует увеличивать скорость обучения. Интуитивно ясно, что чем более точно мы способны посчитать направление локального минимума, тем с большим шагом следует двигаться в этом направлении. Как уже было сказано, применение исследуемых методов эквивалентно обучению одиночной модели с увеличенным батчем, размером в сумму размеров батчей всех узлов сети. Пусть в сети n узлов. В этом случае можно уменьшить локальный размер батча в n раз или применить упомянутое правило и увеличить скорость обучения в n раз. Второй вариант предпочтительнее, ведь уменьшение размера батча повлияет на скорость обучения модели. Однако экспериментальным путем было выяснено, что данный подход следует применять только при малых n. Сильное увеличение скорости обучения влечет за собой нестабильный процесс обучения, а так же искажает исходный набор гиперпараметров, который был подобран специально для достижения близких к state of the art результатов. Возможно, увеличение времени обучения и дополнительный подбор правил уменьшения скорости обучения могут исправить ситуацию, но это еще более серьезное расхождение с оригинальным рецептом обучения.

Успешные и неудачные результаты применения правила увеличения скорости обучения можно наблюдать в Таблице 2. В ней и далее жирным

шрифтом отмечены те результаты экспериментов, которые (с некоторыми оговорками) можно считать успешными, то есть точность моделей агентов приближенно равна точности одиночной модели. Указанные в таблице топологии подробнее представлены в параграфе 4.6.

Таблица 2: Результаты применения LSR.

Топология	**Датасет**	**Метод**	**Размер батча**	**Точность агентов (%)**	**Точность одиночной модели (%)**
Тор 6×6	CIFAR-10	средний консенсус	32	74.88+-1.15	91.5
Плотный граф с 5 вершинами	CIFAR-10	средний консенсус	32	**91.14+-0.01**	91.16
Разреженный граф с 10 вершинами	CIFAR-10	средний консенсус	32	90.17+-0.10	90.77
Цикл из 4 агентов	CIFAR-10	средний консенсус	32	**92.13+-0.09**	91.5
Цикл из 4 агентов	CIFAR-10	сплетни (b=4)	32	**92.02+-0.06**	91.5
Цикл из 4 агентов	CIFAR-10	сплетни (b=2)	32	**91.96+-0.09**	91.5
Цикл из 4 агентов	CIFAR-10	сплетни (b=1)	32	**91.64+-0.00**	91.5
Цикл из 4 агентов	Fashion MNIST	средний консенсус	100	91.80+-0.16	92.18
Цикл из 4 агентов	CIFAR-100	средний консенсус	128	67.67+-0.04	68.21
Цикл из 4 агентов	CIFAR-100	сплетни (b=4)	128	66.63+-0.16	68.21

Цикл из 4 агентов	CIFAR-100	сплетни (b=1)	128	67.23+-0.20	68.21

4.5 Топология сети

Как упоминалось в теории, скорость сходимости консенсуса зависит от алгебраической связности графа. Чем выше связность, тем выше скорость. Следовательно, если есть возможность, стоит выбирать топологию сети с оглядкой на этот факт. В том числе, высокой алгебраической связностью обладают графы:

- с маленьким диаметром
- с высокой реберной связностью

На примере экспериментов с 50 агентами, образующими циклы с разным количеством хорд (Рис. 1), можно проиллюстрировать зависимость скорости сходимости консенсуса от диаметра графа (Рис. 2).

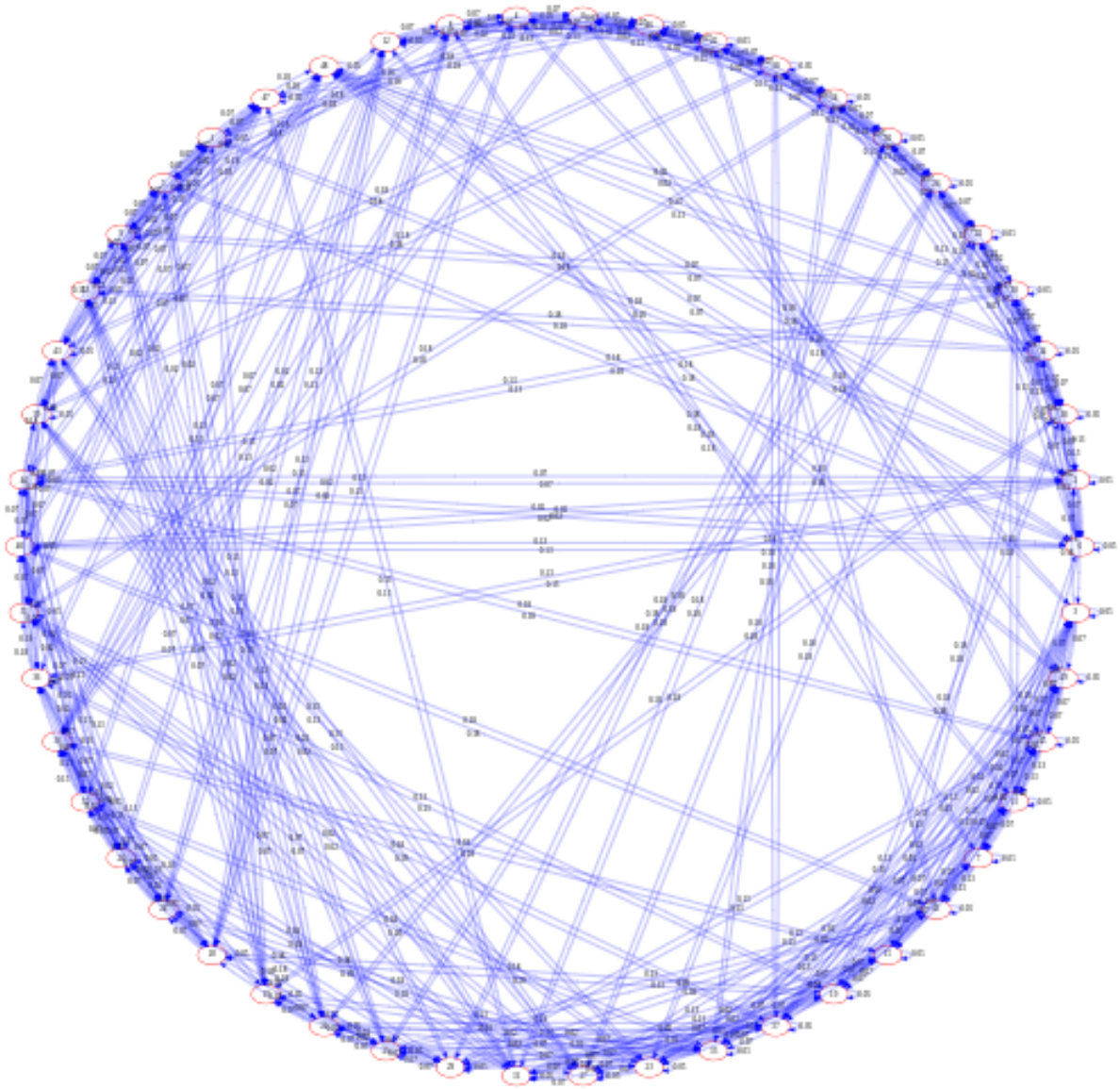

Рис. 1: Цикл из 50 агентов с хордами.

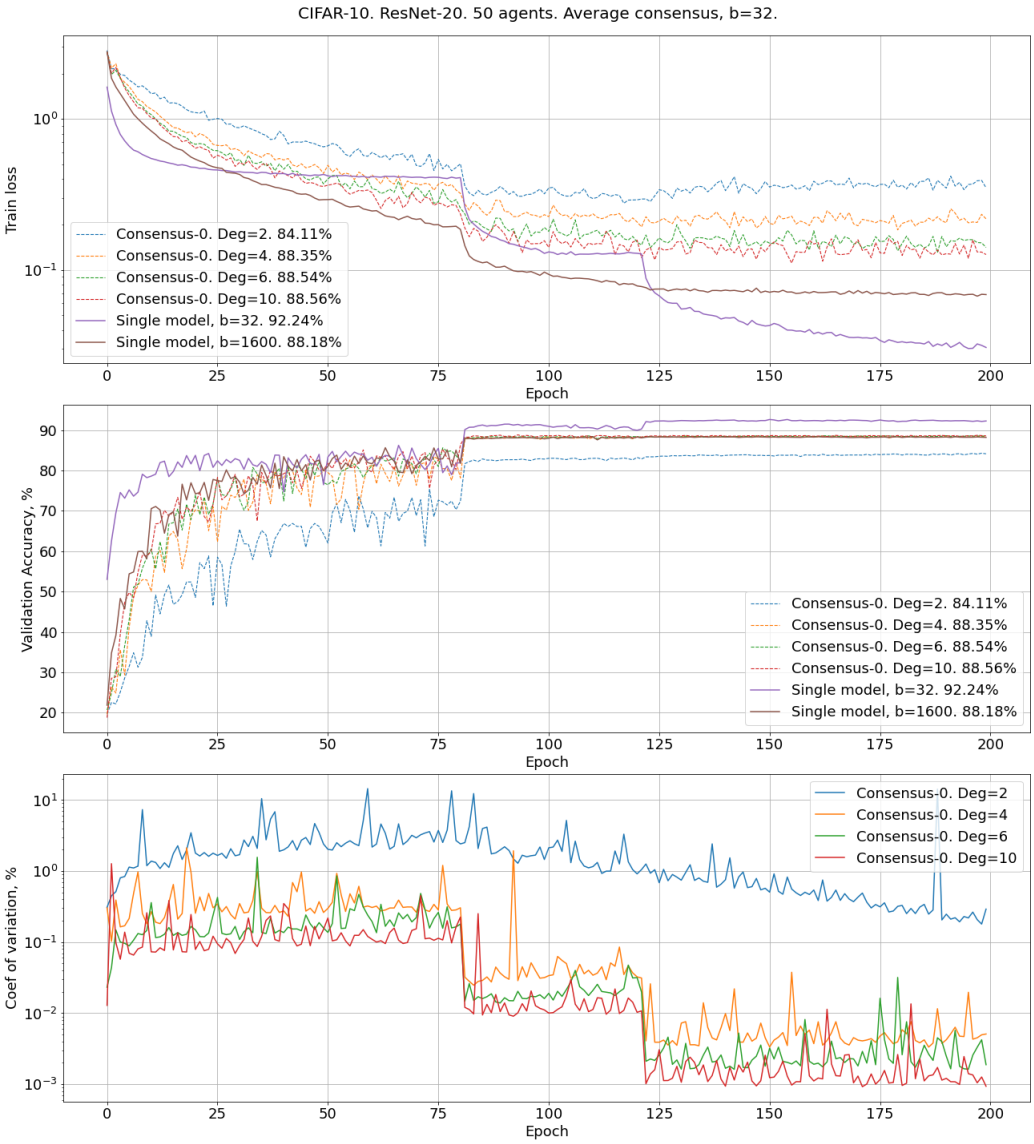

Рис. 2: Эксперимент над циклом из 50 агентов с разным количеством хорд.

Существует особый тип графов, называемых экспандерами. Они обладают высокой реберной связностью и хорошо подходят для данной задачи. Процесс обучения 25 агентов, образующих экспандер (Рис. 3), с разной частотой итерации консенсуса, представлен на (Рис. 4).

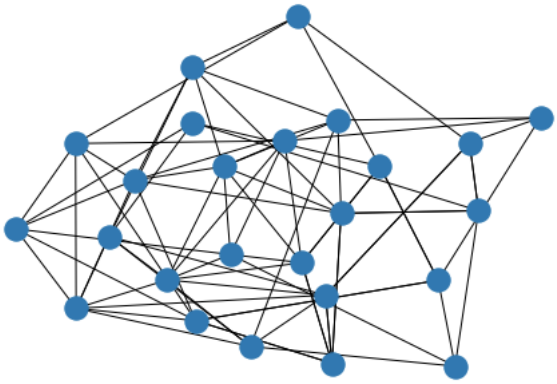

Рис. 3: Экспандер из 25 агентов.

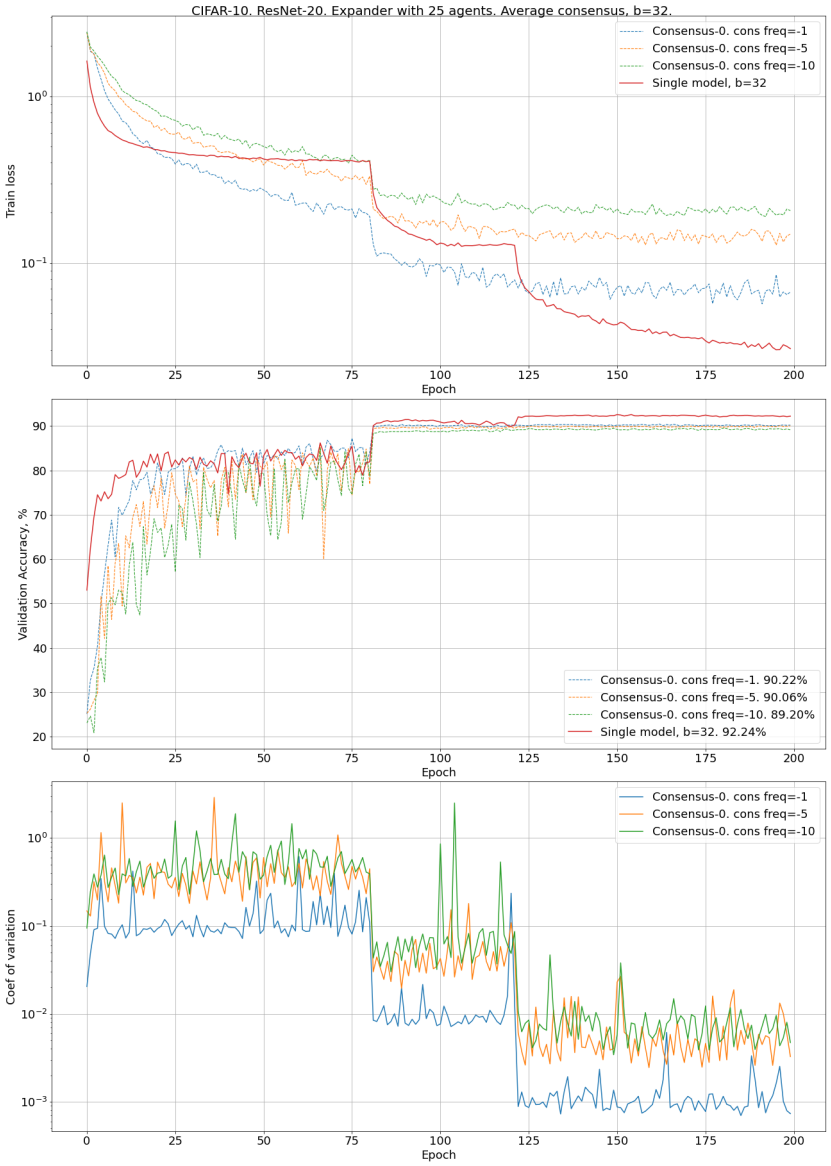

Рис. 4: Эксперимент над экспандером из 25 агентов с разной частотой консенсуса.

4.6 Основные эксперименты

Кроме перечисленных в предыдущем параграфе топологий сети, эксперименты проводились на цикле из 4 агентов (Рис. 5), торе 6×6 (Рис. 6), полной топологии размера 36 (Рис. 7) и случайном графе, для которого среди оптимальных весов, найденных с помощью упомянутой теории для быстрой сходимости, присутствуют отрицательные числа (Рис. 8). Ниже представлены сводные таблицы (Таб. 3 – 7) и графики (Рис. 9 – 12), полученные в процессе проведения экспериментов. В том числе на несбалансированных данных с имитацией разной мощности агентов.

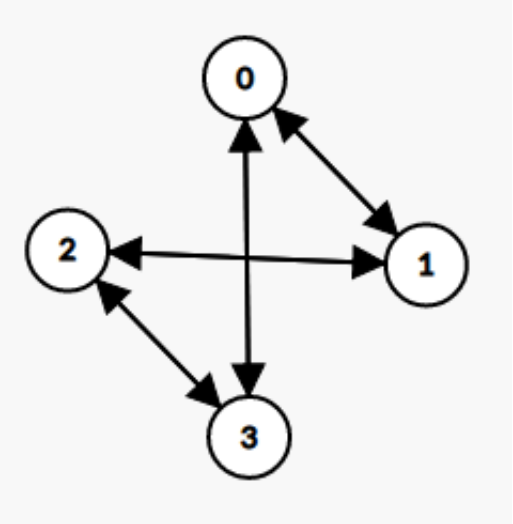

Рис. 5: Цикл из 4 агентов.

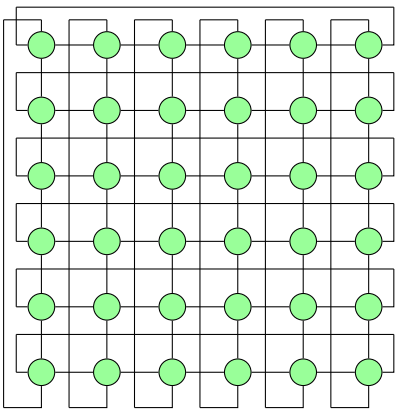

Рис. 6: Тор 6×6.

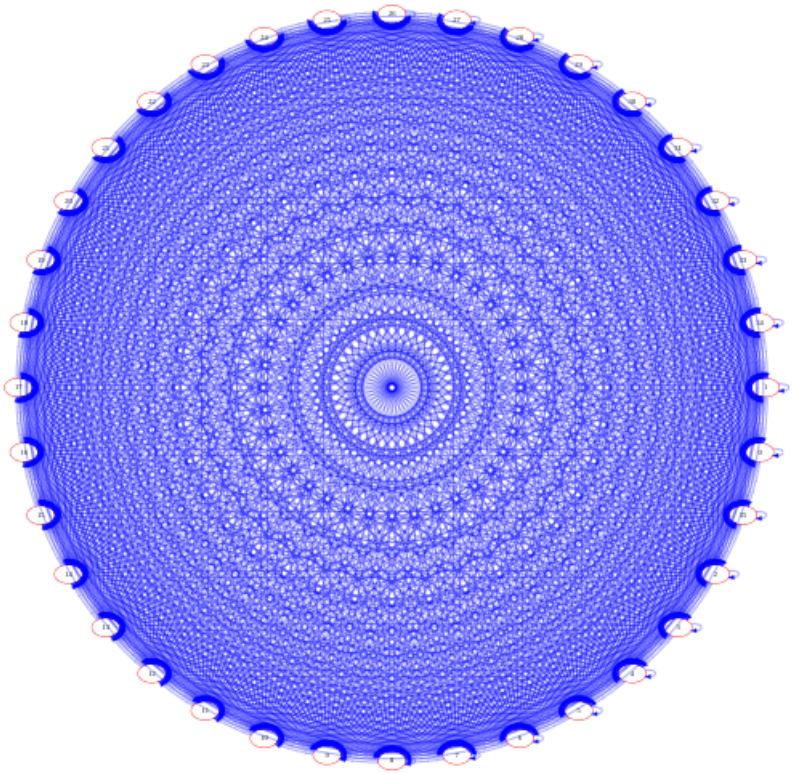

Рис. 7: Полный граф с 36 вершинами.

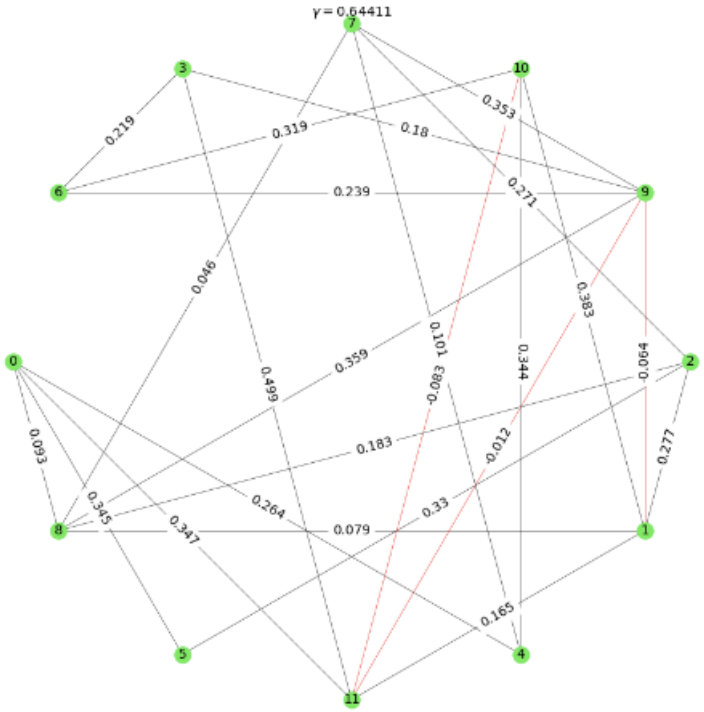

Рис. 8: Граф с отрицательными весами.

Таблица 3: Средний консенсус на сбалансированных данных. Датасет CIFAR-10.

Топология	**Размер батча агента**	**Точность агентов (%)**	**Точность одиночной модели (%)**	**Размер батча одиночной модели**
Цикл из 4 агентов	32	**92.13+-0.04**	91.79	32*4
Экспандер из 25 агентов	32	**90.12+-0.07**	89.31	32*25
Экспандер из 25 агентов, консенсус реже в 5 раз	32	**89.89+-0.10**	89.31	32*25
Экспандер из 25 агентов, консенсус реже в 10 раз	32	**89.36+-0.11**	89.31	32*25
Граф из 50 агентов со 100 ребрами	32	**88.40+-0.12**	88.18	32*50
Граф из 50 агентов с 150 ребрами	32	**88.51+-0.10**	88.18	32*50
Граф из 50 агентов со 250 ребрами	32	**88.61+-0.08**	88.18	32*50
Тор 6×6	32	**89.70+-0.08**	89.21	32*36
Тор 6×6, консенсус реже в 5 раз	32	88.63+-0.08	89.21	32*36
Тор 6×6	6	**91.55+-0.08**	91.12	6*36
Тор 6×6, консенсус реже в 5 раз	6	88.99+-0.08	91.12	6*36
Граф из 12 агентов с отрицательными весами	16	**91.80+-0.03**	91.24	16*12
Полный граф из 36 агентов	32	**89.21+-0.09**	89.21	32*36

Полный граф из 36 агентов, консенсус реже в 3 раза	32	**89.09+-0.09**	89.21	32*36
Полный граф из 36 агентов, консенсус реже в 5 раз	32	**89.18+-0.08**	89.21	32*36
Полный граф из 36 агентов, консенсус реже в 10 раз	32	88.14+-0.09	89.21	32*36

Таблица 4: Средний консенсус на несбалансированных данных. Датасет CIFAR-10.

Топология	**Размер батча агента**	**Точность агентов (%)**	**Точность одиночной модели (%)**	**Размер батча одиночной модели**
Цикл из 4 агентов	32	**92.19+-0.02**	91.79	32*4
Тор 6×6	32	**89.10+-0.14**	89.21	32*36
Тор 6×6	6	90.87+-0.10	91.12	6*36
Тор 6×6, частота консенсуса увеличена	6	**91.13+-0.12**	91.12	6*36

Таблица 5: Средний консенсус на сбалансированных данных. Датасет CIFAR-100.

Топология	**Размер батча у агента**	**Точность агентов (%)**	**Точность одиночной модели (%)**	**Размер батча одиночной модели**

Цикл из 4 агентов	32	**68.62+-0.08**	68.21	32*4
Top 6×6	32	**65.11+-0.11**	63.46	32*36
Top 6×6	6	**67.99+-0.17**	66.92	6*36

Таблица 6: Средний консенсус на сбалансированных данных. Датасет FashionMNIST.

Топология	**Размер батча агента**	**Точность агентов (%)**	**Точность одиночной модели (%)**	**Размер батча одиночной модели**
Цикл из 4 агентов	25	**92.05+-0.09**	92.05	25*4
Top 6×6	6	91.01+-0.17	91.88	6*36

Таблица 7: Алгоритм сплетен.

Топология	**Датасет**	**Параметр b**	**Размер батча агента**	**Точность агентов (%)**	**Точность одиночной модели (%)**	**Размер батча одиночной модели**
Top 6×6	CIFAR-10	36	32	**89.04+-0.10**	89.21	32*36
Top 6×6	CIFAR-10	18	32	88.14+-0.08	89.21	32*36
Цикл из 4 агентов	CIFAR-100	4	32	**68.50+-0.19**	68.21	32*4
Цикл из 4 агентов	CIFAR-100	1	32	**68.45+-0.10**	68.21	32*4

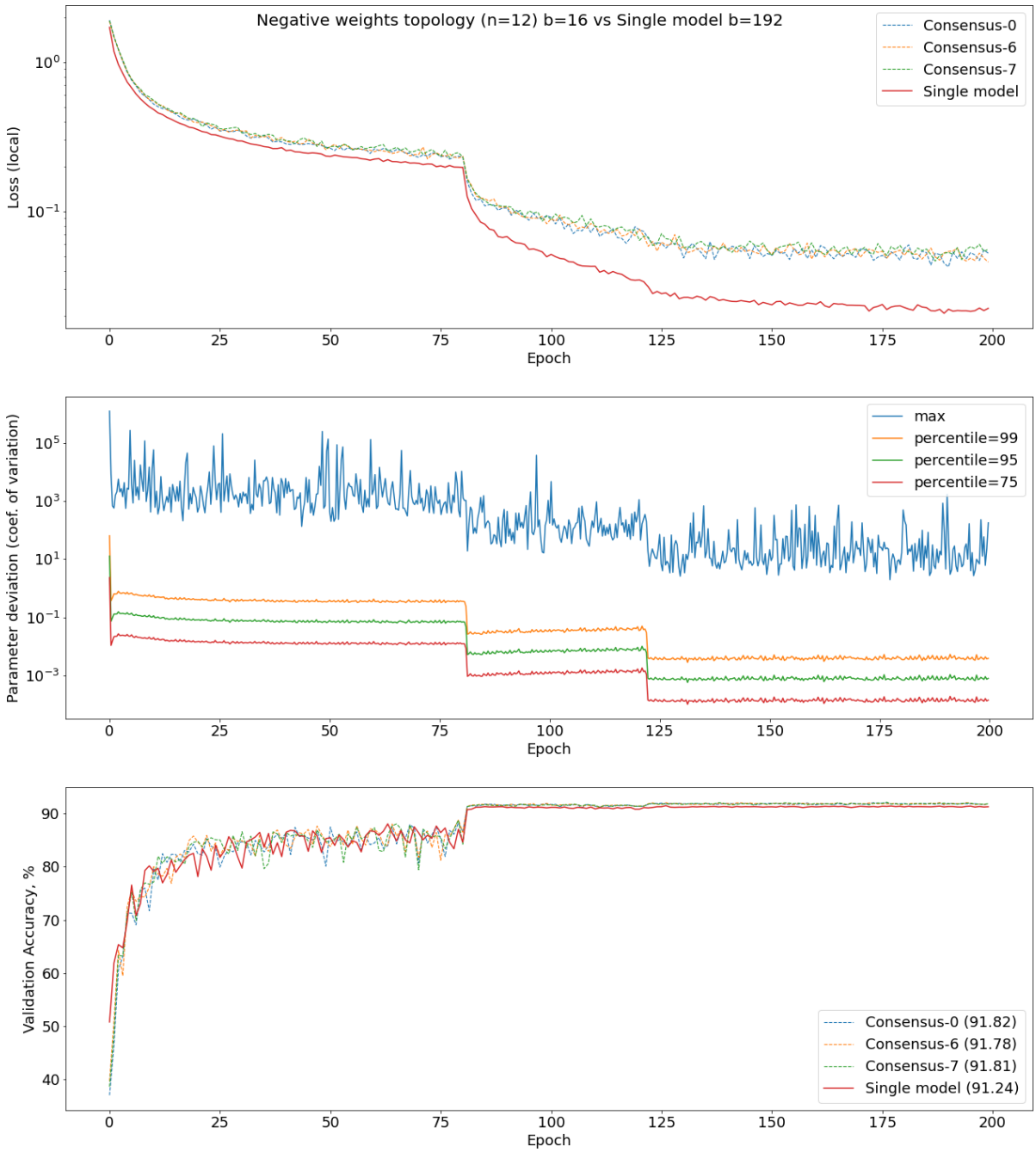

Рис. 9: Результаты эксперимента на графе с отрицательными весами. Датасет CIFAR-10.

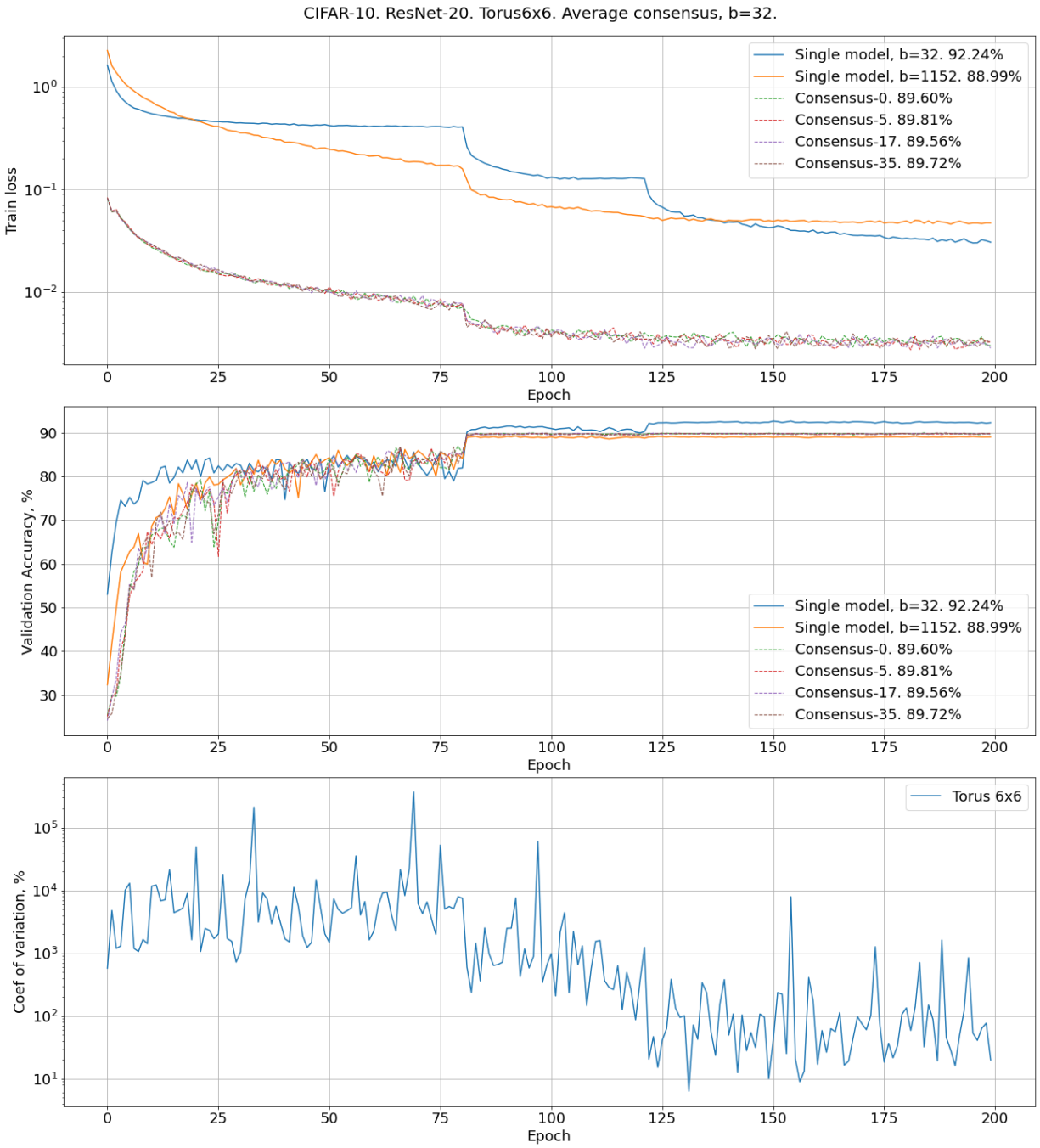

Рис. 10: Результаты эксперимента на торе 6×6 с батчем 32.

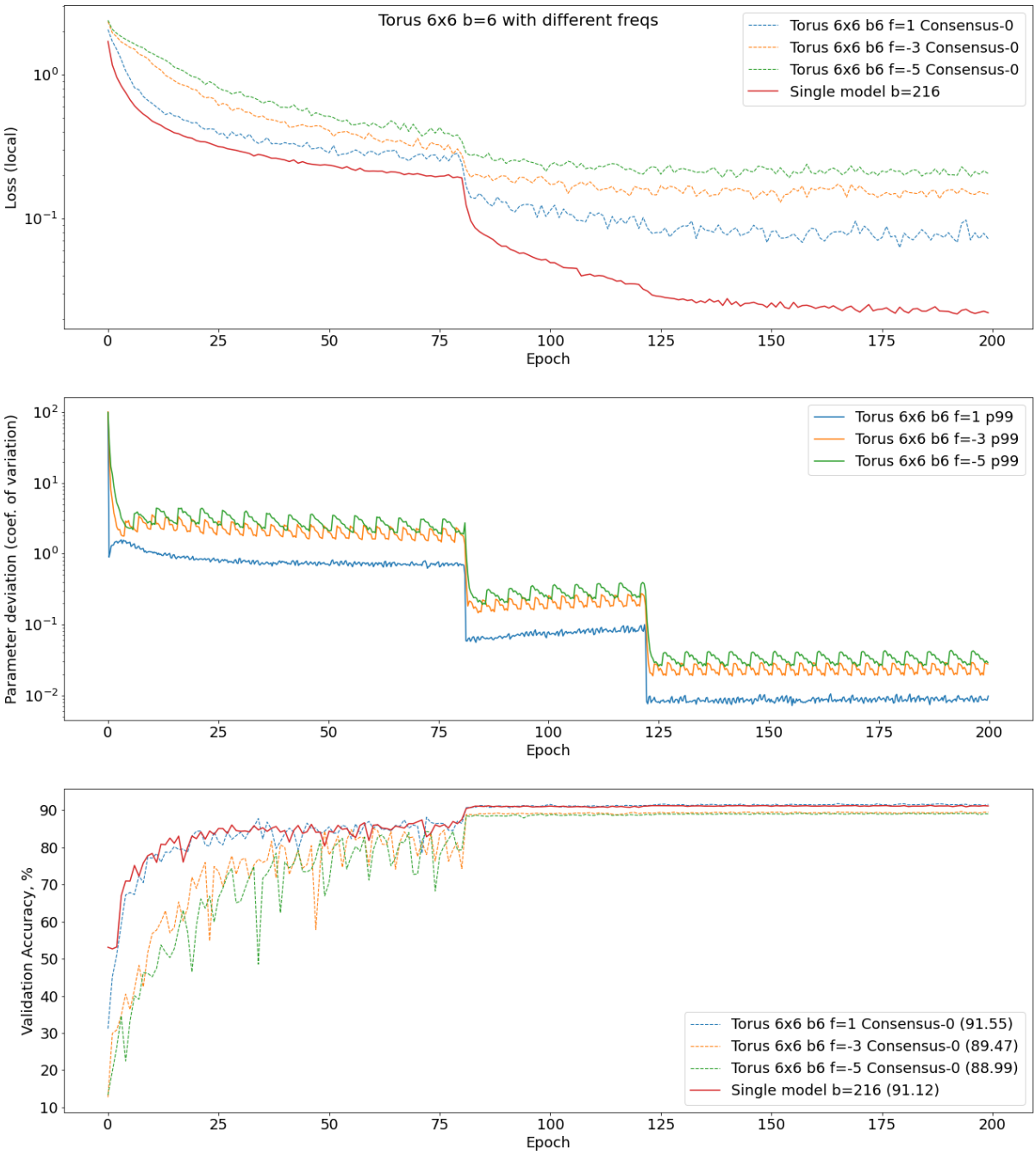

Рис. 11: Результаты эксперимента на торе 6×6 с батчем 6 и разной частотой консенсуса.

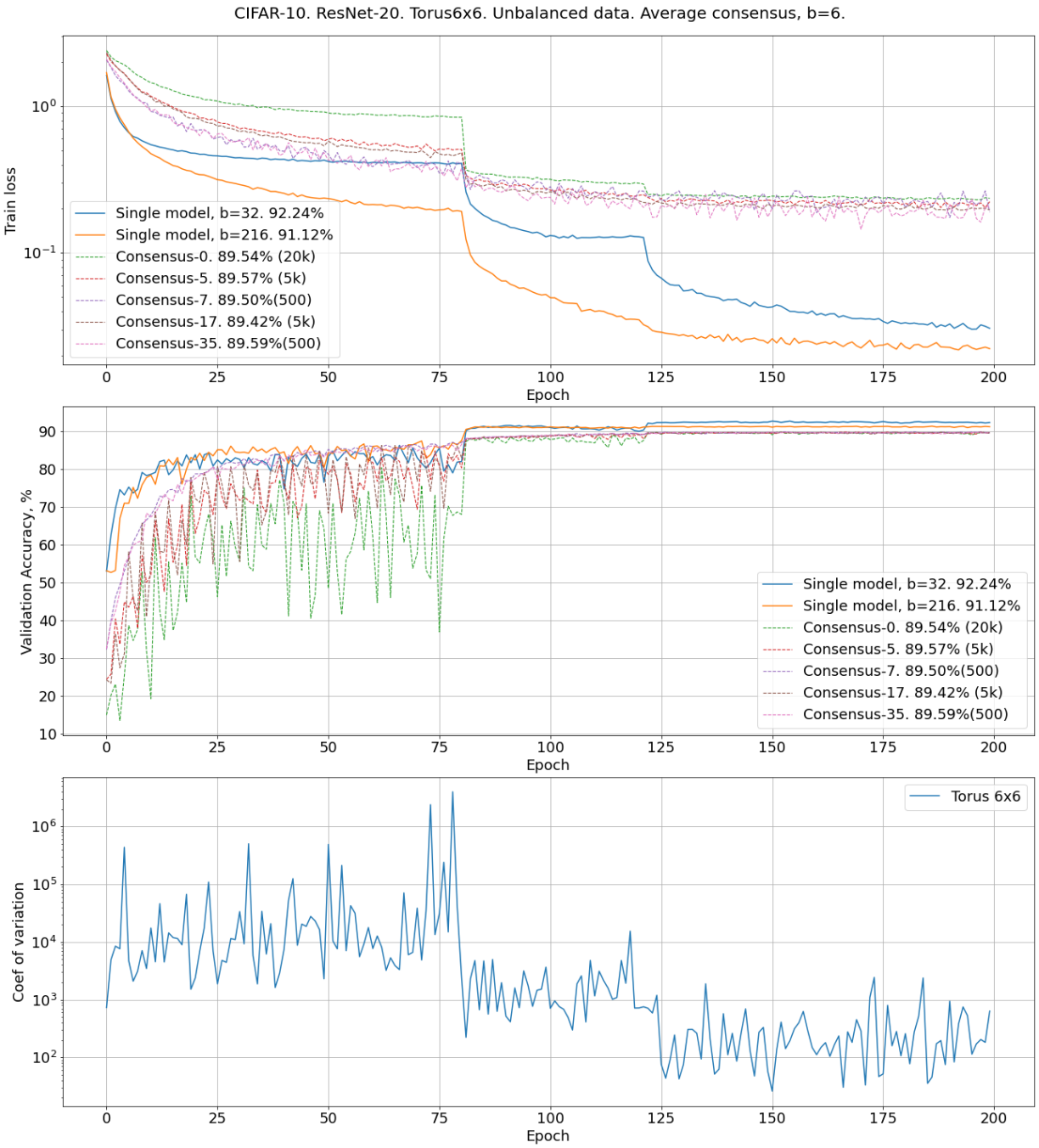

Рис. 12: Результаты эксперимента на торе 6×6 с батчем 6 и несбалансированными данными.

Выводы

Из результатов эксериментов можно сделать несколько выводов. Становится ясно, что алгоритмы среднего консенсуса и сплетен могут быть применены для распределенного обучения нейронных сетей. При условии сбалансированности данных совместная модель агентов сети достигает примерно той же или даже большей точности, чем одиночная модель с увеличенным батчем. При этом остается возможность проводить раунды консенсуса реже, немного проигрывая в качестве, но уменьшая нагрузку на коммуникационную сеть. Также стало понятно, что описанный алгоритм способен корректно работать в графах с малым количеством ребер, что является важным свойством, ведь на практике каналы связи являются узким местом распределенной сети. Если данные несбалансированы, а мощность агентов пропорциональна размеру их локальных данных, то финальная точность совместной модели получается чуть хуже, чем в сбалансированном случае. Но данное отличие сравнительно невелико, оно меньше заявленного разброса исходной одиночной модели. Вдобавок, как выяснилось, увеличив количество итераций консенсуса, можно достигнуть результата более близкого к результату агентов на сбалансированных данных. Помимо всего перечисленного, данный алгоритм основан на том, что локальные данные агентов не пересекаются и не передаются по сети в процессе обучения. Это гарантирует изоляцию данных и расширяет границы применимости рассмотренных методов.

Таким образом, алгоритм среднего консенсуса и алгоритм сплетен успешно применены к задаче распределенного обучения нейронной сети на датасетах «CIFAR-10», «CIFAR-100» и «FashionMNIST».

Заключение

В ходе данной работы исследована возможность применения алгоритма на базе консенсуса и алгоритма сплетен для распределенного обучения нейронных сетей, в частности:

- изучена предметная область и существующие методы
- изучена математическая теория, описывающая исследуемый подход
- реализован алгоритм и прототип распределенной сети для проведения экспериментов
- проведены эксперименты и выполнен анализ результатов

Список литературы

[1] S. Ghadimi, G. Lan, and H. Zhang. «Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization». Mathematical Programming, 2016.

[2] R Olfati-Saber, JA Fax, RM Murray. «Consensus and cooperation in networked multi-agent systems». IEEE, 2007.

[3] R. Olfati-Saber and R. M. Murray, «Consensus problems in networks of agents with switching topology and time-delays». IEEE Transactions on Automatic Control, vol. 49, no. 9, pp. 1520-1533, Sept. 2004, doi: 10.1109/TAC.2004.834113.

[4] Boyd S, «Convex optimization of graph Laplacian eigenvalues ». Proceedings of the International Congress of Mathematicians. – 2006. – Т. 3. – №. 1-3. – С. 1311-1319.

[5] Koloskova A. et al. «A unified theory of decentralized SGD with changing topology and local updates». International Conference on Machine Learning. – PMLR, 2020. – С. 5381-5393.

[6] Boyd S. et al. «Randomized gossip algorithms //IEEE transactions on information theory». – 2006. – Т. 52. – №. 6. – С. 2508-2530.

[7] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. «Optimal distributed online prediction using mini-batches». Journal of Machine Learning Research, 2012.

[8] A. Agarwal and J. C. Duchi. «Distributed delayed stochastic optimization ». In NIPS, 2011.

[9] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson. «An asynchronous mini-batch algorithm for regularized stochastic optimization». IEEE Transactions on Automatic Control, 2016.

[10] T. Paine, H. Jin, J. Yang, Z. Lin, and T. Huang. «Gpu asynchronous stochastic gradient descent to speed up neural network training». arXiv preprint arXiv:1312.6186, 2013.

[11] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: «A lock-free approach to parallelizing stochastic gradient descent». In Advances in neural information processing systems, 2011.

[12] N. Luehr. «Fast multi-gpu collectives with nccl». Nvidia blog, 2016.

[13] P. Patarasuk and X. Yuan. «Bandwidth optimal all-reduce algorithms for clusters of workstations». Journal of Parallel and Distributed Computing, 2009.

[14] Lian X. et al. «Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent». arXiv preprint arXiv:1705.09056. – 2017.

[15] B. Sirb and X. Ye. «Consensus optimization with delayed and stochastic gradients on decentralized networks». In Big Data, 2016.

[16] P. Bianchi, G. Fort, and W. Hachem. «Performance of a distributed stochastic approximation algorithm». IEEE Transactions on Information Theory, 2013.

[17] Xiangru Lian, Wei Zhang, Ce Zhang, Ji Li. «Asynchronous Decentralized Parallel Stochastic Gradient Descent». Proceedings of the 35th International Conference on Machine Learning, PMLR 80:3043-3052, 2018.

[18] GitHub repository with implementing ResNet-20. // URL: https://github.com/akamaster/pytorch_resnet_cifar10

[19] Goyal P. et al. «Accurate, large minibatch sgd: Training imagenet in 1 hour». arXiv preprint arXiv:1706.02677. – 2017.

[20] Smith L. N. «A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay». //arXiv preprint arXiv:1803.09820. – 2018.