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

Курсовая работа на тему

Исследование методов второго порядка для обучения нейронных сетей

Выполнила студентка 311 группы
Кузнецова Мария Павловна

Москва, 2022

Содержание

Введение

В центре исследования лежит метод приближенной кривизны с учетом факторизации Кронекера (Kronecker-factored Approximate Curvature, K-FAC). Это метод, аппроксимирующий метод натурального градиента (natural gradient) в нейронных сетях. K-FAC основан на эффективной обратимой аппроксимации информационной матрицы Фишера [6] нейронной сети. Он работает путем аппроксимации различных больших блоков в блочной аппроксимации матрицы Фишера (соответствующих отдельным слоям нейронной сети) как произведение Кронекера двух гораздо меньших матриц. Вычислений в данном методе всего в несколько раз больше, чем в стохастическом градиентном спуске [1], и на практике этот метод может сходиться намного быстрее, чем стохастический градиентный спуск. Подробнее данный алгоритм описан в статье [6].

В данной работе рассматривается применение метода K-FAC для оптимизации функции потерь нейронной сети при решении задачи классификации и сравнение полученных результатов с результатами решения той же задачи с использованием более простых методов оптимизации, таких как стохастический градиентный спуск и метод Adam [3].

С использованием данных методов были проведены эксперименты на нейронных сетях разной сложности и на различных наборах данных. Это позволило исследовать скорость сходимости, время затраченное на оптимизацию и таким образом выяснить в каких случаях применение K-FAC 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 (нервными импульсами). 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

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

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}.

Разобьём эту формулу:

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

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

где s_i — вектор взвешвнных 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 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.

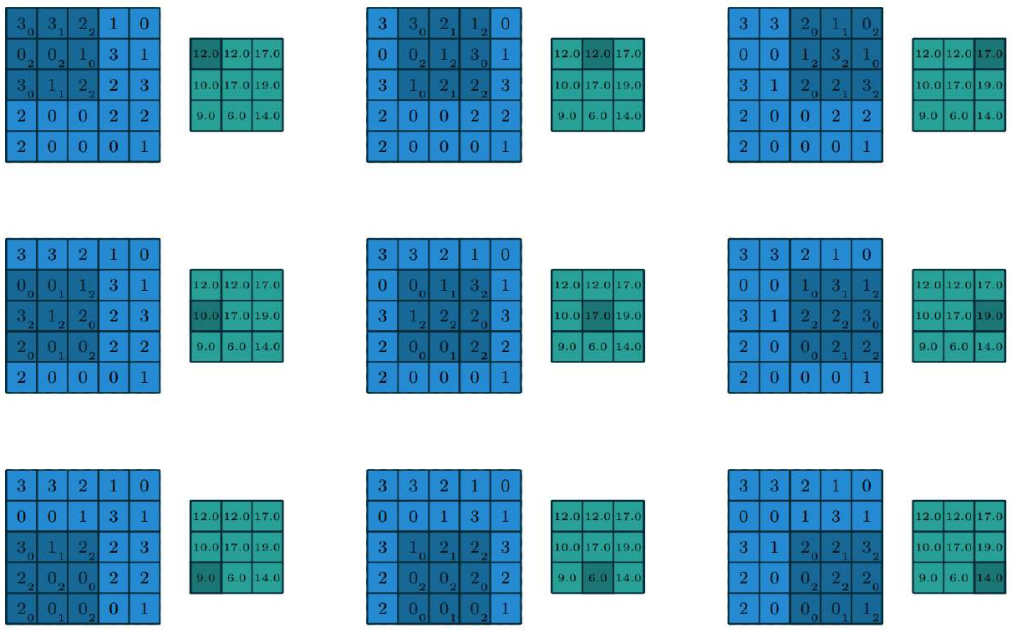

Рис. 1: Пример свертки двух матриц.

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 назвают свёрточным. Такие нейронные сети используются в распознавании и обработке изображений.

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

Свёрточный слой принимает на вход набор из c_1 карт признаков (матриц) a_{i-1}^c размера $n_{i-1} \times m_{i-1}$ и выдаёт c_2 матриц размера $n_i \times m_i$ в соответствии с формулой:

$$a_i^c = \phi(\sum_{j=1}^{c_1} a_{i-1}^j * k^{j,c} + b^c), \quad j = 1, \ldots, c_2, \tag{5}$$

где $k^{j,c}$, b^c — параметры слоя, называемые фильтрами и сдвигами, $*$ — операция свёртки входа a с ядром k.

1.3 Pooling слои

Подвыборочный (Pooling) слой так же, как и свёрточный обрабатывает карты признаков, но количество карт признаков на входе и на выходе для этого слоя совпадает. Операция подвыборки (или MaxPooling — выбор максимального) делается в соответствии с рисунком 2.

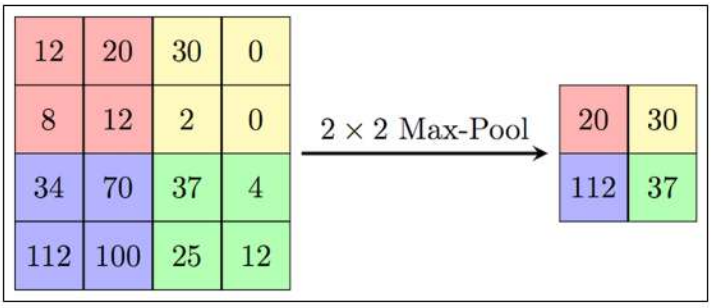

Рис. 2: Операция подвыборки (Max Pooling).

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, a_{i-1} — выход $i-1$ слоя, который подаётся на вход $i-$му слою, a_i — выход из $i-$го слоя.

1.4 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 [1], в свёрточных слоях применяется ReLU.

Softmax — это функция для многомерного случая. Её 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:

$$\phi(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}, \tag{7}$$

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, но приводят к проблемам с затуханием или увеличением градиентов. Однако, можно использовать и гораздо более простой вариант — функцию активации "линейный выпрямитель"(rectified linear unit, ReLU).

2 Постановка задачи

Пусть $X \subset \mathbb{R}^m$ — конечное множество векторов (объектов), $Y = \{1, \ldots, c\}$ — конечное множество допустимых ответов, и существует целевая функция $f^* : X \to Y$, значения которой $y_i = f^*(x_i)$ известны только на конечном множестве объектов $\{x_1, \ldots, x_n\} \subset X$. Пары объект-ответ (x_i, y_i) называются прецедентами. Совокупность таких пар $D^k = \{(x_i, y_i) \in X \times Y | i = \overline{1,k}\}$ и $D^h = \{(x_i, y_i) \in X \times Y | i = \overline{k+1,n}\}$ называются обучающей выборкой и контрольной выборкой соответственно, где $h = n - k$. 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 коэффициэнтов) нейронной сети, объединенных вместе, где vec — это оператор, который векторизирует матрицы, конкатенируя их столбцы, l — количество слоёв нейронной сети.

Пусть $P^c = \{p \in \mathbb{R}^c : p_0 + \ldots + p_{c-1} = 1, p_i \geq 0, i = \overline{0, c-1}\}$ тогда $f(\theta, \cdot) : \mathbb{R}^m \to P^c$ — функция нейронной сети с m-мерным входом и c-мерным выходом вероятностей принадлежности объекта к каждому из классов.

Задача обучения по прецедентам заключается в том, по обучающей выборке D^k подобрать такие параметры θ, чтобы нейронная сеть f приближала зависимость 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 заключатся в подборе таких параметров θ, при которых функция потерь минимальна:

$$L(\theta, D^k) \to \min_{\theta}. \tag{8}$$

Рассмотрим крросс-энтропийную функцию потерь:

$$H(P, Q) = -\sum_{x} P(x) \log(Q(x)), \tag{9}$$

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

Преобразум:

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

$$= -\sum_{i=0}^{k} \log(p(y_i|x_i, \theta)) = -\log(p(Y^k|X^k, \theta)), \tag{10}$$

где $p(y_i|x_i, \theta)$ — это вероятность класса y_i в соответствии с предсказанием модели с весами θ на входных данных x_i.

Таким образом получается следующая задача:

$$L(\theta, D^k) = -\log p(Y^k|X^k, \theta) \to \min_{\theta}. \tag{11}$$

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

$$Dv = \frac{\partial L(\theta, x, y)}{\partial v} = -\frac{\partial \log p(y|x, \theta)}{\partial v}. \tag{12}$$

Тогда производная нейронной сети по взвешенной сумме i-го слоя $g_i = Ds_i$.

2.2 Стохастическая оптимизация

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

Зачастую обучающая выборка делится на несколько частей (батчей). Затем случайным образом выбирается батч, по нему в соответствии с выбранным методом оптимизации делается шаг. Часть выборки, взятую на шаге t обозначим через D_t^k. После снова случайным образом выбирается батч из тех, которые ещё не использовались, и всё повторяется. Процесс перебора всех батчей обучающей выборки и обучения по ним называют эпохой обучения нейронной сети. Количество эпох при обучении может доходить до нескольких сотен.

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 верно предсказанных ответов нейронной сети к количеству объектов в выборке) на контрольной выборке.

2.3 Регуляризация

Также будет использоваться регуляризация весов. Регуляризация — это способ борьбы с 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 $L_2 = \lambda||\theta||_2$, называющихся $L1$-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:

$$-\log p(Y^k|X^k, \theta) + \lambda||\theta||_2, \tag{13}$$

где λ — подбираемый коэффициент регуляризации.

3 Метод стохастического градиентного спуска и его модификации.

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.

3.1 Стохастический градиентный спуск

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 парметров θ_{t+1} выглядит следующим образом:

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t, D_t^k), \tag{14}$$

где функция потерь считается на некотором батче из обучающей выборки.

3.2 Метод адаптивной инерции

Метод Adam использует оценку момента первого порядка и оценку момента второго порядка градиента, чтобы динамически регулировать скорость обучения параметра. Правило обновления параметров выглядит следующим образом:

$$m_{t+1} = \gamma_1 m_t + (1 + \gamma_1) \nabla L(\theta_t, D_t^k), \tag{15}$$

$$g_{t+1} = \gamma_2 g_t + (1 + \gamma_2) \nabla L(\theta_t, D_t^k)^2, \tag{16}$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \gamma_1^{t+1}}, \tag{17}$$

$$\hat{g}_{t+1} = \frac{g_{t+1}}{1 - \gamma_2^{t+1}}, \tag{18}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_{t+1}}{\sqrt{\hat{g}_{t+1} + \varepsilon}}, \tag{19}$$

где m_{t+1} и g_{t+1} — это оценки моментов первого и второго порядков градиента, которые можно рассматривать как приближение к математическим

ожиданиям $E[\nabla L(\theta_t, D_t^k)]$ и $E[\nabla L(\theta_t, D_t^k)^2]$, соответственно $\hat{m}_{t+1}, \hat{g}_{t+1}$ — это поправка m_t и g_t, которую можно рассматривать как несмещенную оценку математического ожидания, γ_1, γ_2 — это экспоненциальные скорости затухания для оценок на данный момент, $0 < \gamma_1 < 1, 0 < \gamma_2 < 1$, ε — это достаточно малая константа, введенная для численной стабильности.

4 Метод приближённой кривизны с учетом факторизации Кронекера

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

$$F = E\left[\frac{\partial \log p(Y_t^k|X_t^k,\theta)}{\partial\theta}\frac{\partial \log p(Y_t^k|X_t^k,\theta)}{\partial\theta}^T\right], \tag{20}$$

где через E обозначается математическое ожидание.

Пусть

$$\frac{\partial \log p(Y_t^k|X_t^k,\theta)}{\partial\theta} = D\theta, \tag{21}$$

тогда

$$F = E[D\theta D\theta^T]. \tag{22}$$

4.1 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:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \dots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \dots & a_{m,n}B \end{bmatrix} \tag{23}$$

.

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.

4.2 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
$\hat{F} = diag(\tilde{F}_{1,1}, \tilde{F}_{2,2}, \ldots, \tilde{F}_{l,l}) = diag(\bar{A}_{0,0} \otimes G_{1,1}, \bar{A}_{1,1} \otimes G_{2,2}, \ldots, \bar{A}_{l-1,l-1} \otimes G_{l,l})$.

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

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

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 [6].

5 Экспериментальные результаты

В данном разделе приводятся результаты работы методов на различных наборах данных. Рассматриваются две задачи с различными по сложности архитектурами нейронных сетей и различными данными. В обоих 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. Шаг обучения подбирается среди значений: 0.1, 0.03, 0.01. Коэффициент регуляризации подбирается среди значений: 0, 0.001, 0.0003, 0.0001.

Рис. 3: Изображения цифр для первого эксперемента.

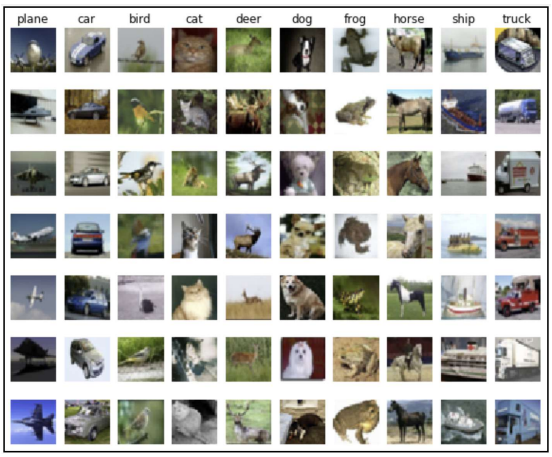

Рис. 4: Цветные изображения для второго эксперемента.

5.1 Задача классификации цифр

В данном эксперименте в качестве данных выступали чёрно-белые изображения цифр от 0 до 9 из набора данных MNIST [5]. Обучающая выборка представлена 60000 экземплярами, а контрольная выборка 10000 экземплярами.

Обучалась свёрточная нейронная сеть, известная как LeNet5 [5]. Структура данной нейронной сети:

1. слой C1 представляет собой свёрточный слой из шести матриц, размер которых составляет 28×28;

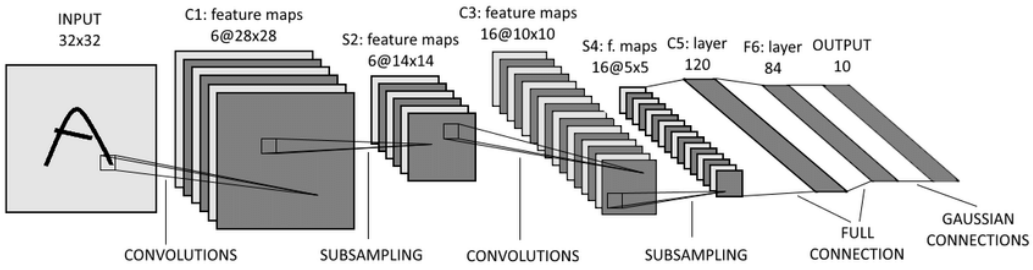

Рис. 5: LeNet5.

2. слой S2 — это слой подвыборки из шести матриц, размер которых составляет 14×14;

3. слой C3 представляет собой свёрточный слой из шестнадцати матриц, размер которых составляет 10×10;

4. слой S4 является слоем подвыборки из шестнадцати матриц, размер которых составляет 5×5;

5. слой C5 представляет собой свёрточный слой из 120 матриц, размер которых составляет 1×1;

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

7. слой OUTPUT возвращает данные о вероятности принадлежности классу.

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

Подбор параметров для метода SGD

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисуноки 6, 7), график зависимости значения функции потерь на тестовом наборе(рисунок 8) и график зависимости значения функции потерь на обучающем наборе(рисунок 9). Исходя из данных графиков делались выводе о лучших параметрах для метода SGD в данной задаче.

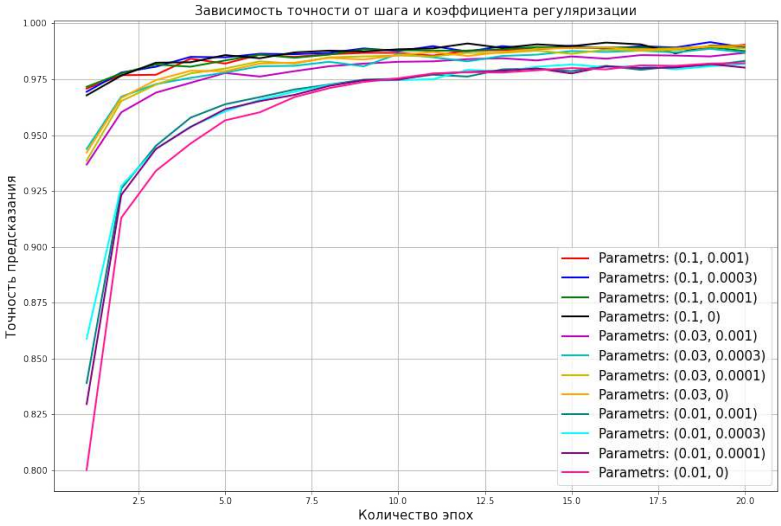

Рис. 6: SGD. График зависимости точности от количества эпох при разных параметрах.

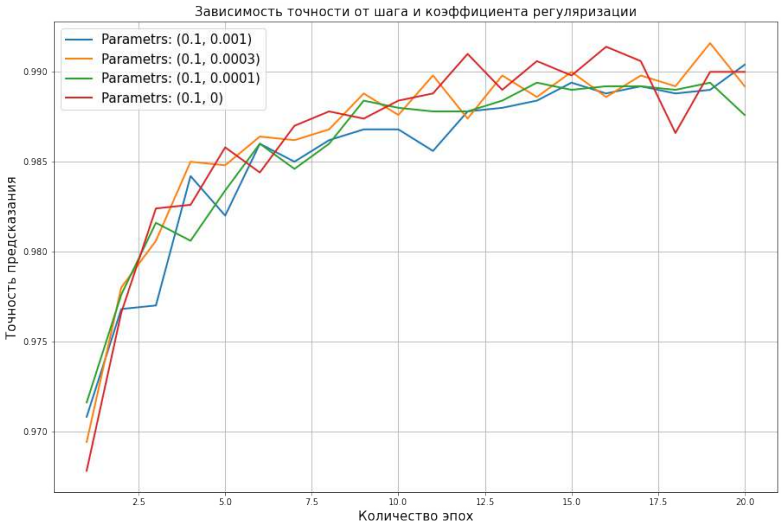

Рис. 7: SGD. График зависимости точности от количества эпох. Лучшие параметры.

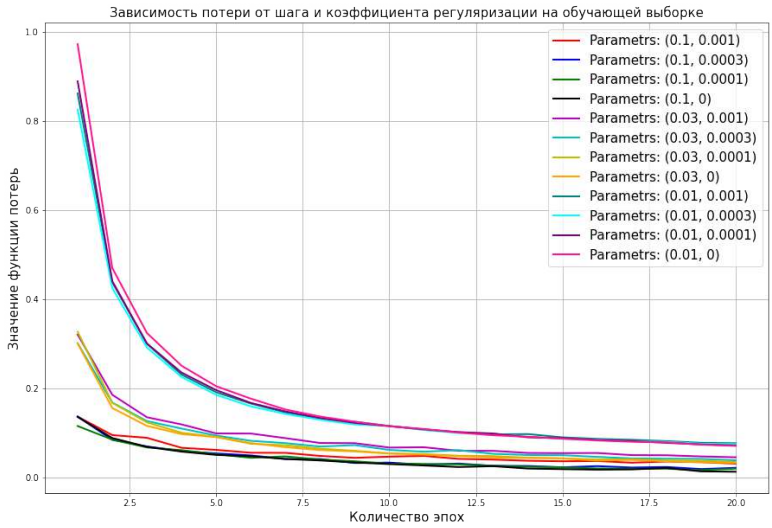

Рис. 8: SGD. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

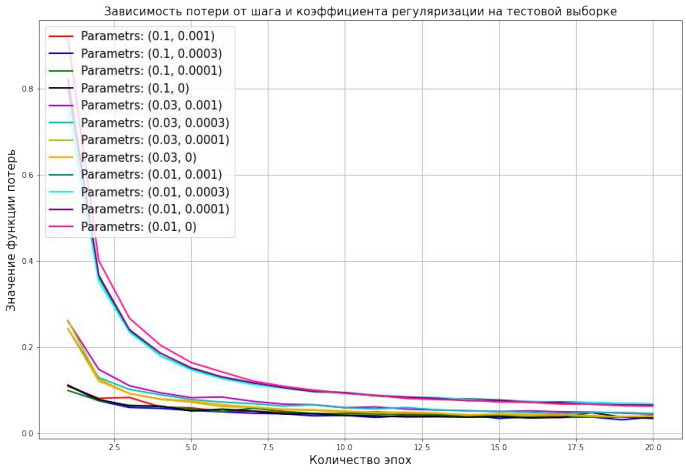

Рис. 9: SGD. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

На графиках можно увидеть, что линии группируются по параметру шага, кривые с шагом 0.1 выше кривых с отличными значениями шага. В

то время как линии с одинаковыми значениями коэффициента регуляризации, но одинаковым шагом, не сильно отличаюся друг от друга. Исходя из этого и того, что всё же чуть лучше остальных кривая с нулевым коэффициентом регуляризации, можно сделать вывод, что регуляризация не требуется, переобучения не появляется в данном случае.

Таким образом, для дальнейшего сравнения метода SGD с другими были выбраны шаг обучения 0.1 и коэффициент регуляризации 0.

Подбор параметров для метода Adam

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 10), график зависимости значения функции потерь на тестовом наборе(рисунок 11) и график зависимости значения функции потерь на обучающем наборе(рисунок 12). Исходя из данных графиков делались выводе о лучших параметрах для метода Adam в данной задаче.

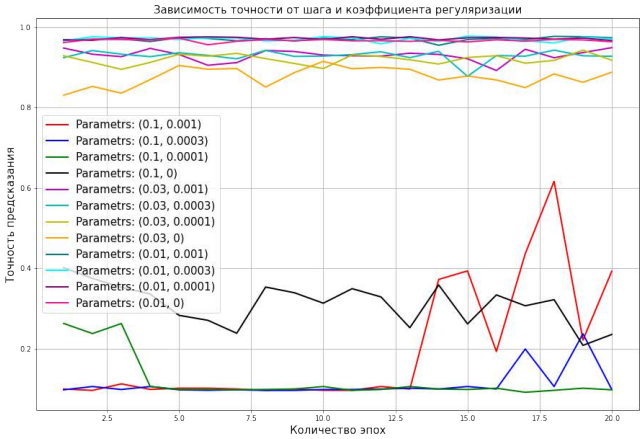

Рис. 10: Adam. График зависимости точности от количества эпох для разных параметров.

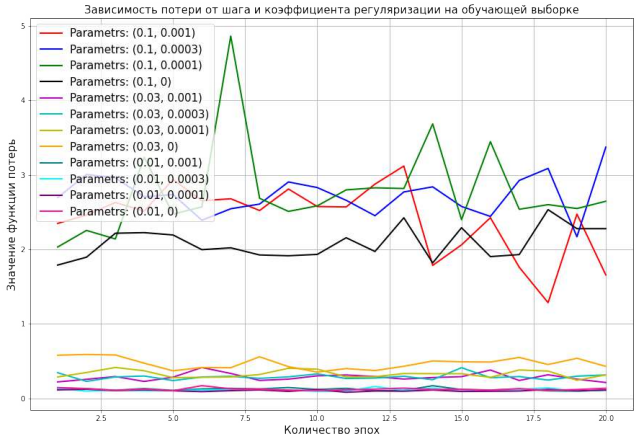

Рис. 11: Adam. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

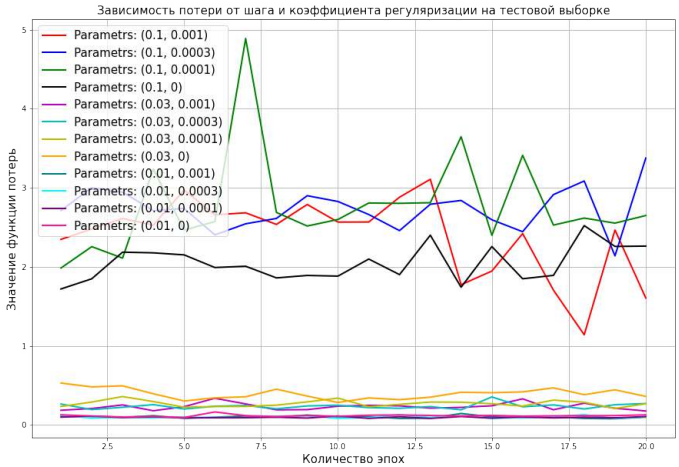

Рис. 12: Adam. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

Модели, обученные с шагом 0.1, показывают очень низкую точность и высокое значение функции потерь, можно сделать вывод, что с таким параметром метод Adam почти не обучается. С параметром 0.0003 модель обучается лучше всего, причём с первых же эпох достигает сво-

его максимума и больше не улучшает свой результат. От коэффициента регуляризации результаты обучения почти не зависят в данном случае.

Исходя из графиков, для дальнейшего сравнения метода Adam с другими были выбраны шаг обучения 0.01 и коэффициент регуляризации 0.0003.

Подбор параметров для метода K-FAC

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 13), график зависимости значения функции потерь на тестовом наборе(рисунок 14) и график зависимости значения функции потерь на обучающем наборе(рисунок 15). Исходя из данных графиков делались выводе о лучших параметрах для метода K-FAC в данной задаче.

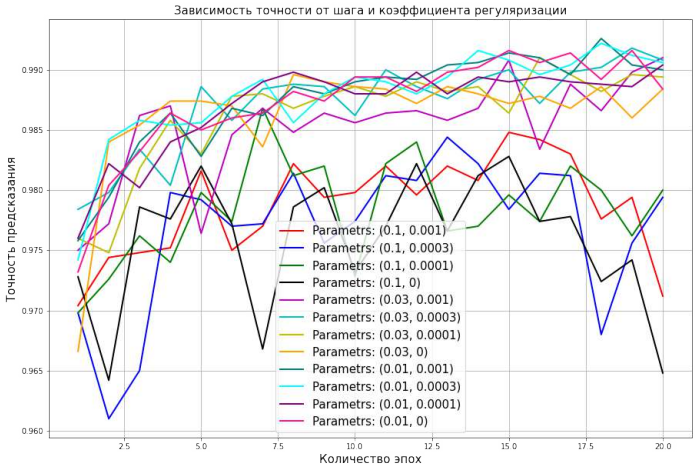

Рис. 13: K-FAC. График зависимости точности от количества эпох для разных параметров.

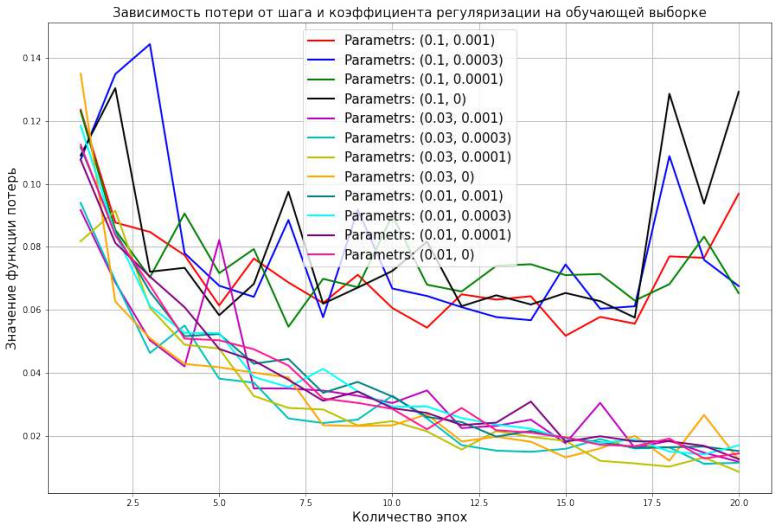

Рис. 14: K-FAC. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

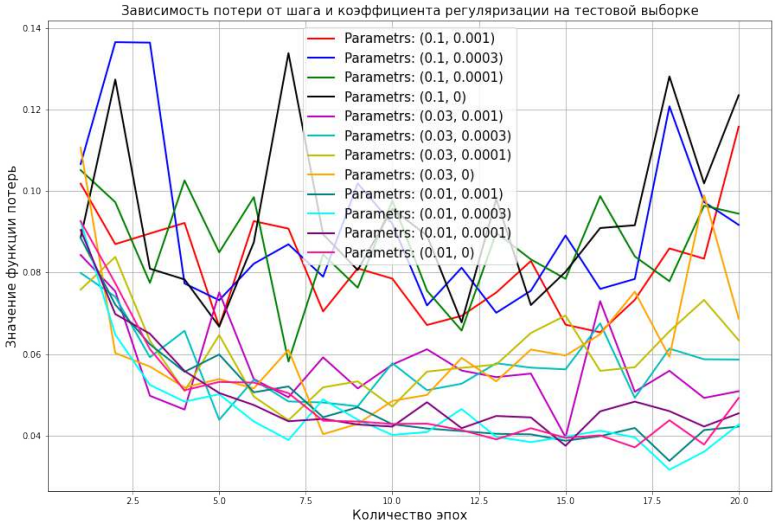

Рис. 15: K-FAC. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

В данном случае на графиках также прослеживается сильная зависимость от параметра шага обучения. Можно обратить внимание, что при шаге 0.1 прослеживается переобучение, так как ближе к двадцатой эпохи на графике зависимости точности от количества эпох кривые с таким шагом начинают снижаться, а на графиках зависимости значения функции потерь от количества эпох наоборот увеличиваться. Лучше всего ведут себя кривые с шагом 0.01. От коэффициента регуляризации работа метода зависит в меньшей степени, но можно заметить, что лучше себя показывают кривые с параметрами 0.001 и 0.0003.

Таким образом, для дальнейшего сравнения метода K-FAC с другими были выбраны шаг обучения 0.01 и коэффициент регуляризации 0.0003.

Итоговые результаты

В данном разделе представлены итоговые графики, на которых сравниваются 4 метода.

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 16), график зависимости значения функции потерь от количества эпох на тестовом наборе(рисунок 17) и на обучающем наборе(рисунок 18) в зависимости от метода, использованного при обучении нейронной сети и диаграмма времени (рисунок 19), затраченного на обучение нейронной сети в течении 20 эпох.

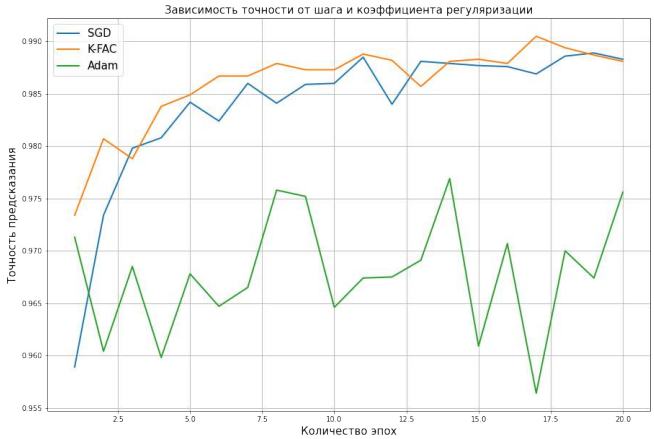

Рис. 16: График зависимости точности от количества эпох для разных методов.

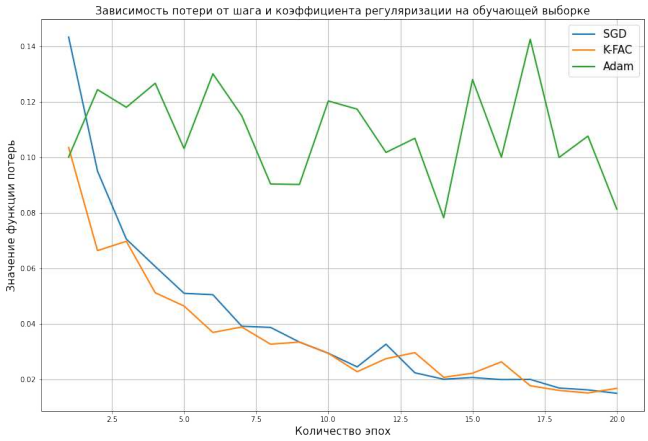

Рис. 17: График зависимости значения функции потерь от количества эпох при обучении для разных методов.

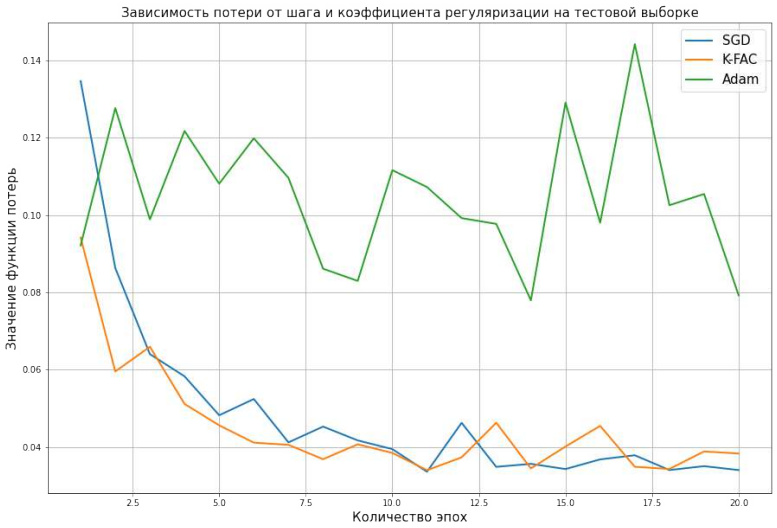

Рис. 18: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

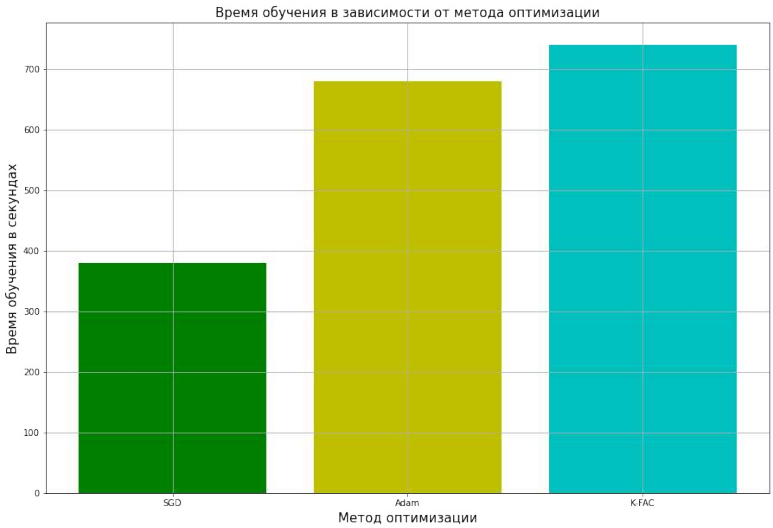

Рис. 19: Диаграмма времени обучения в зависимости от используемого метода.

Из рис. 16 хорошо видно, что K-FAC показывает себя чуть лучше, чем SGD, однако время, потраченное на обучение с использованием метода K-FAC, почти в два раза больше времени, потраченного на обучение с использованием метода SGD. Таким образом, данный эксперимент дает нам неоднозначные результаты.

5.2 Задача классификации цветных изображений

В данном эксперименте в качестве данных выступали цветные изображения из набора данных CIFAR-10 [4]. Обучающая выборка представлена 50000 экземплярами, а тестовая выборка 10000 экземплярами.

В этом эксперименте использовалась свёрточная нейронная сеть, известная как ResNet-18 [2]. В её архитектуре присутствует 18 слоев. Она очень полезна и эффективна при классификации изображений и может классифицировать изображения по 1000 категориям объектов.

Подбор параметров для метода SGD

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисуноки 20, 21), график зависимости значения функции потерь на тестовом наборе(рисунок 22) и график зависимости значения

функции потерь на обучающем наборе(рисунок 23). Исходя из данных графиков делались выводе о лучших параметрах для метода SGD в данной задаче.

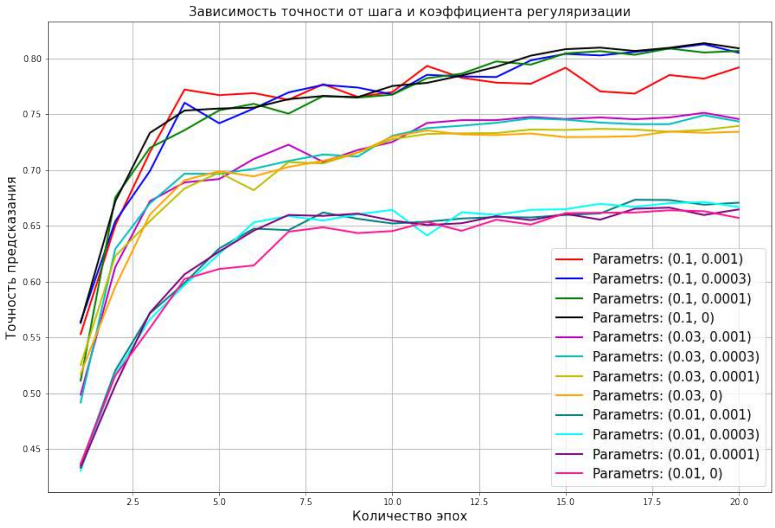

Рис. 20: SGD. График зависимости точности от количества эпох для разных параметров.

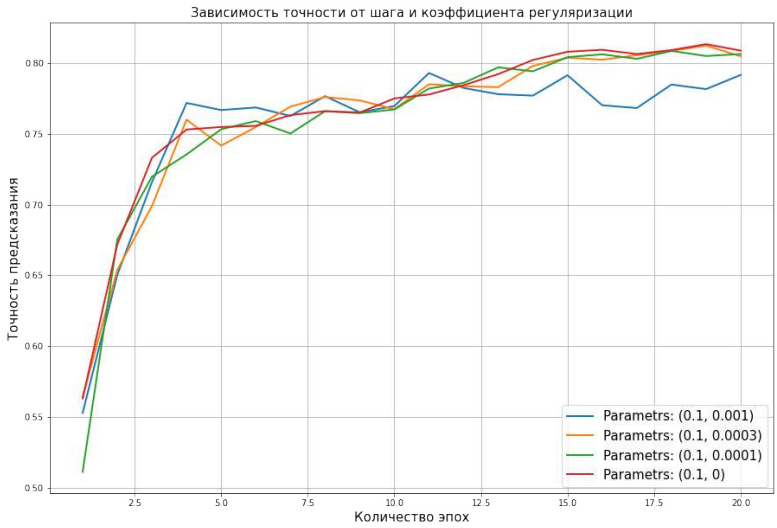

Рис. 21: SGD. График зависимости точности от количества эпох. Лучшие параметры.

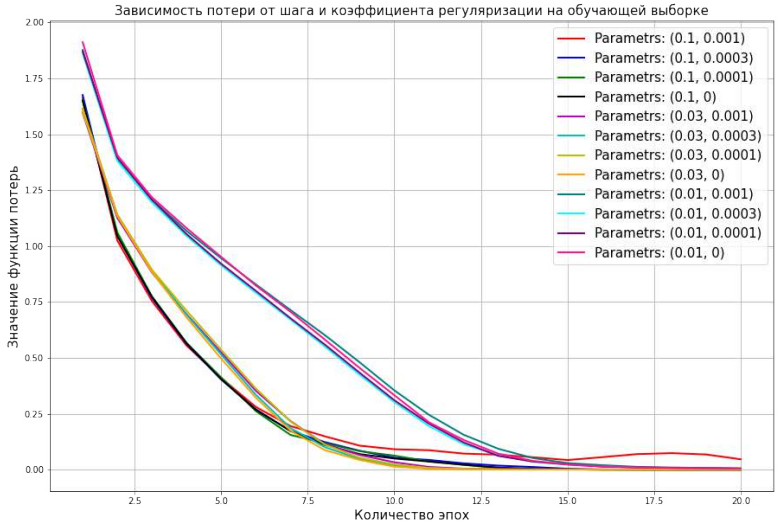

Рис. 22: SGD. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

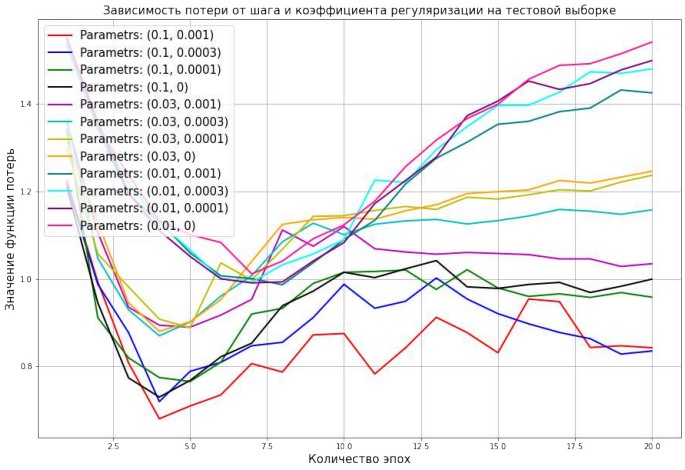

Рис. 23: SGD. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

На рисунке 22 кривые уходят вверх, что связано с особенностями задачи и переобучением. На графиках можно увидеть, что линии группируются по параметру шага, кривые с шагом 0.1 выше кривых с отличными значениями шага. По рисунку 22 видно, что коэффициент регуляризации выполняет свою задачу. При большем коэффициенте регуляризации преобучение уменьшается.

Для дальнейшего сравнения метода SGD с другими были выбраны шаг обучения 0.1 и коэффициент регуляризации 0.001.

Подбор параметров для метода Adam

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 24), график зависимости значения функции потерь на тестовом наборе(рисунок 25) и график зависимости значения функции потерь на обучающем наборе(рисунок 26). Исходя из данных графиков делались выводе о лучших параметрах для метода Adam в данной задаче.

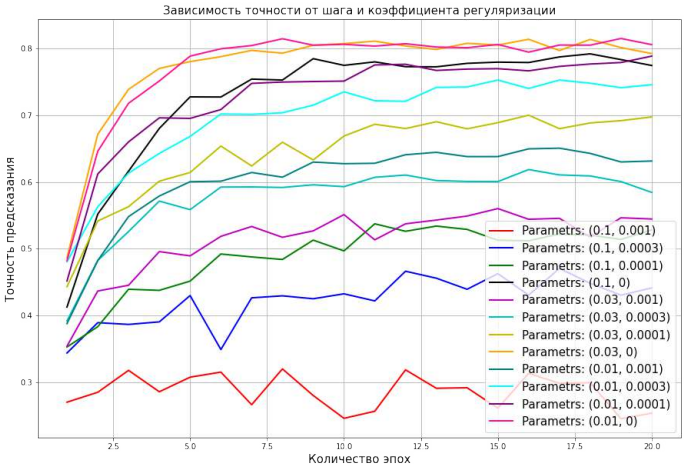

Рис. 24: Adam. График зависимости точности от количества эпох для разных параметров.

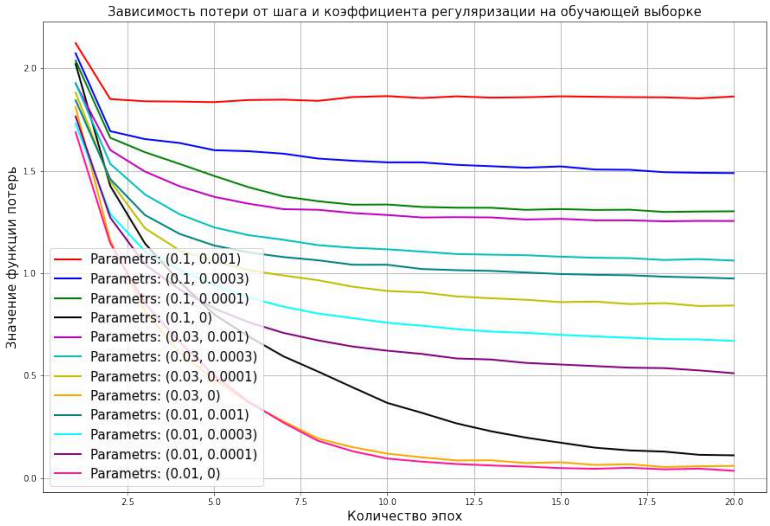

Рис. 25: Adam. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

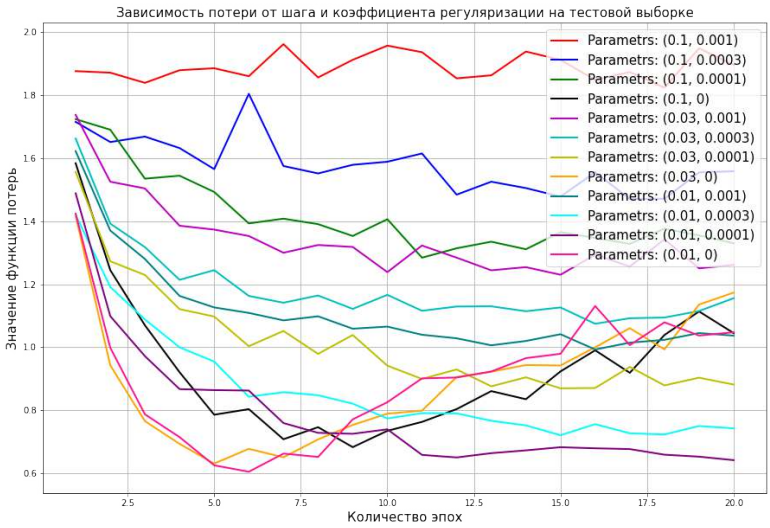

Рис. 26: Adam. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

Из рис. 26 видно, что здесь также присутствует переобучение при некоторых параметрах.

Исходя из графиков, для дальнейшего сравнения метода Adam с другими были выбраны шаг обучения 0.01 и коэффициент регуляризации 0.0001.

Подбор параметров для метода K-FAC

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 27), график зависимости значения функции потерь на тестовом наборе(рисунок 28) и график зависимости значения функции потерь на обучающем наборе(рисунок 29). Исходя из данных графиков делались выводе о лучших параметрах для метода K-FAC в данной задаче.

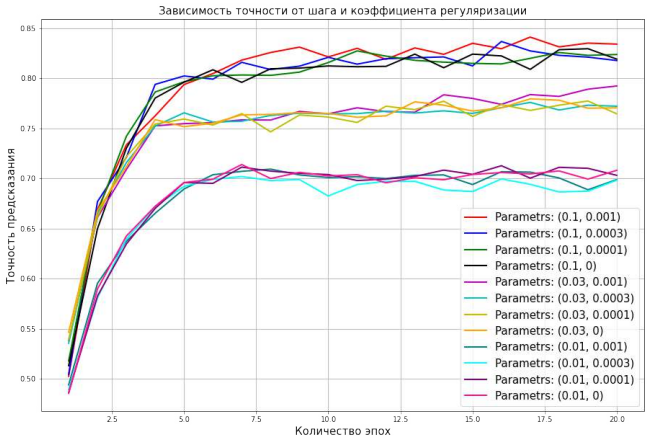

Рис. 27: K-FAC. График зависимости точности от количества эпох для разных параметров.

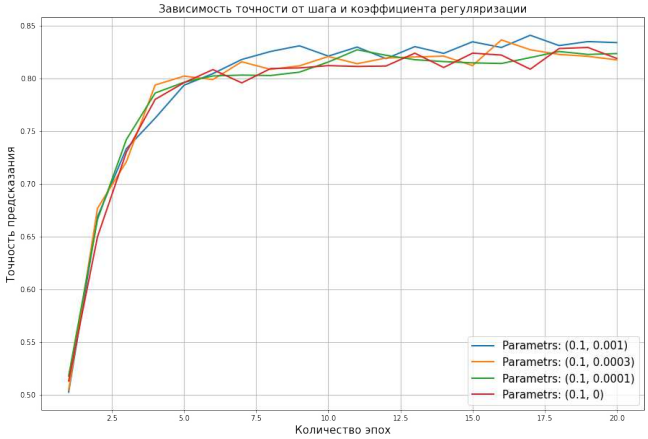

Рис. 28: K-FAC. График зависимости точности от количества эпох. Лучшие параметры.

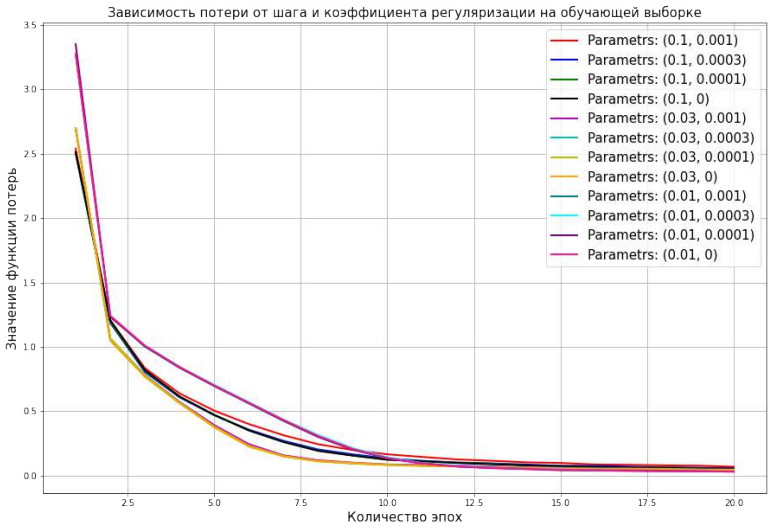

Рис. 29: K-FAC. График зависимости значения функции потерь от количества эпох при обучении для разных параметров.

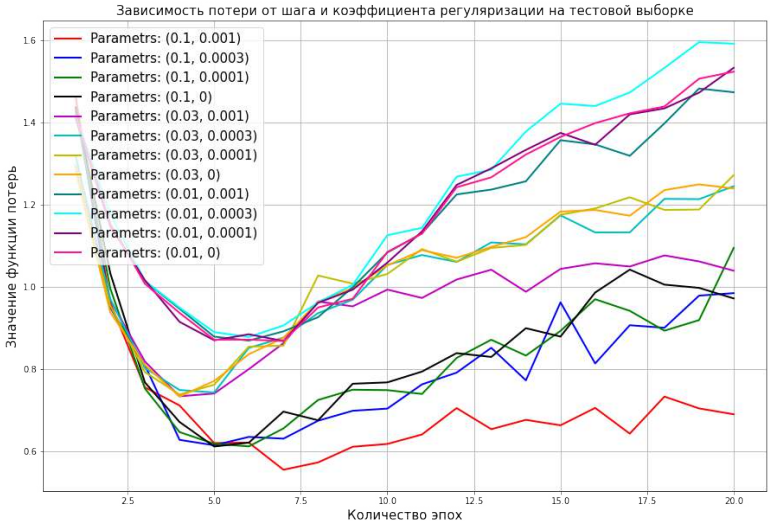

Рис. 30: K-FAC. График зависимости значения функции потерь от количества эпох на тестовой выборке для разных параметров.

При использовании данного метода также наблюдается переобучение. Леше всего ведут себя кривые с шагом 0.1, а больший коэффициент регуляризации лучше борется с переобучением.

Исходя из графиков, для дальнейшего сравнения метода K-FAC с другими были выбраны шаг обучения 0.1 и коэффициент регуляризации 0.001.

Итоговые результаты

На рисунках ниже представлены график зависимости точность на тестовом наборе(рисунок 30), график зависимости значения функции потерь от количества эпох на тестовом наборе(рисунок 31) и на обучающем наборе(рисунок 32) в зависимости от метода, использованного при обучении нейронной сети и диаграмма времени (рисунок 33), затраченного на обучение нейронной сети в течении 20 эпох.

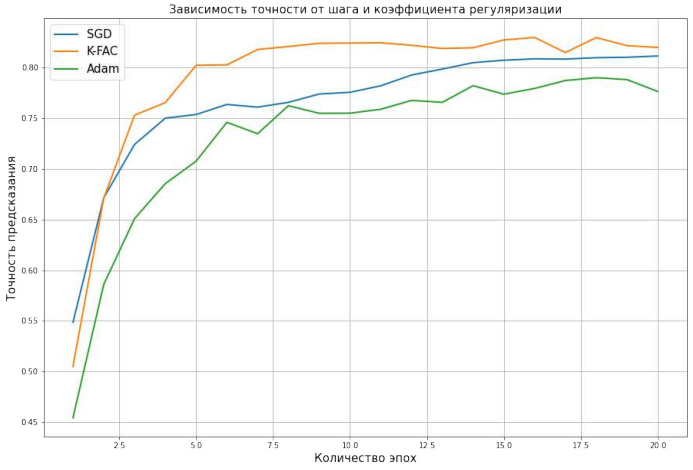

Рис. 31: График зависимости точности от количества эпох для разных методов.

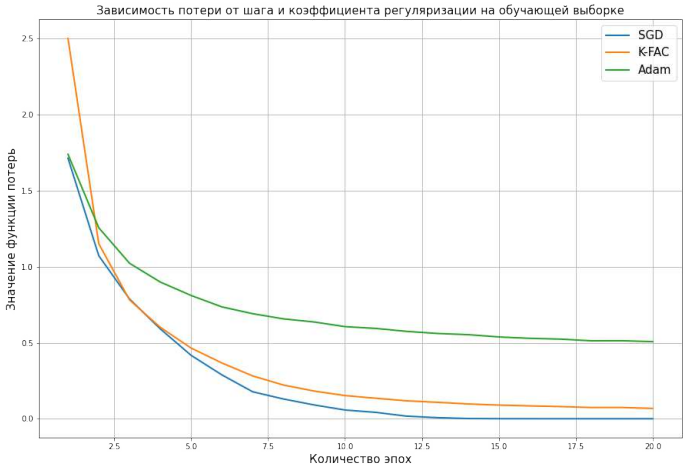

Рис. 32: График зависимости значения функции потерь от количества эпох во при обучении для разных методов.

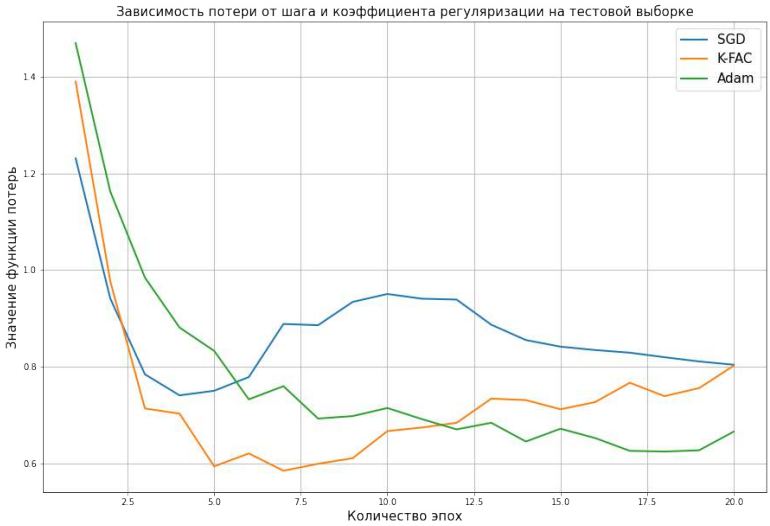

Рис. 33: График зависимости значения функции потерь от количества эпох на тестовой выборке для разных методов.

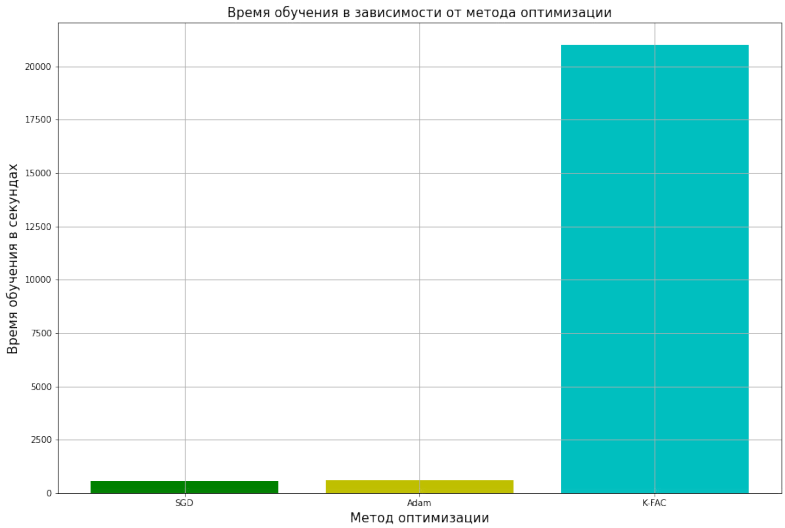

Рис. 34: Диаграмма времени обучения в зависимости от используемого метода.

Вновь можем заметить из рис. 32, что K-FAC показывает себя лучше метода Adam и немного лучше метода градиентного спуска, однако время, затраченное на обучение с использованием данного метода, во много раз превышает время работы других методов, что затрудняет его применение.

Однако, стоит обратить внимание на то, что метод K-FAC достигает высокой точности уже на 5-й эпохе и далее лишь немного улучшает этот результат, в то время как SGD достигает такой точности лишь к 15-й эпохе.

Заключение

В данной работе было проведено сравнение метода K-FAC, являющегося приближением метода второго порядка, с методом стохастического градиентного спуска и методом Adam при обучении нейронных сетей. На основе проведенных экспериментов было замечено быстрое достижение методом K-FAC высокого значения точности и низкого значения функции потерь относительно других методов. Однако, при использовании метода K-FAC затрачивается много времени, что делает его неэффективным для применения в реальных задачах. Из быстрой сходимости на первых эпохах можно сделать предположение о возможности эффективного комбинирования метода K-FAC с другими или применения к другим задачам.

Список литературы

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. — Cambridge: MIT Press, 2016.

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

3. Kingma D. P., Ba J. Adam: A method for stochastic optimization // 3rd International Conference for Learning Representations. — San Diego, 2015.

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

5. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE. — Monterey, 1998. — P. 2278–2324.

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