Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Казанский (Приволжский) федеральный университет

Институт Вычислительной Математики и Информационных Технологий

Кафедра Теоретической Кибернетики
Направление: 01.03.02 - Прикладная математика и информатика
Профиль: Системное программирование

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ С ПОМОЩЬЮ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Работа завершена:
Студент гр. 09-304 ИВМиИТ
"16" июня 2017 г. _____ Н.В. Садчикова

Работа допущена к защите:
Научный руководитель,
ассистент КСАИТ _____ Е.В. Разинков
"16" июня 2017 г.

Заведующий кафедрой,
д.ф-м.н., профессор, ч.-к. АН РТ _____ Ф.М. Аблаев
"16" июня 2017 г.

Казань — 2017

Содержание

Введение

Безопасность информации становится очень значительной и сложной задачей. Существует множество различных технологий идентификации человека, многие из которых используются в коммерческих целях в течение многих лет. Наиболее распространенными методами идентификации людей сегодня являются пароль или PIN-код. Проблема подобных методов заключается в том, что эти данные можно легко потерять, забыть, подделать, вследствие чего, с точки зрения надежности и безопасности – это не лучший подход для идентификации человека. Для преодоления этих проблем возник значительный интерес к системам идентификации, которые используют биометрические данные. Некоторые из этих методов – отпечатки пальцев и распознавание сетчатки и радужной оболочки. Эти методы имеют высокую точность, однако требуют дополнительных действий от пользователей, вследствие чего не всегда могут быть использованы. Распознавание лиц при помощи видео и голоса имеет естественное место в этих интеллектуальных средах следующего поколения, они просты в применении, так как не ограничивают движение пользователей и могут быть применимы к любому человеку, который находится в поле зрения камеры безопасности. Распознавание лиц – это система, используемая для идентификации или верификации человека с цифрового изображения. Под верификацией подразумевается сопоставление лиц людей, изображенных на 2 фотографиях. Идентификация – это сравнение лица, изображенного на фотографии, с некоторой базой данных лиц.

Система распознавания лиц используется во многих сферах, в первую очередь в сфере безопасности. В настоящее время камеры широко распространены в аэропортах, офисах, университетах, банкоматах, банках и в любых местах с системой безопасности. Уже сейчас существуют системы, позволяющие обнаружить людей, которые находятся в списке разыскиваемых. Это позволяет обеспечить безопасность в местах массового скопления. В октябре 2001 года аэропорт Fresno Yosemite International (FYI) в Калифорнии развернул технологию распознавания лиц Viisage для целей безопасности. Система предназначена для оповещения службы безопасности аэропорта FYl, когда человек, который находится в списке подозреваемых в терроризме, входит в контрольно-пропускной пункт аэропорта. Также система распознавания лиц применяется в сфере маректинг. Компания Herta Security предоставляет систему распознавания лиц, которая используется в дорогих магазинах в Европе. На входе в магазин камеры наблюдения ведут съемку за посетителями и отправляют ее на обработку компьютеру, который выделяет каждое лицо в толпе и пытается идентифицировать его. Если есть совпадение, программа передает информацию о предпочтениях клиента продавцам. Если человек ранее был замечен в кражах, информация об этом будет передана охранникам. Facebook с 2010 года использует алгоритм распознавания лиц, чтобы помочь пользователям отметить людей на фотографии. MasterCard также экспериментирует с системой распознавания лиц. Подобно сканерам отпечатков пальцев и другим биометрическим технологиям, распознавание лиц может

быть использовано вместо PIN-кода. Система распознавания лиц используется и в Microsoft Xbox, где пользователи могут получить доступ к своим профилям с помощью распознавания лиц. Также существуют приложения, которые позволяют находить людей, в точности похожих на человека, изображенного на фотографии.

Требования к системе распознавания лиц:

- высокая скорость работы;
- высокая точность;
- масштабируемость.

Цель данной работы: реализовать и обучить глубокую сверточную нейронную сеть с использованием целевой функции N-pair-mc loss, которая учитывает расстояние сразу между несколькими классами. Проанализировать влияние параметров обучения, архитектуры нейронной сети, алгоритмов выбора изображений для обучения на точность распознавания.

1. Методы распознавания лиц

1.1 Обзор методов

Первый алгоритм решения задачи распознавания лиц предложили Вуди Бледсо, Хелен Чан Вольф и Чарльз Биссон в 1960 году. Это была полуавтоматическая система, которая требовала от человека расстановки на изображении ключевых черт лица: носа, глаз, рта, ушей. Далее программа вычисляла расстояния между ними и отношение к общей опорной точке. Эти характеристики использовались для идентификации человека. В 1991 году Тюрк и Пентланд [1] применили анализ главных компонент (англ. principle component analysis), стандартный метод линейной алгебры, для вычисления характеристик ("eigenface"). Данный алгоритм показал, что для точного кодирования правильно выровненного и нормализованного изображения лица требуется менее 100 значений. Самым известным ранним примером системы распознавания лиц, которая использовала нейронные сети, является самоорганизующаяся карта Кохонена [2]. Эта система была простой нейронной сетью, которая способна распознавать выровненные и нормализованные изображения с лицами. Данный алгоритм вычислял описание лица, аппроксимируя собственные векторы автокорреляционной матрицы лица. В алгоритме Elastic Bunch Graph Matching [3] лица представлялись в виде графов с узлами, расположенными в ключевых точках(глаза, нос, рот) и на контуре лица. Каждый узел содержит набор из 40 комплексных вейвлет-коэффициентов Габора в разных масштабах и ориентациях (фаза, амплитуда).

В 2012 году сверточная нейронная сеть AlexNet впервые победила на международном соревновании ImageNet [4]. После этого для вычисления характеристик, описывающих лицо человека, стали исользовать сверточные нейронные сети. Огромный прорыв в этой области сделала компания Google в 2015 году, которая разработала систему распознавания лиц FaceNet [5]. Обучение проводилось таким образом, чтобы для разных изображений одного и того же человека нейронная сеть строила векторы, близкие друг к другу. Для этого выбирались три фотографии: на двух был один и тот же человек, на третьей другой человек. Значение целевой функции зависело от расстояния между векторами, соответствующими этим изображениям. Все дальнейшие инновации в этой области были связаны с улучшением целевой функции:

- в статье [6] значение целевой функции зависело не только от расстояния между классами, но и от расстояния между элементами внутри класса;
- в статье [7] для каждого класса находился центроид и целевая функция зависела от расстояния элемента класса до центроида;
- целевая функция N-pair-mc loss учитывала расстояние сразу между несколькими классами [8].

В данной работе более подробно будут рассмотрены целевая функция triplet loss [5], и N-pair-mc loss [8].

1.2 Система распознавания лиц FaceNet

Систему распознавания лиц FaceNet разработала компания Google в 2015 году [5]. Их идея заключалась в том, что на каждой итерации целевая функция вычислялась на основе трех изображений (триплетов): на двух из них был представлен один и тот же человек, на третьем – другой человек. Для вычисления векторного представления лица использовались глубокие сверточные нейронные сети. Обучение проводилось таким образом, чтобы сблизить вектора соответствующие одному и тому же человеку(положительная пара) и отдалить вектора, соответствующие разным людям(отрицательная пара).

Для этого использовалась триплетная целевая функция, которая имеет следующий вид:

$$L = \sum_{i}^{N} max(\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 - \left\|f(x_i^a) - f(x_i^n)\right\|_2^2 + \alpha, 0), \tag{1}$$

где x_i^a, x_i^p – изображения одного и того же человека, x_i^n – изображение другого человека, α – константа, $f(\cdot)$ – функция отображения входного изображение в d-мерное пространство характеристик.

Большую роль в скорости и качестве обучения нейронной сети играет выбор триплетов. Если создавать все возможные триплеты, то некоторые из них будут с легкостью удовлетворять условию:

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 + \alpha < \left\|f(x_i^a) - f(x_i^n)\right\|_2^2. \tag{2}$$

Эти триплеты не будут способствовать обучению нейронной сети. Поэтому крайне важно подбирать такие триплеты, которые будут нарушать условие (2). Для подготовки этих триплетов необходимо знать их векторное представление. Сначала выбирается некоторое количество классов. Для каждого класса произвольно выбирается заданное количество изображений, для которых будет вычислено векторное представление. На основании полученных векторов уже будут выбираться триплеты. Для каждого элемента выбирается положительный элемент так, чтобы расстояние до него было максимальным, а отрицательный пример - чтобы расстояние было минимальным, то есть:

$$x_i^p = \arg\max_{j \neq i}(\left\|f(x_i^a) - f(x_j^p)\right\|_2^2)$$

$$x_i^n = \arg\min_{j \neq i}(\left\|f(x_i^a) - f(x_j^n)\right\|_2^2)$$

Выбор таких жестких триплетов на практике может привести к плохому локальному минимуму на ранних этапах обучения. Чтобы избежать эту проблему, можно смягчить данное ограничение и выбирать триплеты таким образом, чтобы выполнялось условие:

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 < \left\|f(x_i^a) - f(x_i^n)\right\|_2^2.$$

Выбор триплетов из обучающей выборки требует больших предварительных вычислений, поэтому выполнять данный процесс каждую итерацию – неэффективно. Выбор триплетов осуществляется на фиксированное количество итераций – эпоху.

1.3 Целевая функция N-pair loss

В статье [8] используется аналогичный подход для задачи распознавания лиц: для каждого изображения строится его векторное представление, на основании которого будет приниматься решение. Для обучения предлагается использовать улучшенную целевую функцию, называемую N-pair loss. Сравнивая ее с триплетной целевой функцией, которая использует только один отрицательный пример, было показано, что большее количество отрицательных примеров для каждого элемента дает лучшие результаты.

Главная идея обучения нейронной сети на основе триплетов заключается в следующем: для каждого класса мы пытаемся сократить расстояние между векторами, соответствующими элементам из того же класса, увеличивая расстояния до векторов, соответствующих элементам из других классов. Однако во время одного обновления триплетная целевая функция сравнивает элемент с одним отрицательным примером, игнорируя при этом отрицательные примеры из других классов. Как следствие, после обновления расстояние между элементом и выбранным отрицательным примером будет увеличено, но расстояние до других классов может как уменьшиться, так и увеличиться. Таким образом, увеличивая расстояния для ограниченного количества классов, расстояния для многих классов будет оставаться небольшим. После перебора достаточно большого количества случайно выбранных триплетов, каждый класс будет находиться обособленно от других. В статье [8] предлагается альтернативная функция потерь, которая будет выбирать несколько отрицательных примеров для каждого элемента. В этом случае каждый элемент сравнивается с отрицательными примерами из нескольких классов. При каждом обновлении будет предприниматься попытка увеличить расстояние между этими классами.

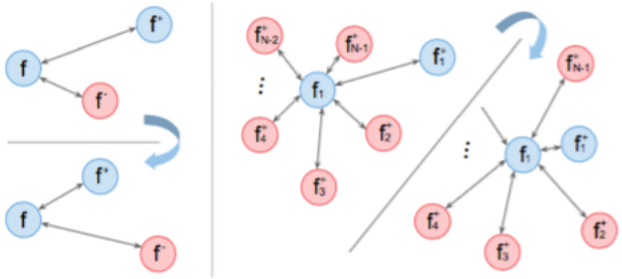

Рис. 1: выбор классов.

Обучение будет происходить на основе одного положительного примера и $N-1$ отрицательных примеров, поэтому каждый элемент обучающей выборки можно представить в виде

$$x, x^+, x_1, ..., x_{N1},$$

где x^+ и x – изображения одного и того же человека, а $\{x_i\}_{i=1}^{N-1}$ - соответствуют изображениям из других классов. Целевая функция, которую предлагают в этой

статье, называется (N+1)-tuplet loss и имеет следующий вид:

$$L(x, x^+, x_1, ..., x_{N1}; f) = log(1 + \sum_{i=1}^{N-1} exp(f^{\mathrm{T}} f_i - f^{\mathrm{T}} f^+)), \tag{3}$$

где под функцией f подразумевается глубокая сверточная нейронная сеть, которая отображает изображение в d-мерный вектор. Можно заметить, что при N=2 мы получим целевую функцию, которая для каждого элемента имеет один положительный пример и один отрицательный пример:

$$L_{(2+1)-tuplet}(x, x^+, x_i; f) = log(1 + exp(f^{\mathrm{T}} f_i - f^{\mathrm{T}} f^+)),$$

$$L_{triplet}(x, x^+, x_i; f) = max(0, f^{\mathrm{T}} f_i - f^{\mathrm{T}} f^+).$$

Решение задачи минимизации $L_{(2+1)-tuplet}$ совпадает с решением задачи минимизации $L_{triplet}$. Пусть N равно количеству классов L, тогда уравнение (3) примет вид:

$$log(1 + \sum_{i=1}^{N-1} exp(f^{\mathrm{T}} f_i - f^{\mathrm{T}} f^+)) = -log \frac{exp(f^{\mathrm{T}} f^+)}{exp(f^{\mathrm{T}} f^+) + \sum_{i=1}^{L-1} exp(f^{\mathrm{T}} f_i)}.$$

Такая запись формулы схожа с многоклассовой логистической функцией потери (softmax loss):

$$P(y = y_j) = -log \frac{exp(f^{\mathrm{T}} w_j)}{exp(f^{\mathrm{T}} w_j) + \sum_{i=1, i \neq j}^{L-1} exp(f^{\mathrm{T}} w_i)},$$

если выход функции f воспринимать как вектор характеристик, а вектора f^+ и f_i - веса. Тогда данная функция будет соответствовать вероятности $P(y = y^+)$. Однако, чем большее количество классов мы пытаемся разделить, тем сложнее становится отнести объект к одному классу. Таким образом, ограничивается количество классов, которые нейронная сеть учится разделять. С другой стороны функция $L_{(N+1)-tuplet}$ является лучшей аппроксимацией, чем триплетная целевая функция.

Если размер порции равен M, и для каждого элемента выбирать N-1 отрицательный класс, то количество элементов, которые должны пройти через нейронную сеть, равно $M(N + 1)$. В этой статье предлагается эффективный алгоритм, который для каждой порции выбирает N классов, по 2 примера из каждого класса. И в качестве отрицательных примеров для каждого элемента x_i будут использоваться положительные примеры из других классов x_j^+, где $i \neq j$. Целевая функция, которая используется для обучения будет иметь следующий вид:

$$L_{N-pair-mc}((x_i, x_i^+)_{i=1}^N; f) = \frac{1}{N} \sum_{i=1}^{N} log(1 + \sum_{i=1}^{N-1} exp(f^T f_i - f^T f^+)). \tag{4}$$

Как и в случае с триплетной функцией потерь, от выбора этих N классов будет зависеть скорость обучения нейронной сети. Эти классы выбираются таким образом, чтобы их векторные представления были близки друг к другу. Однако это требует больших предварительных вычислений, так как необходимо знать векторное представление всех элементов из датасета на каждом этапе выбора N классов. В данной статье предлагается жадный алгоритм выбора этих классов. Сначала произвольным образом выбирается некоторое количество классов C, которое должно быть больше N. В каждом классе выбирается несколько примеров, для которых будут посчитаны вектора. Таким образом, нам необходимо вычислить всего $2C$ векторов представлений. Далее из этих классов на основании векторов представлений будут выбираться N классов:

1. Произвольно из этих классов выбирается первый класс C_1.

2. На основании вычисленных векторов, соответствующих элементам из класса C_1, считается скалярное произведение со всеми векторами, соответствующими еще не выбранным классам. Следующий класс выбирается так, чтобы его скалярное произведение было максимальным. Таким образом мы получаем C_2.

3. Данный механизм повторяется $N - 1$ раз и новый класс добавляется на основании последнего выбранного класса.

Точность, полученная в статье [8] при помощи триплетной целевой функции, равна $95,88\%$, а при помощи целевой функции N-pair-mc loss – $98,33\%$ для N=320. Архитектура нейронной сети, которая использовалась в этой статье – CasiNet [14] без последнего полносвязного слоя.

2. Реализация нейронной сети для распознавания лиц

Задачу распознавания лиц на фотографии усложняют такие факторы, как расположение лица на фотографии, его наклон, размер, освещенность лица, наличие очков, прическа, макияж, эмоции, тень на лице. Некоторые проблемы могут быть решены при помощи применения алгоритмов обнаружения лиц и выравнивания лиц [9]. Алгоритм обнаружения лиц заключается в выделении фрагмента изображения, содержащего лицо. Алгоритм выравнивания преобразует изображение таким образом, чтобы все части лица находились примерно на одной и той же части изображения. Каждое изображение приводилось к размеру $130 \times 130 \times 3$. Для увеличения разнообразия данных на этапе обучения для каждого изображения применялись две техники:

- нарезка исходного изображения с некоторым сдвигом;
- зеркальное отображение.

На этапе тестирования изображения вырезалось по центру. Последний этап обработки каждого изображения – стандартизация, которая центрирует значения пикселей относительно 0 между [-1,1].

Для обучения нейронной сети использовались две базы данных лиц, которые находятся в открытом доступе. Первый – WebFace [10] включает в себя 10575 классов, всего изображений – 494 414. Второй – MsCeleb [11] включает в себя 100 000 классов (знаменитостей), в каждом классе в среднем по 100 изображений. В итоге он состоит из 10 миллионов изображений.

Проверка точности работы алгоритма проводилась по окончанию каждой эпохи на базе данных лиц Labeled faces in the Wild [12]. Эта база данных лиц содержит 13234 изображения, всего человек, представленных в нем, 1680. По каждому человеку есть как минимум два изображения. Файл pairs.txt содержит пути к изображениям и информацию: эти изображения относятся к одному человеку или нет. На основании этого файла проводится оценка точности для двух классов: на изображениях разные люди (класс C_0) или один и тот же человек (C_1). Для каждой пары считается расстояние на основании косинуса угла между вектормаи:

$$dist(x, y) = \frac{x^{\mathrm{T}} y}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}}.$$

Пусть x – элемент, состоящий из двух векторов характеристик, где x_0 соответствует первому изображению пары, x_1 – второму. Тогда

$$dist(x) = \frac{x_0^{\mathrm{T}} x_1}{\sqrt{\sum_i x_{0,i}^2} \cdot \sqrt{\sum_i x_{1,i}^2}}.$$

Пусть функция классификации имеет следующий вид:

$$f(x,\theta) = \begin{cases} 1, dist(x) \geq \theta \\ 0, dist(x) < \theta \end{cases},$$

где θ - пороговое значение. Необходимо подобрать такое пороговое значение, чтобы точность классификации была максимальной. Для этого используется кросс-валидация. Все пары изображений делятся на 10 частей, 9 частей идут в обучающую выборку, одна часть – в тестовую выборку. На обучающей выборке подбирается пороговое значение таким образом, чтобы максимизировать точность классификации на ней. Пороговое значение выбирается из промежутка $[-1, 1]$ с шагом 0,01 по формуле:

$$\theta^* = \underset{\theta \in [-1,1]}{\arg\max} \left(\frac{1}{|X|} \sum_{x \in X} f(x,\theta) \right).$$

По этому пороговому значению будет происходить оценка точности на тестовой выборке. Данный алгоритм запускается 10 раз, итоговая точность будет найдена как среднее по всем полученным на тестовой выборке значениям точности.

2.1 Архитектура нейронной сети

В данной работе реализована архитектура нейронной сети CasiaNet [14] при помощи библиотеки с открытым исходным кодом Tensorflow. Эта архитектура на вход принимала изображения размера $100 \times 100 \times 3$ и на выходе выдавала вектор размерности 320. Ниже приводится описание исходной архитектуры.

Таблица 1: Архитектура нейронной сети CasiaNet [14]

Тип слоя	Размер фильтра, шаг	Размер выхода	Количество параметров	Функция активации
convolution	$3 \times 3, 1$	$100 \times 100 \times 32$	280	ReLU
convolution	$3 \times 3, 1$	$100 \times 100 \times 64$	18 тыс.	ReLU
max pooling	$2 \times 2, 2$	$50 \times 50 \times 64$	0	–
convolution	$3 \times 3, 1$	$50 \times 50 \times 64$	36 тыс.	ReLU
convolution	$3 \times 3, 1$	$50 \times 50 \times 128$	72 тыс.	ReLU
max pooling	$2 \times 2, 2$	$25 \times 25 \times 128$	0	–
convolution	$3 \times 3, 1$	$25 \times 25 \times 96$	108 тыс.	ReLU
convolution	$3 \times 3, 1$	$25 \times 25 \times 192$	162 тыс.	ReLU
max pooling	$2 \times 2, 2$	$13 \times 13 \times 192$	0	–
convolution	$3 \times 3, 1$	$13 \times 13 \times 128$	216 тыс.	ReLU
convolution	$3 \times 3, 1$	$13 \times 13 \times 256$	288 тыс.	ReLU
max pooling	$2 \times 2, 2$	$7 \times 7 \times 256$	0	–
convolution	$3 \times 3, 1$	$7 \times 7 \times 160$	216 тыс.	ReLU
convolution	$3 \times 3, 1$	$7 \times 7 \times 320$	288 тыс.	ReLU
average pooling	$7 \times 7, 1$	$1 \times 1 \times 320$	0	–
L2-norm	–	320	0	–

Convolution – слой, в котором все нейроны связаны с небольшой окрестностью предыдущего слоя и имеют один и тот же тензор весов (фильтр). Max pooling и average pooling – слой, который находит максимальное и среднее значение соответственно в области, соответствующей размеру фильтра. В качестве функции активации использовалась функция $ReLU(x) = max(0, x)$. Слой L2-norm нормализует вектор по формуле:

$$\overline{x}_{norm} = \frac{\overline{x}}{\sqrt{\sum\limits_i x_i^2}}.$$

Обучение нейронной сети осуществлялось при помощи стохастического градиентного спуска – алгоритма Adam [13].

Все веса сверточных слоев нейронной сети, а также веса полносвязного слоя инициализировались значениями из нормального распределения со средним значением равным 0 и дисперсией равной $\frac{1}{n_{in}}$ для линейной функции активации и $\frac{2}{n_{in}}$

для нелинейной функции активации. Данный вид инициализации предложен в статье [16]. Рассмотрим, как она влияет на процесс обучения на примере нейронной сети, состоящей из одного нейрона. Пусть на вход подается вектор $X \in \mathbb{R}^n$, вектор весов $W \in \mathbb{R}^n$ и

$$Y = \sum_{i=1}^{n} W_i \cdot X_i.$$

Из теории вероятности известно, что если две случайные величины независимы, то дисперсия их произведения задается формулой:

$$\mathsf{D}(W_i \cdot X_i) = [E(W_i)]^2 \, \mathsf{D}(X_i) + [E(X_i)]^2 \, \mathsf{D}(W_i) + \mathsf{D}(W_i) \, \mathsf{D}(X_i).$$

Если веса W и входной вектор X имеют среднее значение 0, то формула примет вид:

$$\mathsf{D}(W_i \cdot X_i) = \mathsf{D}(W_i) \, \mathsf{D}(X_i).$$

Тогда дисперсия выходного значения Y будет равна:

$$\mathsf{D}(Y) = \sum_{i=1}^{n} \mathsf{D}(W_i) \, \mathsf{D}(X_i) = n \cdot \mathsf{D}(W_i) \, \mathsf{D}(X_i).$$

Таким образом, если мы хотим, чтобы распределения входного вектора X и выходного значения Y было одинаковым, то дисперсия весов должна быть равна $\frac{1}{n}$. При использовании нелинейной функции активации ReLU, половина выходов станет равной 0, поэтому инициализация весов должна быть из нормального распределения с дисперсией равной $\frac{2}{n_{in}}$. Для сверточного слоя, в котором веса имеют размерность $L \times M \times D$, формула дисперсии будет иметь вид $\frac{2}{L \cdot M \cdot D}$

Усложнение архитектуры глубокой сверточной нейронной сети усложняет процесс обучения. Добиться ускорения при обучении глубоких сетей помогает нормализация активаций нейронов (англ. batch normalization) [15]. Данный алгоритм использует несколько обучаемых параметров: β и γ с начальными значениями 0 и 1 соответственно. На каждой итерации вычисляется среднее значение и дисперсия для элементов x_i из порции:

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} x_i,$$

$$\sigma_i^2 = \frac{1}{n} \sum_{j=1}^{n} (x_i - \mu_i)^2.$$

На основании полученных значений каждый элемент порции нормируется:

$$\hat{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}.$$

Для каждого нормализованного элемента далее применялось линейное преобразование:

$$BN(x_i) = \gamma \hat{x_i} + \beta.$$

Среднее значение и дисперсия, полученные на этапе обучения, использовались для нормализации порции данных на этапе тестирования. Ниже будут приведены эксперименты с добавлением нормализации активаций нейронов после сверточных слоев.

2.2 Выбор классов для обучения

Важным этапом в этих алгоритмах является выбор классов, используемых в одной эпохе. Чем более сложные классы будут подобраны, тем быстрее нейронная сеть будет обучаться. Под сложными классами подразумеваются такие классы, на которых значение целевой функции будет максимальным. Пусть N – количество классов, выбранных на одну эпоху. Процесс выбора этих классов включает следующие этапы:

Algorithm 1

1: Случайным образом выбирается M классов, $M > N$.
2: Из этих классов выбирается по одному изображению и считается вектор характеристик.
3: Случайно выбирается первый класс C_1
4: $I = \{C_1\}$
5: **for** $i := 1$ to $N - 1$ **do**
6: $\quad C_{i+1} = \underset{j \notin I}{\arg\max}(f_j^\top f_{C_i})$
7: $\quad I = I \cup \{C_{i+1}\}$
8: **end for**

Данный метод требует минимальных предварительных вычислений. Но вывод относительно близости двух классов делается по одному изображению из каждого класса. Увеличение количества изображений для каждого класса приведет к увеличению предварительных вычислений. Однако чем больше векторов будет вычислено для каждого класса, тем лучше будет представление о близости классов. Следующий алгоритм выбирает по два изображения на класс.

Algorithm 2

1: Случайным образом выбирается M классов, $M > N$.
2: Из этих классов выбирается по два изображению и считается вектор характеристик.
3: Случайно выбирается первый класс C_1
4: $I = \{C_1\}$
5: **for** $i := 1$ to $N - 1$ **do**
6: $\quad C_{i+1} = \underset{j \notin I}{\arg\max} \left(\frac{1}{4} \sum_{s=0}^{1} \sum_{t=0}^{1} f_{s,j}^\top f_{t,C_i} \right)$
7: $\quad I = I \cup \{C_{i+1}\}$
8: **end for**

Расстояния между классами, вычисленные на основании двух произвольных изображений из этих классов, точнее описывают их реальное расположение, чем расстояния между классами, вычисленные на основании одного изображения. Поэтому все эксперименты будут проводиться с использованием двух изображений

из каждого класса. В Algorithm 2 максимально приближенный класс вычисляется только на основании последнего добавленного класса. Оптимизировать этот алгоритм можно, если выбирать максимально близкий класс не только для последнего добавленного, но и для всех уже добавленных классов. Рассмотрим данный алгоритм.

Algorithm 3

1: Случайным образом выбирается M классов, $M > N$.
2: Из этих классов выбирается по два изображению и считается вектор характеристик.
3: Случайно выбирается первый класс C_1
4: $I = \{C_1\}$
5: **for** $i := 1$ to $N - 1$ **do**
6: $\quad C_{i+1} = \underset{i \in I \quad j \notin I}{\operatorname{argmax}} \left(\frac{1}{4} \sum_{s=0}^{1} \sum_{t=0}^{1} f_{s,j}^{\top} f_{t,i} \right)$
7: $\quad I = I \cup \{C_{i+1}\}$
8: **end for**

Эксперименты

3.3 Сравнение алгоритмов выбора классов

Размер порции изображений, на которой производилось обучение в течении одной итерации, во всех экспериментах было установлено равным 320. В этой порции было по 2 изображения из каждого класса. На каждую эпоху выбиралось $N = 960$ классов. Количество классов M варьировалось от 3200 до количества классов в обучающей выборке. В этих экспериментах для вычисления ветора характеристик использовалась архитектура CasiaNet с добавление полносвязного в конце. После каждого сверточного слоя и полносвязного слоя применялась нормализация активаций нейронов (англ. batch normalization).

Таблица 2: Архитектура нейронной сети CasiaNet с нормализацией

Тип слоя	Размер фильтра, шаг	Размер выхода	Количество параметров	Функция активации
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 32$	282	ReLU
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 64$	18 тыс.	ReLU
max pooling	$2 \times 2, 2$	$50 \times 50 \times 64$	0	–
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 64$	36 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 128$	72 тыс.	ReLU
max pooling	$2 \times 2, 2$	$25 \times 25 \times 128$	0	–
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 96$	108 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 192$	162 тыс.	ReLU
max pooling	$2 \times 2, 2$	$13 \times 13 \times 192$	0	–
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 128$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 256$	288 тыс.	ReLU
max pooling	$2 \times 2, 2$	$7 \times 7 \times 256$	0	–
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 160$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 320$	288 тыс.	ReLU
average pooling	$7 \times 7, 1$	$1 \times 1 \times 320$	0	–
fully connected + BN	320	320	102 тыс.	–
L2-norm	–	320	0	–

После average pooling тензор размера $1 \times 1 \times 320$ вытягивается в вектор размера 320. Полносвязный слой включает в себя матрицу весов W. К выходу полносвязного слоя применяется нормализация:

$$y = BN(Wx).$$

В качестве целевой функции использовалась N-pair-mc loss. Для регуляризации полносвязного слоя к исходной целевой функции добавлялась регуляризационная ошибка $\frac{\beta}{2}\sum_i |w_i|^2$, где $\beta = 0,0005$, а w_i - веса полносвязного слоя. Начальное значение коэффициента скорости обучения ставилось равным 0,05 и применялся метод уменьшения коэффициента exponential decay. Размер вектора характеристик выбирался равным 320.

1. Для выбора классов использовался жадный алгоритм на основании двух примеров. Из каждого класса выбиралось 50 пар изображений. Рассмотрим алгоритм генерации данных на эпоху:

Algorithm 4

$n = 160$ – количество классов в каждой порции
$N = 960$ – количество классов, выбранных на эпоху
$C_1, C_2, .., C_N$ - классы, полученные при помощи Algorithm 2
$stride = 160$ - шаг
$RepeatCount = 50$
for $j := 1$ to $RepeatCount$ **do**
    $start = 0$
    while $start < N - n$ **do**
        Выбираются пары изображений из классов $C_{start}, .., C_{start+n}$
        $start = start + stride$
    end while
end for

При размере шага равного размеру порции, 960 классов разобьются на 6 непересекающихся множеств. В течение одной эпохи обучение будет происходить таким образом, чтобы разделить классы внутри каждого множества. Из каждого класса в порции произвольным образом будет выбрано 50 различных пар на итерацию. Маленькое значение целевой функции на первых итерациях эпохи будет означать, что классы плохо подобраны. Такой эффект может наблюдаться только спустя некоторое время, когда нейронная сеть будет достаточно хорошо обучена. Обучение нейронной сети продолжалось 42 эпохи по 300 итераций. Максимальная точность $85,8\%$ была достигнута на 42 эпохе. Значение регуляризационной ошибки уменьшилось с 0,123 до 0,002.

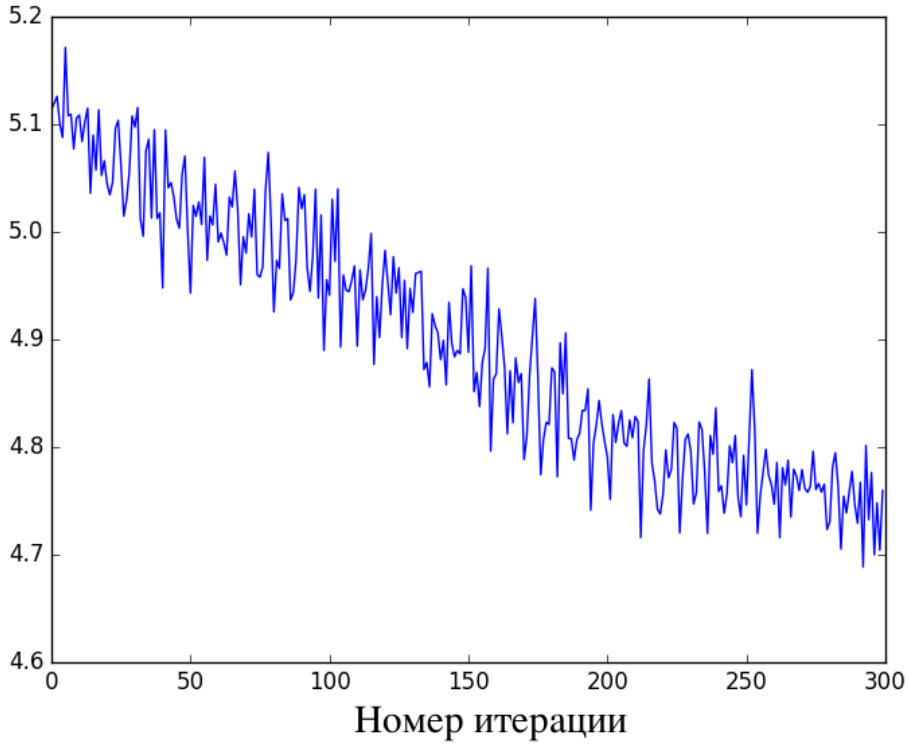

Рис. 2: график целевой функции на первой эпохе.

Значение целевой функции на первых итерациях около $5,1$. Это максимальное значение, которое принимала целевая функция на протяжении всего процесса обучения. К концу первой эпохи значение целевой функции уменьшилось до $4,75$. Точность после первой эпохи на валидационной выборке (Labeled Faces in the Wild) равна $66,7\%$.

Значение целевой функции на первых итерациях 47 эпохи $4,55$. График целевой функции на 47 эпохе начал медленне понижаться. Чем дальше будет идти обучение, тем медленнее будет уменьшаться значение целевой функции.

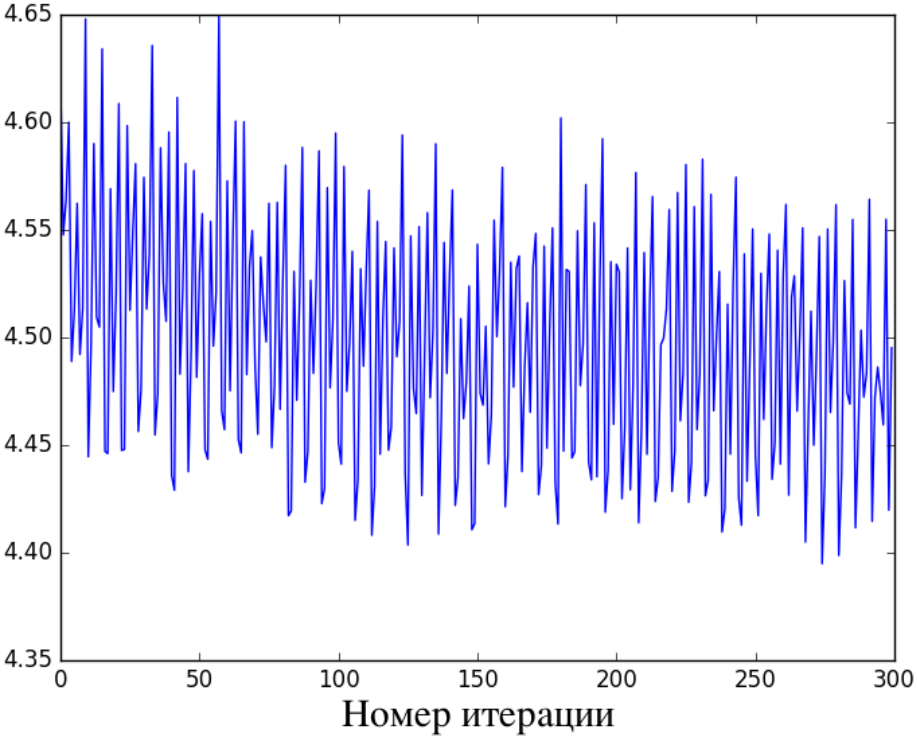

Рис. 3: график целевой функции на 47 эпохе.

После 42 эпохи точность на валидационной выборке перестала увеличиваться и к концу 46 эпохи была равна 85%. Для этого обучение было продолжено начиная с 42 эпохи с меньшим значением коэффициента скорости обучения: 0,03. Количество классов M было увеличено до 6400. Обучение длилось 12 эпох.

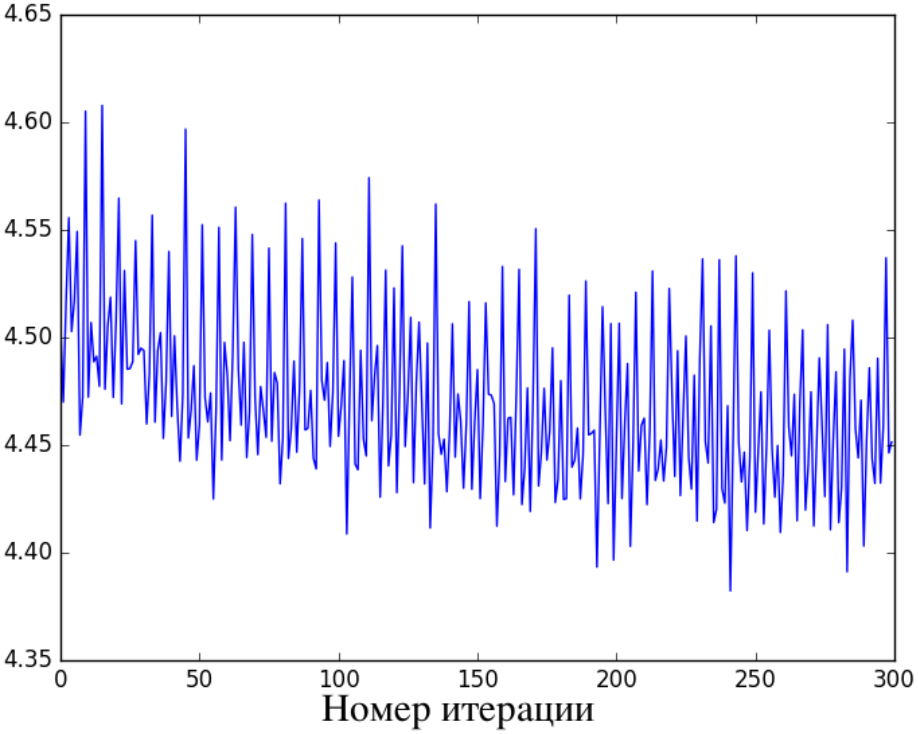

Рис. 4: график целевой функции на 12 эпохе.

После первых 6 эпох была достигнута точность $86,3\%$. График ошибки в течение этих итераций был практически ровным. Это могло свидетельствовать о том, что значение скорости обучение было достаточно велико. Обучение было продолжено с 6 эпохи с коэффициентом скорости обучения, равным 0,008. M увеличилось до 9071. Обучение продолжалось 28 эпох по 120 итераций. На 25 эпохе была достигнута точность $87,9\%$. Обучение было продолжено с 25 эпохи, значение скорости обучения было уменьшено до 0,005. Количество классов M осталось равным 9071. Обучение проолжалось в течение 52 эпох. На 7 эпохе была достигнута точность $88,2\%$. На протяжении 8 – 52 эпохи точность так и не увеличилась.

Итоговая точность, которая была достигнута при помощи этого метода – $88,2\%$.

2. Недостаток предыдущего алгоритма заключается в том, что он решает задачу разделения классов в 6 подмножествах. Решение данной проблемы – выбирать порцию классов начиная с произвольного класса. Так как все классы были набраны последовательно, то с какого бы класса не начали выбирать классы на одну порцию, все они будут близки к соседним. Рассмотрим алгоритм подготовки данных на одну эпоху.

Algorithm 5

1: $n = 160$ – количество классов в каждой порции
2: $N = 960$ – количество классов, выбранных на эпоху
3: $C_1, C_2, .., C_N$ - классы, полученные при помощи Algorithm 2
4: $stride = 160$
5: $NrofIterations = 300$ – количество итераций
6: **for** $j := 1$ to NrofIterations **do**
7: Произвольным образом выбирается $start \in [1, N - n]$
8: Выбираются пары изображений из классов $C_{start}, .., C_{start+n}$
9: **end for**

Обучение нейронной сети продолжалось 211 эпох по 300 итераций. Максимальная точность $87,5\%$ была достигнута на 157 эпохе.

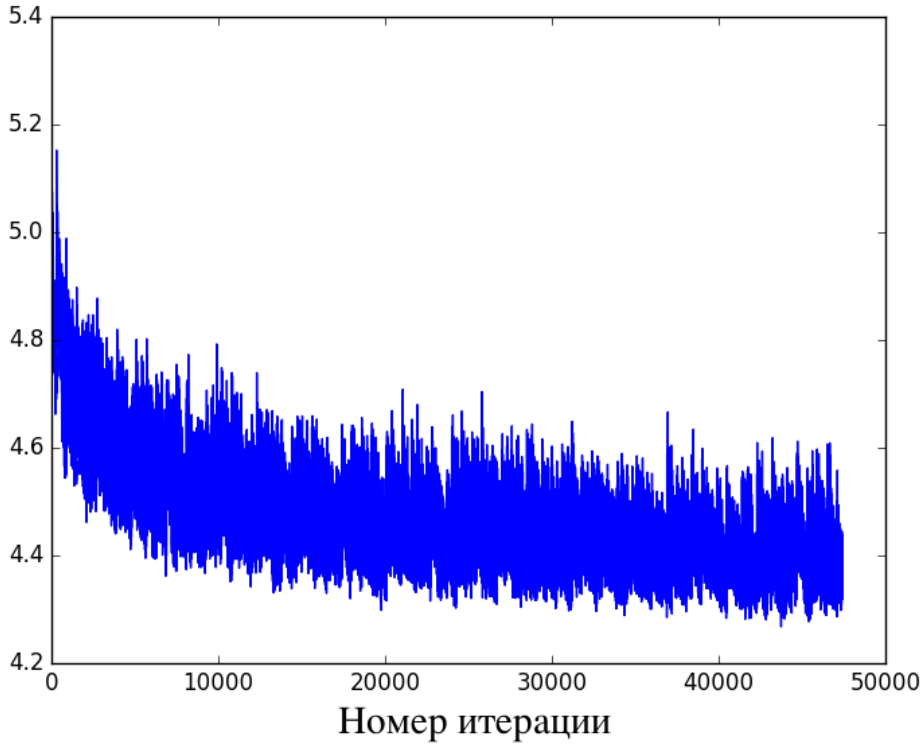

Рис. 5: график целевой функции в течение 211 эпох.

Рассмотрим, как меняется значение целевой функции на первых трех эпохах. Первые 300 итераций значение целевой функции уменьшилось с $5,15$ до $4,75$. Ска-

чок на 300 итерации связан с тем, что для этой итерации были выбраны новые классы. Высокое значение целевой функции в начале 300 итерации свидетельствуют о том, что классы на эту эпоху были выбраны хорошо. На 600 итерации такого скачка графика целевой функции не наблюдается. Это говорит о том, что классы на эту эпоху были выбраны не самые сложные. Значение регуляризационной ошибки уменьшилось с 0,124 до 0,002.

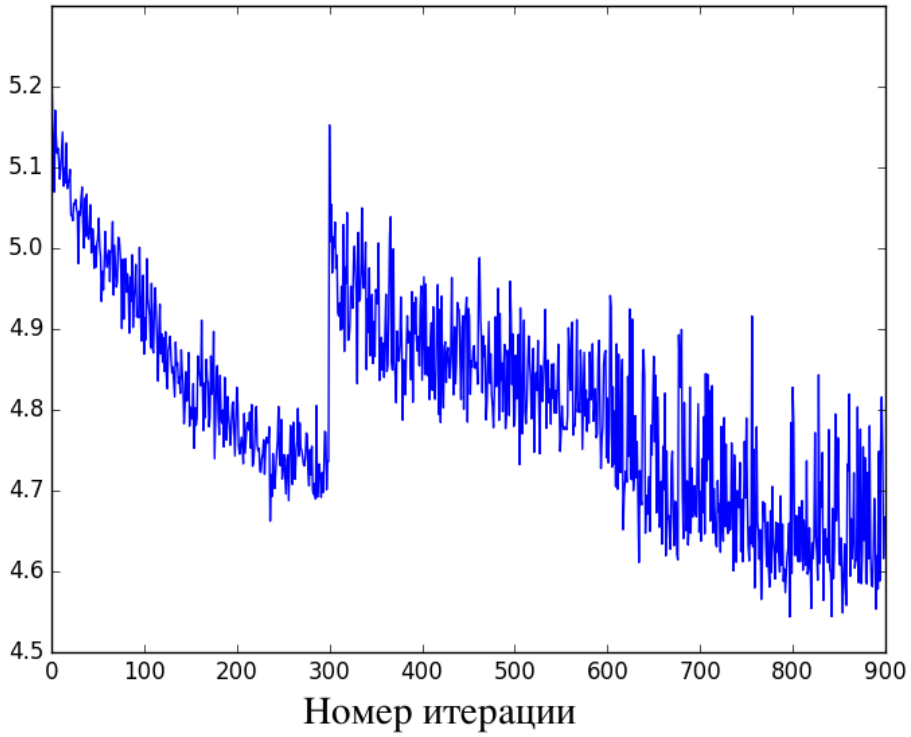

Рис. 6: график целевой функции на первых 3 эпохах.

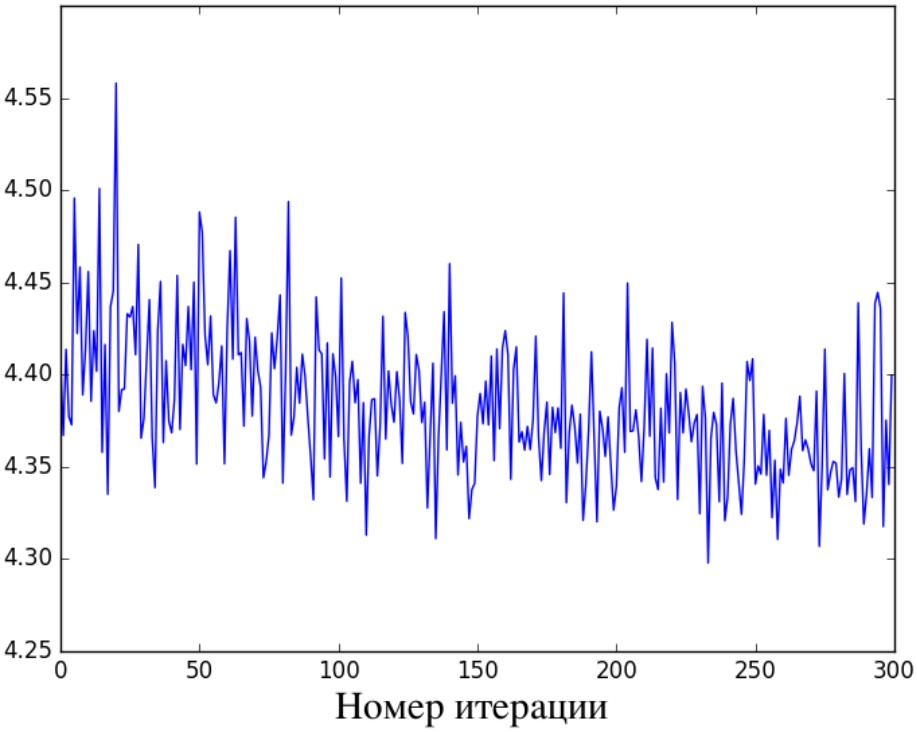

Рис. 7: график целевой функции на 157 эпохе.

Как видно по графику, значение целевой функции продолжает уменьшаться. После 157 эпохи точность на валидационной выборке стала немного меньше. Обучение продолжилось начиная со 158 эпохи. Количество итераций в эпохе было уменьшенно до 120. Это было сделано для того, чтобы нейронная сеть не успевала слишком сильно обучиться под эти классы. Количество классов M было установлено равным 9071. Ниже приведен график целевой функции начиная со 158 эпохи.

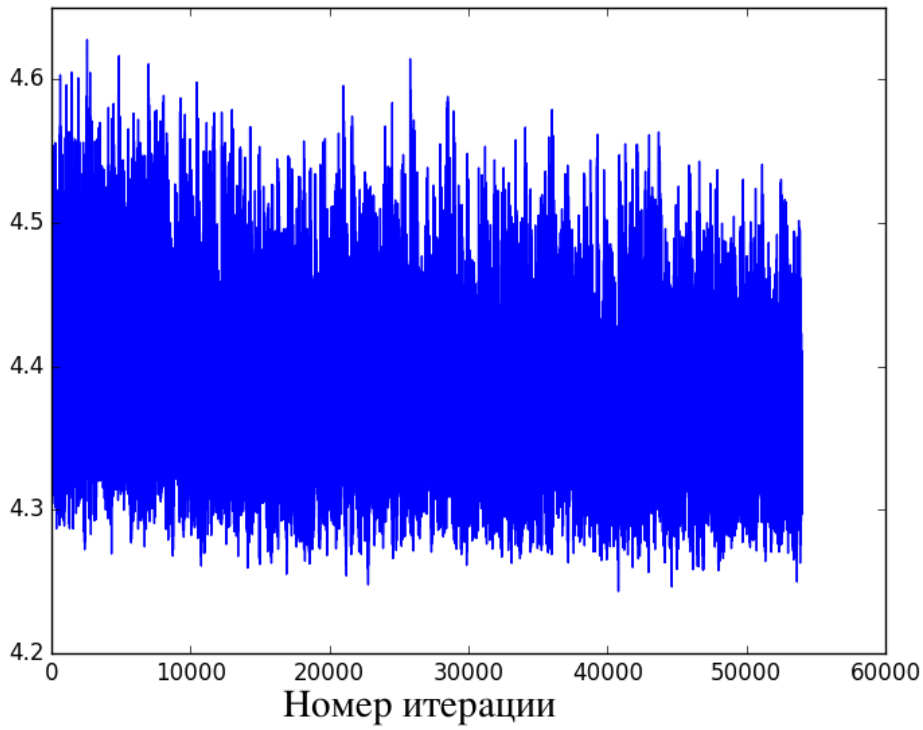

Рис. 8: график целевой функции в течение 450 эпох.

График целевой функции продолжил понижаться, но более плавно. Обучение продолжалось в течение 450 эпох по 120 итераций. На 94 эпохе была достигнута точность $87,9\%$. После 94 эпохи точность на валидационной выборки не поднималась больше $87,2\%$. Обучение было продолжено с 94 эпохи с коэффициентом скорости обучения равным 0,003. Спустя 133 эпохи была достигнута точность $88,2\%$. Итоговая точность данного эксперимента, равная $88,3\%$, была получена после уменьшения коэффициента скорости обучения до 0,001 через 8520 итераций.

3. Выбор классов на эпоху в двух предыдущих экспериментах основывался на жадном алгоритме. В этом эксперименте используется Algorithm 3 для выбора классов на одну порцию, в каждой порции по 160 самых близких классов.

Algorithm 6

Произвольным образом выбирается M классов
$n = 160$ – количество классов в каждой порции
$I = \varnothing$
for $k := 1$ to 6 **do**
    Выбираются классы $C_1^k, ..., C_n^k$ при помощи Algorithm 3 так, чтобы $\forall i \in [1, n]$: $C_i^k \notin I$
    $I = I \cup \{C_1^k, ..., C_n^k\}$
end for
for $t := 1$ to 50 **do**
    for $k := 1$ to 6 **do**
        Случайным образом выбираются пары изображений из классов $C_1^k, ..., C_n^k$
    end for
end for

Всего на каждую эпоху набирается 960 классов. В этом эксперименте, как и в первом, классы, выбранные на эпоху, будут разделены на 6 множеств. Из каждого класса в множестве будет выбрано 20 пар изображений. Таким образом, размер эпохи будет равен 120. Коэффициент скорости обучения равен 0,05, количество классов $M = 9071$. Обучение продолжалось 44 эпохи. Всего прошло 5280 итераций. Значение целевой функции уменьшилось с 5,1 до 4,6.

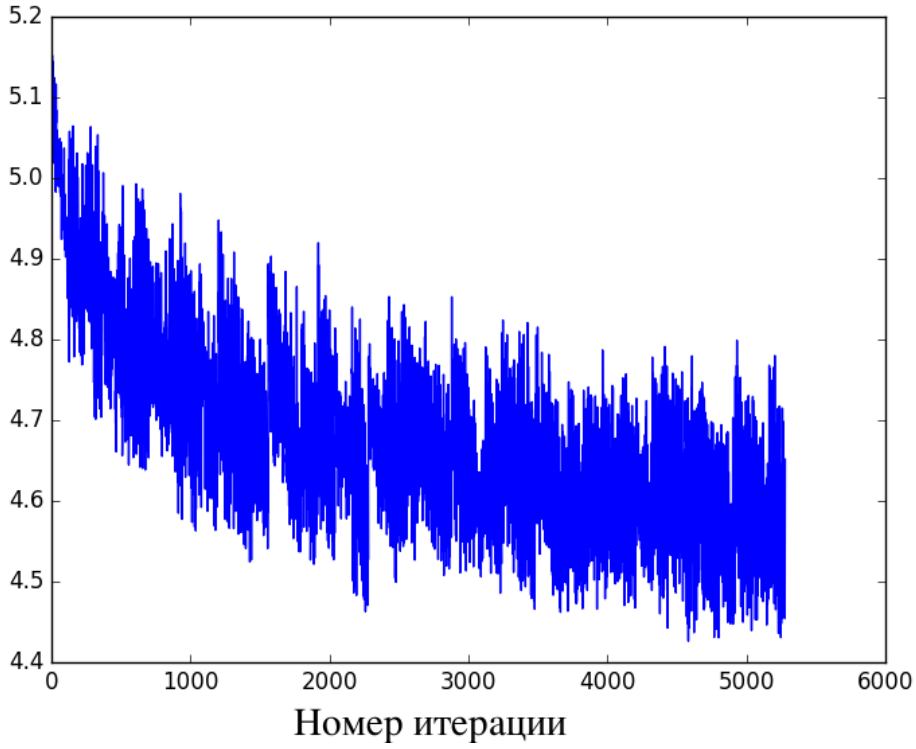

Рис. 9: график целевой функции в течение первых 44 эпох.

Максимальная точность $83,8\%$ достигнута на 43 эпохе. Коэффициент скорости обучения был уменьшен до 0,01. Остальные параметры остались без изменения. Обучение длилось 18 эпох по 120 итераций. На 9 эпохе была достигнута точность $85,5\%$. Начиная с 10 эпохи точность начала понижаться и к концу 18 эпохи понизилась до $81,6\%$. Значение целевой функции продолжало уменьшаться (с 4,6 до 4,55). В связи с этим значение коэффициента скорости обучения было уменьшено до 0,008. Итоговая точность равна $86,3\%$ и была достигнута через 840 итераций. Всего итераций в данном эксперименте – 7080.

3.4 Сравнение архитектур нейронной сети

1. Первый эксперимент был поставлен без использования полносвязного слоя. После каждого сверточного слоя применялась сначала нормализация активаций нейронов, и нелинейная функция активации ReLU. Для выбора классов на итерации использовался Algorithm 5, так как он показал лучший результат среди всех алгоритмов. Количество классов M равно 9071. На каждую эпоху выбиралось 960 классов. Из каждого класса по 20 различных пар изображений. В одной эпохе 120 итераций. Значение коэффициента скорости обучения – 0,01. Выход каждого сверточного слоя нормализовался (BN).

Таблица 3: Архитектура нейронной сети CasiaNet без полносвязного слоя

Тип слоя	Размер фильтра, шаг	Размер выхода	Количество параметров	Функция активации
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 32$	282	ReLU
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 64$	18 тыс.	ReLU
max pooling	$2 \times 2, 2$	$50 \times 50 \times 64$	0	–
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 64$	36 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 128$	72 тыс.	ReLU
max pooling	$2 \times 2, 2$	$25 \times 25 \times 128$	0	–
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 96$	108 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 192$	162 тыс.	ReLU
max pooling	$2 \times 2, 2$	$13 \times 13 \times 192$	0	–
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 128$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 256$	288 тыс.	ReLU
max pooling	$2 \times 2, 2$	$7 \times 7 \times 256$	0	–
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 160$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 320$	288 тыс.	ReLU
average pooling	$7 \times 7, 1$	$1 \times 1 \times 320$	0	–
L2-norm	–	320	0	–

Для обучения использовалась исходная целевая функция N-pair-mc loss без регуляризации весов свертоных слоев.

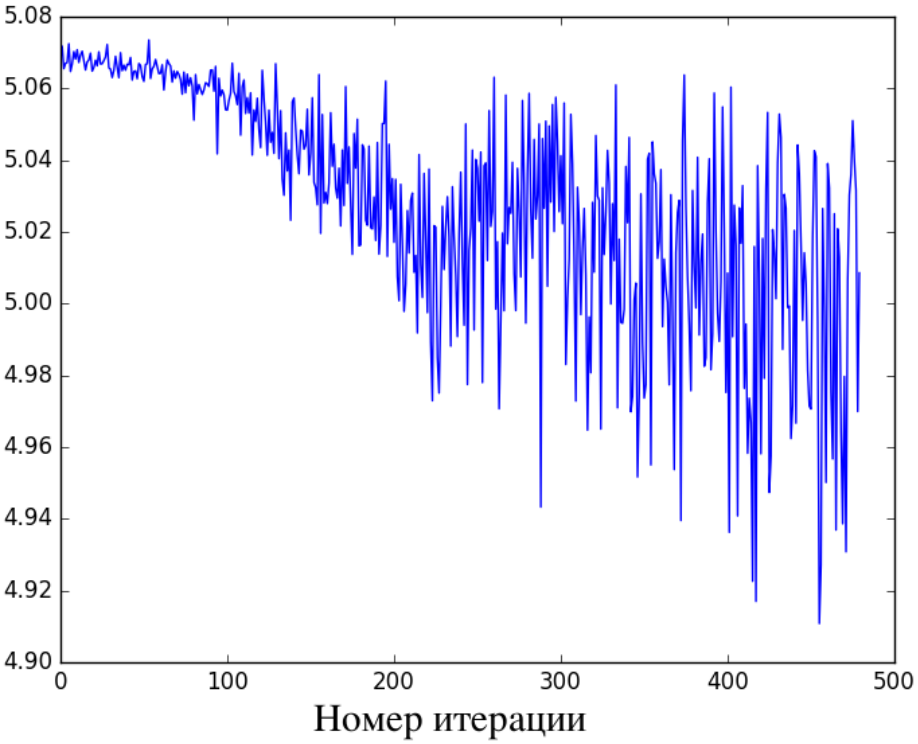

Рис. 10: график целевой функции в течение первых 4 эпох.

Как видно на Рис. 10, значение целевой функции меньшилось с 5,07 до ~ 5. Первые 200 итераций график плавно понижается, большая амплитуда на итерациях 300 – 480 может свидетельствовать о том, что значение скорости обучения слишком велико. Точность на валидационной выборк после 480 итераций равна $58,9\%$. Далее точность постепенно увеличивалась и достигла своего максимума, равного 84%, на 114 эпохе. Обучение продолжалось еще 57 эпох, но точность уже не превышала $83,7\%$.

2. Данный эксперимент был поставлен с добавлением полносвязного слоя после average pooling. После каждого слоя, в том числе и после полносвязного слоя, применялась нормализация активаций нейронов. Целевая функция вместе с N-pair-mc loss учитывала значение регуляризации весов полносвязного слоя. Коэффициент регуляризации равен 0,0005.

Таблица 4: Архитектура нейронной сети CasiaNet без полносвязного слоя

Тип слоя	Размер фильтра, шаг	Размер выхода	Количество параметров	Функция активации
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 32$	282	ReLU
convolution + BN	$3 \times 3, 1$	$100 \times 100 \times 64$	18 тыс.	ReLU
max pooling	$2 \times 2, 2$	$50 \times 50 \times 64$	0	–
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 64$	36 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$50 \times 50 \times 128$	72 тыс.	ReLU
max pooling	$2 \times 2, 2$	$25 \times 25 \times 128$	0	–
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 96$	108 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$25 \times 25 \times 192$	162 тыс.	ReLU
max pooling	$2 \times 2, 2$	$13 \times 13 \times 192$	0	–
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 128$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$13 \times 13 \times 256$	288 тыс.	ReLU
max pooling	$2 \times 2, 2$	$7 \times 7 \times 256$	0	–
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 160$	216 тыс.	ReLU
convolution + BN	$3 \times 3, 1$	$7 \times 7 \times 320$	288 тыс.	ReLU
average pooling	$7 \times 7, 1$	$1 \times 1 \times 320$	0	–
fully connected	320	320	102 тыс.	–
L2-norm	–	320	0	–

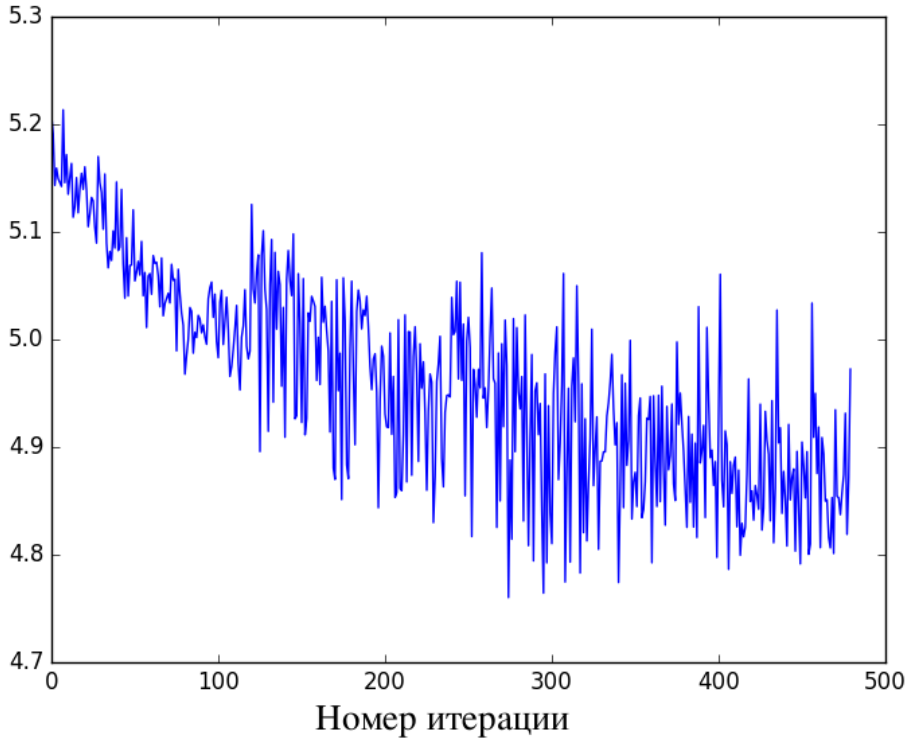

Рис. 11: график целевой функции в течение первых 4 эпох.

Значение регуляризации весов уменьшилось с $0,124$ до $0,018$. Максимальная точность на валидационной выборке была достигнута на 115 эпохе и равна $85,1\%$.Обучение длилось еще 56 эпох по 120 итераций, точность не превышала $84,9\%$.

Заключение

Была реализована и обучена глубокая сверточная нейронная сеть, основанная на архитектуре CasiaNet [14]. Были поставлены эксперименты, направленные на определение влияния архитектуры и параметров сети, алгоритма выбора изображений в процессе обучения на точность распознавания. Лучший результат – $88,3\%$ был получен при использовании Algorithm 5, в котором последовательно выбираются ближайшие классы на несколько итераций, и архитектуре нейронной сети CasiaNet с полносвязным слоем. После каждого сверточного слоя, а также после полносвязного слоя применялась нормализация активаций нейронов. В качестве функции активации использовалась функция ReLU. Обучение длилось 82860 итераций.

Список литературы

[1] C. A. Hansen, "Face Recognition", Institute for Computer Science University of Tromso, Norway.
[2] Kohonen, T. (1985). Self-Organizing Maps, Springer-Verlag, Berlin
[3] L. Wiskott, J.-M. Fellous, N. Krueuger, C. von der Malsburg, Face Recognition by Elastic Bunch Graph Matching, Chapter 11 in Intelligent Biometric Techniques in Fingerprint and Face Recognition, eds. L.C. Jain et al., CRC Press, 1999, pp. 355-396
[4] $http://www.image-net.org$
[5] Schroff F., Kalenichenko D., Philbin J. Facenet: A unified embedding for face recognition and clustering //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2015. – C. 815-823.
[6] Cheng D. et al. Person re-identification by multi-channel parts-based cnn with improved triplet loss function //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2016. – C. 1335-1344.
[7] Wen Y. et al. A discriminative feature learning approach for deep face recognition //European Conference on Computer Vision. – Springer International Publishing, 2016. – C. 499-515.
[8] Sohn K. Improved deep metric learning with multi-class n-pair loss objective //Advances in Neural Information Processing Systems. – 2016. – C. 1849-1857.
[9] $https://github.com/davidsandberg/facenet/blob/master/src/align/align\_dataset\_$
[10] $http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html$
[11] $https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/$
[12] $http://vis-www.cs.umass.edu/lfw/$
[13] Kingma D., Ba J. Adam: A method for stochastic optimization //arXiv preprint arXiv:1412.6980. – 2014.
[14] Yi D. et al. Learning face representation from scratch //arXiv preprint arXiv:1411.7923. – 2014.
[15] Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift //arXiv preprint arXiv:1502.03167. – 2015. [16] Glorot X., Bengio Y. Understanding the difficulty of training deep feedforward neural networks //Aistats. – 2010. – T. 9. – C. 249-256.