НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук

НАУЧНЫЙ ДОКЛАД
по результатам подготовленной научно-квалификационной работы
(диссертации)

«Методы сжатия рекуррентных нейронных сетей для задач обработки естественного языка»

Грачев Артем Михайлович

Направление подготовки: 02.06.01 Компьютерные и информационные науки

Профиль программы: 05.13.17 Теоретические основы информатики

Аспирантская школа по компьютерным наукам

Аспирант
А. М. Грачев _____
Научный руководитель
к.т.н. Д. И. Игнатов _____
Директор аспирантской школы
по компьютерным наукам
к.т.н. С. А. Объедков _____

Москва, 2018

1 Актуальность темы исследования

В последнее десятилетие быстро развивается направление в машинном обучении, называемое глубинным обучением (deep learning) [2, 26], связанное с успешным обучением нейронных сетей и использованием в них сложных архитектур. Подходы, связанные с глубинным обучением, вывели сразу несколько направлений в компьютерных науках на новый уровень. В первую очередь эти направления связаны со сложно формализуемыми задачами, такими как обработка изображений, понимание текста, распознавание речи. Во многих областях и конкретных задачах подходы, основанные на методах глубинного обучения, стали признанным индустриальным решением [7]. При этом появляются всё новые задачи [2, 9], а потенциал этих подходов далеко не исчерпан.

Вместе с тем актуальной становится задача сжатия нейронных сетей для размещения их на различных устройствах. Модели, основанные на нейронных сетях, могут требовать много места на диске и в памяти, а также большого времени для вычислений во время использования. Это может быть критично для устройств, у которых памяти не так много и для которых важно быстродействие, например, для мобильных телефонов. Из целого ряда современных приложений нейронных сетей отдельно стоит выделить задачу моделирования естественного языка. Самые первые методы для решения этой задачи были основаны на сохранении всех вариантов продолжения цепочки слов (предложения) по заданному началу. В момент предсказания выбирался наиболее вероятный вариант. У такого подхода есть проблемы, связанные с учетом дальних зависимостей. Все возможные цепочки длины 4 уже занимают несколько гигабайт. А цепочки длины 20 уже просто физически невозможно хранить в памяти компьютера. Нейронные сети решают эту проблему, во многом за счёт использования специальных рекуррентных архитектур. Но они всё ещё слишком большого размера для их повсеместного использования на устройствах, ко-

торые, как уже упоминалось выше, обычно очень требовательны к памяти и быстродействию. Особенностью построения языковых моделей является необходимость хранить словарь из нескольких тысяч слов (например, около 20 тысяч слов покрывают лишь 80% русского языка). Кроме того, часто необходимо решать задачу прогнозирования следующего слова или, выражаясь терминами машинного обучения, решать задачу классификации на несколько тысяч (или даже десятков тысяч) классов. Несмотря на то, что нейронные сети занимают гораздо меньше места, чем простые статистические модели, задача сжатия является актуальной для их широкого применения.

В данном диссертационном исследовании была поставлена цель изучить текущие техники и развить их в контексте сжатия нейронных сетей для задачи моделирования языка.

2 Цель и задачи исследования

В данном разделе сформулированы основные цели и задачи исследования.

2.1 Цель исследования

Изучение алгоритмов сжатия нейронных сетей для применения их в задаче обработки естественного языка. Создание новых алгоритмов, решающих эту задачу более эффективно с точки зрения использования ресурсов устройства.

2.2 Основные задачи

Для достижения цели в рамках работы предполагается сравнить различные методы сжатия нейронных сетей и построить на их базе архитектуры, имеющие сравнительно маленький размер и эффективно решающие прикладные задачи. В качестве подзадач этого исследования можно выделить:

- Изучение и применение алгоритмов сжатия нейронных сетей, основанных на техниках разреживания и квантизации, разложении матриц, байесовских методах.
- Построение алгоритмов сжатия нейронных сетей с учётом специфики задачи обработки естественного языка и типичных архитектур рекуррентных нейронных сетей.
- Создание новых видов алгоритмов, отвечающих заданным характеристикам, для применения их на различных устройствах.

3 Степень разработанности темы исследования

Существует целый ряд подходов к сжатию нейронных сетей. Верхнеуровнево их можно разделить на две категории: методы, базирующиеся на разреженных вычислениях, и методы, основанные на различных матричных разложениях.

К первой группе методов можно отнести прореживание нейронных сетей (pruning) и квантизацию. Одна из первых работ на эту тему это [12]. В этой статье авторы показали, что подобное прореживание может удалять большое количество весов без потери качества. Так было показано, что можно удалить 67% весов в сверточных слоях и вплоть до 90% в полносвязных слоях. В статье это было показано на таких известных нейронных сетях, как LeNet, AlexNet, VGGNet. Ещё более высокого уровня сжатия можно достичь комбинируя прореживание с использованием смешанной точности [11].

Более полное математическое обоснование техника прореживания получила в терминах вариационного дропаута. Изначально вариационный дропаут был представлен в [15] как метод автоматического подбора вероятности дропаута, иными словами, вероятности, что нейрон будет неактивным в нейронной сети. В [20, 22] авторы адаптировали эту технику для сжатия нейронных сетей. В их работах была применена параметризация, которая позволяет вероятности дропаута быть единицей, что равносильно полному удалению этого нейрона из нейронной сети. Также эта техника вариационного дропаута была применена к рекуррентным нейронным сетям в статье [17]. Однако данная техника до сих пор не применялась к естественно возникающей проблеме высокоразмерного выхода в задачах работы с текстом, которую мы обсудим ниже.

С помощью квантизации и прореживания можно достичь существенно-

го уменьшения размера натренированной сети при хранении её на диске. Однако когда мы начинаем использовать такую сжатую модель непосредственно в вычислениях, возникают проблемы. Они обусловлены высокой сложностью разреженных вычислений (после прореживания) или необходимостью делать вычисления в 32-битовом формате (для квантизации), что не ведет к желаемому ускорению. Для прореживания одним из возможных решений является использование так называемого структурированного прореживания, как это было показано в статье [17, 22], где удалялись сразу целые столбцы или строчки матриц в слоях нейронной сети. Для квантизации решением является использование специальных процессоров, которые поддерживают 16-битные вычисления, но это возможно только при использовании 16-битного сжатия.

Другое направление методов основано на использовании различных матричных преобразований в нейронных сетях. Например, эти методы могут использовать матрицы специальных типов. Так, использование матрицы Теплица в [18] дает около 40% сжатия для RNN в задаче голосового поиска. В [1] предлагается использовать новый тип RNN, основанный на унитарных матрицах, называемый Unitary Evolution Recurrent Neural Networks. В [29] авторы предлагают использовать так называемое преобразование FastFood для полносвязных и сверточных слоев в нейронной сети (они достигли порядке 90% сжатия в терминах хранимых параметров). К этому классу методов можно отнести и методы, основанные на различных разложениях матриц, используемых в нейронной сети. Эти методы могут быть как простым, вроде обычного низкорангового разложения так и более сложными, как например разложение Tensor Train [8, 23, 28, 30]. Методы, основанные на матричных разложениях, позволяют сжимать нейронные сети и по-прежнему иметь удобные (не разреженные) матрицы для умножения, что в итоге приводит к выигрышу в скорости.

4 Рекуррентные нейронные сети для задачи моделирования языка

Рассмотрим задачу моделирования языка. Обычно в ней требуется оценить вероятность предложения или последовательности слов $(w_1, \ldots, w_T)$ в языке $\mathbb{L}$.

$$\mathsf{P}(w_1, \ldots, w_T) = \mathsf{P}(w_1, \ldots, w_{T-1})\,\mathsf{P}(w_T | w_1, \ldots, w_{T-1}) = \\ = \prod_{t=1}^{T} \mathsf{P}(w_t | w_1, \ldots, w_{t-1}) \quad (1)$$

Для того чтобы использовать такую модель напрямую необходимо вычислять вероятность $\mathsf{P}(w_t | w_1, \ldots, w_{t-1})$. В общем случае, то есть для произвольных t эта вероятность невычислима. Поэтому обычно данную вероятность аппроксимируют с помощью $\mathsf{P}(w_t | w_{t-n}, \ldots, w_{t-1})$ — вероятности следующего слова, при длине предыдущего контекста n. Это приводит нас к n-граммным моделям [14, 16], которые представляют из себя частотные распределения всех цепочек длины n, встречаемых в обучающем тексте. Как уже упоминалось, у такого подхода есть естественные проблемы. Например, все цепочки длины 4 уже занимают несколько гигабайт в памяти компьютера.

Использование рекуррентных нейронных сетей [3, 19] стало новой вехой в развитии статистического моделирования языка. Рассмотрим RNN, где N — это число шагов по времени, L — число скрытых рекуррентных слоев, x_ℓ^t — это выходной вектор для слоя ℓ в момент времени t. Здесь $t \in \{1, \ldots, N\}$, $\ell \in \{1, \ldots, L\}$, и x_0^t — это вектор эмбеддингов. При таких обозначениях мы можем описать каждый слой в следующем виде:

$$z_\ell^t = W_\ell x_{\ell-1}^t + U_\ell x_\ell^{t-1} + b_l \tag{2}$$

$$x_\ell^t = \sigma(z_\ell^t), \tag{3}$$

где W_ℓ и V_ℓ — это матрицы весов и σ — это функция активации. Выход нейронной сети задать следующим образом:

$$y^t = \text{Softmax}\left[W_{L+1} x_L^t + b_{L+1}\right]. \tag{4}$$

Тогда интересующая нас вероятность (1) будет выражаться следующим образом:

$$\mathsf{P}\left(w_t | w_{t-N}, \ldots, w_{t-1}\right) = y^t. \tag{5}$$

В то время как n-грамные модели даже с не очень большими n требуют большого количества памяти из-за комбинаторного взрыва, рекуррентные нейронные сети могут выучить представления слов и последовательностей без непосредственного запоминания всех слов в контексте.

В настоящее время широко используются различные вариации рекуррентных сетей, которые решают проблему затухающего градиента [6, 13]. Наиболее популярные из них это сети с длинно-короткой памятью (Long-short term memory, сокращенно LSTM) и GRU (Gated Recurrent Unit). Далее опишем один слой LSTM сети:

$$i_\ell^t = \sigma\left[W_l^i x_{l-1}^t + U_l^i x_l^{t-1} + b_l^i\right] \qquad \text{input gate} \tag{6}$$

$$f_\ell^t = \sigma\left[W_l^f x_{l-1}^t + U_l^f x_l^{t-1} + b_l^f\right] \qquad \text{forget gate} \tag{7}$$

$$c_\ell^t = f_l^t \cdot c_l^{t-1} + i_l^t \tanh\left[W_l^c x_{l-1}^t + U_l^c x_l^{t-1} + b_l^c\right] \qquad \text{cell state} \tag{8}$$

$$o_\ell^t = \sigma\left[W_l^o x_{\ell-1}^t + U_l^o x_l^{t-1} + b_l^o\right] \qquad \text{output gate} \tag{9}$$

$$x_\ell^t = o_\ell^t \cdot \tanh[c_l^t], \tag{10}$$

где опять $t \in \{1, \ldots, N\}$, $\ell \in \{1, \ldots, L\}$, c_ℓ^t — это вектор памяти для слоя ℓ и временного шага t. Выход сети задается той же формулой 4 как и для RNN.

Подходы к моделированию языка, основанные на нейронных сетях, эффективны и широко используются, но все еще требуют много места. Из формул (6–10) видно, что каждый LSTM с входной и выходной размернорстью k включает в себя восемь матриц $k \times k$. Кроме того, обычно в задаче моделирования языка мы хотим использовать слова (а не символы) как фундаментальную единицу входа и выхода. Это требование ведет нас к большим размерам входного и выходного слоя. Входной слой или слой эмбеддингов переводит номер слова в словаре $\mathbb{V}$ в числовой вектор. Выходной слой — аффинное преобразование из скрытого представления в номер слова в словаре, для которого обычно используется функция софтмакс. Здесь надо отметить, что размер словаря $|\mathbb{V}|$ — это величина порядка тысяч или даже десятка тысяч. Складывая все это, получаем, что количество параметров во всей сети с L слоями равно

$$n_{total} = 8Lk^2 + 2|\mathbb{V}|k. \tag{11}$$

4.1 Анализ числа параметров в нейронной сети

Проанализируем, насколько велик вклад каждого слагаемого в (11). В датасаете PTB размер словаря $|\mathbb{V}| = 10,000$. Рассмотрим LSTM сеть с двумя скрытыми слоями $k = 650$ нейронов в каждом. Тогда получаем, что каждый LSTM слой имеет восемь матриц размером 650×650, то есть $650 \times 650 \times 8 \times 2 = 6.76$M параметров. Выходной слой такой сети имеет $650 \times 10000 = 6.5$M параметров. Похожие вычисления для LSTM сети с размерами скрытых слоев $k = 1500$ дают нам 36 M параметров суммарно в скрытом слое и 15 M параметров для выходного слоя. Таким образом, мы можем видеть, что выходной (софтмакс) слой может занимать до од-

ной трети от всей сети. Заметим, что эмбеддинги сети занимают такой же размер, если не используются техники наподобие 'tied softmax" [25].

Проблема высокоразмерных выходов в задаче моделирования языка широко обсуждается в литературе. Например, в [10] рассматривается проблема редких или неизвестных слов. Большая вычислительная сложность и размер выходного слоя обсуждается в [4,21]. В [21] разрабатывается идея иерархического софтмакса. Авторы показывают, что возможно вычислять софтмакс со скоростью $O(\sqrt{|\mathbb{V}|})$. Bengio et al. [4] предлагают ускорять вычисление софтмакса с помощью сэмплирования подмножества словаря на каждой итерации в течении вычисления выходов сети. Мы также обращаемся к проблеме высокоразмерного выхода в софтмакс слое. Мы используем низкоранговое разложение и Tensor Train разложение для уменьшения размера матрицы софтмакса и ускорения ее расчета.

4.2 Базовые техники сжатия нейронных сетей

В данном разделе рассматриваются базовые техники сжатия нейронных сетей, которые мы применили в контексте задачи моделирования языка.

Прореживание (pruning) нейронной сети —это метод уменьшения числа параметров нейронной сети, значение которых близко к нулю. При применении этого метода обычно выставляется какой-то порог и веса, по модулю меньшие этого порога, обнуляются. После этого также можно дообучить сеть на оставшихся разреженных нейронах.

Квантизация — это ещё один технический метод для уменьшия размера нейронной сети на диске. Далее изложим основную идею этой техники. Обычно в памяти компьютера нейронная сеть хранится в виде набора весов, каждый из которых, в свою очередь, представлен числом с плавающей точкой и занимает 32 бита. Возьмём весь диапазон весов, который у нас получается в нейронной сети, и разделим его на 256 интервалов. У каждого интервала будет свой порядковый номер, который мы закодируем целым

числом, а также значение, которое ему отвечает, например, среднее значение в интервале. Для такой кодировки нам нужно всего лишь хранить массив из 256 средних значений интервала. Теперь заменим значение веса порядковым номером интервала в зависимости от того, в какой интервал оно попадает. Теперь вместо 32 битного представления всех нейронов мы получили их 8-битные представления и массив для их хранения. Другими словами, мы сжали нейронную сеть в 4 раза. Аналогично с такими же целями иногда применяются 16-битные вычисления.

Несмотря на то, что на первый взгляд кажется, что эти техники просты и эффективны, они обе имеют похожие недостатки. Первым недостатком является невозможность применения таких техник при обучении с нуля. То есть сначала необходимо обучить нейронную сеть, а после применить один из этих методов, для того чтобы получить сжатие.

Другим недостатком является ряд проблем с непосредственным использованием таких сетей в тестовом режиме. В случае квантизации мы всё ещё вынуждены использовать 32-битное представление в оперативной памяти компьютера для проведения операций, несмотря на то что храним модель на диске в 8-битном представлении. Это означает, что мы не получаем никакого выигрыша во время вычислений. В случае с 16-битной квантизацией мы можем использовать 16-битные вычисления, если процессоры их поддерживают, что на данный момент всё ещё не имеет достаточно широкого распространения.

Вычислительные проблемы с прореженными сетями связаны с необходимостью использовать разреженные матрицы для вычислений. И несмотря на то что их можно эффективно хранить в памяти, мы не получаем выигрыша по скорости при их умножении, а до какого-то порога сжатия получаем даже проигрыш.

Несмотря на указанные недостатки мы исследовали степень сжатия, достигаемую с помощью этих методов в задаче моделирования языка. Для

этих и всех последующих экспериментов мы использовали модели из статьи [31] в качестве базовых моделей, с которыми сравнивались. Это три LSTM модели, которые также называются Small, Medium, и Large по размеру их скрытых слоёв, которые соответственно составляют 200, 650, and 1500. Достигаемое сжатие мы сравнивали, измеряя количество парметров, размеры модели, время работы на мобильном устройстве (для некоторых моделей) и перплексию. Перплексия – это одна из стандартных метрик качества для языковой модели. Она показывает насколько хорошо данная модель предсказывает следующий элемент языковой последовательности. Меньшее значение перплексии означает лучшую модель.

Для тестирования техники прореживания и квантизации мы использовали Small LSTM модель. Результаты экспериментов представлены в Таблице 1.

Таблица 1: Результаты прореживания и квантизации на PTB датасете (Small LSTM)

Model	**Size, Mb**	**No. of parameters, M**	**Test perplexity**
LSTM 200-200	18.6	4.64	117.659
Pruning output layer 90% w/o additional training	5.5	0.5	149.310
Pruning output layer 90% with additional training	5.5	0.5	121.123
Quantization (1 byte per number)	4.7	4.64	118.232

5 Основные результаты диссертационного исследования

Сформулируем основные результаты исследования и положения, выносимые на защиту

- Применение техник матричных разложений в нейронных сетях для задачи моделирования языка.
- Эффективное решение задачи классификации на большое число классов с использованием техник матричных разложений.
- Адаптация техник сжатия рекуррентных нейронных сетей для задачи моделирования языка с помощью байесовских методов.
- Портирование эффективной реализации рекуррентных нейронных сетей на мобильные устройства и лабораторное тестирование

5.1 Техники матричного разложения для сжатия рекуррентных нейронных сетей

В этой секции рассмотрим техники матричных разложений для задачи сжатия нейронных сетей и их возможные модификации в контексте задачи моделирования естественного языка. Опишем низкоранговое (Low-rank, LR) разложение матриц весов. Простейщее разложение для RNN может быть выполнено в следующей форме:

$$x_l^t = \sigma\left[W_\ell^a W_\ell^b x_{\ell-1}^t + U_l^a U_l^b x_\ell^{t-1} + b_l\right] \tag{12}$$

Но в [18] при применении рекуррентной нейронной сети к задаче распознавания голоса авторы предложили использовать следующее ограничение: $W_l^b = U_{\ell-1}^b$. В таком случае уравнение для слоя RNN можно переписать

следующим образом:

$$x_l^t = \sigma\left[W_l^a m_{l-1}^t + U_l^a m_l^{t-1} + b_l\right] \tag{13}$$

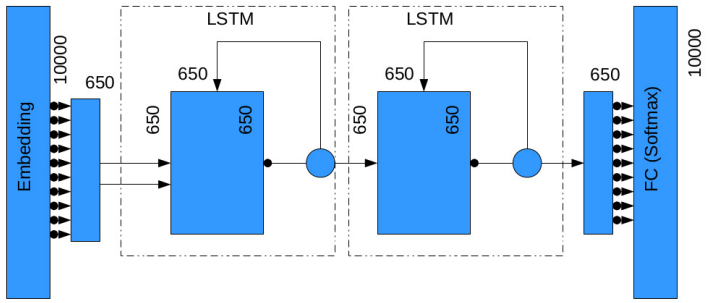
$$m_l^t = U_l^b x_l^t \tag{14}$$

$$y_t = \text{Softmax}\left[W_{L+1} m_L^t + b_{L+1}\right] \tag{15}$$

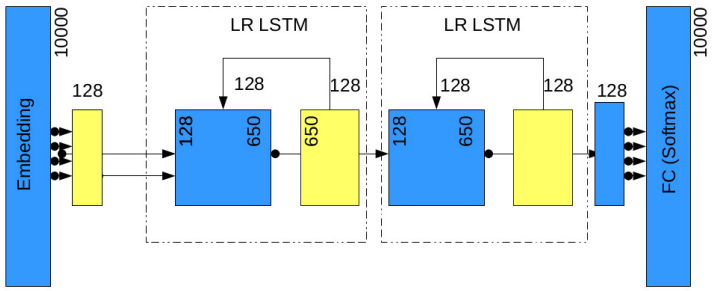
Сжатие LSTM слоя может быть реализовано таким же образом, но с более сложными выкладками. Схематично такое разложение можно изобразить как на Рис. 1. Главным преимуществом такой техники сжатия является потенциально маленькие размеры $r \times k$ and $k \times r$ матриц W_l^a / U_l^b и U_l^a соответственно. Они намного меньше, чем $k \times k$ оригинальных матриц W_l and V_l if $r \ll k$. С маленьким r мы получаем выигрыш и в размере сети, и в скорости вычисления. Все эти выводы подтверждены экспериментами, представлеными в Таблице 2 и Таблице 3. В Таблице 2 также приведены контрольные замеры скорости работы таких рекуррентных нейронных сетей на мобильном телефоне Samsung S7 Edge. Как видно из приведенных результатов, с помощью низкорангового разложения удалось достичь сжатия модели LSTM 650-650 почти в 5 с некоторой потерей качества, но при этом с показателем перплексии всё ещё ниже, чем у модели LSTM 200-200.

Таблица 2: Результаты применения низкорангового матричного разложения для LSTM моделей на PTB датасете

	Model	**Size, Mb**	**No. of parameters, M**	**Test perplexity**	**Inference time, ms**
PTB Benchmarks	LSTM 200-200	18.6	4.64	117.659	9.63
	LSTM 650-650	79.1	19.7	82.07	16.13
	LSTM 1500-1500	264.1	66.02	78.29	45.47
Ours	LR LSTM 650-650	16.8	4.2	92.885	9.68
	TT LSTM 600-600	50.4	12.6	168.639	16.75
	LR LSTM 1500-1500	94.9	23.72	89.462	15.70



(a)



(b)

Рис. 1: Архитектуры нейронных сетей: (a) оригинальная сеть LSTM 650-650, (b) Модель, сжатая с помощью низкорангового разложения.

5.2 Применение разложения в тензорный поезд

В этом разделе мы опишем ещё одну технику для сжатия нейронных сетей, которая, по сути, близка к низкоранговому матричному разложению. Разложение в так называемый тензорный поезд (Tensor Train, TT) изначально было предложено как более эффективная форма представления тензора [24]. Опишем, как такое разложение может быть применено к нейронным сетям. Рассмотрим, например, матрицу весов $W \in \mathbb{R}^{k \times k}$ RNN слоя (2). Далее мы можем выбрать такие числа $k_1, ..., k_d$, чтобы $k_1 \times \ldots \times k_d = k \times k$ и сделать преобразование матрицы W в тензор $\vec{W} \in \mathbb{R}^{k_1 \times ... \times k_d}$. Здесь d – это размерность нового тензора, $k_1, \ldots, k_d$ – это внутренние размеры каждого измерения. Далее мы можем выполнить TT-разложение тензора $\vec{W}$ и получить набор матриц $G_m[i_m] \in \mathbb{R}^{r_{m-1} \times r_m}, i_m = 1, \ldots, k_m,\ m = 1, \ldots, d$

Таблица 3: Матричное низкоранговое разложение и Tensor Train разложение для выходного слоя нейронной сети

	Model	**Size, Mb**	**No. of parameters, M**	**No. of output layer parameters, M**	**Test perplexity**
PTB Benchmarks	LSTM 200-200	18.6	4.64	2.0	117.659
	LSTM 650-650	79.1	19.7	6.5	82.07
	LSTM 1500-1500	264.1	66.02	15.0	78.29
LR for Softmax layer	LSTM 200-200	12.6	3.15	0.51	112.065
	LSTM 650-650	57.9	14.48	1.193	84.12
	LSTM 1500-1500	215.4	53.85	2.829	89.613
TT for Softmax layer	LSTM 200-200	11.8	2.95	0.304	116.588
	LSTM 600-600	51.12	12.8	1.03	88.551
	LSTM 1500-1500	215.8	53.95	2.92	85.63

и $r_0 = r_d = 1$ такой, что каждый элемент тензора может быть представлен как $\vec{W}(i_1, i_2, \ldots, i_d) = G_1[i_1]G_2[i_2]\ldots G_d[i_d]$. Числа $r_0, \ldots r_m$ называются рангами Tensor Train разложения. Подобное TT разложение может быть эффективно имплементировано с помощью TT-SVD алгоритма, описанного в [24]. Фактически каждый $G_m \in \mathbb{R}^{r_{m-1} \times k_m \times r_m}$ это трехмерный тензор с измерением k_m соответствующему измерению в оригинальном тензоре и двумя рангами r_{m-1}, r_m, которые в некотором смысле отвечают за внутреннее представление этого измерения внутри разложения. Также необходимо подчеркнуть, что даже если мы зафиксируем размерность тензора, в который мы конвертируем матрицу весов, у нас все еще будет богатый выбор рангов для TT-разложения в качестве гиперпараметров.

Обозначим эти две операции: конвертации матрицы W в тензор $\vec{W}$ и её последующее разложение в Tensort Train формат как одну операцию $\mathrm{TT}(W)$. Применяя её к обеим матрицам W and V в рекуррентном слое (2), мы получаем TT-RNN слой в следующей форме:

$$z_\ell^t = \sigma(\mathrm{TT}(W_l)x_{\ell-1}^t + \mathrm{TT}(U_l)x_\ell^{t-1} + b_\ell). \tag{16}$$

Похожим образом мы можем применить TT разложение для каждой матрицы LSTM слоя (6–9) или для матрицы выходного слоя (4).

Сжатие с помощью такого разложения может быть достигнуто за счет выбора внутренних рангов $r_1, \ldots, r_{d-1}$. Пусть $R = \max\limits_{m=0,\ldots,d} r_m$, $K = \max\limits_{m=0,\ldots,d} k_m$. Тогда количество параметров в TT разложение $N_{\text{TT}} = \sum r_{m-1} k_m r_m \leq dR^2K$. Фактически, каждый множитель в этом произведении может быть на порядок меньше, чем оригинальное k.

Результаты применения TT разложения также можно найти в Таблице 2 и Таблице 3. При применении этого метода для выходного слоя с большим числом классов Tensor Train в основном даёт качество, сравнимое с применением низкорангового разложения.

5.3 Адаптация техник вариационного байесовского вывода

В этом разделе будет описано, как к задаче моделирования естественного языка с помощью нейронных сетей можно применить техники байесовского вывода для получения сжатия модели.

Начнем с описания алгоритма дважды стохастического вариационного вывода для автоматического определения значимости признаков, который изначально был предложен в [27]. Для этого опишем проблему многоклассовой классификации скозь призму байесовского подхода. Метод, изложенный далее, представляет нам возможность автоматически выбирать значимые признаки и называется алгоритм автоматического определения значимости (Automatic Relevance Determination, ARD)

Рассмотрим дискриминативную вероятностную модель

$$\begin{aligned} p(\Theta, t \mid \mathbf{x}, \Lambda) &= p(\Theta \mid \Lambda) p(t \mid \Theta, \mathbf{x}) = \\ &= \left[\prod_{i=1}^{D} \prod_{j=1}^{K} \mathcal{N}\left(\theta_{ij} \mid 0, \lambda_{ij}\right) \right] \text{Softmax}\left(\mathbf{x}^T \Theta\right)[t], \end{aligned} \tag{17}$$

здесь $\Theta \in \mathbb{R}^{D \times K}$ — это матрица параметров модели, $\mathbf{x} \in \mathbb{R}^D$ – это вектор признаков, который описывает текущий объект, $t \in \{1, \ldots, K\}$ – метка класса объекта и $\Lambda \in \mathbb{R}^{D \times K}$ — это матрица гиперпараметров, определяющих априорное распределине $p(\Theta \mid \Lambda)$ для параметров Θ.

Априорное распределение $p(\Theta \mid \Lambda)$ — это поэлементное факторизованное нормальное распределение для каждого θ_{ij} с нулевым средним и дисперсией λ_{ij}. Функция правдоподобия $p(t \mid \Theta, \mathbf{x})$ (плотность распределения над всеми возможными классами объекта при условии их описания $\mathbf{x}$ и параметров Θ) это t элементный вектор, полученный в результате применения софтмакс преобразования $\text{Softmax}(\mathbf{a}) = \frac{1}{\sum_{i=1}^{K} e^{a_i}} \left(e^{a_1}, \ldots, e^{a_n}\right)^T, \mathbf{a} \in \mathbb{R}^K$ к скалярному произведению $\mathbf{x}^T\Theta$.

Пусть $(X, T) = \{(\mathbf{x}_l, t_l)\}_{l=1}^N$ — это обучающая выборка из N независимых объектов. Мы можем вывести совместную плотность из определенной нами модели (17) следующим образом:

$$\begin{aligned} p(\Theta, T \mid X, \Lambda) &= p(\Theta \mid \Lambda) p(T \mid \Theta, X) = \\ &= p(\Theta \mid \Lambda) \prod_{l=1}^{N} p(t_l \mid \Theta, \mathbf{x}_l) = \\ &= \prod_{i=1}^{D} \prod_{j=1}^{K} \mathcal{N}\left(\theta_{ij} \mid 0, \lambda_{ij}\right) \prod_{l=1}^{N} \text{Softmax}\left(\mathbf{x}_l^T \Theta\right)[t_l], \end{aligned} \tag{18}$$

Во всём последующем выводе мы преследуем две цели. Первая – это вывод апостериорного распределения для параметров модели при условии обучающей выборки $p(\Theta \mid X, T, \Lambda) = \frac{p(\Theta, T \mid X, \Lambda)}{p(T \mid X, \Lambda)}$, и вторая – это выбор оптимальной модели, то есть выбор оптимальных гиперпараметров. Вычисление апостериорного распределения напрямую невозможно, так как интеграл $p(T \mid X, \Lambda) = \int p(\Theta, T \mid X, \Lambda) d\Theta$ невычислим. На сегодняшний день популярным является метод аппроксимации апостериорного распределе-

ния с помощью максимизации ELBO (Evidence Lower Bound):

$$\begin{aligned}\mathcal{L}(q,\Lambda) &= \log p(T \mid X,\Lambda) - KL\left(q(\Theta)\|p(\Theta \mid X,T,\Lambda)\right) = \\ &= \mathbb{E}_{\Theta\sim q(\Theta)}\left[\log p(T \mid \Theta, X)\right] - KL\left(q(\Theta)\|p(\Theta \mid \Lambda)\right)\end{aligned} \tag{19}$$

ELBO – это функция двух переменных: произвольного вариационного распределения над параметрами модели $q(\Theta)$ и гиперпараметрами Λ. Заметим, что $\mathcal{L}(q,\Lambda) \leq \log p(T \mid X,\Lambda), \forall q,\Lambda$ и $\mathcal{L}(q,\Lambda) = \log p(T \mid X,\Lambda) \Leftrightarrow q(\Theta) = p(\Theta \mid X,T,\Lambda)$, то есть максимизация ELBO по q для фиксированных Λ эквивалентна тому, что мы приближаем распределение q к нужному нам апостериорному распределению $p(\Theta \mid X,T,\Lambda)$, таким образом решая первую проблему.

Максимизация обоснованности $p(T \mid X,\Lambda)$ по гиперпараметрам Λ широко используется как байесовский выбор модели и также известен как эмпирический Байес (empirical Bayes [5]). Модель с наибольшим значением обоснованности рассматривается как "лучшая" в терминах подстройки под данные и сложности модели, что решает вторую упомянутую проблему. Мы предлагаем [5] для дальнейшего ознакомления с этой концепцией. Максимизация обоснованности может быть выполнена через максимизацию ELBO в следующем виде:

$$\begin{aligned}\max_{\Lambda} \log p(T \mid X,\Lambda) &= \max_{\Lambda}\max_{q} \mathcal{L}(q,\Lambda) = \\ &= \max_{q,\Lambda} \mathcal{L}(q,\Lambda) = \max_{q}\max_{\Lambda} \mathcal{L}(q,\Lambda)\end{aligned} \tag{20}$$

Эта оптимизационная процедура решает проблему выбора модели и также наилучшим образом приближает q к апостериорному распределению.

Рассмотрим подробнее функционал ELBO (19). Видно, что только KL слагаемое $KL\left(q(\Theta)\|p(\Theta \mid \Lambda)\right)$ зависит от Λ и следовательно:

$$\mathcal{L}(q,\Lambda) \longrightarrow \max_{\Lambda} \Longleftrightarrow KL\left(q(\Theta)\|p(\Theta \mid \Lambda)\right) \longrightarrow \min_{\Lambda}.$$

Зафиксируем класс произвольного вариационного распределения q. Пусть это будет факторизованное нормальное распределение, то есть:

$$q(\Theta \mid \boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_{i=1}^{D} \prod_{j=1}^{K} \mathcal{N}(\theta_{ij} \mid \mu_{ij}, \sigma_{ij}^2), \tag{21}$$

где $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{D \times K}$ *вариационные параметры.*

Задача минимизации KL дивергенции по Λ может быть решена аналитически (как показано в [27]) с точкой аргминимума в:

$$\lambda_{ij}^* = \mu_{ij}^2 + \sigma_{ij}^2, \tag{22}$$

и затем, после подстановки Λ^* в уравнение для ELBO (19) и учета того, что мы зафиксировали семейство вариационного распределения (21), мы можем переписать уравнение (20) как:

$$\begin{aligned} \mathcal{L}\left(q\left(\Theta \mid \boldsymbol{\mu}, \boldsymbol{\sigma}\right), \Lambda^*\right) = \mathbb{E}_{\Theta \sim q(\Theta \mid \boldsymbol{\mu}, \boldsymbol{\sigma})} \left[\log p(T \mid \Theta, X)\right] + \\ + \frac{1}{2} \sum_{i=1}^{D} \sum_{j=1}^{K} \log \frac{\sigma_{ij}^2}{\mu_{ij}^2 + \sigma_{ij}^2} \longrightarrow \max_{\boldsymbol{\mu}, \boldsymbol{\sigma}} \end{aligned} \tag{23}$$

Это финальное выражение для метода релевантных векторов. Мы можем видеть, что первое слагаемое, называемое Data-term, отвечает за то, чтобы вариационные параметры хорошо описывали наблюдаемые данные, вынуждая варицинные параметры консолидироваться в точке максимума правдоподобия. В то же время второе слагаемое (KL-слагаемое) обрезает ненужные слагаемые, потому что, если $\mu_{ij}^2 + \sigma_{ij}^2 \longrightarrow 0$, то отвечающий им параметр θ_{ij} не несёт практической пользы, и поэтому может быть удалён.

Мы применили такую схему к нашей задаче следующим образом. Мы зафиксировали всю нейронную сеть и заново обучили только выходной слой с помощью описанного выше метода. То есть на вход этому алгоритму мы подавали выходы последнего рекуррентного слоя нейронной сети и предсказывали класс следующего слова. Результаты экспериментов представлены в Таблице 4. Видно, что удалось добиться сжатия выходного слоя

до 8 раз в сравнении с изначальной моделью. Это больше, чем с помощью техник матричного разложения, но при худшей перплексии.

Таблица 4: Результаты применения DSVI ARD

	Model	**Size, Mb**	**No. of parameters, M**	**No. of output layer parameters, M**	**Test perplexity**
PTB Benchmarks	LSTM 200-200	18.6	4.64	2.0	117.659
	LSTM 650-650	79.1	19.7	6.5	82.07
	LSTM 1500-1500	264.1	66.02	15.0	78.29
DSVI ARD for Softmax layer	LSTM 650-650	55.38	13.845	0.845	108.588

Отметим, что этот раздел на текущий момент не закончен и находится в активной стадии исследований. Проводятся численные эксперименты, по результатам которых планируется подготовить публикацию.

6 Апробация результатов исследования

Работа проходит апробацию. Основные результаты работы были опубликованы в следующих статьях:

1. Artem M. Grachev and Dmitry I. Ignatov and Andrey V. Savchenko. Neural Networks Compression for Language Modeling. — *International Conference on Pattern Recognition and Machine Intelligence. Springer, Cham, 2017.*

2. Artem M. Grachev and Dmitry I. Ignatov and Andrey V. Savchenko. Compression of Recurrent Neural Networks for Efficient Language Modeling. — *Applied Soft Computing, 2018; статья находится в процессе ревью.*

Сейчас ведётся работа над статьёй о применении техник байесовского сжатия для уменьшения числа параметров в рекуррентных нейронных сетей в задаче моделирования языка. Также за время обучения в аспирантуре были опубликованы следующие статьи:

1. A. Grachev , A. Shiriy Clustering techniques versus binary thresholding for detection of signal tracks in ionograms. *in: SCAKD 2016 – The Second International Workshop on Soft Computing Applications and Knowledge Discovery.*

2. Elena Andreeva, Dmitry I. Ignatov, Artem M. Grachev, Andrey V. Savchenko. Extraction of Visual Features for Recommendation of Products via Deep Learning — *LNCS 11179, AIST 2018*

Результаты диссертационного исследования докладывались на аспиранском семинаре факультета компьютерных наук ВШЭ и внутренних семинарах в компании Samsung R&D Russia.

Список литературы

[1] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1120–1128, 2016.

[2] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[4] Y. Bengio and J. Senecal. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*, 2003.

[5] B. P. Carlin and T. A. Louis. BAYES AND EMPIRICAL BAYES METHODS FOR DATA ANALYSIS. *Statistics and Computing*, 7(2):153–154, 1997.

[6] H. Deng, L. Zhang, and X. Shu. Feature memory-based deep recurrent neural network for language modeling. *Appl. Soft Comput.*, 68:432–446, 2018.

[7] L. Deng and D. Yu. Deep learning: Methods and application. *Foundations and Trends in Signal Processing*, 7:197–387, 2013.

[8] T. Garipov, D. Podoprikhin, A. Novikov, and D. P. Vetrov. Ultimate tensorization: compressing convolutional and FC layers alike. *CoRR*, abs/1611.03214, 2016.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] Ç. Gülçehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[12] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015.

[13] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *S. C. Kremer and J. F. Kolen, eds. A Field Guide to Dynamical Recurrent Neural Networks*, 2001.

[14] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[15] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[16] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pages 181–184, 1995.

[17] E. Lobacheva, N. Chirkova, and D. Vetrov. Bayesian sparsification of recurrent neural networks. *arXiv preprint arXiv:1708.00077*, 2017.

[18] Z. Lu, V. Sindhwani, and T. N. Sainath. Learning compact recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 5960–5964, 2016.

[19] T. Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[20] D. Molchanov, A. Ashukha, and D. P. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2498–2507, 2017.

[21] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, 2005.

[22] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6778–6787, 2017.

[23] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 442–450, 2015.

[24] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Scientific Computing*, 33(5):2295–2317, 2011.

[25] O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 157–163, 2017.

[26] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[27] M. K. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1971–1979, 2014.

[28] A. Tjandra, S. Sakti, and S. Nakamura. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4451–4458, 2017.

[29] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. J. Smola, L. Song, and Z. Wang. Deep fried convnets. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1476–1483, 2015.

[30] R. Yu, S. Zheng, A. Anandkumar, and Y. Yue. Long-term forecasting using tensor-train rnns. *CoRR*, abs/1711.00073, 2017.

[31] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.