Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра исследования операций

Кузнецова Мария Павловна

Экспериментальное исследование современных методов оптимизации, учитывающих кривизну функции потерь, для обучения нейронных сетей

Выпускная квалификационная работа

Научный руководитель
д.ф-м.н., профессор
А. Ф. Измаилов

Москва, 2023

Содержание

Введение

Нейронные сети широко используются в современном мире, с их помощью решаются различные задачи: распознавание изображений, перевод или продолжение текста, прогнозирование, и многие другие. Один из центральных вопросов состоит в оптимизации нейронных сетей, т. е. определении оптимальных весовых параметров сетей. Часто для оптимизации нейронных сетей используются методы первого порядка, они просты в реализации, однако недостаточно эффективны, и требуют большого количества оптимизационных шагов для отыскания решения нужного качества. Методы второго порядка служат основой наиболее эффективных алгоритмов в математической оптимизации. В то время как методы второго порядка часто обладают значительно лучшими свойствами сходимости, чем методы первого порядка, размер типичных задач препятствует их использованию на практике, поскольку они требуют хранения большого количества информации и тратят много времени для выполнения каждого шага оптимизации. Возможно, одна из важнейших задач современной оптимизации состоит в том, чтобы преодолеть этот разрыв между теоретической и практической оптимизацией, сделав методы второго порядка возможными для реализации и применения в задачах огромной размерности.

Пока этого не произошло, однако постоянно появляются новые методы оптимизации, которые совмещают быструю сходимость с относительно простой реализацией. В основе этой работы лежит исследование и сравнение современных методов оптимизации, а именно метода приближенной кривизны с учетом факторизации Кронекера (Kronecker-factored Approximate Curvature, K-FAC) [10], метода минимизации с учётом остроты минимума (Sharpness-Aware Minimization, SAM) [2], метода Evo**L**ved **S**ign M**o**me**n**tum (Lion) [1] и метода предобусловленной стохастической оптимизации (Shampoo) [4].

Основными целями работы являются численное сравнение и тестирование современных методов оптимизации для обучения нейронных сетей на задачах классификации. Для достижения данных целей в работе были поставлены следующие задачи.

- Сформулировать задачу классификации как задачу оптимизации.

- Программно реализовать обучение нейронных сетей с разными архитектурами с помощью упомянутых выше методов.
- Визуализировать результаты работы разных методов и сделать выводы.

В связи с этим в работе рассматривается использование указанных методов для оптимизации функции потерь нейронной сети при решении задач классификации изображений и сравнение полученных результатов с результатами решения той же задачи с использованием более простого метода оптимизации, а именно стохастического градиентного спуска (SGD). С использованием современных методов проводятся эксперименты на нейронных сетях разной глубины и на различных наборах данных. Это позволяет исследовать скорость сходимости, время, затрачиваемое на оптимизацию, и таким образом выяснить в каких случаях применение современных методов оптимизации может быть эффективно.

1 Нейронные сети

Искусственные нейронные сети (artificial neural networks) — это семейство моделей, созданных по подобию центральной нервной системы у животных. Они представляют собой систему связанных между собой нейронов, обменивающихся друг с другом сигналами. Нейронные сети используются для аппроксимации функций, которые в общем случае неизвестны, и могут зависеть от большого количества признаков.

Нейрон — это узел сети, имеющий n входов $x = (x^{(1)}, x^{(2)}, ..., x^{(n)})$ и один выход a. Со входами связан вектор вещественных параметров веса $w = (w^{(1)}, w^{(2)}, ..., w^{(n)})$. Помимо этого, нейрон имеет также параметр смещения b. Выход нейрона вычисляется следующим образом:

$$a = \phi(b + w^{(1)}x^{(1)} + ... + w^{(n)}x^{(n)}), \tag{1}$$

где к результату суммы применяется функция активации ϕ. Зачастую используют сигмоидную функцию активации $\phi(x) = 1/(1 + e^{-x})$ и функцию ReLU $\phi(x) = \max(x, 0)$.

Нейронные сети состоят из нескольких слоёв, каждый из которых состоит из нейронов. Нейронная сеть может быть отображена как ориентированный граф. Первый слой нейронной сети называют входным слоем (input layer), последний — выходным слоем (output layer), а промежуточные слои называются скрытыми слоями (hidden layers). Математически функция нейронной сети определяется как композиция функций, соответствующих каждому из слоёв сети.

1.1 Полносвязные слои

Полносвязными нейронными сетями называют такие нейросети, в которых в паре соседних слоев все нейроны связаны между собой. Тогда формулу i-го полносвязного слоя можно записать как

$$a_i = \phi_i(W_i a_{i-1} + b_i), \tag{2}$$

где n_{i-1} и n_i — количество нейронов в $i-1$ слое и i-м слое соответственно, a_{i-1} — выход $i-1$ слоя, который подаётся на вход i-му слою размера n_{i-1}, W_i — матрица

весов i-го слоя размера $n_i \times n_{i-1}$, b_i — вектор смещения i-го слоя размера n_{i-1}, a_i — выход из i-го слоя размера n_{i-1}, ϕ_i — функция активации i-го слоя.

Разобьём эту формулу:

$$s_i = W_i \bar{a}_{i-1}, \tag{3}$$

$$a_i = \phi_i(s_i), \tag{4}$$

где s_i — вектор взвешенных сумм, а $\bar{a}_{i-1}$ — определяется как вектор, сформированный путем добавления к a_{i-1} дополнительной однородной координаты со значением 1. Это нужно, чтобы не включать явные параметры смещения, поскольку они фиксируются неявно при использовании однородных координат.

1.2 Свёрточные нейронные сети. Свёрточные слои

Для начала рассмотрим операцию свёртки. Свёртка — операция $*$ над парой матриц A (размера $n_1 \times n_2$) и B (размера $m_1 \times m_2$), результатом которой является матрица $C = A * B$ размера $(n_1 - m_1 + 1) \times (n_2 - m_2 + 1)$. Каждый элемент результата

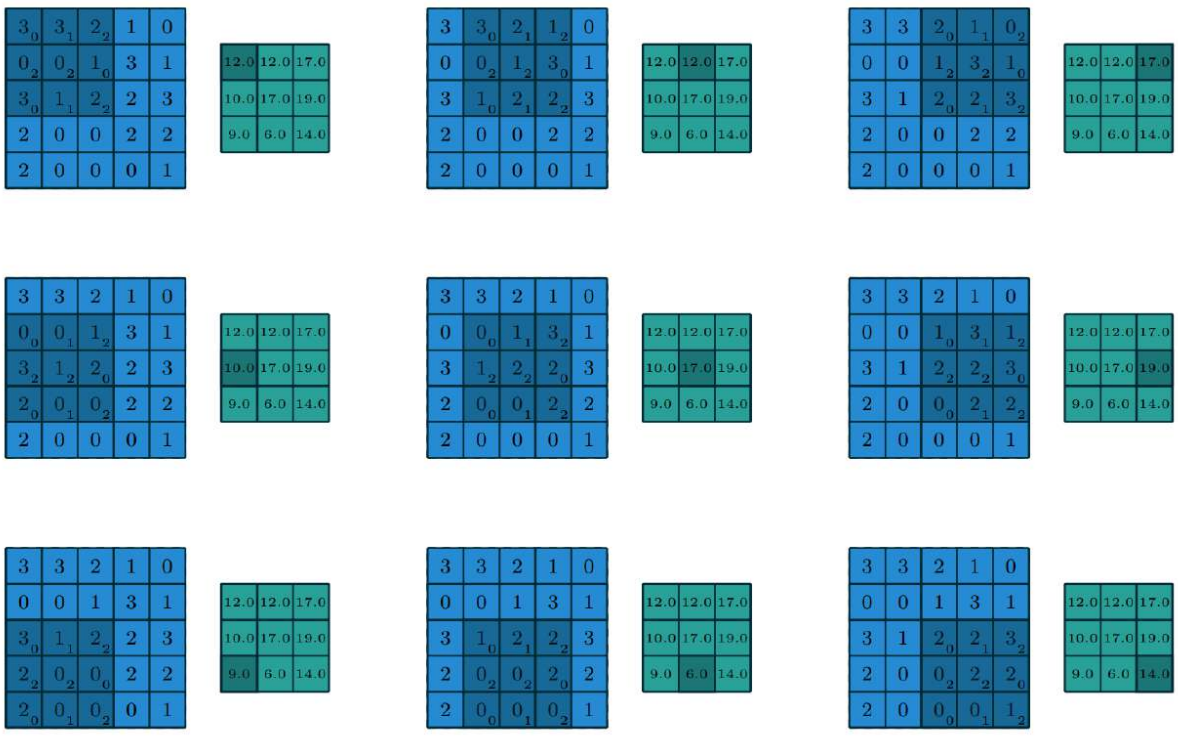


Рис. 1: Пример свертки двух матриц.

вычисляется как скалярное произведение матрицы B и некоторой подматрицы A такого же размера (подматрица определяется положением элемента в результате). То есть, $C_{i,j} = \sum_{u=0}^{m_1-1} \sum_{v=0}^{m_2-1} A_{i+u,j+v} B_{u,v}$. На рисунке 1 можно видеть, как матрица B «двигается» по матрице A, и в каждом положении считается скалярное произведение матрицы B и той части матрицы A, на которую она сейчас наложена. Получившееся число записывается в соответствующий элемент результата.

Свёрточные нейронные сети (Convolutional Neural Networks, CNN) — тип нейронных сетей, которые используют операцию свертки вместо общего матричного умножения по крайней мере в одном из своих слоев, а такой слой называют свёрточным. Такие нейронные сети используются в распознавании и обработке изображений.

CNN используют три типа слоёв: свёрточный слой, подвыборочный слой и полносвязный слой.

Свёрточный слой принимает на вход набор из n_{i-1} карт признаков (матриц) одного размера и выдаёт n_i матриц другого размера в соответствии с формулой:

$$a_i^{(t)} = \phi_i \left(\sum_{j=1}^{n_{i-1}} a_{i-1}^{(j)} * k^{j,t} + b^t \right), \quad t = 1, \ldots, n_i, \tag{5}$$

где $k^{j,t}$, b^t — параметры слоя, называемые фильтром и сдвигом, $*$ — операция свёртки входа a с ядром k.

1.3 Pooling слои

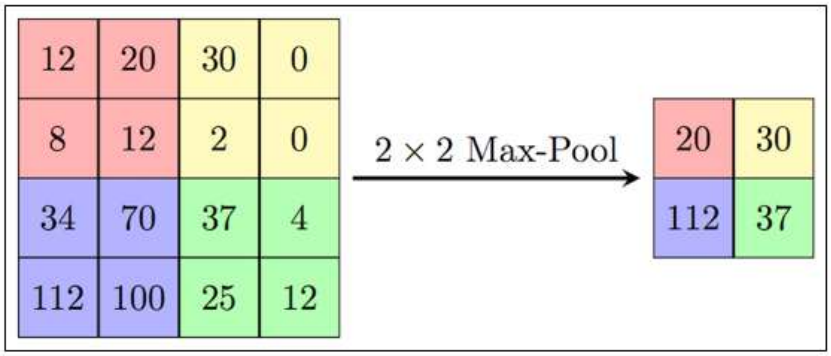


Рис. 2: Операция подвыборки (Max Pooling).

Подвыборочный (Pooling) слой так же, как и свёрточный обрабатывает матрицы признаков, но количество матриц признаков на входе и на выходе для этого слоя

совпадает. Операция подвыборки (или MaxPooling — выбор максимального) делается в соответствии с рисунком 2.

Тогда формулу i-го подвыборочного слоя можно записать как

$$a_i = subsample(a_{i-1}), \tag{6}$$

где $subsample(\cdot)$ — операция выборки максимальных значений в подматрицах матрицы входа.

1.4 Рекуррентные нейронные сети. Рекуррентные слои

Рекуррентные нейронные сети (RNN) — это класс нейронных сетей, которые хороши для моделирования последовательных данных, таких как временные ряды или естественный язык. Идея RNN заключается в последовательном использовании информации. Если нужно предсказать следующее слово в предложении, лучше учитывать предшествующие ему слова. RNN называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем выход зависит от предыдущих вычислений.

Рассмотрим формулу i-го рекуррентного слоя. Для каждого $a_{i-1}^{(t)}$ элемента вектора a_{i-1}, $t = \overline{1, n_{i-1}}$, в t-й момент генерируется выход $a_i^{(t)}$ в соответствии с формулами:

$$a_0^{(t)} = x^{(t)}, \tag{7}$$

$$a_i^{(t)} = \phi_i(W_i^{hx} a_{i-1}^{(t)} + W_i^{hh} a_i^{(t-1)} + b_i^h), \tag{8}$$

$$a_{Last}^{(t)} = \phi_{Last}(W_L^{ah} a_L^{(t)} + b^a). \tag{9}$$

Здесь матрица весов представлена в трёх частях — W_i^{hx}, W_i^{hh}, W^{ah}, вектор смещения представлен двумя частями — b_i^h, b^a. В соответствии с формулой (8) t-й элемент i-го слоя хранит информацию о предыдущих $t-1$ элементах. Для последнего рекуррентного слоя дополнительно применяется формула (9).

1.5 Функции активации

Одним из этапов разработки нейронной сети является выбор функции активации нейронов. Вид функции активации во многом определяет функциональные

возможности нейронной сети и метод обучения этой сети. В данной работе в качестве функции активации в выходном слое применяется Softmax [3], в свёрточных слоях применяется ReLU, а в рекуррентных слоях — гиперболический тангенс $\phi(x) = (e^{2x} - 1)/(e^{2x} + 1)$.

Softmax используют в задачах многоклассовой классификации. Для подобной классификации сеть строят таким образом, что на последнем слое количество нейронов оказывается равным количеству искомых классов. При этом каждый нейрон должен выдавать значение вероятности принадлежности объекта к соответствующему классу, то есть значение между нулём и единицей, а все нейроны в сумме должны дать единицу.

Вид формулы softmax:

$$\phi(x)^{(i)} = \frac{e^{x^{(i)}}}{\sum_{k=1}^{K} e^{x^{(k)}}}, \tag{10}$$

где K — количество классов.

Известно, что нейронные сети способны приблизить сколь угодно сложную функцию, если в них достаточно слоев, и функция активации является нелинейной. Функции активации, такие как логистическая (сигмоидная) функция или гиперболический тангенс, являются нелинейными и приводят к проблемам с затуханием или увеличением градиентов. Однако можно использовать и гораздо более простой вариант — функцию активации "линейный выпрямитель"(rectified linear unit, ReLU), которая чаще всего используется в свёрточных сетях.

2 Постановка задачи

Рассмотрим задачу классификации. Пусть $X \subset \mathbb{R}^n$ — конечное множество векторов (описаний объектов), $Y = \{1, \ldots, c\}$ — конечное множество допустимых ответов, и существует функция правильных ответов $f^* : X \to Y$, значения которой $y_i = f^*(x_i)$ известны только на конечном множестве объектов $\{x_1, \ldots, x_m\} \subset X$. Совокупность пар (x_i, y_i) объект-ответ $D^k = \{(x_i, y_i) \in X \times Y | i = \overline{1,k}\}$ и $D^h = \{(x_i, y_i) \in X \times Y | i = \overline{k+1,m}\}$ называются обучающей выборкой и контрольной выборкой соответственно, где $h = m - k$. Тогда $X^k = \{x_i \in X | i = \overline{1,k}\}$ и $Y^k = \{y_i \in Y | i = \overline{1,k}\}$, X^h и Y^h аналогично.

Определим $\theta = [vec(W_1)^T, vec(W_2)^T, \ldots, vec(W_l)^T]^T$, который представляет собой вектор, состоящий из всех параметров (весовых коэффициентов) нейронной сети, объединенных вместе, где vec — это оператор, который векторизует матрицы, конкатенируя их столбцы, l — количество слоёв нейронной сети.

Пусть $P^c = \{p \in \mathbb{R}^c : p_1 + \ldots + p_c = 1, p_i \geq 0, i = \overline{1,c}\}$, $f(\theta, \cdot) : \mathbb{R}^n \to P^c$ — функция нейронной сети с n-мерным входом и c-мерным выходом вероятностей принадлежности объекта к каждому из классов.

Задача обучения нейронной сети заключается в том чтобы, по обучающей выборке D^k найти такие параметры θ, чтобы нейронная сеть f приближала функцию правильных ответов f^*.

2.1 Функция потерь

Функция потерь — это неотрицательная функция $L(\theta, D^k)$, характеризующая величину ошибки функции нейронной сети при данном наборе параметров θ на обучающей выборке.

Тогда задача обучения нейронной сети будет заключаться в поиске таких параметров θ, при которых функция потерь минимальна:

$$L(\theta, D^k) \to \min_{\theta}. \tag{11}$$

Рассмотрим крросс-энтропийную функцию потерь. Положим

$$H(P, Q) = -\sum_{i=1}^{k} P(x_i) \log(Q(x_i)), \tag{12}$$

где P — распределение истинных ответов, Q — распределение вероятностей прогнозов модели (то, что получается на выходе нейронной сети).

Пусть $\bar{y}_i$ — это $c-$мерный вектор из 0 и 1, имеющий единицу на месте j, если объект x_i принадлежит к классу j. Тогда кросс-энтропийная функция потерь имеет вид

$$L(\theta, D^k) = H(Y^k, f(\theta, X^k)) = -\sum_{i=0}^{k} \bar{y}_i \log(f(\theta, x_i)) =$$

$$= -\sum_{i=0}^{k} \log(p(y_i|x_i, \theta)) = -\log(p(Y^k|X^k, \theta)), \tag{13}$$

где $p(y_i|x_i, \theta)$ — это вероятность, с которой нейросеть относит объект x_i к классу y_i (то, что сеть с параметрами θ выдает для класса y_i на входе x_i).

Таким образом, получается следующая задача:

$$L(\theta, D^k) = -\log p(Y^k|X^k, \theta) \to \min_{\theta}. \tag{14}$$

Дополнительно, для некоторого вектора v, от которого зависит функция потерь, введём производную функции потерь по этому вектору:

$$Dv = \nabla_v L(\theta, x, y). \tag{15}$$

Тогда обозначим производную функции потерь по взвешенной сумме s_i из уравнения (3) как $g_i = Ds_i$.

2.2 Стохастическая оптимизация

Для обучения нейронных сетей используют разные методы оптимизации, которые пошагово делают некоторые преобразования параметров сети, приближающие их к оптимальному значению для задачи (11). Шаги делаются до тех пор, пока значение функции потерь не станет достаточно мало.

Зачастую обучающая выборка делится на несколько непересекающихся частей (батчей). Тогда случайным образом выбирается батч, с использованием которого делается шаг обучения в соответствии с выбранным методом оптимизации. Таким же образом выполняются шаги обучения с использованием оставшихся батчей. Эпоха — последовательность шагов, в ходе которой используются все батчи. Количество

эпох при обучении может доходить до нескольких сотен. Батч, взятый на шаге t, обозначим через D_t^k.

Помимо значения функции потерь на обучающей выборке для оценки того, насколько хорошо нейронная сеть приближает исходную зависимость, изучают значение функции потерь и точность (отношение количества верных ответов нейронной сети к количеству объектов в выборке) на контрольной выборке.

2.3 Регуляризация

При поиске оптимальных параметров будет использоваться регуляризация весов. Регуляризация — это способ борьбы со слишком большими значениями параметров сети и как следствие этого переобучением (излишне точным соответствием нейронной сети конкретному набору обучающих примеров, при котором сеть теряет способность к обобщению).

Данный способ состоит в прибавлении манхеттенской нормы $L_1 = \lambda||\theta||_1$ или Евклидовой нормы $L_2 = \lambda||\theta||_2^2$, называющихся L_1-регуляризацией и L_2-регуляризацией соответственно. При обучении нейронных сетей в основном используется L_2-регуляризация. То есть с учетом регуляризации задача будет состоять в минимизации функции вида

$$-\log p(Y^k|X^k,\theta) + \lambda||\theta||_2^2, \tag{16}$$

где λ — настраиваемый коэффициент регуляризации.

3 Методы оптимизации

3.1 Метод стохастического градиентного спуска (SGD)

Стандартным методом обучения нейронных сетей является метод стохастического градиентного спуска. Однако он может расходиться, или сходиться очень медленно, если шаг обучения настроен плохо. Поэтому существует много альтернативных методов, разработанных с целью улучшить сходимость.

Стохастический градиентный спуск обновляет параметр, вычитая градиент оптимизируемой функции по соответствующему параметру и масштабируя его на шаг обучения α. Если α слишком большой, то метод будет расходиться; если слишком маленький — будет сходиться медленно. Правило вычисления параметров θ_{t+1} выглядит следующим образом:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(\theta_t, D_t^k), \tag{17}$$

где функция потерь считается на некотором батче из обучающей выборки.

3.2 Метод приближённой кривизны с учетом факторизации Кронекера (K-FAC)

K-FAC является аппроксимацией метода натурального градиента (natural gradient), шаг которого определяется как $F^{-1}\nabla_\theta L(\theta, D_t^k)$, где через F обозначается так называемая информационная матрица Фишера. Она используется для вычисления ковариационных матриц, связанных с оценками максимального правдоподобия. Для нашей задачи информационная матрица фишера будет выглядеть следующим образом:

$$F = E\left[\nabla_\theta L(\theta, D_t^k) \nabla_\theta L(\theta, D_t^k)^T\right], \tag{18}$$

где через E обозначается математическое ожидание относительно случайной величины Y_t^k распределённой как $p(Y_t^k | X_t^k, \theta)$.

Пусть

$$\nabla_\theta L(\theta, D_t^k) = D\theta, \tag{19}$$

тогда

$$F = E[D\theta D\theta^T]. \tag{20}$$

3.2.1 Блочное приближение Фишера с помощью факторизации Кронекера

Произведение Кронекера — бинарная операция над матрицами A, B произвольного размера, которая обозначается как $A \otimes B$. Если A — матрица размера $m \times n$, B — матрица размера $p \times q$, тогда произведение Кронекера есть блочная матрица размера $mp \times nq$, имеющая вид

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \dots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \dots & a_{m,n}B \end{bmatrix}. \tag{21}$$

Основной вычислительной задачей, связанной с использованием натурального градиента, является вычисление обратной матрицы Фишера F^{-1}. В этом разделе разрабатывается начальное приближение, которое будет ключевым компонентом в получении эффективно вычислимого приближения F^{-1}.

Заметим, что $D\theta = [vec(DW_1)^T, vec(DW_2)^T, ..., vec(DW_l)^T]^T$. Тогда
$F = E[D\theta D\theta^T] = E[[vec(DW_1)^T, ..., vec(DW_l)^T]^T [vec(DW_1)^T, ..., vec(DW_l)^T]] =$

$$= \begin{bmatrix} E[vec(DW_1)vec(DW_1)^T] & \dots & E[vec(DW_1)vec(DW_l)^T] \\ \vdots & \ddots & \vdots \\ E[vec(DW_l)vec(DW_1)^T] & \dots & E[vec(DW_l)vec(DW_l)^T] \end{bmatrix}.$$

Таким образом, можно видеть, что матрицу F можно рассматривать как квадратную блочную матрицу размера $l \times l$, с (i,j)-м блоком $F_{i,j}$, заданным как $F_{i,j} = E[vec(DW_i)vec(DW_j)^T]$.

Заметим, что $DW_i = g_i \bar{a}_{i-1}^T$, а также, что $vec(uv^T) = v \otimes u$, тогда $vec(DW_i) = vec(g_i \bar{a}_{i-1}^T) = \bar{a}_{i-1} \otimes g_i$. Перепишем $F_{i,j}$, используя это:

$$F_{i,j} = E[vec(DW_i)vec(DW_j)^T] = E[(\bar{a}_{i-1} \otimes g_i)(\bar{a}_{j-1} \otimes g_j)^T] =$$

$$= E[(\bar{a}_{i-1} \otimes g_i)(\bar{a}_{j-1}^T \otimes g_j^T)] = E[\bar{a}_{i-1}\bar{a}_{j-1}^T \otimes g_i g_j^T].$$

Сделаем приближение блока $\tilde{F}$ к F:

$$F_{i,j} = E[\bar{a}_{i-1}\bar{a}_{j-1}^T \otimes g_i g_j^T] \approx E[\bar{a}_{i-1}\bar{a}_{j-1}^T] \otimes E[g_i g_j^T] = \bar{A}_{i-1,j-1} \otimes G_{i,j} = \tilde{F}_{i,j},$$

где $\bar{A}_{i,j} = E[\bar{a}_i \bar{a}_j^T]$ и $G_{i,j} = E[g_i g_j^T]$.

Тогда начальное приближение $\tilde{F}$ к F будет определяться следующим блочным приближением:

$$\tilde{F} = \begin{bmatrix} \bar{A}_{0,0} \otimes G_{1,1} & \bar{A}_{0,1} \otimes G_{1,2} & \dots & \bar{A}_{0,l-1} \otimes G_{1,l} \\ \bar{A}_{1,0} \otimes G_{2,1} & \bar{A}_{1,1} \otimes G_{2,2} & \dots & \bar{A}_{1,l-1} \otimes G_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{A}_{l-1,0} \otimes G_{l,1} & \bar{A}_{l-1,1} \otimes G_{l,2} & \dots & \bar{A}_{l-1,l-1} \otimes G_{l,l} \end{bmatrix}.$$

Данное приближение имеет форму, известную как произведение Хатри-Рао в многомерной статистике.

Математическое ожидание произведения Кронекера, как правило, не равно произведению математических ожиданий сомножителей, поэтому это приближение, вероятно, не будет точным при любом реалистичном наборе допущений. Тем не менее на практике оно оказывается полезным.

3.2.2 Аппроксимация $\tilde{F}^{-1}$ как блочно-диагональной матрицы

Далее нужно эффективное приближение обратной матрицы Фишера. Существует два способа реализовать это: приближение с помощью блочно-диагональной матрицы и приближение с помощью блочно-трехдиагональной. Здесь подробнее рассмотрим первый более строгий способ.

Аппроксимация $\tilde{F}^{-1}$ блочно-диагональной матрицей эквивалентна аппроксимации $\tilde{F}$ блочно-диагональной матрицей. Естественным выбором для такого приближения $\hat{F}$ к $\tilde{F}$ является принятие блочной диагонали $\hat{F}$ за диагональ $\tilde{F}$. Это дает матрицу

$$\hat{F} = diag(\tilde{F}_{1,1}, \tilde{F}_{2,2}, \dots, \tilde{F}_{l,l}) = diag(\bar{A}_{0,0} \otimes G_{1,1}, \bar{A}_{1,1} \otimes G_{2,2}, \dots, \bar{A}_{l-1,l-1} \otimes G_{l,l}).$$

Используя следующее свойство произведения Кронекера: $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, можно легко вычислить обратную матрицу к $\hat{F}$ как

$$\hat{F}^{-1} = diag(\bar{A}_{0,0}^{-1} \otimes G_{1,1}^{-1}, \bar{A}_{1,1}^{-1} \otimes G_{2,2}^{-1}, \dots, \bar{A}_{l-1,l-1}^{-1} \otimes G_{l,l}^{-1}).$$

Таким образом, вычисление $\hat{F}^{-1}$ равносильно вычислению обратных к $2l$ меньшим матрицам $\bar{A}_{i,i}$ и $G_{i,i}$.

Затем, чтобы вычислить $u = \hat{F}^{-1}v$, можно использовать известное тождество $(A \otimes B)vec(X) = vec(BXA^T)$, чтобы получить

$$U_i = vec(G_{i,i}^{-1} V_i \bar{A}_{i-1,i-1}^{-1}),$$

где v и u некоторые векторы, соответствующие $(V_1, V_2, \ldots, V_l)$ и $(U_1, U_2, \ldots, U_l)$, аналогично тому, как θ соответствует $(W_1, W_2, \ldots, W_l)$.

Также в этом методе используется эффективный подбор некоторых внутренних коэффициентов и другие эвристики, о которых подробнее можно узнать в статье [10].

3.3 Метод минимизации с учётом остроты минимума (SAM)

Зачастую методы оптимизации направлены на поиск значений параметров θ, в которых потери при обучении $L(\theta, D^k)$ имеют низкое значение. Вместо этого данный метод предлагает искать значения параметров, в некоторой окрестности которых функция потерь имеет значения близкие к минимальным.

Чтобы прояснить термин «острота», рассмотрим

$$[\max_{||\epsilon||_2 \leq \rho} L(\theta + \epsilon, D^k) - L(\theta, D^k)] + L(\theta, D^k) + \lambda ||\theta||_2^2,$$

где $\rho \geq 0$. Выражение в квадратных скобках отражает остроту минимума функции $L(\theta, D^k)$ в точке θ, измеряя, насколько быстро потери при обучении могут быть увеличены путем перехода от θ к близким значениям параметра. Это выражение затем суммируется с самим значением потерь при обучении и L_2-регуляризатором величины θ. Таким образом, предлагается выбирать значения параметров, решая следующую задачу минимизации с учетом остроты:

$$L^{SAM}(\theta, D^k) + \lambda ||\theta||_2^2 \to \min_{\theta}, \tag{22}$$

где

$$L^{SAM}(\theta, D^k) = \max_{||\epsilon||_2 \leq \rho} L(\theta + \epsilon, D^k). \tag{23}$$

Чтобы минимизировать $L^{SAM}(\theta, D^k)$, необходимо использовать приближение к $\nabla_\theta L^{SAM}(\theta, D^k)$, которого можно достичь с помощью дифференцирования через внутреннюю максимизацию. Сначала вычислим приближение задачи внутренней максимизации с помощью разложения Тейлора первого порядка $L(\theta + \epsilon, D^k)$, для достаточно малого ϵ. Получаем

$$\epsilon^*(\theta) = \arg\max_{||\epsilon||_2 \leq \rho} L(\theta, D^k) \approx \arg\max_{||\epsilon||_2 \leq \rho} L(\theta, D^k) + \epsilon^T \nabla_\theta L(\theta, D^k) =$$

$$= \arg\max_{||\epsilon||_2 \leq \rho} \epsilon^T \nabla_\theta L(\theta, D^k).$$

В свою очередь, значение $\bar{\epsilon}(\theta)$ в правой части этой формулы задается выражением

$$\bar{\epsilon}(\theta) = \rho \nabla_\theta L(\theta, D^k) / ||\nabla_\theta L(\theta, D^k)||_2. \tag{24}$$

Тогда

$$\nabla_\theta L^{SAM}(\theta, D^k) \approx \nabla_\theta L(\theta + \bar{\epsilon}(\theta), D^k) = \frac{\partial(\theta + \bar{\epsilon}(\theta))}{\partial \theta} \nabla_{\theta + \bar{\epsilon}(\theta)} L(\theta + \bar{\epsilon}(\theta), D^k) =$$

$$= \nabla_{\theta + \bar{\epsilon}(\theta)} L(\theta + \bar{\epsilon}(\theta), D^k)| + \frac{\partial \bar{\epsilon}(\theta)}{\partial \theta} \nabla_{\theta + \bar{\epsilon}(\theta)} L(\theta + \bar{\epsilon}(\theta), D^k). \tag{25}$$

Для ускорения вычисления, отбросим члены второго порядка. Тогда приближение градиента принимает окончательный вид

$$\nabla_\theta L^{SAM}(\theta, D^k) \approx \nabla_{\theta + \bar{\epsilon}(\theta)} L(\theta + \bar{\epsilon}(\theta), D^k). \tag{26}$$

Обновления весов происходят в соответствии с формулой:

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta + \bar{\epsilon}(\theta)} L(\theta + \bar{\epsilon}(\theta), D^k). \tag{27}$$

Подробнее о методе SAM можно см. в [2].

3.4 Shampoo: Предобусловленная стохастическая оптимизация

Предобусловленные градиентные методы являются одними из наиболее общих и мощных инструментов оптимизации. Однако предобуславливание требует хранения непомерно больших матриц и манипулирования ими. В данном методе реализуется новый алгоритм предобуславливания, называемый Shampoo, для стохастической оптимизации в многомерных пространствах. Shampoo использует набор предобуславливателей, каждый из которых работает с одним измерением. Эксперименты с самыми современными моделями глубокого обучения показывают, что Shampoo способен сходиться значительно быстрее, чем обычно используемые оптимизаторы, хотя он и включает в себя более сложное правило обновления весов.

3.4.1 Shampoo для матриц

Рассмотрим частный случай применения метода Shampoo. Параметры образуют матрицу $\theta \in \mathbb{R}^{n \times c}$. В методах первого порядка обновление θ_t происходит на основе градиента $\nabla_\theta L(\theta_t, D_t^k)$, который также является матрицей размера $n \times c$.

Схема предобуславливания полной матрицы, не учитывающая структуру, преобразовала бы пространство параметров в многомерный вектор и использовала бы предобуславливатели H_t размера $nc \times nc$. Напротив, Shampoo использует меньшие предобуславливатели: левый Hl_t размера $n \times n$ и правый Hr_t размера $c \times c$, содержащие информацию о втором моменте накопленных градиентов. На каждой итерации предобуславливатели формируются из Hl_t и Hr_t и перемножаются с матрицей градиента слева и справа соответственно. Размер пространства, который данный метод использует для матрицы, равен $n^2 + c^2$ вместо n^2c^2. Более того, поскольку предварительная обработка включает обращение матрицы (и часто собственное разложение), объем вычислений, необходимый для построения левой и правой матриц предобуславливания, составляет $O(n^3 + c^3)$, что существенно ниже, чем полноматричные методы, которые требуют $O(n^3c^3)$. Обновления весов происходят в соответствии с формулами:

$$Hl_t = Hl_{t-1} + \nabla_\theta L(\theta_t, D_t^k)(\nabla_\theta L(\theta_t, D_t^k))^T, \tag{28}$$

$$Hr_t = Hr_{t-1} + (\nabla_\theta L(\theta_t, D_t^k))^T \nabla_\theta L(\theta_t, D_t^k), \tag{29}$$

$$\theta_{t+1} = \theta_t - \alpha Hl_t^{-1/4} \nabla_\theta L(\theta_t, D_t^k) Hr_t^{-1/4}. \tag{30}$$

Резюмируя, Shampoo поддерживает две разные матрицы: матрицу $Hl_t^{1/4}$ для предобуславливания строк $\nabla_\theta L(\theta_t, D_t^k)$ и $Hr_t^{1/4}$ для ее столбцов. Использование показателя $1/4$ обосновано в [4]. Алгоритм можно рассматривать как поддерживающий «структурированную» матрицу, которая неявно используется для предобусловленного градиента, без формирования полной матрицы и без явного выполнения произведения с вектором сглаженного градиента. Подробнее о методе Shampoo и его обобщении на многомерный случай см. в [4].

3.5 EvoLved Sign Momentum (Lion)

Метод Lion был недавно разработан исследователями из Google с использованием эволюционного алгоритма автоматического машинного обучения (AutoML) [11]. Одними из широко используемых алгоритмов для обучения нейронных сетей являются Adam [6] и Adafactor [13]. Хотя с помощью этих оптимизаторов получаются хорошие результаты, команда Google задалась вопросом, можно ли использовать AutoML для их улучшения. Результатом исследования стал метод Lion.

EvoLved Sign Momentum — это стохастический алгоритм оптимизации для глубокого обучения, основанный на концепции оптимизации знакового импульса. Lion направлен на устранение ограничений традиционных алгоритмов оптимизации, таких как стохастический градиентный спуск и Adam. Ключевое отличие Lion от других оптимизаторов заключается в том, что Lion учитывает только знак компонент градиента. Обновление весов происходит в соответствии с формулами:

$$\widetilde{m}_{t+1} = \beta_2 m_t + (1 - \beta_2)\nabla_\theta L(\theta_t, D_t^k), \tag{31}$$

$$\theta_{t+1} = \theta_t - \alpha(\text{sign}(\widetilde{m}_t) + \lambda\theta_t), \tag{32}$$

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1)\nabla_\theta L(\theta_t, D_t^k), \tag{33}$$

где m_{t+1}, $\widetilde{m}_{t+1}$ — это оценки момента первого порядка градиента, λ — коэффициент регуляризации, β_1, β_2 — это экспоненциальные скорости затухания для оценок на данный момент, $0 < \beta_1 < 1$, $0 < \beta_2 < 1$, задаваемые в начале обучения.

В целом, Lion — многообещающий алгоритм оптимизации, который показал конкурентоспособную производительность по сравнению с другими современными алгоритмами оптимизации в различных задачах глубокого обучения, включая классификацию изображений, обнаружение объектов и обработку естественного языка. Подробности об открытии метода Lion и его особенностях см. в [1].

3.6 Планировщик скорости обучения

Одним из основных гиперпараметров методов оптимизации является шаг обучения α, который определяет величину изменения весов модели. От него зачастую сильно зависят результаты. В простейшем случае α является фиксированным значением от 0 до 1.

Однако, выбор удачного значения α может оказаться сложной задачей. С одной стороны, большой шаг обучения может привести к более быстрой сходимости алгоритма, так как для этого потребуется меньше шагов, а значит, и времени. Но это также может привести к тому, что алгоритм будет не в состоянии достичь минимума, так как растояние до него будет меньше длины шага. С другой стороны, используя более низкую скорость обучения, алгоритм сможет лучше приблизиться к минимуму. Однако, оптимизатору потребуется больше времени для сходимости, если шаг слишком мал.

Одним из решений, помогающих алгоритму быстро и эффективно сходиться к оптимуму, является использование планировщика скорости обучения, который уменьшает шаг после каждой эпохи обучения. Благодаря этому, в начале обучения алгоритм делает большие шаги, что позволяет не тратить много времени на приближение к области вокруг минимума, а в конце, когда шаг стоновится малым, алгоритм даёт возможность более точно приблизиться к минимуму. Изменение шага обучения происходит в соответствии с формулой

$$\alpha_j = \alpha_{min} + \frac{1}{2}(\alpha_{max} - \alpha_{min}) \left(1 + \cos\left(\frac{j}{J_{max}}\pi\right)\right), \tag{34}$$

где α_{max} — начальный шаг, α_{min} — минимальный шаг (часто используется $\alpha_{min} = 0$), J_{max} — количество эпох, j — номер текущей эпохи, $1 \leq j \leq J_{max}$.

Обычно скорость обучения устанавливается на более высокое значение в начале обучения, и с каждой эпохой уменьшается. Уменьшение шага в процессе обучения также известно как «отжиг». Планировщик скорости обучения используется в одном из экспериментов ниже для улучшения результатов. Мотивация такого подхода к изменению шага обучения приводится в [9].

4 Экспериментальные результаты

В данном разделе приводятся результаты работы методов на различных наборах данных. Рассматриваются три задачи с различными по сложности архитектурами нейронных сетей и различными данными. Во всех случаях функция потерь нейронной сети отдельно минимизируется каждым из рассмотренных выше методов. Далее сравниваются результаты обучения такие, как точность, значение функции потерь, время обучения. Перед итоговым сравнением для каждого метода подбираются шаг обучения и коэффициент регуляризации. Подбираются именно эти параметры, так как они сильнее всего влияют на результаты обучения. Для части методов шаг обучения подбирается среди значений: 0.1, 0.03, 0.01. Для метода Shampoo шаг подбирается среди больших значений, для метода Lion наоборот подбирается из меньших значений. Коэффициент регуляризации подбирается среди значений: 0, 0.001, 0.0003, 0.0001.

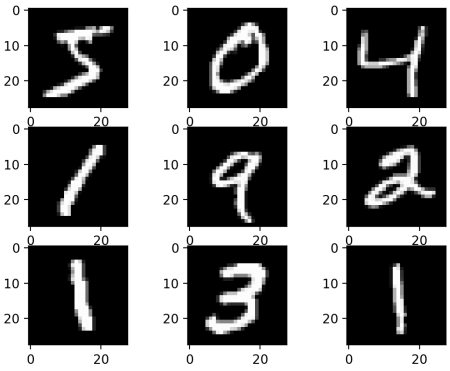


Рис. 3: Пример изображений цифр для первого эксперемента.

Рис. 4: Пример цветных изображений для второго эксперемента.

4.1 Задача классификации цифр

В данном эксперименте в качестве данных выступали чёрно-белые изображения цифр от 0 до 9 из набора данных MNIST [8]. Обучающая выборка представлена 60000 экземплярами, а контрольная выборка 10000 экземплярами.

Обучалась свёрточная нейронная сеть, известная как LeNet5 [8]. Структура данной нейронной сети:

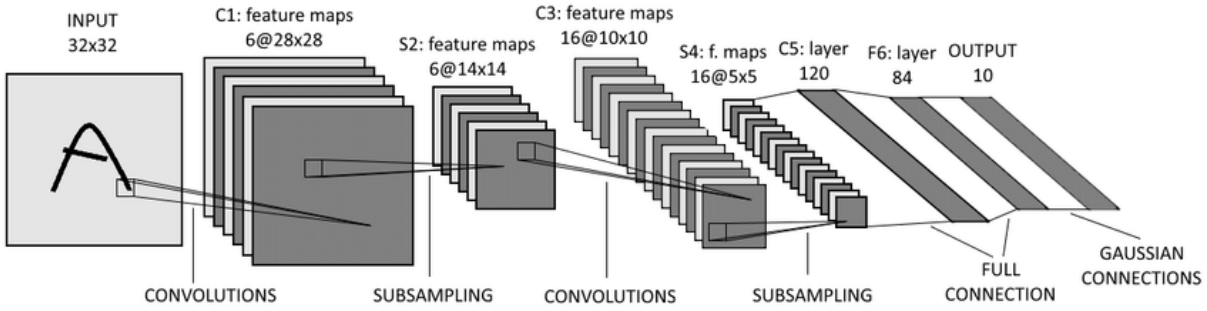


Рис. 5: LeNet5.

1) слой C1 представляет собой свёрточный слой с шестью фильтрами, который преобразует входную матрицу размера 32×32 в 6 матриц размера 28×28;

2) слой S2 — это слой подвыборки, уменьшающий 6 матриц с предыдущего слоя до размера 14×14;

3) слой C3 представляет собой свёрточный слой с шестнадцатью фильтрами, который преобразует 6 матриц с предыдущего слоя в 16 матриц размера 10×10;

4) слой S4 является слоем подвыборки, уменьшающий 16 матриц с предыдущего слоя до размера 5×5;

5) слой C5 представляет собой свёрточный слой с 120 фильтрами, который преобразует 16 матриц с предыдущего слоя в 120 матриц размера 1×1;

6) полносвязный слой F6 содержит 84 нейрона и полностью связан со свёрточным слоем C5;

7) полносвязный слой OUTPUT возвращает данные о вероятности принадлежности классу.

Процесс минимизации функции проводился в 20 эпох, размер пакета (батча) 100, то есть 20 раз производился проход по обучающей выборке, в каждой эпохе было $60000/100 = 600$ итераций оптимизации одним из методов.

Подбор параметров

Для итоговых сравнений для каждого метода были подобраны шаг обучения α и коэффициент регуляризации λ, для метода SAM был также подобран параметр ρ:

Метод оптимизации	α	λ	ρ
SGD	0.1	0	—
SAM	0.03	0	0.1
Lion	0.0001	0	—
K-FAC	0.01	0.0003	—
Shampoo	0.1	0.0001	—

Подробнее с графиками, использовавшимися для подбора параметров, можно ознакомиться в приложении 1.

Итоговые результаты

С подобранными параметрами нейронная сеть была обучена каждым из пяти методов. На рисунках ниже представлены график зависимости точности от количества эпох на тестовом наборе (рисунок 6), график зависимости значения функции потерь от количества эпох на тестовом наборе (рисунок 7) и на обучающем наборе (рисунок 8) в зависимости от метода, использованного при обучении нейронной сети.

При помощи всех методов были получены схожие результаты. Модели, обученные с использованием метода Lion и Shampoo, раньше других достигли высокой точности. Было установлено, что при использовании Shampoo, точность равномерно увеличивается, а значение функции потерь равномерно уменьшается, без каких-либо скачков.

Также было замерено время, затраченное на обучение нейронной сети в течении 20 эпох каждым из методов (рисунок 9). Из него можно видеть, что применение методов K-FAC и Shampoo в практических задачах может быть неэффективно, на них уходит слишком много времени, а результаты находятся на уровне результатов метода SGD. Метод SAM тратит примерно в два раза больше времени чем SGD, а при использовании Lion было затрачено лишь чуть больше времени чем на SGD.

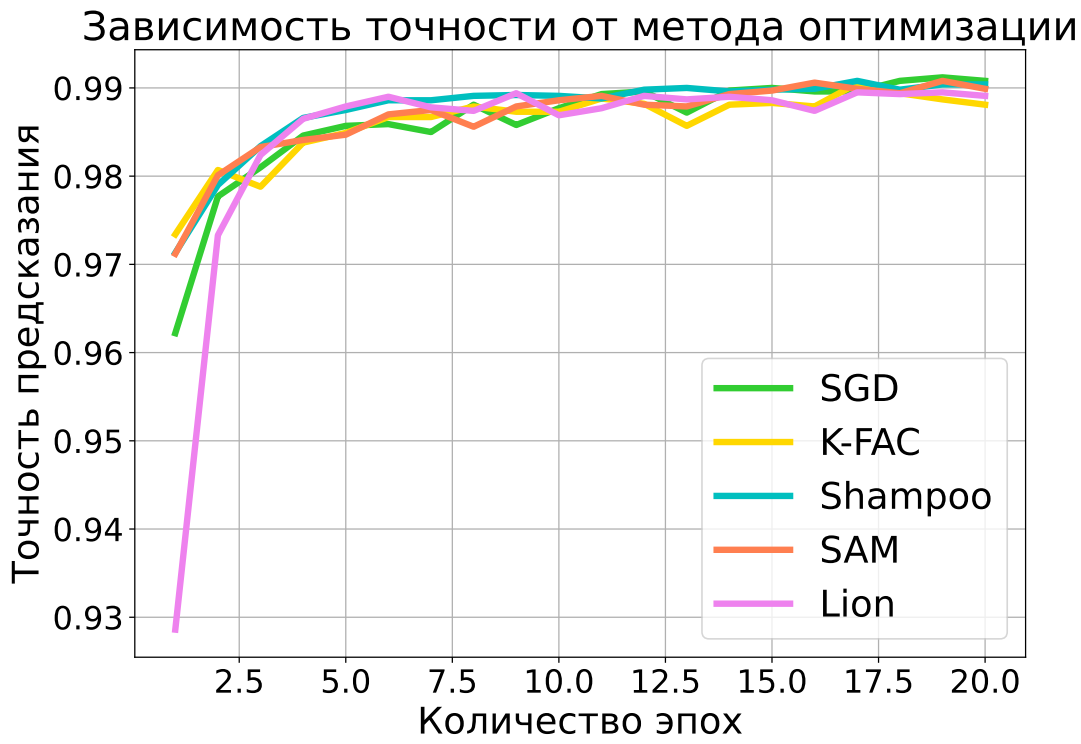


Рис. 6: График зависимости точности от количества эпох для разных методов.

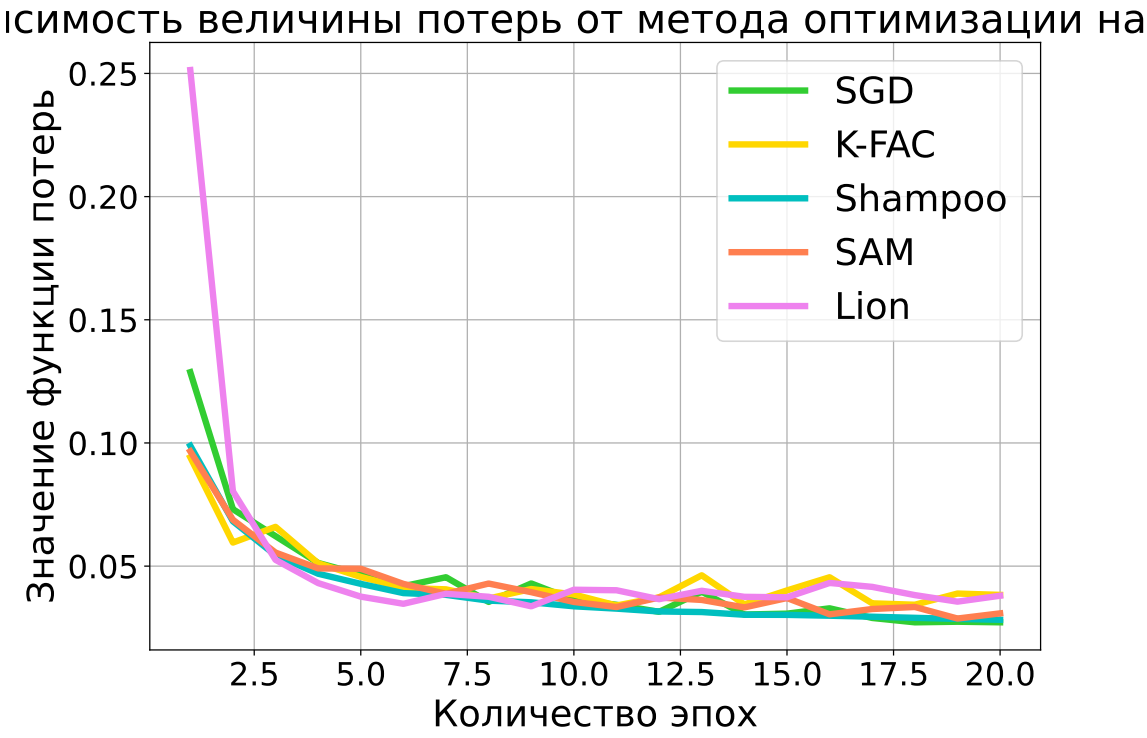


Рис. 7: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

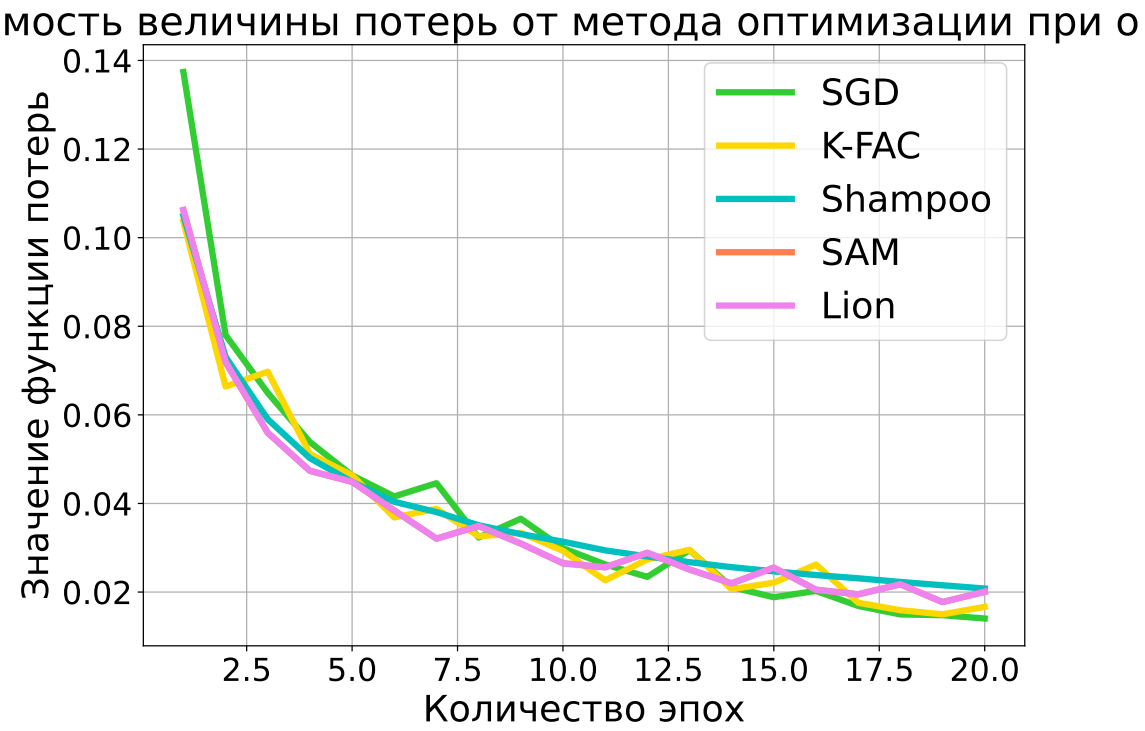


Рис. 8: График зависимости значения функции потерь от количества эпох при обучении для разных методов.

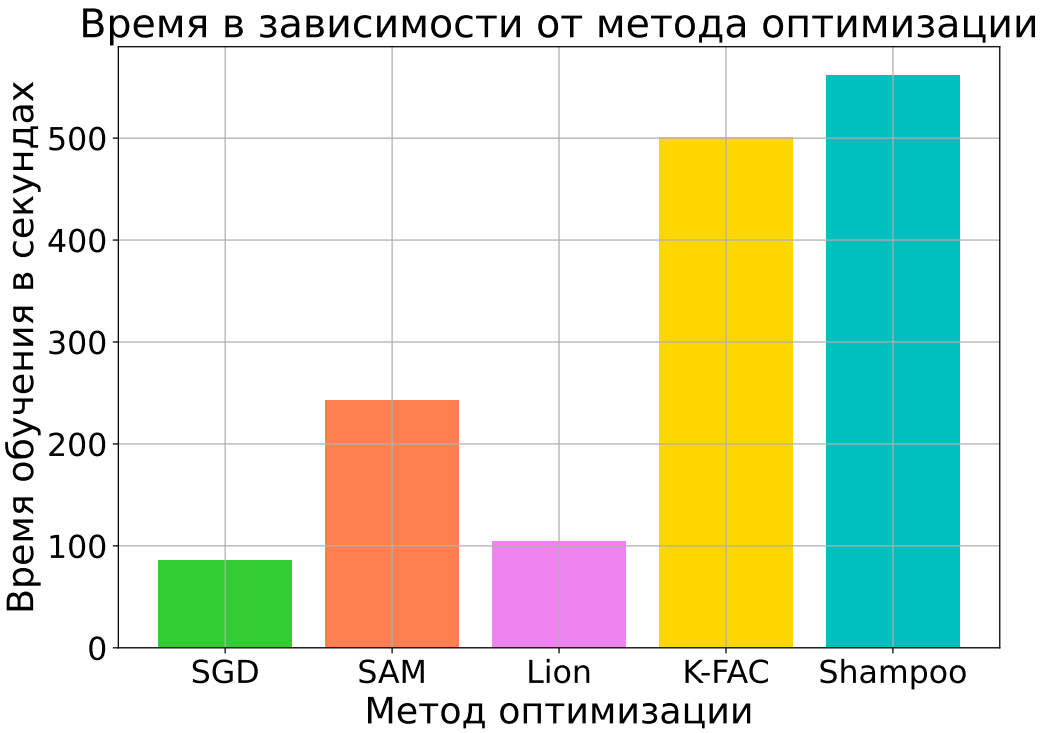


Рис. 9: График времени, затраченного на обучение, в зависимости от метода.

Для более точного сравнения нейронная сеть была обучена до точности 0.985 с помощью разных методов, и были произведены замеры времени, понадобившегося для этого обучения. Ниже представлены график зависимости точности обучения от количества эпох (рисунок 10) и график зависимости времени (рисунок 11), потраченного на обучение до точности 0.985, от метода, выбранного для обучения. Исходя из них можно видеть, что наиболее эффективным методом оказался SAM, он за меньшее количество эпох и меньшее время достиг нужной точности. Методы K-FAC и Shampoo в этом эксперименте показывают себя наиболее неэффективными. Однако K-FAC достигает нужной точности быстрее чем Shampoo. При обучении с помощью Lion затрачивается примерно столько же времени сколько и при использовании SGD.

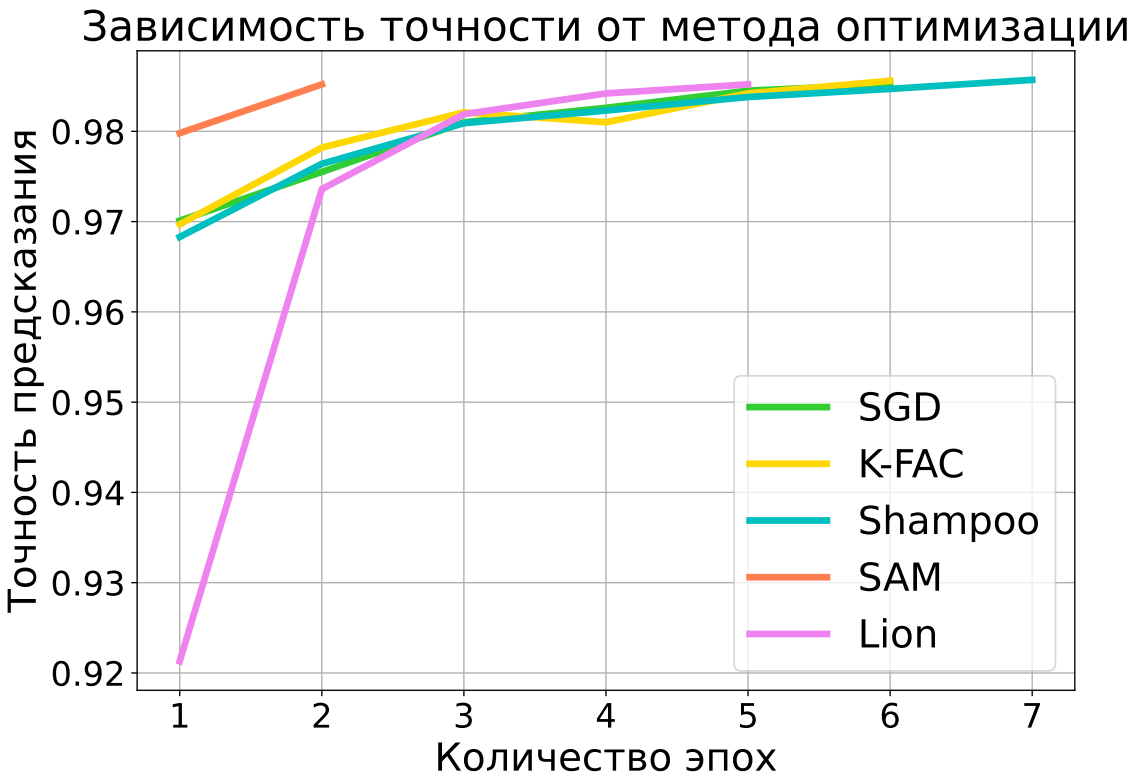


Рис. 10: График зависимости точности от количества эпох для разных методов.

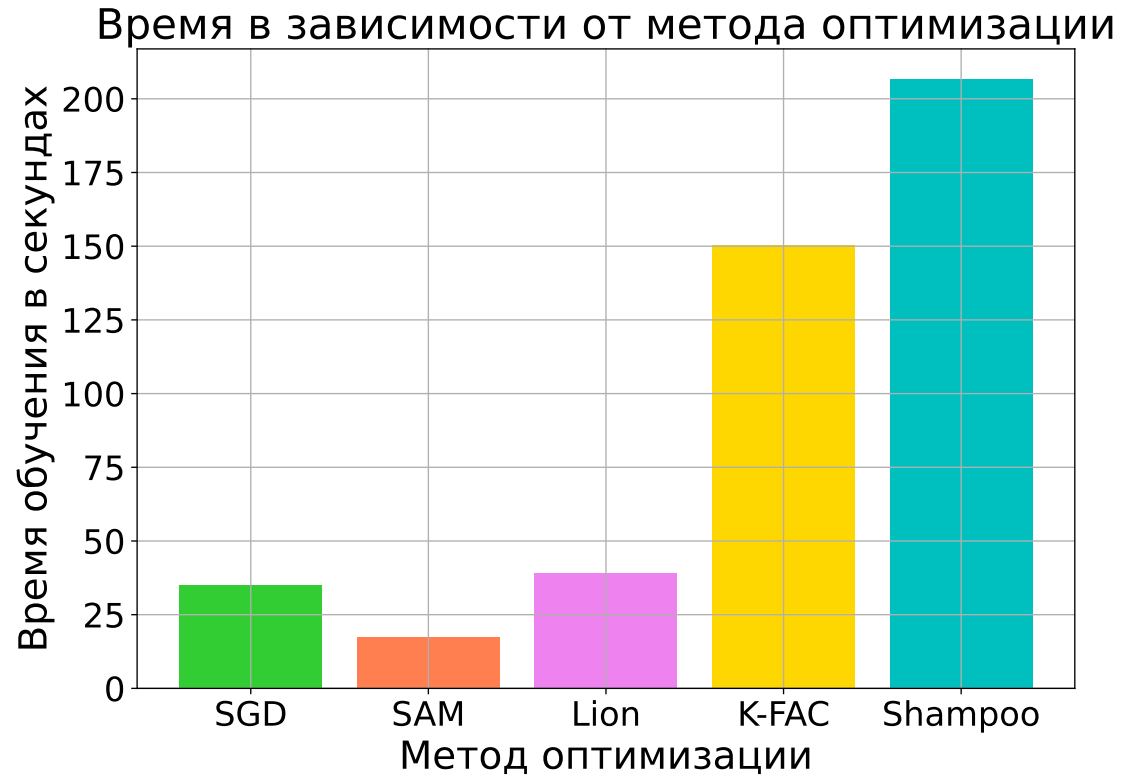


Рис. 11: График зависимости времени, затраченного на обучение, от используемого метода.

4.2 Задача классификации цветных изображений

В данном эксперименте в качестве данных выступали изображения из набора данных CIFAR-10 [7]. Он состоит из цветных изображений размером 32×32 в 10 классах: самолёт, автомобиль, птица, кот, олень, собака, лягушка, лошадь, корабль, грузовик. Классы полностью взаимоисключающие. Между легковым и грузовым транспортом нет пересечения. Обучающая выборка представлена 50000 экземплярами, а тестовая выборка 10000 экземплярами.

В этом эксперименте использовалась свёрточная нейронная сеть, известная как ResNet-18 [5]. В её архитектуре присутствует 18 слоев. Она очень полезна и эффективна при классификации изображений и может классифицировать изображения по 1000 категориям объектов. Процесс минимизации функции проводился в 20 эпох, размер пакета (батча) 100.

Подбор параметров

Для итоговых сравнений для каждого метода были подобраны шаг обучения α и коэффициент регуляризации λ, для метода SAM был также подобран параметр ρ:

Метод оптимизации	α	λ	ρ
SGD	0.1	0.001	—
SAM	0.03	0.0003	0.1
Lion	0.0001	0.01	—
K-FAC	0.1	0.001	—
Shampoo	2	0	—

Подробнее с графиками, использовавшимися для подбора параметров, можно ознакомиться в приложении 2.

Итоговые результаты

С подобранными параметрами нейронная сеть была обучена каждым из пяти методов. На рисунках ниже представлены график зависимости точности от количества эпох на тестовом наборе (рисунок 12), график зависимости значения функции потерь от количества эпох на тестовом наборе (рисунок 13) и на обучающем наборе (рисунок 14) в зависимости от метода, использованного при обучении нейронной сети.

С помощью метода SGD точность, обученной им, модели достигает 0.82. Исходя из рисунка 13 можно сделать вывод, что присутствуют элементы переобучения, так как в окрестности 9-й эпохи значение функции потерь на тесте увеличивается. Результаты модели, обученной с помощью метода K-FAC, достигают своего максимума к 7-й эпохе, а после, что видно по рисунку 13, начинается переобучение. Максимальная точность в районе 0.82. Модель, обученная с помощью метода Shampoo, достигает точности лишь 0.76 к восемнадцатой эпохе, что хуже результатов, полученных с помощью SGD. А после 9-й эпохи модель сильно переобучается, значение функции потерь на тесте увеличивается, а на контрольной выборке почти в нуле. При использовании метода Lion нейронная сеть быстрее всего достигает высокой точности 0.82, но после 7-й эпохи переобучается.

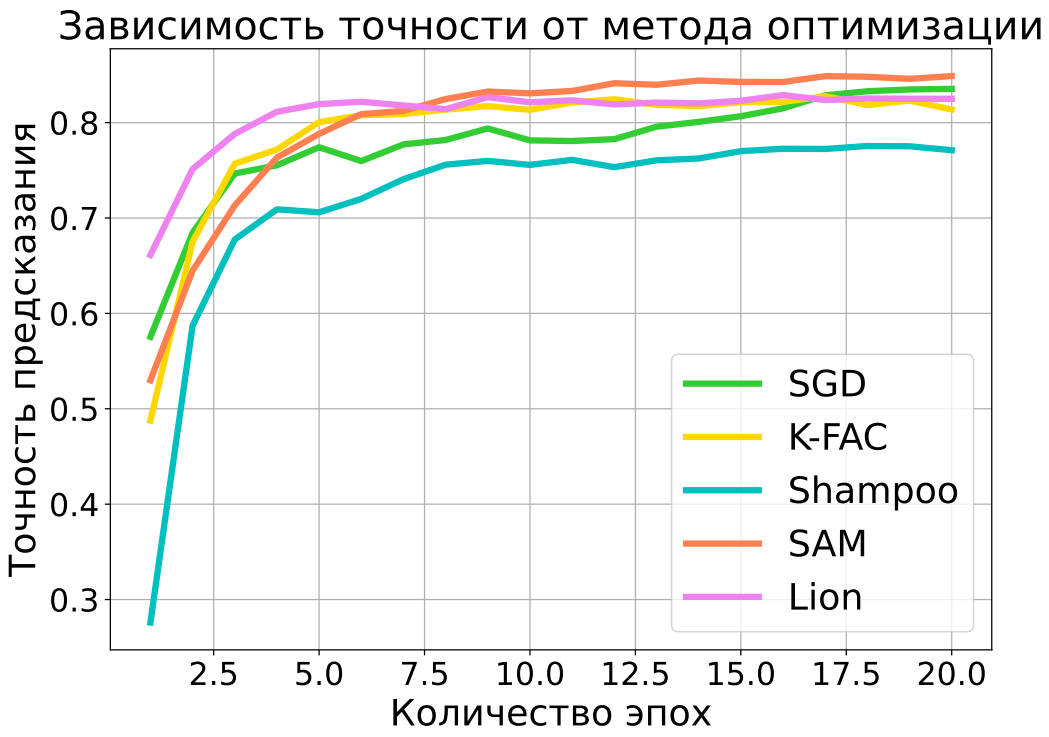


Рис. 12: График зависимости точности от количества эпох для разных методов.

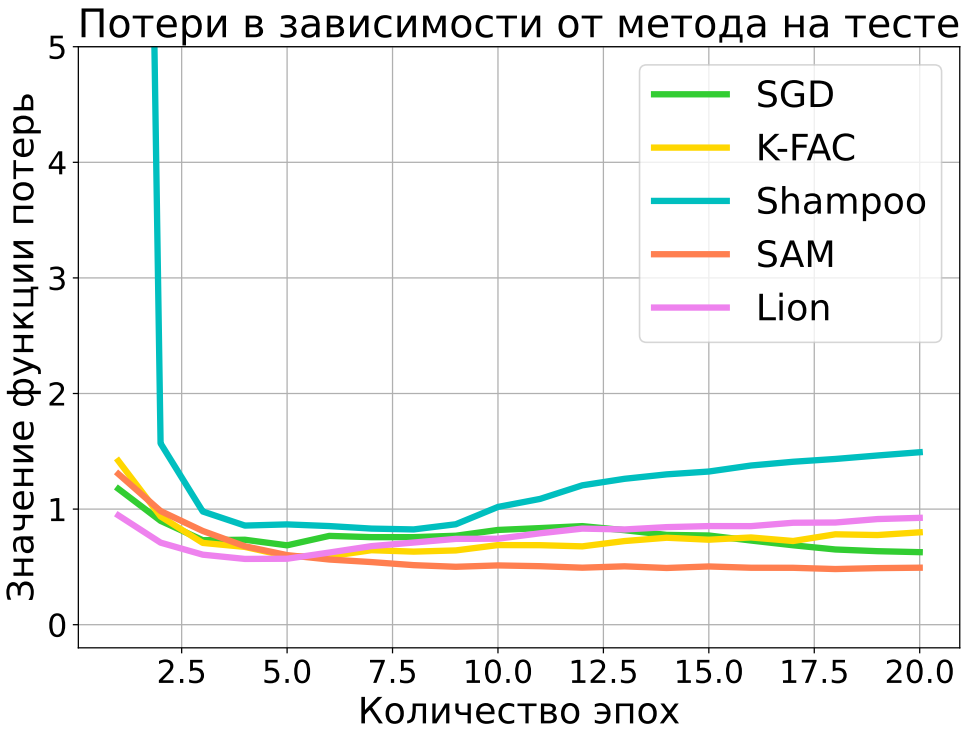


Рис. 13: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

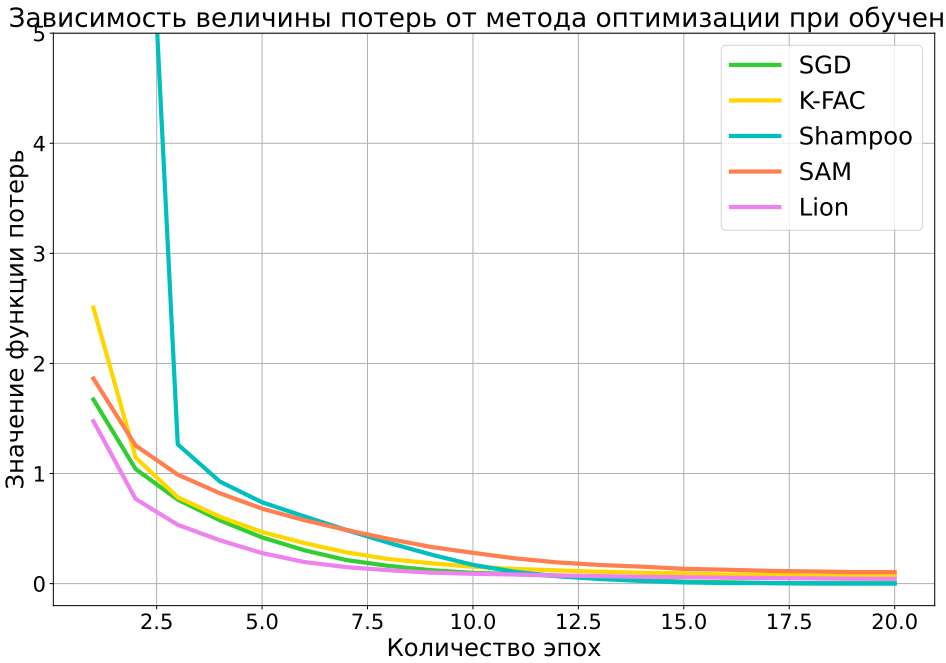


Рис. 14: График зависимости значения функции потерь от количества эпох при обучении для разных методов.

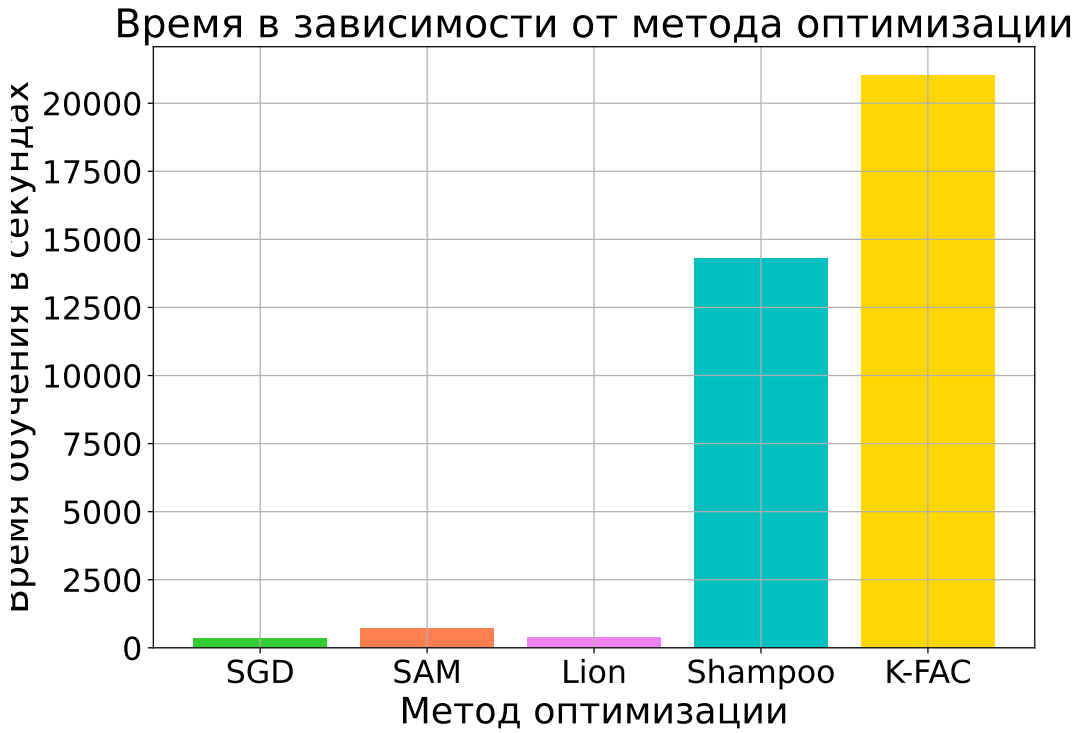


Рис. 15: График времени, затраченного на обучение, в зависимости от метода.

Лучше всего показывает себя метод SAM. Точность, достигнутая с помощью него, выше всех остальных, в окрестностях 0.83. Исходя из рисунка 13, при использовании данного метода модель не подвергается переобучению. И при продолжении обучения есть перспектива улучшить полученный результат.

На графике выше также было показано время, затраченное на обучение нейронной сети в течении 20 эпох каждым из методов (рисунок 15). Из него можно видеть, что применение методов K-FAC и Shampoo оказывается слишком затратно. Минимальное количество времени затрачивается при использовании методов SGD и Lion. При использовании метода SAM тратится в два раза больше времени, чем при использовании SGD, но при этом получается лучший результат.

Для более точного сравнения нейронная сеть была обучена до точности 0.75 с помощью разных методов, и были произведены замеры времени, понадобившегося для этого обучения. Ниже представлены график зависимости точности обучения от количества эпох (рисунок 16) и график зависимости время (рисунок 17), потраченного на обучение до точности 0.75, от метода, выбранного для обучения.

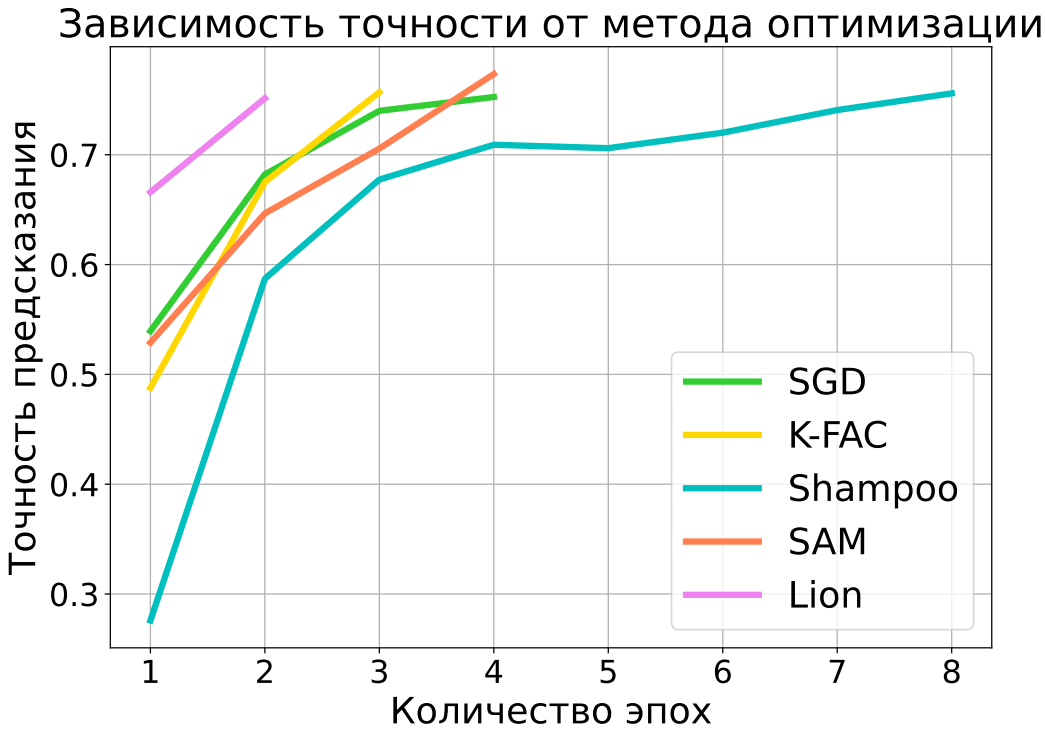


Рис. 16: График зависимости точности от количества эпох для разных методов.

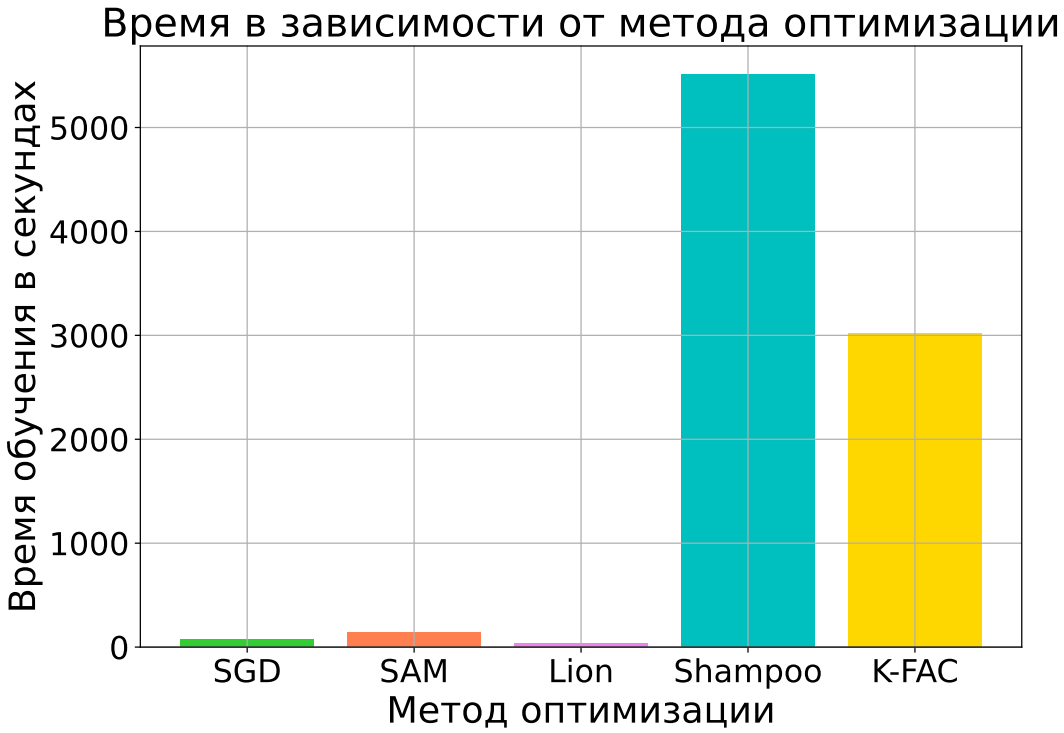


Рис. 17: График времени, затраченного на обучение, в зависимости от метода.

Исходя из них, можно видеть, что меньшее количество эпох и меньшее количество времени было затрачено при использовании метода Lion, а большее при использовании Shampoo. Модель, обученная с помощью K-FAC, всего за три эпохи достигла необходимой точности, но из-за дороговизны шагов данного метода даже на это было потрачено достаточно много времени. При использовании SGD и SAM необходимая точность была достигнута за 4 эпохи, но при использовании метода SAM было затрачено больше времени.

В ходе данного эксперимента была получена точность лишь в районе 0.84. Чтобы улучшить данный результат, был использован планировщик скорости обучения. Было интересно исследовать поведение методов при изменении шага обучения с течением эпох. Для данного эксперимента необходимо было подобрать больший шаг обучения α, чем в предыдущем эксперименте, для каждого метода:

Метод оптимизации	α	λ	ρ
SGD	1	0.001	—
SAM	0.2	0.0003	0.1
Lion	0.0003	0.01	—
K-FAC	0.5	0.001	—
Shampoo	2	0	—

С подобранными параметрами и с использованием планировщика скорости обучения была обучена нейронная сеть каждым из пяти методов. На рисунках 18, 19, 20 представлены графики с результатами.

Результаты, полученные с помощью всех методов кроме Shampoo, улучшились. Лучше всего вновь показал себя метод SAM, чуть хуже него K-FAC, ещё хуже Lion и SGD. Модель, обученная с помощью метода Lion, быстрее всех показала хороший результат. Точность удалось поднять до 0.87.

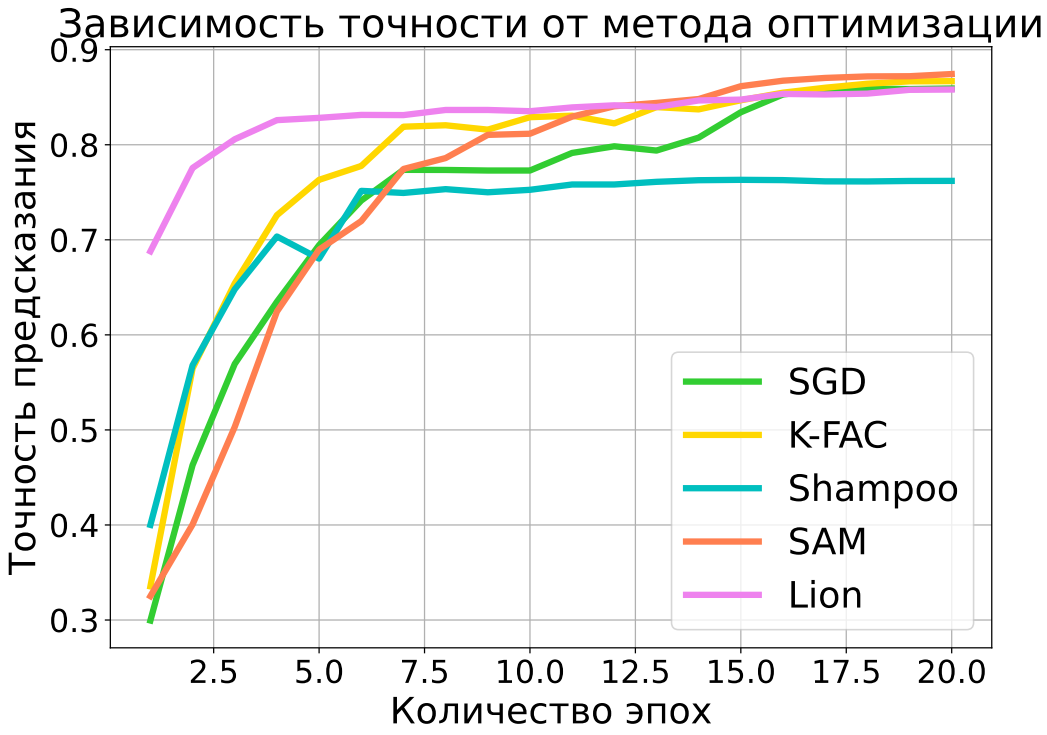


Рис. 18: График зависимости точности от количества эпох для разных методов.

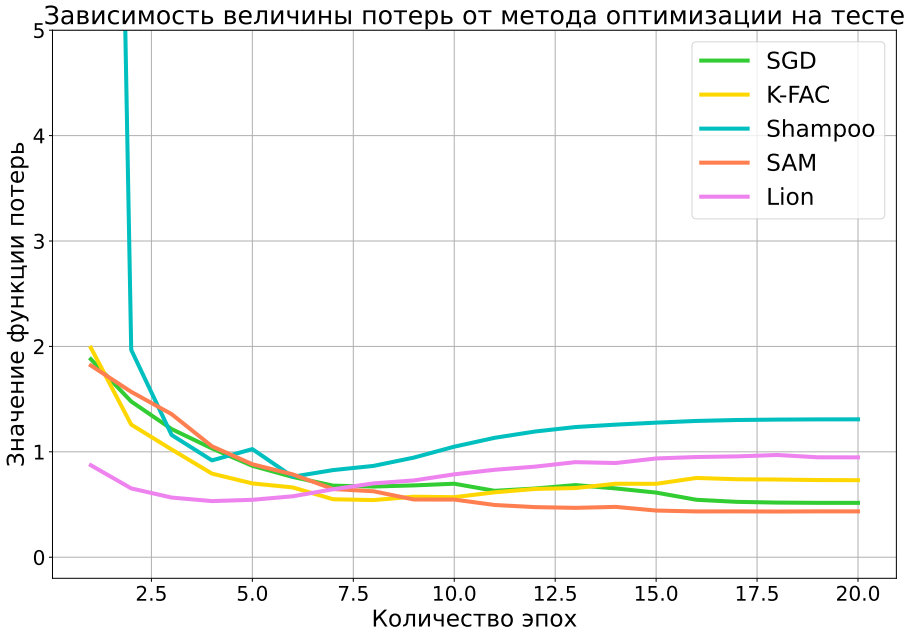


Рис. 19: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

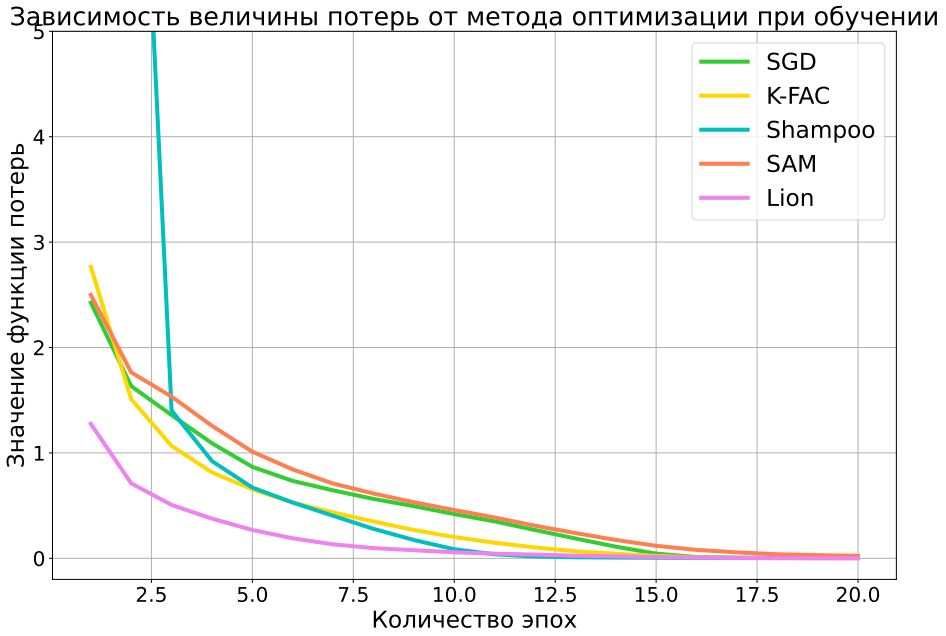


Рис. 20: График зависимости значения функции потерь от количества эпох при обучении для разных методов.

4.3 Задача классификации именованных сущностей

В данном эксперименте в качестве данных выступали предложения на английском языке из набора данных CoNLL [14]. Общая задача касается распознавания именованных объектов. Рассматривались следующие типы именованных сущностей: лица, места, организации и названия различных сущностей, не принадлежащих к предыдущим трем группам. Каждому слову из предложения нужно было сопоставить тип.

Обучающая выборка представлена 15000 экземплярами, а контрольная выборка 3500 экземплярами. Нейронная сеть, использовавшаяся в данном эксперименте, состояла из модификации рекуррентного слоя LSTM [12] и полносвязного слоя. Процесс минимизации функции проводился в 20 эпох, размер пакета (батча) 2 предложения.

Подбор параметров

В данном эксперименте не участвовал метод Shampoo, из-за его неэффективности и сложности его применения к рекуррентным нейронным сетям. Для итоговых сравнений для каждого метода были подобраны шаг обучения α и коэффициент регуляризации λ, для метода SAM был также подобран параметр ρ:

Метод оптимизации	α	λ	ρ
SGD	0.1	0.0003	—
SAM	0.03	0.0001	0.1
Lion	0.0001	0.001	—
K-FAC	0.03	0.001	—

Подробнее с графиками, использовавшимися для подбора параметров, можно ознакомиться в приложении 3.

Итоговые результаты

С подобранными параметрами нейронная сеть была обучена каждым из четырех методов. На рисунках ниже представлены график зависимости точности от количества эпох на тестовом наборе (рисунок 21), график зависимости значения функции потерь от количества эпох на тестовом наборе (рисунок 22) и на обучающем наборе (рисунок 23) в зависимости от метода, использованного при обучении нейронной сети.

Модель, обученная методом K-FAC, показывает худшие результаты. Кривая K-FAC делает резкие скачки на рисунках 21 и 22. Можно сделать предположение, что данный метод не стоит использовать с рекуррентными нейронными сетями. При использовании метода Lion модель с первых эпох достигает высокой точности, однако после 4-й эпохи лишь переобучается, так как точность не увеличивается, а даже немного уменьшается, значение функции потерь на контрольной выборке лишь увеличивается (рисунок 22), значение функции потерь на обучающей выборке после 8-й эпохи почти в нуле. Методы SAM и SGD показывают схожие хорошие результаты, оба не подвергаются переобучению, однако метод SAM всё же показывает лучший результат.

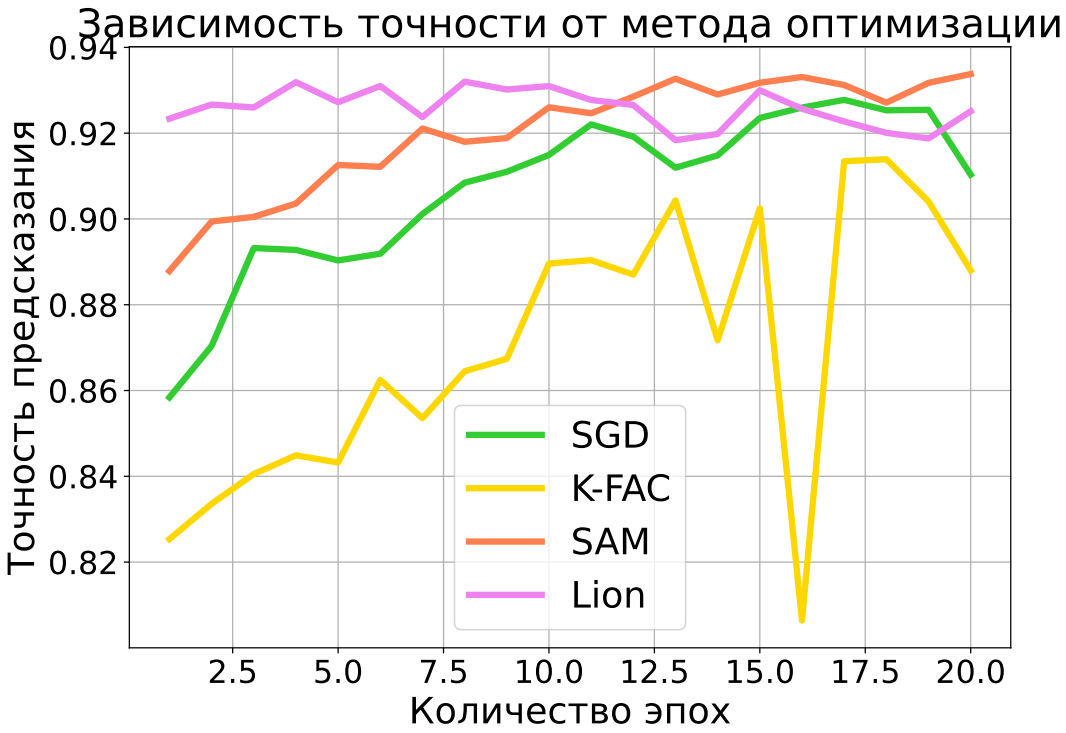


Рис. 21: График зависимости точности от количества эпох для разных методов.

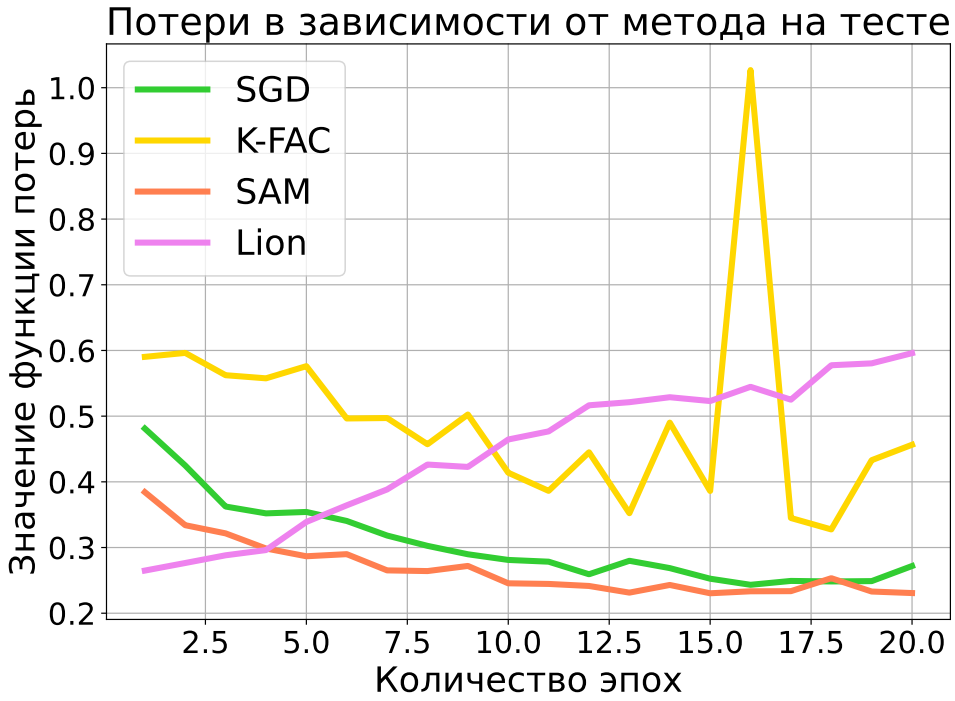


Рис. 22: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

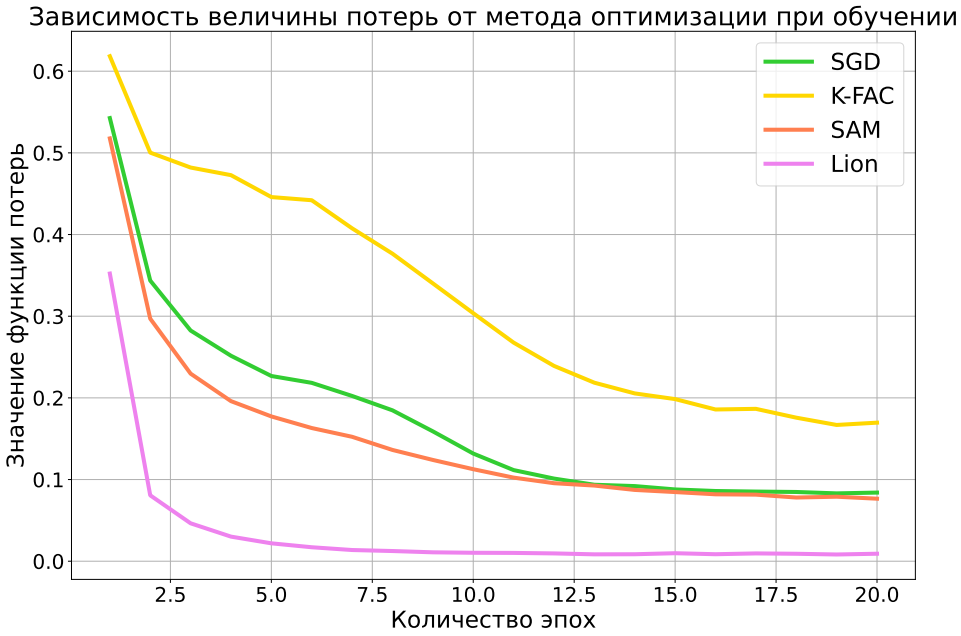


Рис. 23: График зависимости значения функции потерь от количества эпох при обучении для разных методов.

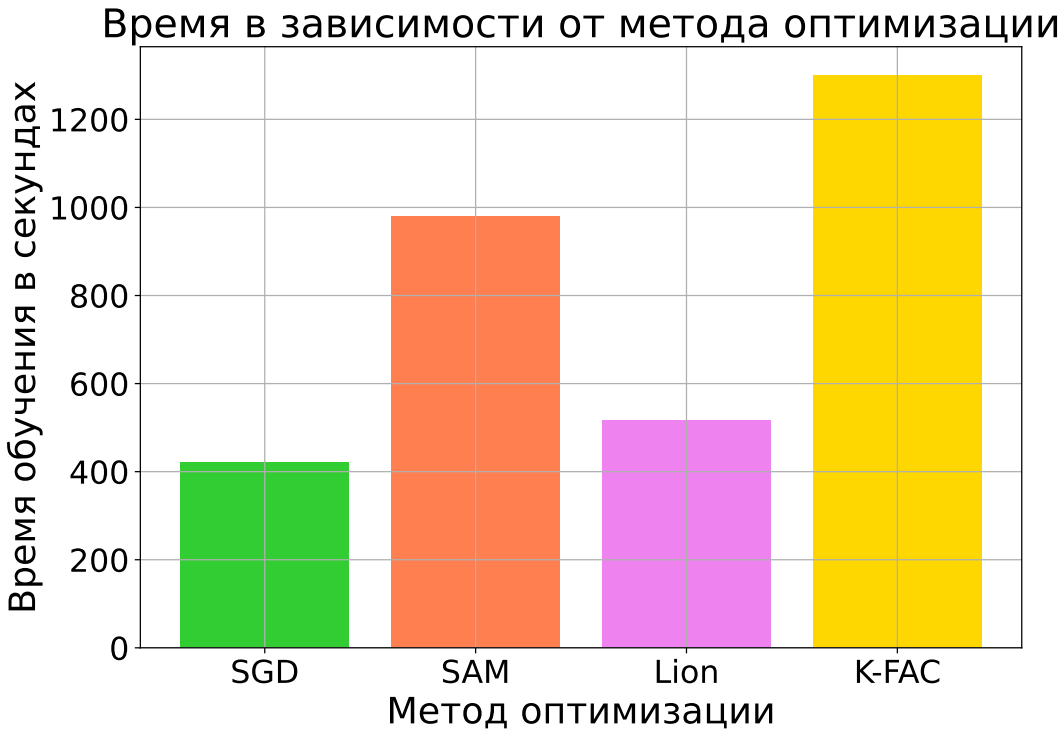


Рис. 24: График времени, затраченного на обучение, в зависимости от метода.

На рисуноке 24 показано время, затраченное на обучение нейронной сети в течении 20 эпох каждым из методов. Из него можно видеть, что при использовании метода Lion было затрачено почти столько же времени, сколько при использовании метода SGD. Больше всего времени было затрачено на K-FAC, а с учётом его плохих результатов он оказывается неэффективным для этой задачи. Метод SAM, хоть и показал хорошие результаты, затратил в два раза больше времени чем SGD.

Для более точного сравнения нейронная сеть была обучена до точности 0.91 с помощью разных методов, и были произведены замеры времени, понадобившегося для этого обучения. На рисунках 25 и 26 показаны график зависимости точности обучения от количества эпох и график зависимости времени, потраченного на обучение, от метода, выбранного для обучения. Метод Lion на рисунке 25 показан точкой, так как достиг необходимой точности всего за одну эпоху и потратил минимальное количество времени. Метод K-FAC показал себя неэффективным. Методу SAM понадобилось меньше эпох, но больше времени чем SGD.

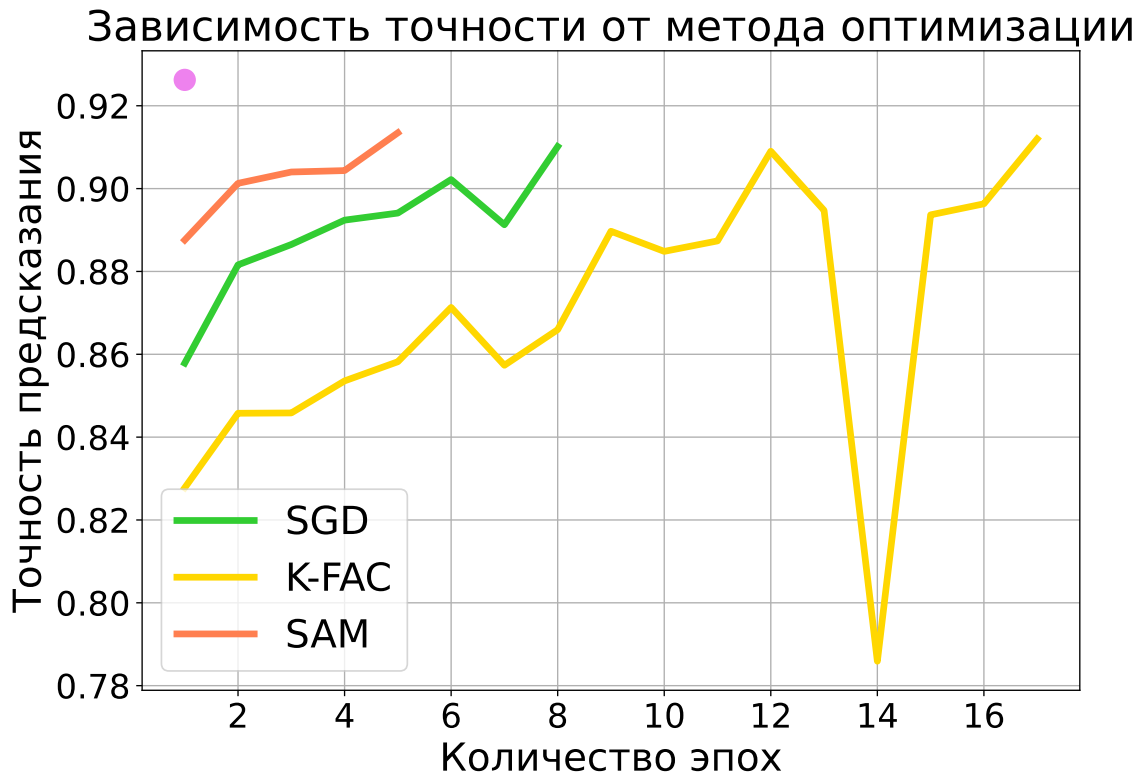


Рис. 25: График зависимости точности от количества эпох для разных методов.

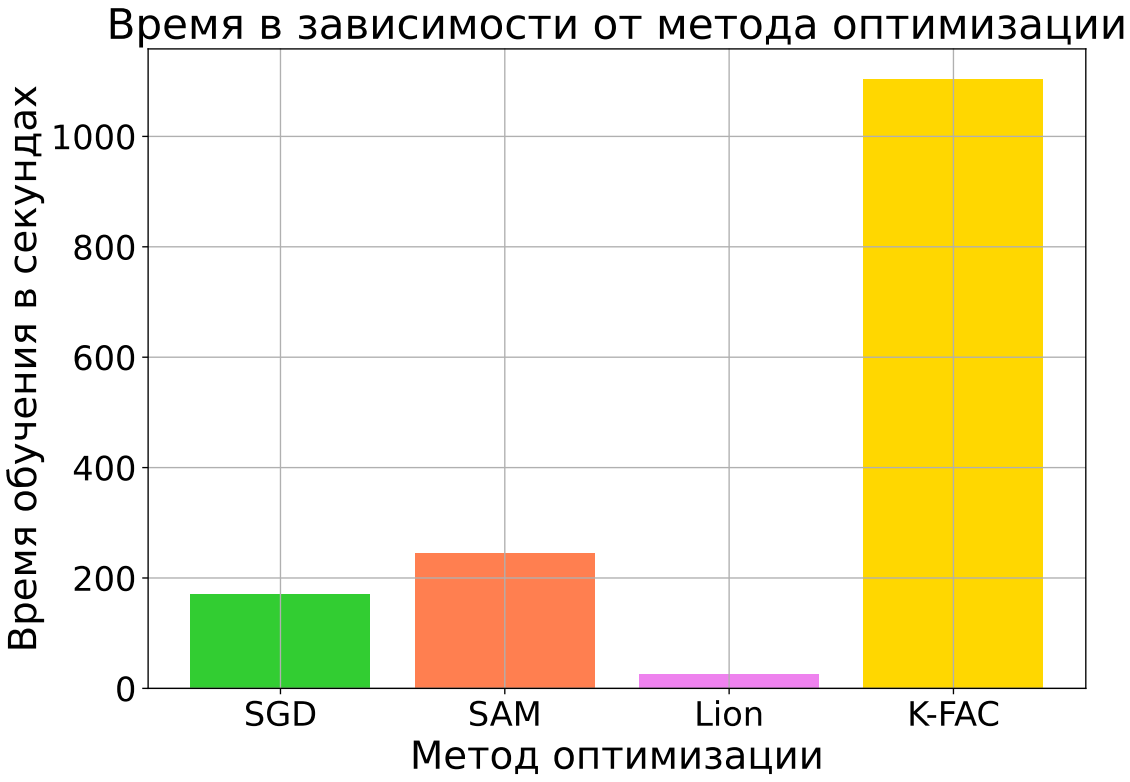


Рис. 26: График времени, затраченного на обучение, в зависимости от метода.

Заключение

В данной работе было проведено сравнение методов SGD, K-FAC, Shampoo, Lion и SAM при обучении нейронных сетей с разными архитектурами. На основе проведённых экспериментов было установлено, что K-FAC и Shampoo достаточно неэффективны, они затрачивают слишком много времени на обучение, не давая лучшие результаты относительно других методов. Результаты для метода Lion оказались достаточно интересными. При его использовании затрачивается столько же времени, сколько при использовании метода SGD, при этом с его помощью достигается высокая точность уже после небольшого количества эпох, однако ещё через несколько эпох он себя исчерпывает и модель переобучается. Несмотря на это метод Lion может быть вполне эффективно использоваться в реальных задачах и давать хорошие результаты. Во всех типах задач при помощи метода SAM были получены одни из лучших результатов, а также не наблюдалось переобучение. Хотя времени при использовании метода SAM было затрачено больше чем при использовании SGD и Lion, это время не настолько велико и позволяет использовать данный метод в реальных задачах и ожидать хороших результатов.

Список литературы

[1] Chen X., Liang C., Huang D., Real E., Wang K., Liu Y., Pham H., Dong X., Luong T., Hsieh C., Lu Y., Le Q. Symbolic Discovery of Optimization Algorithms // arXiv.org. 2023. URL: https://arxiv.org/abs/2302.06675.

[2] Foret P., Kleiner A., Mobahi H., Neyshabur B. Sharpness-Aware Minimization for Efficiently Improving Generalization // 9th International Conference on Learning Representations. 2021.

[3] Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge: MIT Press, 2016.

[4] Gupta V., Koren T., Singer Y. Shampoo: Preconditioned Stochastic Tensor Optimization // Proceedings of the 35th International Conference on Machine Learning. Stockholm, 2018. V. 80. P. 1837–1845.

[5] He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, 2016. P. 770–778.

[6] Kingma D. P., Ba J. Adam: A method for stochastic optimization // Proceedings of the 3rd International Conference for Learning Representations. San Diego, 2015.

[7] Krizhevsky A. Learning multiple layers of features from tiny images // Technical Report TR-2009, University of Toronto. Toronto, 2009.

[8] LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE. Monterey, 1998. V. 86, №11. P. 2278–2324.

[9] Loshchilov I., Hutter F. SGDR: Stochastic Gradient Descent with Warm Restarts // Proceedings of the 5th International Conference on Learning Representations. Toulon, 2017.

[10] Martens J., Grosse R. B. Optimizing neural networks with kronecker-factored approximate curvature // Proceedings of the 32nd International Conference on Machine Learning. Lille, 2015. V. 37. P. 2408–2417.

[11] Real E., Liang C., So D. R., Le Q. V. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch // Proceedings of the 37th International Conference on Machine Learning. 2020. V. 119. P. 8007–8019.

[12] Sak H., Senior A. W., Beaufays F. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition // arXiv.org. 2014. URL: https://arxiv.org/abs/1402.1128.

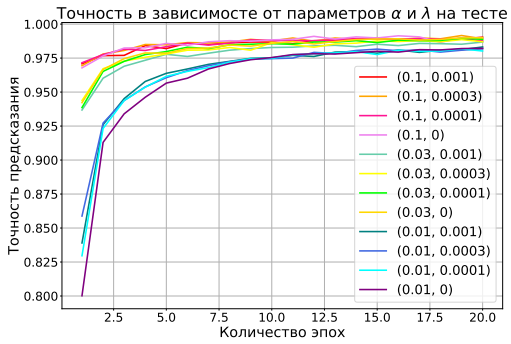
[13] Shazeer N., Stern M. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost // Proceedings of the 35th International Conference on Machine Learning. Sweden, 2018. V. 80. P. 4603–4611.

[14] Wang Z., Shang J., Liu L., Lu L. Liu J., Han J. CrossWeigh: Training Named Entity Tagger from Imperfect Annotations // Proceedings of the 9th International Joint Conference on Natural Language Processing. Hong Kong, 2019. P. 5153–5162.

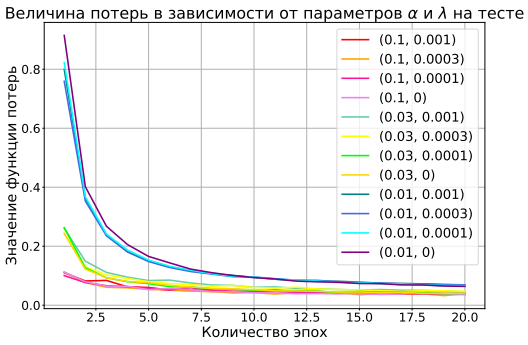
Приложения

Приложение 1

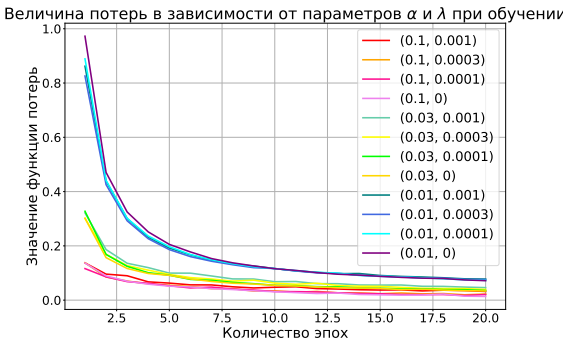
Ниже представлен подбор параметров для эксперимента классификации цифр. На графиках изображены зависимости точности от количества эпох на контрольной выборке, значения функции потерь от количества эпох на контрольной и обучающей выборках для каждого из методов. При помощи данных на графиках подбирается шаг и коэффициент регуляризации, для метода SAM также подбирается параметр ρ.



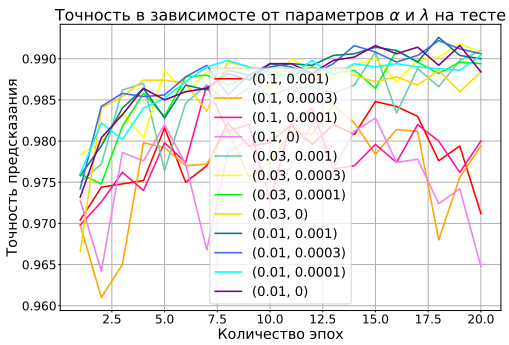
SGD. Точность на тесте.



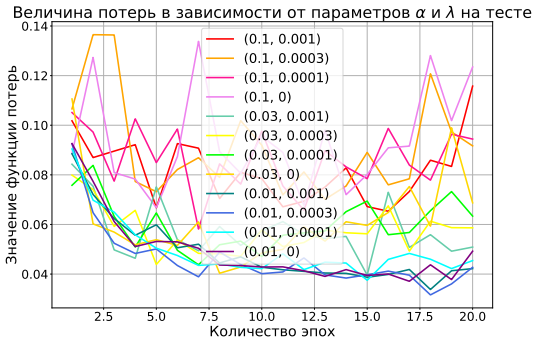
SGD. Потери на тесте.



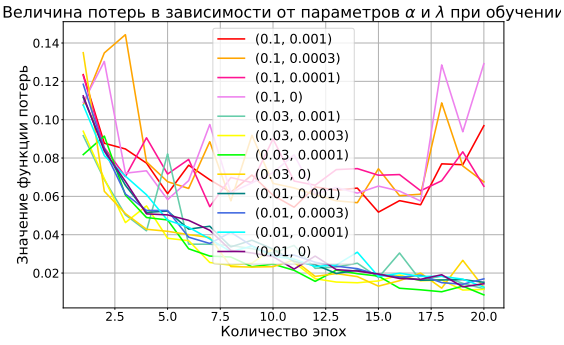
SGD. Потери на обучающей выборке.



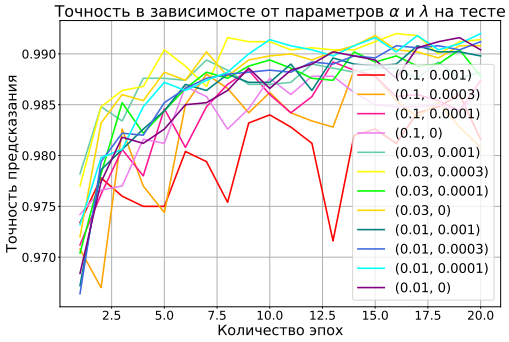
K-FAC. Точность на тесте.



K-FAC. Потери на тесте.

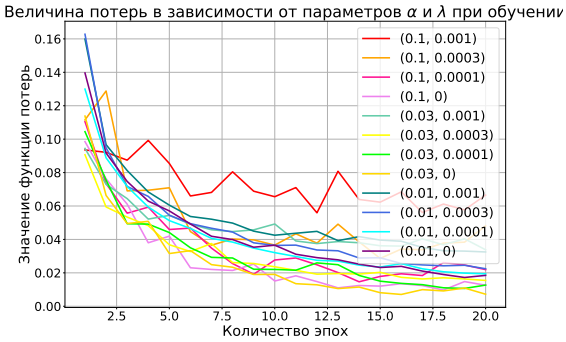


K-FAC. Потери на обучающей выборке.

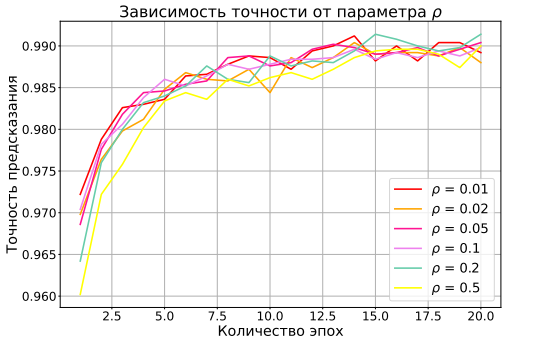


SAM. Точность на тесте.

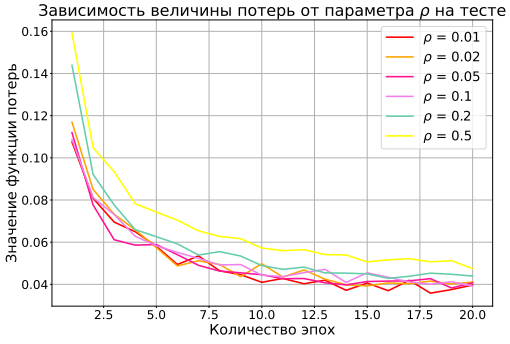
SAM. Потери на тесте.



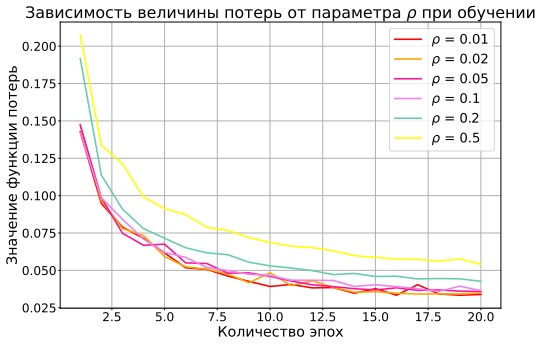
SAM. Потери на обучающей выборке.



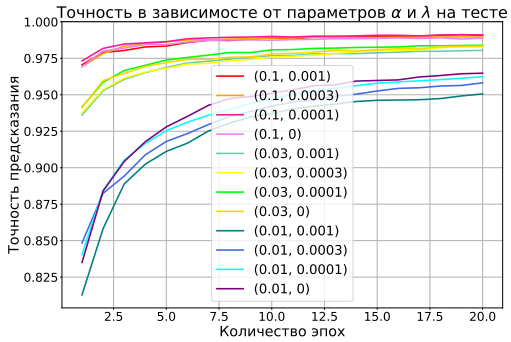
SAM. Подбор ρ. Точность на тесте.



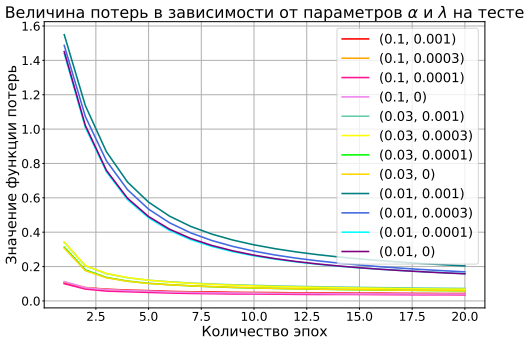
SAM. Подбор ρ. Потери на тесте.



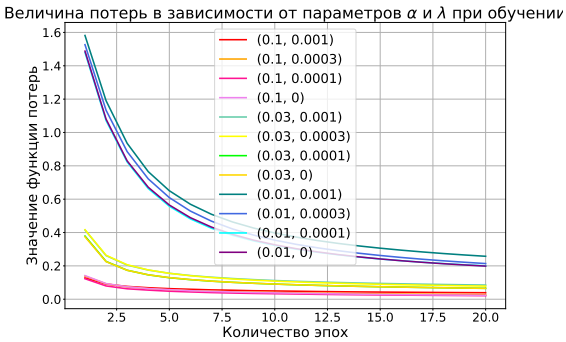
SAM. Подбор ρ. Потери на обучающей выборке.



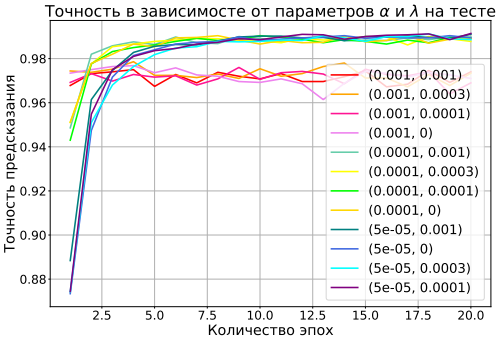
Sampoo. Точность на тесте.



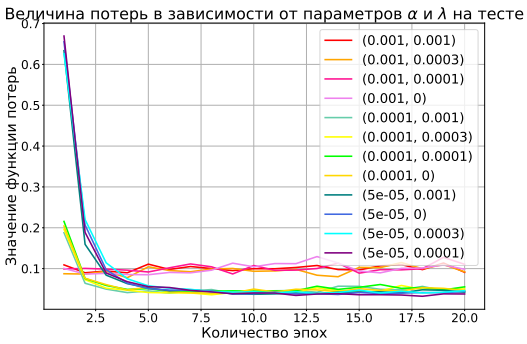
Sampoo. Потери на тесте.



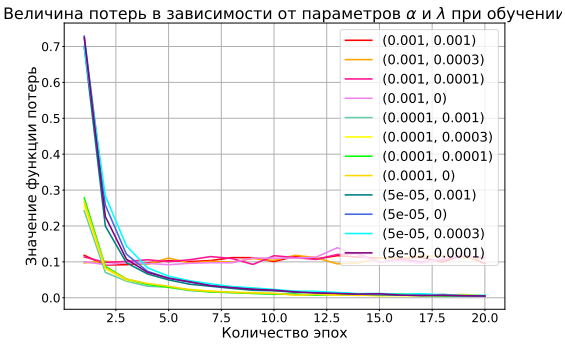
Sampoo. Потери на обучающей выборке.



Lion. Точность на тесте.



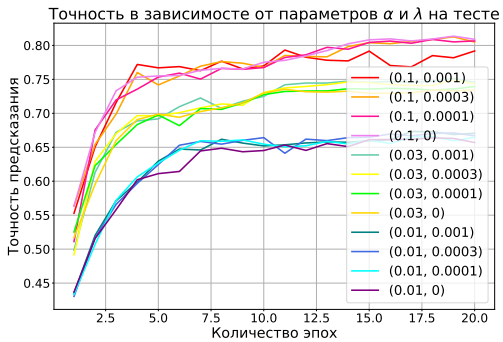
Lion. Потери на тесте.



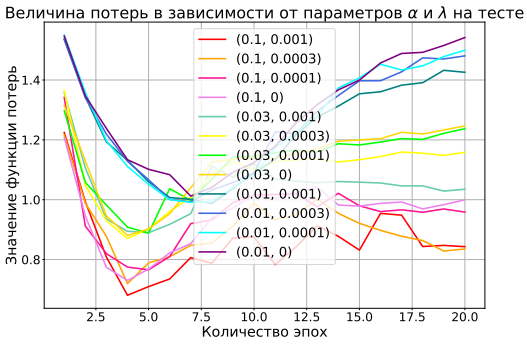
Lion. Потери на обучающей выборке.

Приложение 2

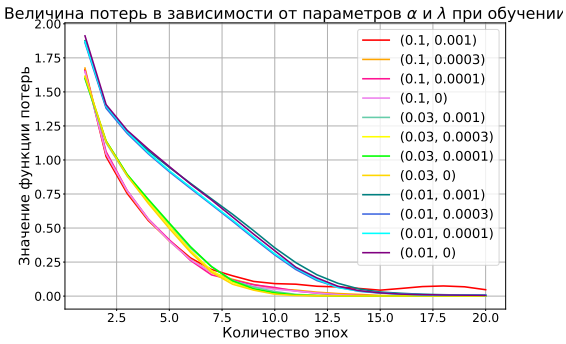
Ниже представлен подбор параметров для эксперимента классификации изображений. На графиках изображены зависимости точности от количества эпох на контрольной выборке, значения функции потерь от количества эпох на контрольной и обучающей выборках для каждого из методов. При помощи данных на графиках подбирается шаг и коэффициент регуляризации, для метода SAM также подбирается параметр ρ.



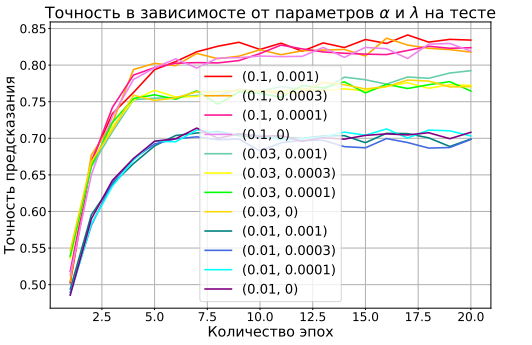
SGD. Точность на тесте.



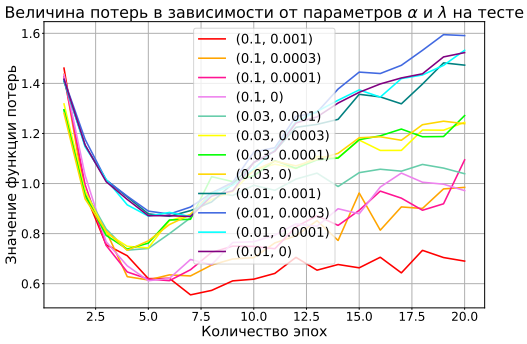
SGD. Потери на тесте.



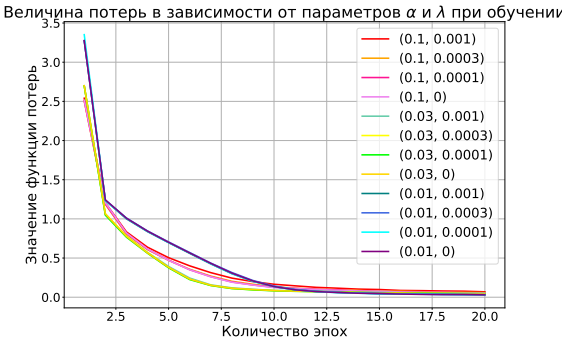
SGD. Потери на обучающей выборке.



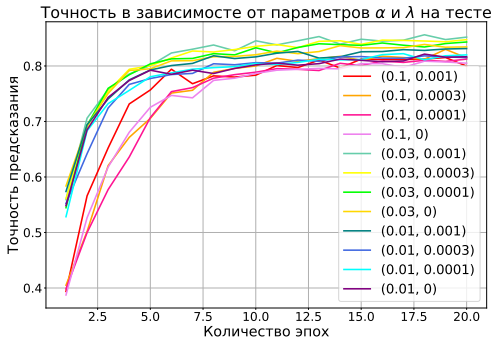
K-FAC. Точность на тесте.



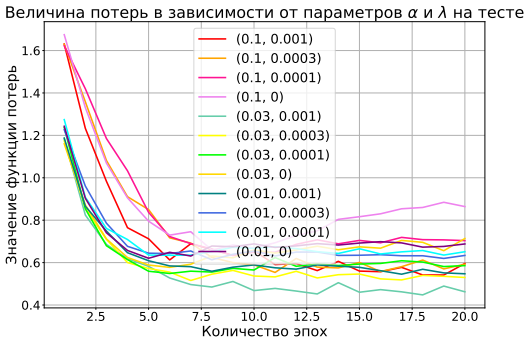
K-FAC. Потери на тесте.



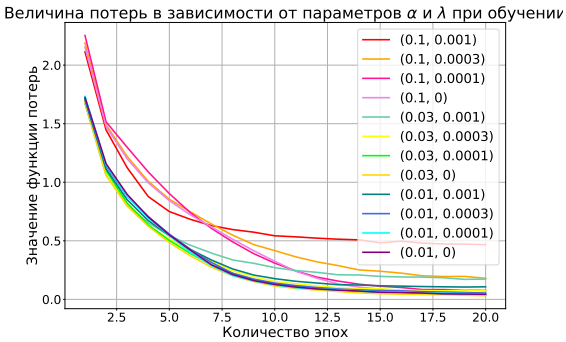
K-FAC. Потери на обучающей выборке.



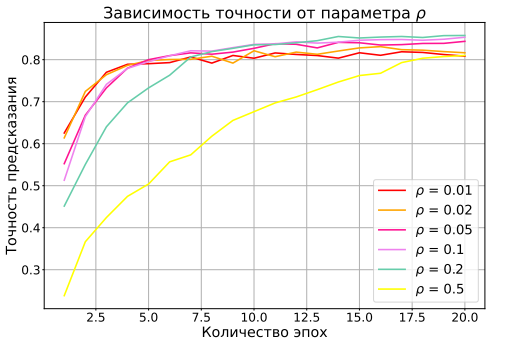
SAM. Точность на тесте.



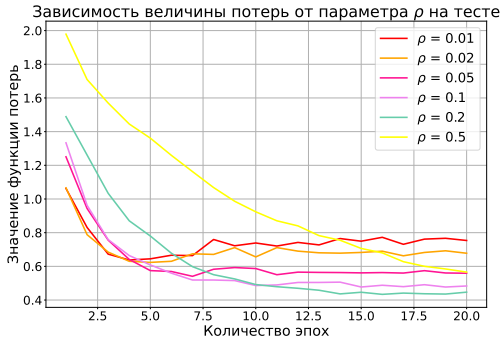
SAM. Потери на тесте.



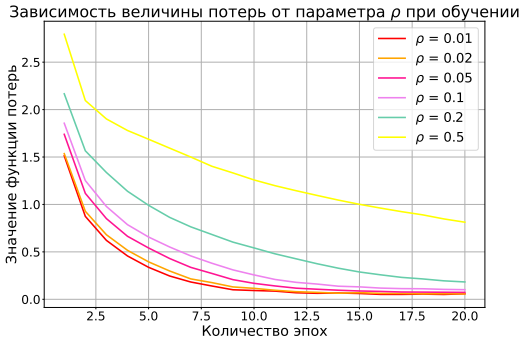
SAM. Потери на обучающей выборке.



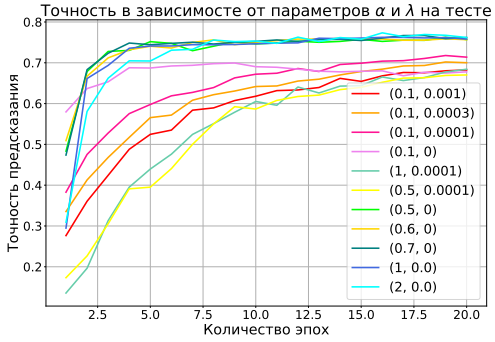
SAM. Подбор ρ. Точность на тесте.



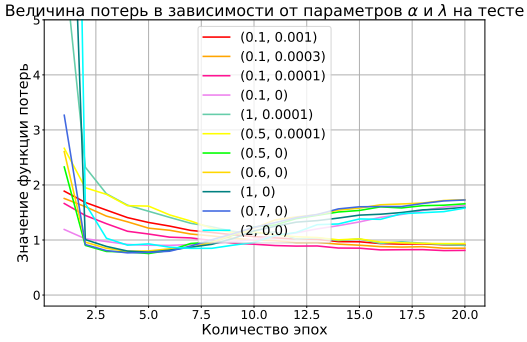
SAM. Подбор ρ. Потери на тесте.



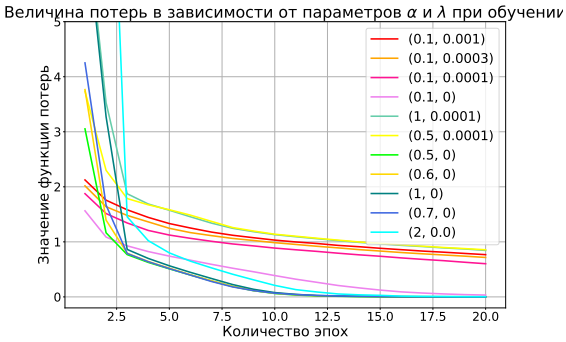
SAM. Подбор ρ. Потери на обучающей выборке.



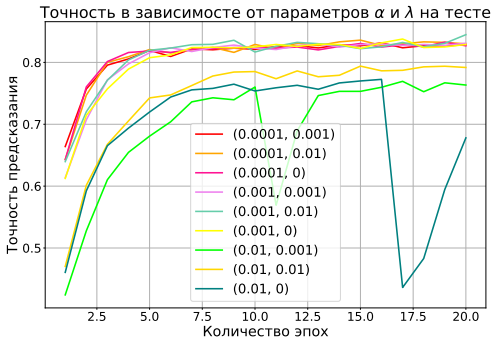
Sampoo. Точность на тесте.



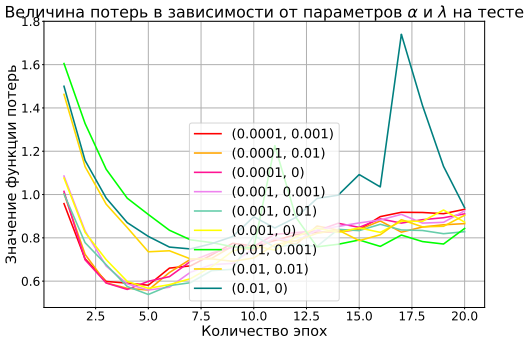
Sampoo. Потери на тесте.



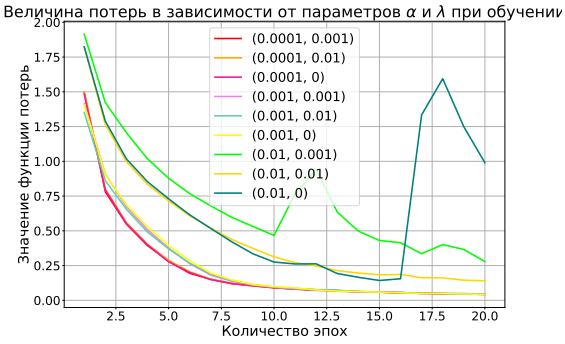
Sampoo. Потери на обучающей выборке.



Lion. Точность на тесте.



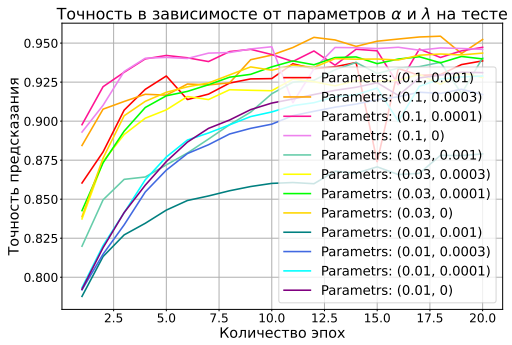
Lion. Потери на тесте.



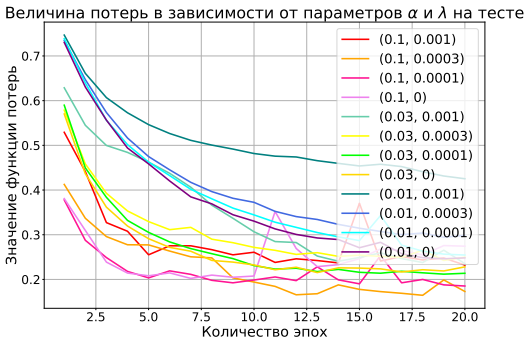
Lion. Потери на обучающей выборке.

Приложение 3

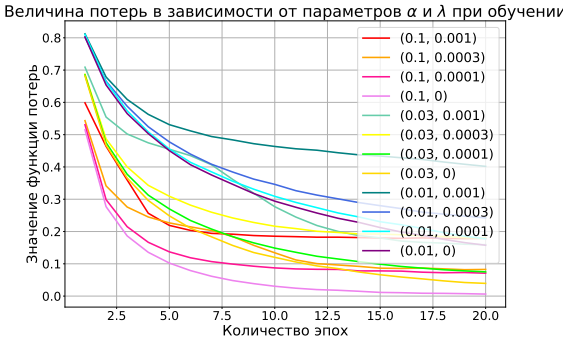
Ниже представлен подбор параметров для эксперимента классификации именованных сущностей. На графиках изображены зависимости точности от количества эпох на контрольной выборке, значения функции потерь от количества эпох на контрольной и обучающей выборках для каждого из методов. При помощи данных на графиках подбирается шаг и коэффициент регуляризации, для метода SAM также подбирается параметр ρ.



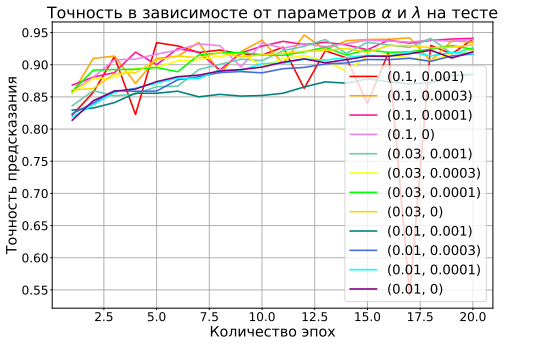
SGD. Точность на тесте.



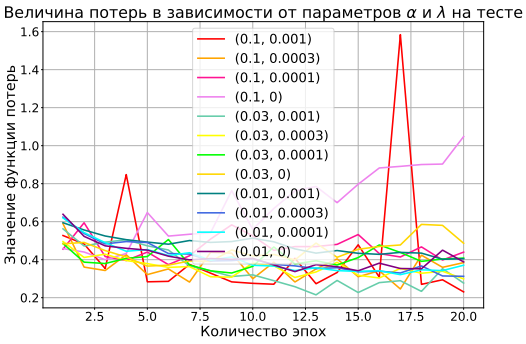
SGD. Потери на тесте.



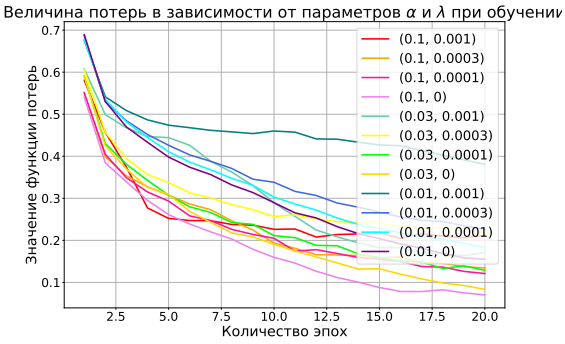
SGD. Потери на обучающей выборке.



K-FAC. Точность на тесте.

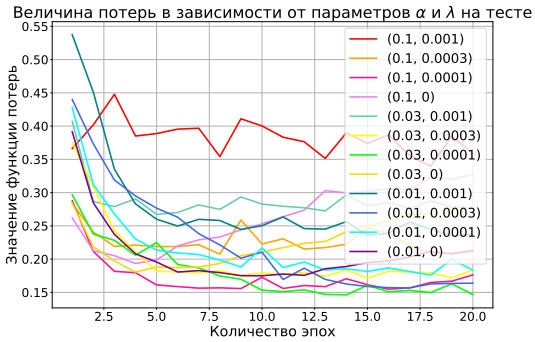


K-FAC. Потери на тесте.

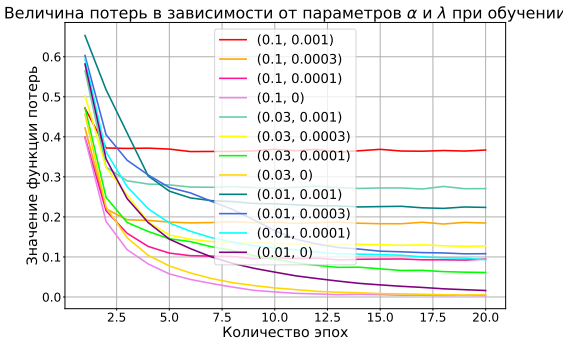


K-FAC. Потери на обучающей выборке.

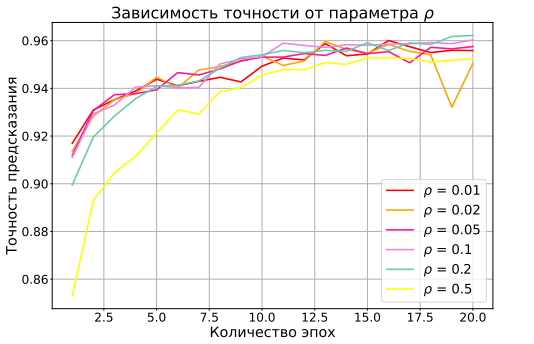
SAM. Точность на тесте.



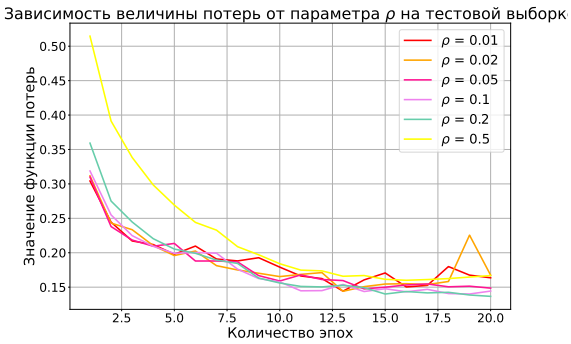
SAM. Потери на тесте.



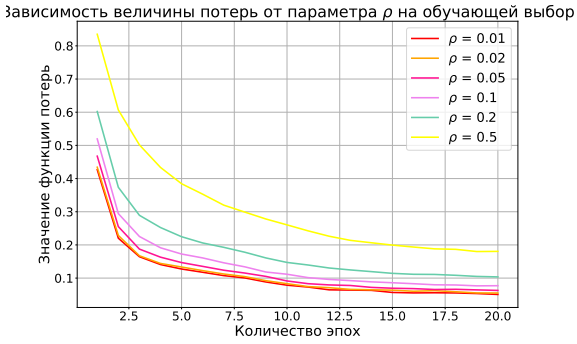
SAM. Потери на обучающей выборке.



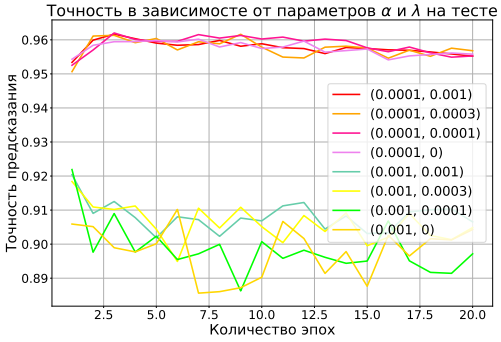
SAM. Подбор ρ. Точность на тесте.



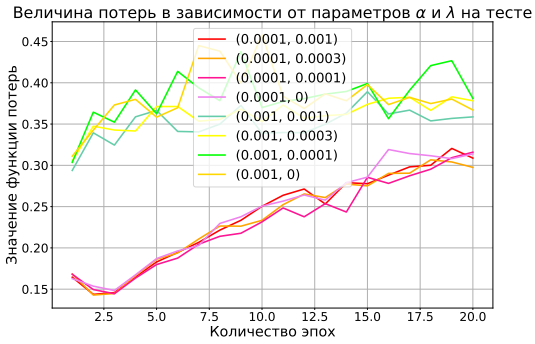
SAM. Подбор ρ. Потери на тесте.



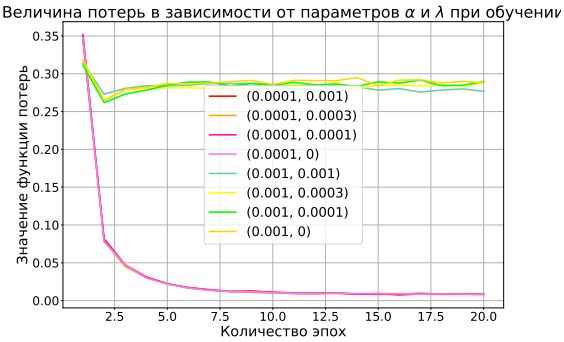
SAM. Подбор ρ. Потери на обучающей выборке.



Lion. Точность на тесте.



Lion. Потери на тесте.



Lion. Потери на обучающей выборке.