Заключительный научно технический отчет

о выполнении НИР

по теме

Разработка алгоритмов прогнозирования индивидуального поведения на основе визуального распознавания эмоций

договор №11320ГУ/2017 (код 0033562)

от 14.04.2017 , этап 2 , УМНИК 16-12

Исполнитель: Ивановский Леонид Игоревич

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Эмоции – выразительные движения мышц лица, являющиеся одной из форм проявления тех или иных чувств человека. Выражение лица помогает определить не только физическое, но и психологическое состояние людей. Благодаря эмоциям, можно определить те или иные психологические проблемы человека, предугадать его дальнейшие действия, определить болезни и даже следить за динамикой реабилитации. В эпоху повсеместной роботизации автоматическое распознавание эмоций является важным компонентом для создания систем взаимодействия человека и компьютера.

Автоматическое распознавание выражений лица – одна из передовых областей информатики. На сегодняшний день системы, анализирующие эмоции, используются в ряде коммерческих решений для мониторинга эффективности маркетинговых и рекламных компаний, а также для оценки работы персонала при общении с клиентом [1]. Отдельным направлением в области анализа выражения лица человека является детектирование улыбки, находящее свое применение в системе Smile To Pay, в программном обеспечении фото- и видеокамер, а также в приложениях для выбора лучших снимков [2].

Несмотря на то, что человек способен определять эмоции людей вне зависимости от их пола, возраста или расовой принадлежности, для компьютера распознавание выражений лица является трудной задачей. Её сложность определяется разнообразием лиц, наличием оптических препятствий, а также различной мимикой людей.

В системах автоматического распознавания эмоций, в общем случае реализуются следующие этапы работы:

1. Захват изображений лица
2. Предварительная обработка изображений: снижение помех, фильтрация, повышение четкости.

3. Извлечение оптимального набора признаков. Данная стадия является ключевым этапом, поскольку отобранные признаки должны минимизировать внутриклассовые отличия и максимизировать дисперсию между классами.
4. Классификация каждого лица на заданное число классов эмоций.
5. Вывод или сохранение результатов классификации.

Многие из этих этапов, такие как доработка алгоритмов нормализации изображения, определения лица и выделения на нем особых точек, а также доработка алгоритма детектирования улыбки уже были реализованы на первом этапе НИР. Данный научно-технический задел послужил опорой для выполнения второго этапа НИР. Здесь, для построения высокоэффективного алгоритма распознавания эмоций требовалось решить следующие задачи:

- Разработка алгоритма компьютерного определения эмоций человека по изображениям.
- Обучение классификации на основе свёрточных нейронных сетей и классических алгоритмов машинного обучения.
- Проведение тестирования алгоритмов.
- Апробация на пилотных объектах.

1. РАЗРАБОТКА АЛГОРИТМА КОМПЬЮТЕРНОГО ОПРЕДЕЛЕНИЯ ЭМОЦИЙ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЯМ

Автоматический анализ выражения лица требует применения новых эффективных подходов. Одним из таких подходов является использование алгоритмов глубокого машинного обучения – сверточных нейронных сетей, позволяющих полностью отойти от этапа формирования признаков, поскольку при этом подходе признаки формируются алгоритмом самостоятельно в процессе обучения модели. Автоматически сгенерированные алгоритмом дескрипторы дают наилучшие результаты для многих задач классификации [3].

Прогресс в области создания высокопроизводительных компьютеров позволил исследователям работать со сверточными нейронными сетями, которые имеют миллионы параметров. Если быть точным, сверточные нейронные сети стали повсеместными в области компьютерного зрения с тех пор, как архитектура глубокой сети AlexNet [4] выиграла в конкурсе ImageNet Challenge: ILSVRC 2012 [5]. В последние несколько лет глубокие нейронные сети показали выдающиеся результаты в задачах обнаружения и классификации объектов на изображениях из-за их относительно высокой точности и скорости работы. При решении современных задач компьютерного зрения подходы, основанные на использовании сверточных нейронных сетей, превосходят не только классические методы, но и в некоторых случаях даже экспертов в соответствующих областях. Поэтому в настоящее время почти все новые задачи компьютерного зрения пытаются решить именно с помощью сверточных нейронных сетей [6].

В статье [7] приведены различные типы глубокой сверточной архитектуры нейронной сети. Эти типы архитектуры позволяют решать проблемы распознавания пола и прогнозирования возраста человека по изображению его лица. В работе [8] описано использование сверточной нейронной сети в задаче классификации объектов разных типов в

изображении. В статье [9] описывается двухфазная гибридная система, в которой соединены воедино сверточная нейронная сеть и логистический классификатор. На первом этапе сеть распознает изображения лиц. На втором этапе логистический классификатор используется для классификации картинок, полученных на предыдущем этапе. Доля правильных ответов такой модели составила 86,06%. В статье [10] также представлен двухфазный классификатор, чтобы классифицировать изображение лица человека в соответствии с одним из семи типов эмоций. Сверточная нейронная сеть используется для классификации нейтральных типов эмоций, а гауссова модель применяется для классификации оставшихся типов эмоций. Доля правильных ответов такой модели 71%. В статье [11] описывается алгоритм для задачи распознавания лиц. Сначала из входного изображения извлекаются значения признаков, а затем с помощью гауссовой модели и сверточной нейронной сети извлекается необходимая информация. Доля правильных ответов такой модели составила около 85%.

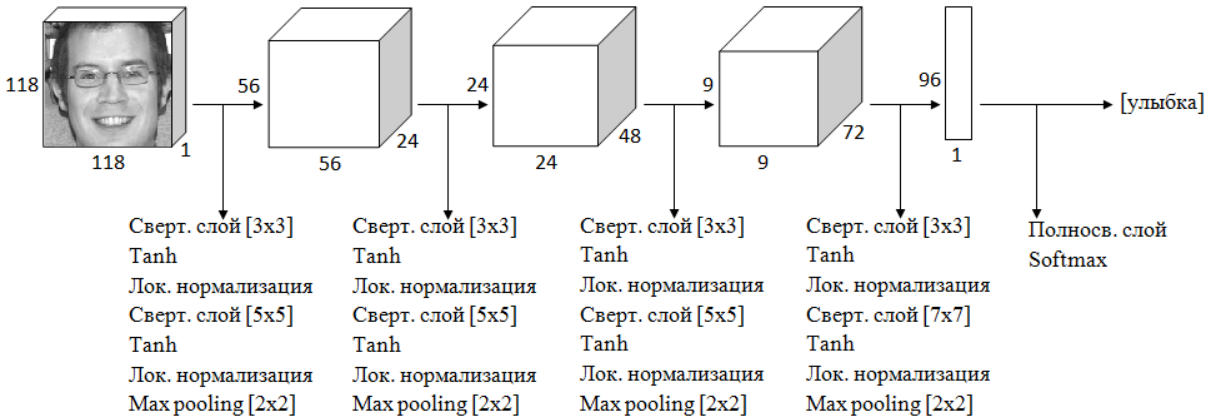


Рисунок 1 – Архитектура сверточной нейронной сети для задачи классификации эмоций.

В данном разделе представлена реализация сверточной нейронной сети для решения задачи классификации лиц человека согласно различным эмоциям. В основе реализации нейронной сети лежала модель, реализованная на первом этапе НИР для автоматического детектирования улыбки на лицах людей. Архитектура сверточной нейронной сети для решения задачи классификации эмоций показана на рисунке 1. Как видно

из этой схемы, в разработанной модели использовались операции свертки, слои локальной нормализации и понижения дискретизации с помощью операции maxpooling, функции активации, а также один полносвязный слой и обобщенная логистическая функция Softmax для предсказания одного из шести классов эмоционального состояния человека.

Сверточные слои являются самыми важными строительными блоками глубоких нейронных сетей. При работе этого слоя набор признаков I, поступивших на вход с предыдущего слоя с помощью матричного умножения на различные окна свертки K фиксированного размера, преобразуется в другой, преобразованный набор признаков $I \times K$. Другими словами, происходит взвешенное суммирование значений интенсивности признаков по мере прохождения окна по всем данным. Тем самым, сверточные слои на самом деле применяют взаимную корреляцию, похожую на математическую операцию свертки [12]. Схема работы сверточного слоя показана на рисунке 2.

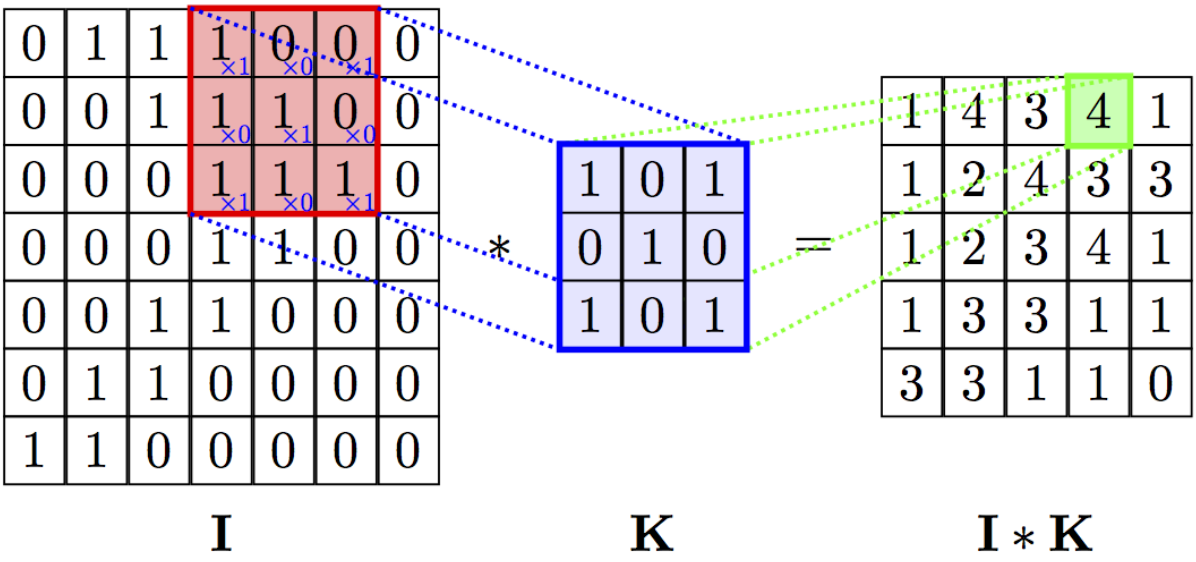


Рисунок 2 – Схема работы сверточного слоя.

Функция активации, примененная к набору признаков, способствует уменьшению значения ошибки при обучении сети. Этой слой позволяет применить к каждому элементу матрицы, поступившей на вход с предыдущего слоя, некоторую нелинейную функцию, активируя тем самым определенные признаки [13]. Для задачи распознавания эмоций был

выбран гиперболического тангенс $y = tanh(x)$, который часто используется при решении задач компьютерного зрения. График этой функции изображен на рисунке 3.

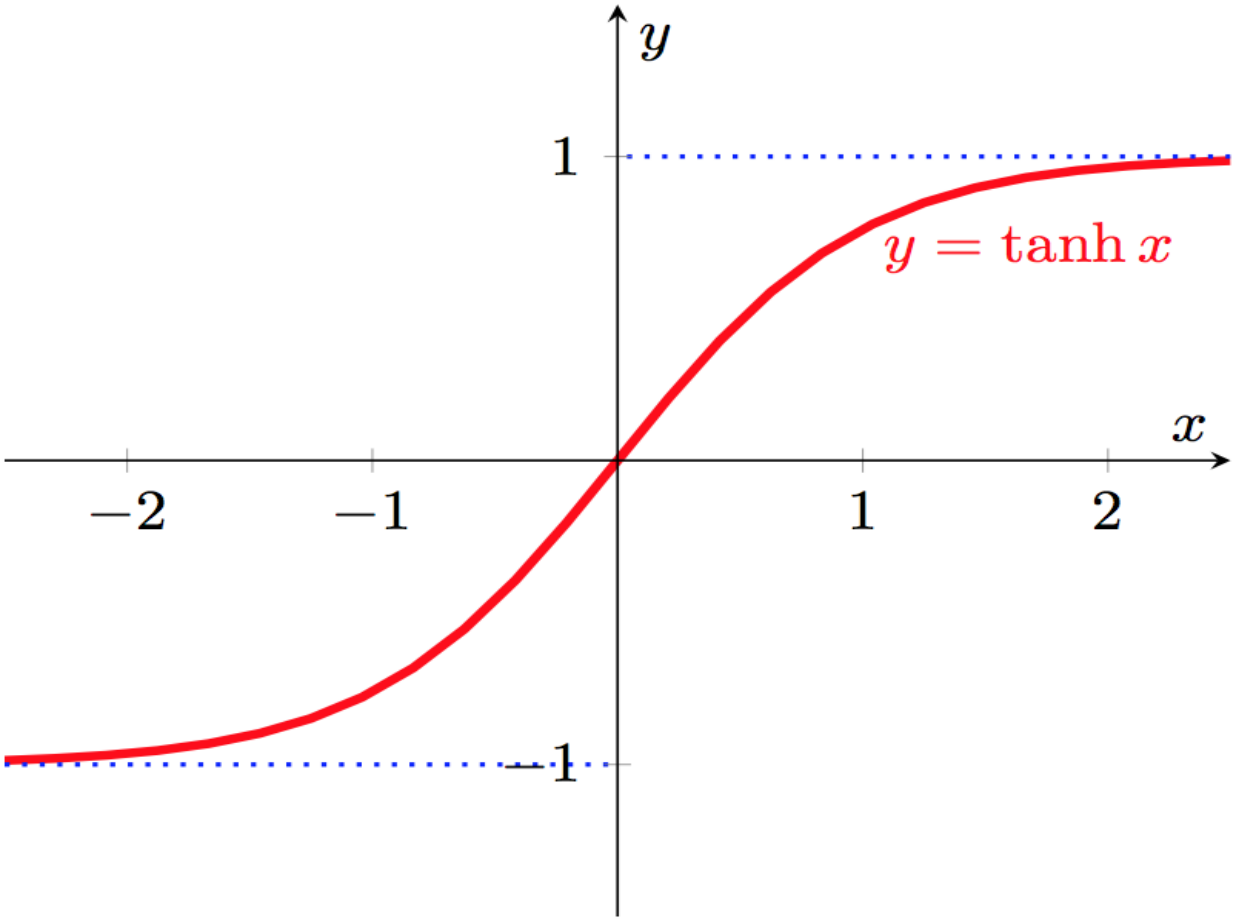


Рисунок 3 – График функции активации $y = tanh(x)$.

Слой локальной нормализации признаков так же, как и функция активации, позволяет уменьшить значение ошибки при обучении сети. Однако в отличие от функции активации, этот слой, реализует процесс нормализации матрицы, поступившей на вход, снижая тем самым достаточно высокие значения признаков – выбросы, поступившие на вход алгоритму машинного обучения.

Слой понижения дискретизации является уплотнением карты признаков, поступившей на вход с предыдущего слоя. Сканируя скользящим окном непересекающиеся участки матрицы, группа признаков уменьшается в несколько раз, позволяя тем самым уменьшить количество признаков, теряя при этом минимум информации. Такое

перемасштабирование и выбор определенных свернутых выходов помогает исключить возможность переобучения [12]. Операция max pooling характеризуется тем, что из набора данных выбирается значение признака с максимальной величиной. Схема работы операции maxpooling показана на рисунке 4.

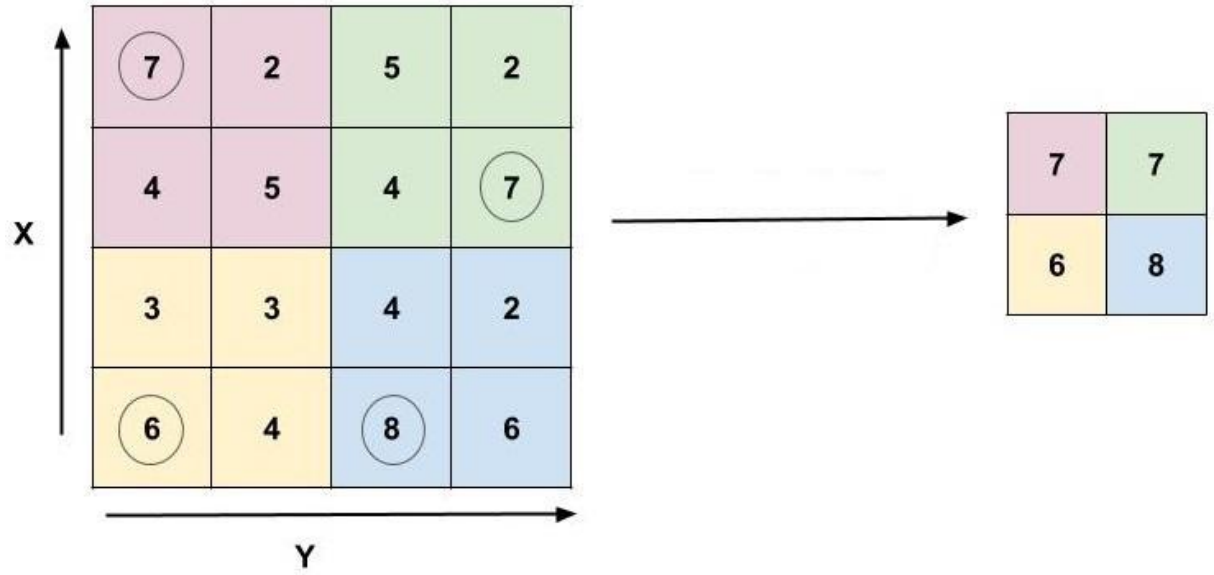


Рисунок 4 – Схема работы операции max pooling.

Полносвязный слой представляет собой простой двухслойный персептрон – нейронную сеть без скрытых слоев, на вход которой подается некоторый набор признаков, а уже на выходе формируется вектор, с заданным количеством элементов [14]. Схема работы полносвязного слоя показана на рисунке 5. При разработке алгоритмов глубокого обучения полносвязный слой обычно применяется в конце нейронной сети, непосредственно перед этапом классификации.

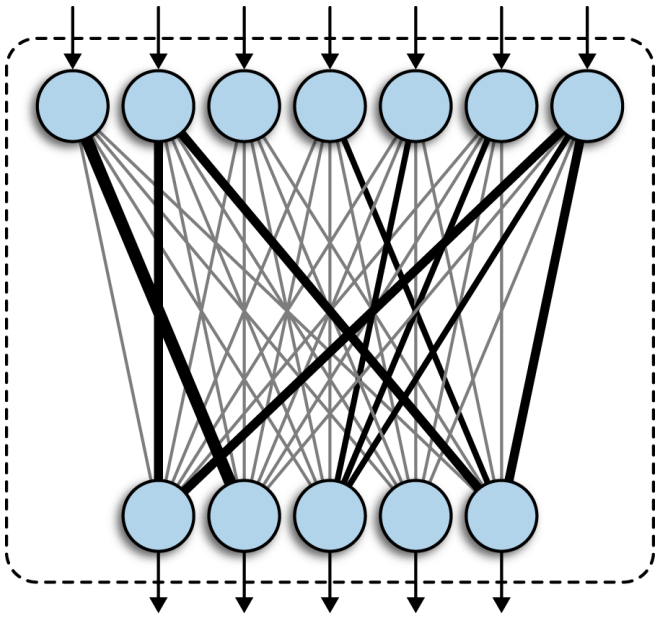


Рисунок 5 – Схема работы полносвязнного слоя.

Обобщенная логистическая функция Softmax преобразует один вектор данных в массив того же размера. К особенностям выходного вектора можно отнести то, что каждая его координата является вещественным числом из интервала $[0, 1]$, а сумма всех полученных элементов равна 1. Таким образом, в контексте алгоритмов глубокого обучения, входным массивом данных обычно является карта признаков некоторого изображения, а выходной последовательностью – значения вероятностей принадлежности к тому или иному классу. Благодаря такому вектору значений вероятностей исходное изображение сопоставляется с соответствующим выражением, обнаруженным на лице человека. Другими словами, с помощью обобщенной логистической функции, нейронная сеть классифицирует входные данные. Значение обобщенной логистической функции Softmax вычисляется по формуле

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}},$$

где x_i – значение i –ого признака n –мерного вектора, поданного с предыдущего слоя на вход обобщенной логистической функции, а $\sigma(x_i)$ – значения вероятности, которые получились после применения функции Softmax для признака x_i. В машинном обучении обобщенная

логистическая функция обычно применяется для последнего слоя сверточной нейронной сети [13].

Нейронная сеть для задачи распознавания эмоций состоит из восьми сверточных слоев (четыре со сверточным окном размера 3×3, три – 5×5 и один – 7×7), восьми слоев локальной нормализации, восьми слоев с функцией активации $y = tanh(x)$, четырех слоев понижения дискретизации с помощью операции maxpooling (с окном размера 2×2 и шагом 2), одного полносвязнного слоя и слоя, реализующего обобщенную логистическую функцию Softmax. По сравнению с архитектурой сети для решения задачи детектирования улыбок, реализованной на первом этапе НИР, разработанный алгоритм машинного обучения отличается тем, что для него до применения слоя понижения дискретизации дополнительно применялась одна операция свертки с бóльшим размером окна, одна функция активации и одна операция локальной нормализации. Таким образом, было вдвое увеличено количество сверточных слоев, слоев понижения дискретизации с помощью операции maxpooling, а также функций активации и локальной нормализации. Ко всему прочему, были изменены такие параметры сверточной нейронной сети, как размер изображения, поступающего на вход модели (со 122×122 на 118×118), количество нейронов в полносвязном слое (с 80 на 96), глубина карты признаков на скрытых слоях модели и размеры окон свертки. Значения всех этих параметров были получены в ходе эмпирического исследования качества работы классификатора.

Реализация архитектуры сети осуществлялась с помощью фреймворка Caffe. Эта кросс-платформенная библиотека позволяет достаточно просто, описать сверточную нейронную сеть и параметры для ее обучения в файлах формата prototxt, схожих по своей структуре с файлами формата json. Библиотека Caffe содержит уже готовые реализации различных слоев сверточной нейронной сети. Главными достоинствами этой библиотеки является простота разработки моделей глубокого машинного обучения, а

также то, что полученными результатами, можно легко воспользоваться, встроив их в проект, написанный на C++ или Python. На сегодняшний день, фреймворк Caffe активно применяется для решения практических задач машинного обучения [15].

2. ОБУЧЕНИЕ КЛАССИФИКАЦИИ НА ОСНОВЕ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ И КЛАССИЧЕСКИХ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Подход, основанный на использовании сверточных нейронных сетей, требует немалых вычислительных ресурсов. Для ускорения работы нейронных сетей, этапы обучения и тестирования выполнялись на большом количестве независимых потоков графического процессора. Причина такого выбора связана с тем, что архитектура графического процессора хорошо приспособлена для задач, в которых требуется запуск большого количество потоков, в каждом из которых производятся несложные однотипные операции. Среди технологий параллельных вычислений на графическом процессоре была выбрана технология CUDA от компании NVIDIA. Эта технология является кросс-платформенной и поддерживается всеми современными видеокартами NVIDIA [16].

Помимо файла с описанием структуры сверточной нейронной сети, для обучения разработанной на Caffe модели необходим также файл со значениями параметров обучения.

В качестве алгоритма численной оптимизации была выбрана адаптивная оценка моментов Adam. Она объединяет в себе идеи моментной оптимизации и регулирования скорости обучения. В отличие от стохастического градиентного спуска, упрощенного метода последовательного поиска локального экстремума функции с помощью движения вдоль вектора градиента, который применялся при обучении алгоритма детектирования улыбки, на каждом шаге изменения весов Adam с помощью вычисляемых величин импульсов, подстраивает скорость обучения алгоритма, учитывая изменения градиента на последних шагах. Другими словами, Adam использует значения моментов градиентов для регулирования скорости обучения, что повышает вероятность найти глобальный экстремум функции, а значит и повысить точность работы

алгоритма машинного обучения. Adam считается устойчивой к выбору гиперпараметров, поэтому скорость обучения модели необходимо редактировать лишь в крайних случаях, только при плохом обобщении на определенных наборах данных [17]. Согласно последним достижениям в области машинного обучения, именно Adam показывает наилучшую и самую эффективную и быструю сходимость при решении задач компьютерного зрения.

Перед тем, как картинка из выборки поступала на вход сверточной нейронной сети, к ней применялась операция crop: из изображения случайным образом вырезался участок размера 118×118. Далее, эта картинка могла быть зеркально отражена относительно своей центральной вертикальной оси. Подобного рода преобразования над изображениями применялись с целью повышения робастности алгоритма машинного обучения, а также для повышения вариативности данных, поступающих на вход классифицирующей модели.

В качестве функции потерь была выбрана категориальная кросс-энтропия. В области машинного обучения такая функция потерь характеризует собой стоимость неправильно принятых решений для задачи многоклассовой классификации. На каждой итерации обучения модель обновляла свои веса после прогона через сеть 32 образцов из обучающей выборки. Классификатор заканчивал свое обучение после совершения 70000 итераций.

Для выполнения численного эксперимента была подготовлена выборка из 210000 случайно отобранных снимков (по 35000 изображений для каждого из 6 типов выражений лиц) из базы данных Multi-PIE [18]. Эта база содержит более 750000 изображений 337 различных людей. Изображения представляют собой снимки, сделанные на фотокамеру, под углом обзора, не превышающим 90°, и с разной степенью освещенности сцены. База Multi-Pie использовалась и при обучении алгоритма детектирования улыбок. Выбор был сделан именно в пользу базы Multi-

Pie, поскольку именно в ней для каждого из типов эмоций содержится большое количество снимков. Примеры изображений из базы данных Multi-Pie показаны на рисунке 6.

Рисунок 6 – Примеры изображений из базы данных Multi-Pie.

Для обучения и тестирования разработанного алгоритма машинного обучения, были выбраны картинки с разной степенью освещенности сцены и с углом обзора камеры, не превышающим 45°. Эти изображения были помечены в соответствии с одним из 6 типов выражений лиц, представленных в наборе снимков из Multi-Pie: спокойствие, улыбка, удивление, заинтересованность, отвращение, спокойствие и крик. Примеры снимков различных классов из базы данных Multi-Pie представлены на рис. 7.

а) Спокойствие б) Улыбка в) Удивление

г) Заинтересованность д) Отвращение е) Крик

Рисунок 7 – Типы выражений лиц, представленные в базе изображений Multi-Pie

Разметка выборки хранилась в текстовом файле. Над каждым элементом выборки осуществлялись следующие преобразования: из картинки вырезался участок размером 128x128 с изображением лица и переводился в черно-белый формат. Такое преобразование осуществлялось с помощью модификации классического алгоритма Виолы-Джонса – вычислительно эффективного алгоритма PICO для детектирования лиц, который был разработан на первом этапе выполнения НИР. В отличие от метода Виолы-Джонса, PICO-алгоритм, не требует масштабирования признаков. Другими словами, нет никакой необходимости в предварительной обработке изображения (например, редактирования контрастности картинки или изменения ее размеров). PICO-алгоритм является более быстрым и менее чувствительным к шуму в данных [19].

Полученное множество картинок в свою очередь разбивалось на тренировочный и тестовый наборы данных в соотношении 80/20. Таким образом, обучающий набор данных содержал 168000 примеров, а тестовый – 42000. Тренировочный и тестовый наборы данных не содержали одинаковых изображений. Более того, для чистоты эксперимента, снимки с одним и тем же человеком не находились одновременно в тренировочном и тестовом наборе данных. Все данные были размечены в соответствии с классом, которому они принадлежали (0 – улыбка, 1 – удивление, 2 – отвращение, 3 – заинтересованность, 4 – крик, 5 – спокойствие). Разметка изображений из тренировочной и тестовой выборки хранилась в текстовых

файлах формата txt. Сгенерированные таким образом данные использовались для процессов обучения и тестирования модели.

Процесс обучения алгоритма распознавания эмоций осуществлялся на суперкомпьютере NVIDIA DGX-1 [20]. Обучение на нем длилось около 50 минут.

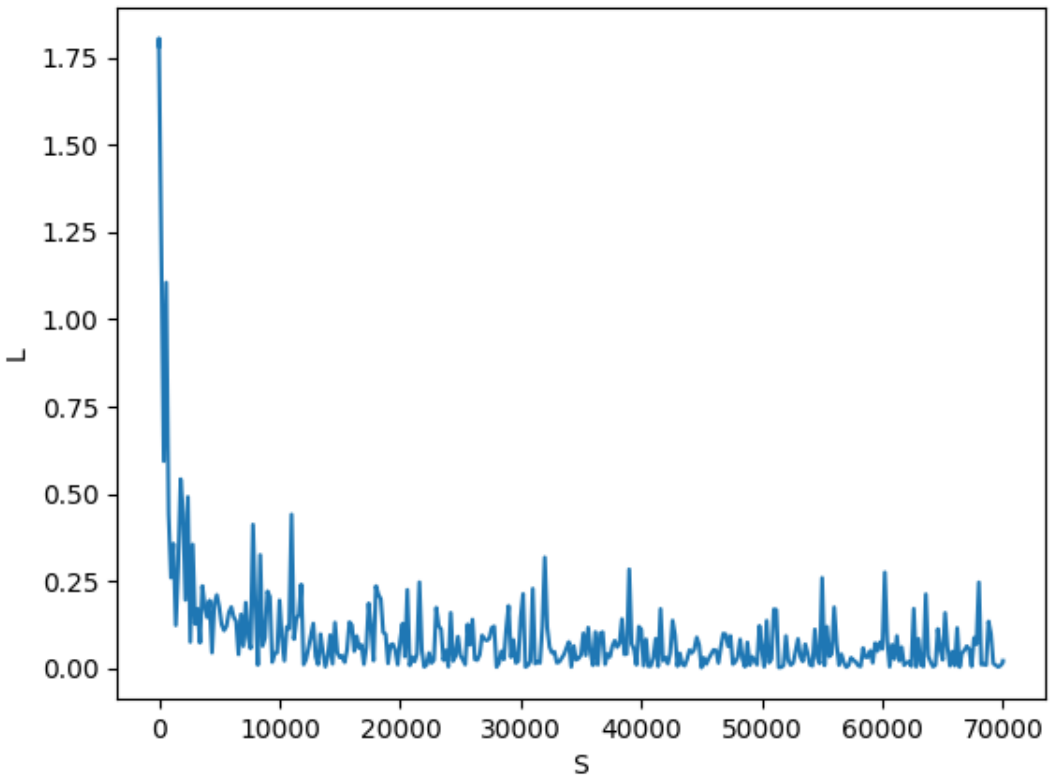


Рисунок 8 – Зависимость функции потерь от количества проделанных итераций обучения.

Как видно из рисунка 8, с ростом числа проделанных итераций (S) обучения, значение функции потерь (L) убывает. Это позволяет сделать вывод о том, что алгоритм сходится, т.е. он устойчив. О хорошем качестве разработанной модели говорит и тот факт, что величина функции потерь незначительна: ее значение колеблется от 0.03 до 0.26.

3. ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ АЛГОРИТМОВ

Как уже упоминалось в предыдущем разделе, тестирование алгоритма распознавания эмоций осуществлялось параллельно, на большом числе независимых потоков графического процессора суперкомпьютера NVIDIA DGX-1. Процесс тестирования в среднем занимал порядка 9 минут.

Для оценки работы качества сверточной нейронной сети на тестовом множестве из 42000 картинок, выбранных из базы данных Multi-Pie, использовалось несколько метрик расчета качества. Одна из них – доля правильных ответов классификатора (A). Эта величина представляет общую информацию о том, сколько образцов оказалось классифицировано правильно. Доля правильных ответов классификатора рассчитывается по формуле:

$$A = \frac{P}{N},$$

где P – количество изображений, по которым классификатор принял верное решение, а N – размер выборки [21]. Для задачи распознавания эмоций доля правильных ответов классификатора составила 94,48%.

На рисунке 9 представлена графическая интерпретация результатов тестирования усложненной модели сверточной нейронной сети. Как видно из него, с ростом числа проделанных итераций обучения значение доли правильных ответов достигало величины приблизительно равной 95%, а дальше существенно не менялось.

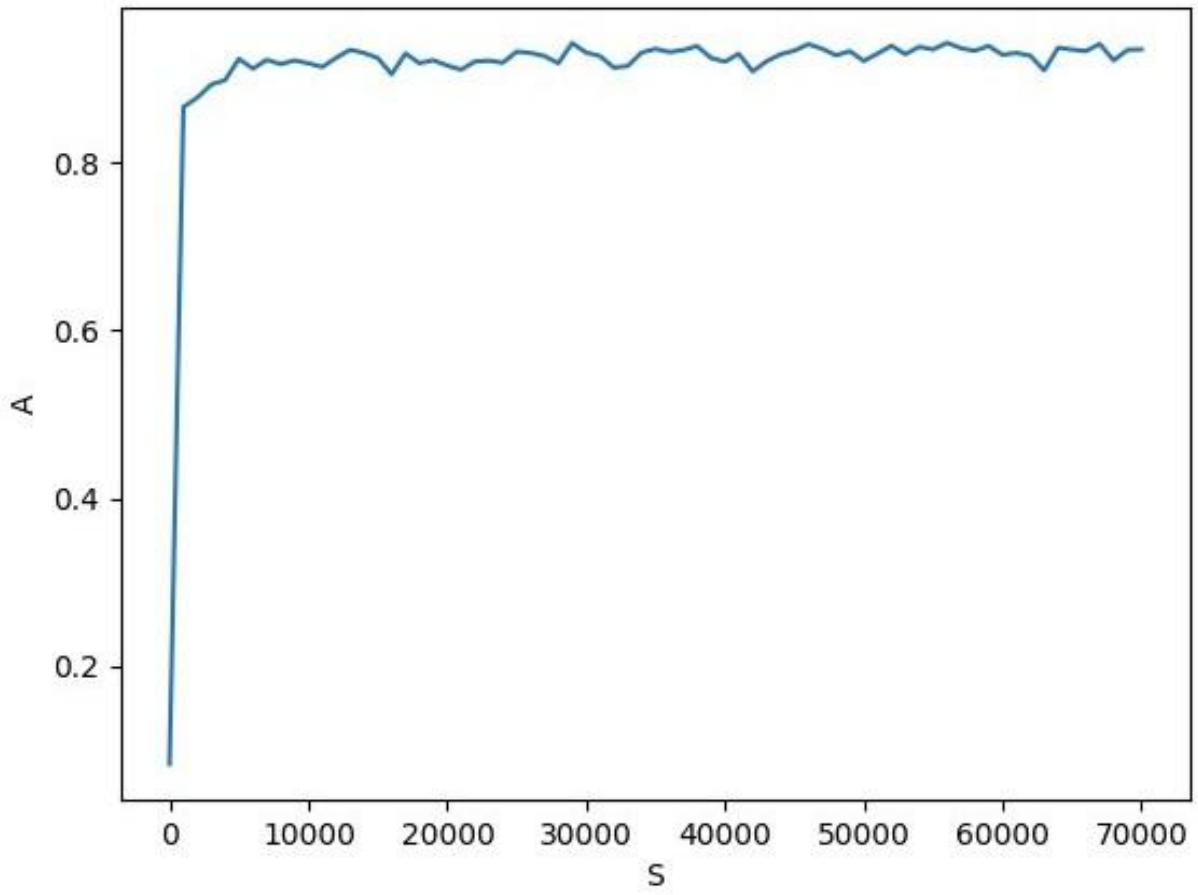


Рисунок 9 – Зависимость величины доли правильных ответов классификатора от количества проделанных итераций обучения для тестовой выборки

Вторая метрика, которая дает более детальную оценку эффективности классификатора – это матрица неточностей. Она позволяет не только оценить качество работы алгоритма машинного обучения, и то, как часто модель путает классы при попытке классификации данных, но и определить ошибки классификации первого и второго рода, количество ложно-положительных (FP) и ложно-отрицательных срабатываний (FN) классификатора соответственно [22]. Эти величины рассчитывались по следующим формулам:

$$FP_c = \sum_{i=1, i \neq c}^{n} A_{i,c}, \qquad FN_c = \sum_{i=1, i \neq c}^{n} A_{c,i}$$

где $A_{i,j}$ является элементом матрицы неточности размера $n \times n$, c – некоторый класс. Как видно из величин FP и FN, величины, которые расположены на главной диагонали матрицы неточностей, обозначают те случаи, когда для картинок из тестовой выборки класс был предсказан правильно, а все величины, что лежат вне главной диагонали матрицы –

неверные срабатывания алгоритма. В области машинного обучения, в зависимости от задачи классификации, разработчики могут сделать упор в сторону уменьшения значений ошибки первого или второго рода. В случае задачи классификации эмоций они имеют совершенно одинаковую цену.

Таблица 2

Матрица неточностей

Категории		**Фактический класс**					
		Улыбка	**Удив-ление**	**Отвра-щение**	**Заинте-ресо-ванность**	**Крик**	**Спокой-ствие**
Предсказанный класс	**Улыбка**	6526	37	53	6	1	145
	Удив-ление	140	6767	2	0	16	8
	Отвра-щение	72	2	6322	334	33	43
	Заинте-ресо-ванность	148	2	541	6479	9	166
	Крик	21	156	18	7	6940	1
	Спокой-ствие	93	36	64	174	1	6637

Из таблицы 2 видно, что разработанный алгоритм показал достаточно высокие результаты: количество правильно детектированных эмоций для каждого класса из выборки достаточно велико. Однако стоит отметить тот факт, что модель чуть хуже детектировала эмоции «отвращение» и «заинетресованность». Лучше всех, разработанный алгоритм детектировал эмоции класса «крик». Как видно из результатов, представленных в таблице 2 только 60 картинок из 7000 были классифицированы неверно, что составляет менее 1% от числа всех объектов данного класса, представленных в тестовой выборке.

Хорошие результаты работы классифицирующей модели подтверждаются также и третьим набором метрик по подсчету качества

работы алгоритмов машинного обучения: точность модели (P), ее полнота (R) и значение F-меры (F). Значения этих метрик качества вычислялись по следующим формулам:

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}, \qquad F = 2\frac{P \times R}{P + R}.$$

По своему смыслу величина точности классификатора (P) отражает в себе меру вероятности того, что положительный прогноз окажется правильным. Значение полноты (R) является мерой частоты истинно-положительных предсказаний. F-мера представляет собой среднее гармоническое значений точности и полноты классификатора и фактически является величиной, объединяющей в себе оба значения, точности и полноты классификатора [23]. Значения этих метрик оценки качества алгоритма определения выражения лица для каждого класса эмоций представлены в таблице 3.

Таблица 3

Анализ ошибок классификатора для задачи детектирования улыбок

Категории		Метрики		
		Точность (P)	**Полнота (R)**	**F-мера (F)**
Классы	**Улыбка**	0,96	0,93	0,95
	Удивление	0,98	0,97	0,97
	Отвращение	0,93	0,9	0,92
	Заинтересованность	0,88	0,93	0,9
	Крик	0,97	0,99	0,98
	Спокойствие	0,95	0,95	0,95

Как видно из таблицы 3 величина F-меры для каждого класса составляет как минимум 0,9, что является показателем отличной работы алгоритма. Для четырех классов из 6 величина F-меры достигла значения минимум в 0.95, тогда как для классов «крик» и «удивление», значения этих метрик оказались близки к 1. Хуже всего разработанной модели удавалось детектировать эмоции типа «отвращение» и «заинтересованность»: для них значения F-меры составили 0,92 и 0,9

соответственно. Этот факт объясняется тем, что в базе данных Multi-Pie снимки с этими классами были труднее всего различимы друг от друга. При выражении этих типов эмоций, очень часто люди имеют схожий взгляд, а отличия заметны лишь в области лба и в уголках рта. Примеры трудноразличимых снимков с эмоциями типа «отвращение» и «заинтересованность» из тестовой выборки представлены на рисунке 10.

а) Отвращение б) Заинтересованность

Рисунок 10 – Пример наиболее трудно различимых картинок с эмоциями

4. АПРОБАЦИЯ НА ПИЛОТНЫХ ОБЪЕКТАХ

После обучения и тестирования алгоритма распознавания эмоций на изображении лица человека, требовалось провести апробацию разработанной модели. Для этого было решено использовать следующее аппаратное обеспечение:

- Веб-камера Logitech B525, которая позволяет снимать фото- и видеопоследовательности в формате Full HD 720p (до 1280×720) с частотой кадров до 30 Гц. Преимуществами Web-камеры являются высокое качество съемки, автофокусировка, легкость, компактная складная конструкция и поле обзора по диагонали в 69°. Ее главными преимуществами, вследствие чего был сделан именно в пользу этого устройства, являются вращение на 360°, что позволяет настроить ее на любое пространство в помещении, соединение при помощи USB-кабеля, а также совместимость с операционными системами Linux и Windows (от 7 и выше).

Рисунок 11 – Веб-камера Logitech B525.

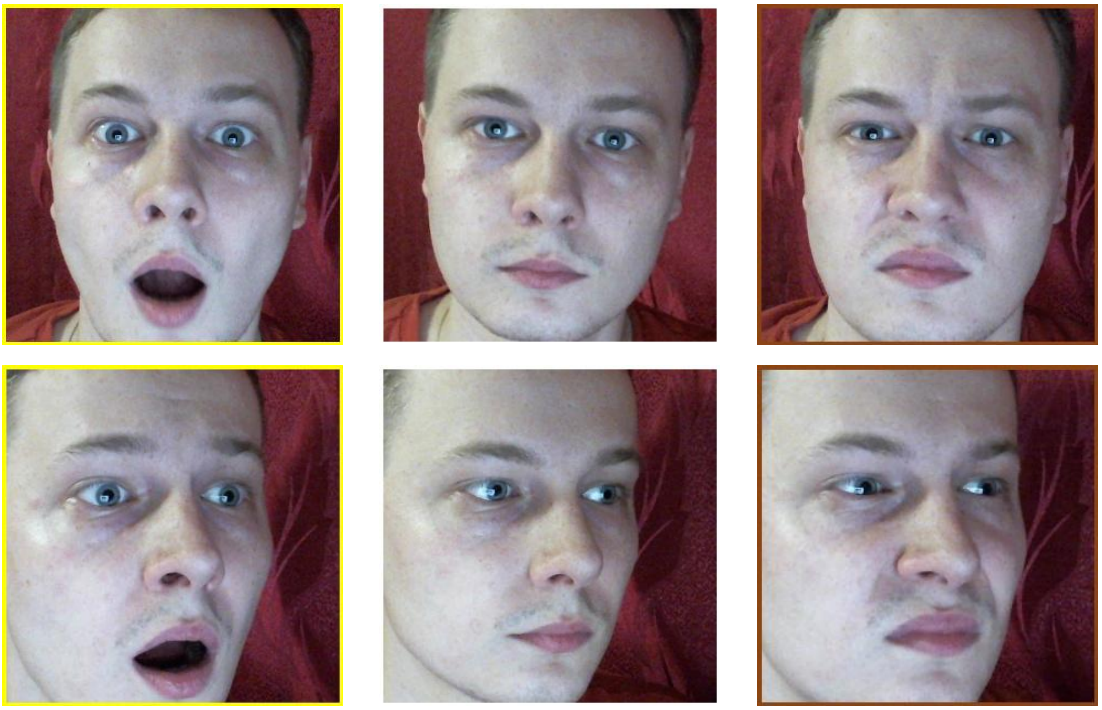
- ПК с предустановленной операционной системой Windows 7, который имеет следующие технические характеристики: процессор Intel Core i5 3 поколения с тактовой частотой 3.2 ГГц, оперативная память 8Гб.

Изображения лица извлекались из видеопоследовательности, снятой веб-камерой в режиме реального времени. Этот процесс осуществлялся с помощью специально разработанного приложения, написанного на языке Python с использованием библиотеки компьютерного зрения OpenCV. В нем использовались алгоритмы, которые были реализованы на первом этапе выполнения НИР. Подготовленные кадры с лицами поступали на вход разработанной модели, а после результаты детектирования эмоций сохранялись. Эта информация может использоваться для определения эмоционального состояния человека в тот или иной момент времени. Результаты собранной статистики могут использовать в своих целях:

- рекламные агентства – для оценки эффективности рекламы, мониторинга ее воздействия и выбора стратегий продвижения товаров и услуг,
- отделы развития бизнеса – для оценки качества предоставленных услуг, возможного изменения предоставления услуг клиентам, увеличения объема продаж,
- кадровые агентства – для оценки качества работы персонала и помощи при ассесмент-исследованиях,
- фонды социальных исследований – для сбора статистика в масштабах ТЦ, в местах массового скопления людей.

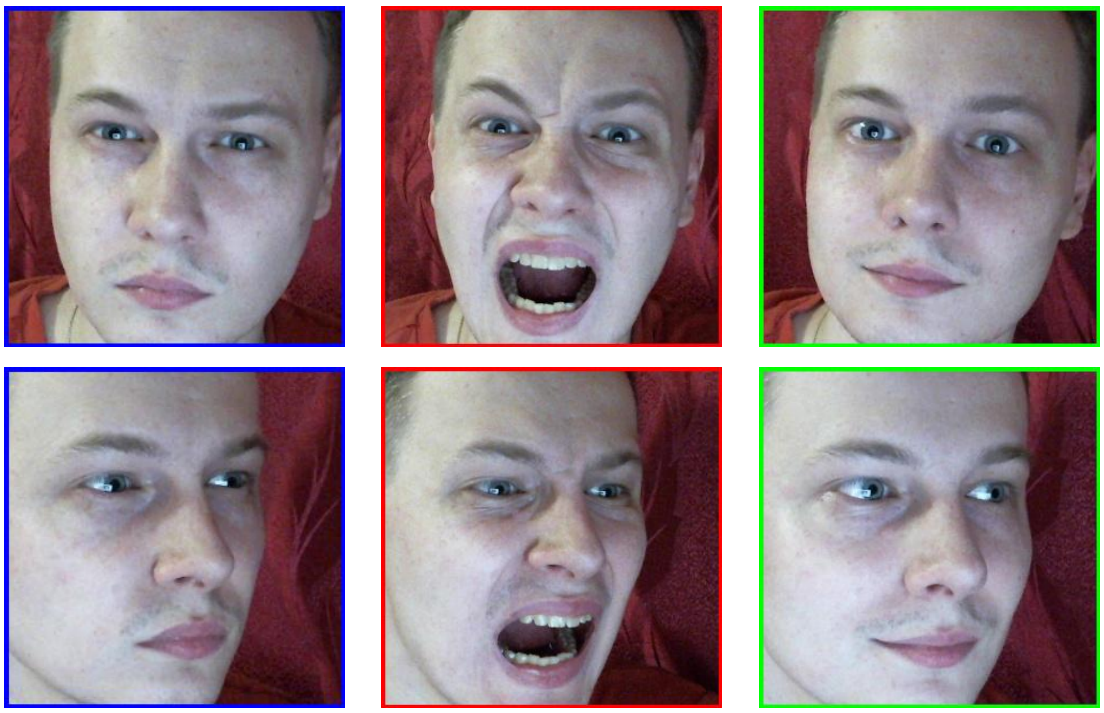
В результате проведенных исследований работы алгоритма по определению эмоций на изображении лица человека, оказалось, что модель уверенно справляется с поставленными задачами. Ошибки оказались незначительными: как и в случае с тестированием сверточной нейронной сети наиболее трудно различимыми эмоциями оказались

выражения лица, типа «заинтересованность» и «отвращение». Примеры работы алгоритма распознавания эмоций приведены на рисунках 12 - 14.



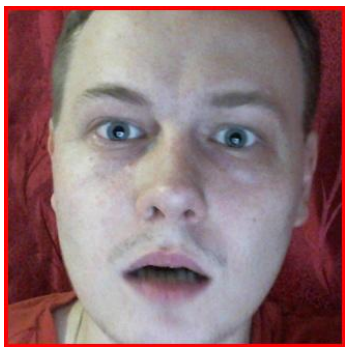
а) удивление б) спокойствие в) отвращение

Рисунок 12 – Примеры успешно распознанных эмоций



а) заинтересованность б) крик в) улыбка

Рисунок 13 – Примеры успешно распознанных эмоций



а) удивление (распознано: крик)

б) отвращение (распознано: улыбка)

Рисунок 14 – Примеры неверно распознанных эмоций

ЗАКЛЮЧЕНИЕ

В ходе работы над данным проектом были использованы результаты из первой части НИР, проанализированы литературные источники по программированию и машинному обучению, а также были изучены сверточные нейронные сети, которые были разработаны для решения схожих задач.

Реализация алгоритма распознавания эмоций человека происходила с использованием фреймворка Caffe. Тестирование разработанной сверточной нейронной сети осуществлялась с помощью специально разработанного скрипта, написанного на языке программирования Python. Апробация модели на пилотных проектах осуществлялась с помощью инструментов языка Python, а также библиотеки компьютерного зрения OpenCV и фреймворка Caffe. В результате работы над проектом

- Была разработана несложная по реализации, но высокоэффективная сверточная нейронная сеть для решения задачи компьютерного определения эмоций человека по изображению лица.
- Разработанная сверточная нейронная сеть была обучена на графическом процессоре суперкомпьютера NVIDIA DGX-1 с использованием технологии параллельных вычислений NVIDIA CUDA. Обучение разработанной модели длилось около 50 минут. Оно осуществлялось на выборке из случайно отобранных примеров из базы данных Multi-Pie, содержащей снимки с разной степенью освещенности 6 различных типов выражения эмоций: спокойствие, улыбка, удивление, заинтересованность, отвращение, спокойствие и крик.
- После обучения сверточной нейронной сети, для нее было проведено тестирование. Этот процесс, как и обучение модели, также осуществлялся на графическом процессоре

суперкомпьютера NVIDIA DGX-1. Тестирование длилось порядка 9 минут. Оно осуществлялось на выборке из 42000 случайно отобранных примеров из базы данных Multi-Pie (по 7000 примеров на каждый класс). На этапе тестирования модели была произведена оценка эффективности работы алгоритма с помощью специально рассчитанных метрик качества. Посчитанные из матрицы неточностей показатели точности и полноты классификатора для каждого из 6 имеющихся классов эмоций, а также высокие значения доли правильных ответов и F-меры подтверждают эффективность работы классификатора.

– В результате проведенных исследований работы алгоритма по определению эмоций на изображении лица человека, оказалось, что модель уверенно справляется с поставленными задачами. Ошибки оказались незначительными: как и в случае с тестированием сверточной нейронной сети наиболее трудно различимыми эмоциями оказались выражения лица, типа «заинтересованность» и «отвращение».
– В ходе выполнения второго этапа НИР был получен документ о прохождении преакселерационной программы в аккредитованном преакселераторе «Инновационные проекты», а также свидетельство о государственной регистрации программы для ЭВМ №2019611919 «CNN – Facial Expression Recognition – Прогнозирование индивидуального поведения человека на основе визуального распознавания эмоций». В ходе прохождения преакселерационной программы также был разработан бизнес-план инновационного проекта, который прошел проверку бизнес-экспертами Ассоциации Брокеров Инноваций и Технологий.

Таким образом, разработанный алгоритм машинного обучения для классификации эмоций на изображении лица человека уже можно

встраивать в приложения с использованием искусственного интеллекта, актуальные в области NeuroNet рынка НТИ.

Уже подтверждено участие данного проекта на конкурсе по поддержке стартапов СТАРТ-1 (заявка С1-57297, направление Н1. Цифровые технологии).

СПИСОК ЛИТЕРАТУРЫ

[1] M.S. Bartlett, G. Littlewort, I. Fasel, J.R. Movellan, *Real time face detection and facial expression recognition: development and applications to human computer interaction*. IEEE Conference on Computer Vision and Pattern Recognition, 2003. pp. 53.

[2] V. Bettadapura, *Face Expression Recognition and Analysis: The State of the Art. Tech Report* [Электронный ресурс]. URL: https://arxiv.org/abs/1203.6722.

[3] С. Николенко, А. Кадурин, Е. Архангельская, *Глубокое обучение. Погружение в мир нейронных сетей*. СПб.: Питер, 2018, 480 с.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, 2012, pp. 1097–1105.

[5] ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [Электронный ресурс]. URL: http://image-net.org/challenges/LSVRC/2012/.

[6] Я. Гудфеллоу, И. Бенджио, А. Курвилль, *Глубокое обучение*. М.: ДМК Пресс, 2017, 652 с.

[7] Z. Niu, M. Zhou, L. Wang, X. Gao, G. Hua, *Ordinal Regression with Multiple Output CNN for Age Estimation*. IEEE Conference on Computer Vision and Pattern Recognition, 2016. pp. 4920–4928.

[8] S. Paisitkriangkrai, J. Sherrah, P. Janney, A. Van-Den Hengel, *Effective Semantic Pixel labeling with Convolutional Networks and Conditional Random Fields*. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015. pp. 36–43.

[9] H. Khalajzadeh, M. Mansouri, M. Teshnehlab, *Face Recognition using Convolutional Neural Network and Simple Logistic Classifier*. Soft Computing in Industrial Applications. Advances in Intelligent Systems and Computing, vol. 223. Springer, Cham, 2013. pp. 197–207.

[10] Z. Wen, T. Huang, *Capturing subtle facial motions in 3d face tracking*. Proceedings of the 9th IEEE International Conference on Computer Vision, 2003. pp. 1343-1350.

[11] H. Singh, R. Patel, *Facial Expression Analysis using Deep Learning*. International Research Journal of Engineering and Technology, vol. 4, issue 10, 2017. pp. 66–69.

[12] Ф. Шолле, *Глубокое обучение на Python*. СПб: Питер, 2018, 400 с.

[13] Т. Рашид, *Создаем нейронную сеть*. Вильямс, 2018, 272 с.

[14] С. Рашка, *Python и машинное обучение*. ДМК Пресс, 2017, 418 с.

[15] Caffe Framework [Электронный ресурс]. URL: http://caffe.berkleyvision.org.

[16] S. Bajpai, *Implementing Convolutional Neural Network with Parallel Computing Using CUDA*. International Journal of Innovative Science, Engineering & Technology, Vol. 2, Issue 11, 2015. pp. 517 – 520.

[17] D. P. Kingma, J. Ba, *Adam, A Method for Stochastic Optimization* [Электронный ресурс]. URL: https://arxiv.org/abs/1412.6980.

[18] The CMU Multi-PIE Face Database [Электронный ресурс]. URL: http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html.

[19] N. Markus, M. Frljak, I.S. Pandzic, J. Ahlberg, R. Forchheimer, *Object Detection with Pixel Intensity Comparisons Organized in Decision Trees* [Электронный ресурс]. URL: https://arxiv.org/pdf/1305.4537.pdf.

[20] NVIDIA DGX-1 [Электронный ресурс]. URL: https://www.nvidia.com/ru-ru/data-center/dgx-1/.

[21] П. Флах, *Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных*. М.: ДМК Пресс, 2015.

[22] Дж. Вандер Плас, *Python для сложных задач: наука о данных и машинное обучение*. СПб.: Питер, 2018. 576 с.

[23] А. Мюллер, С. Гвидо, *Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными*. СПб: ООО "Алфа-книга", 2018, 480 с.

Приложение А. Исходный код архитектуры сверточной нейронной сети для решения задачи визуального распознавания эмоций.

all_emotions_deploy.prototxt

```
name: "EmotionsClassification"

layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param {
    shape: {
      dim: 1
      dim: 1
      dim: 118
      dim: 118
    }
  }
}

# STRATUM 0
layer {
  name: "conv00"
  type: "Convolution"
  bottom: "data"
  top: "conv00"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
```

```
    num_output: 12
    kernel_size: 3
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh00"
  type: "TanH"
  bottom: "conv00"
  top: "tanh00"
}
layer {
  name: "norm00"
  type: "LRN"
  bottom: "tanh00"
  top: "norm00"
}

layer {
  name: "conv0"
  type: "Convolution"
  bottom: "norm00"
  top: "conv0"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
```

```
  }
  convolution_param {
    num_output: 24
    kernel_size: 5
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh0"
  type: "TanH"
  bottom: "conv0"
  top: "tanh0"
}
layer {
  name: "norm0"
  type: "LRN"
  bottom: "tanh0"
  top: "norm0"
}
layer {
  name: "pool0"
  type: "Pooling"
  bottom: "norm0"
  top: "pool0"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
```

```
  }
}

# STRATUM 1
layer {
  name: "conv10"
  type: "Convolution"
  bottom: "pool0"
  top: "conv10"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 36
    kernel_size: 3
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh10"
  type: "TanH"
  bottom: "conv10"
  top: "tanh10"
}
layer {
```

```
  name: "norm10"
  type: "LRN"
  bottom: "tanh10"
  top: "norm10"
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "norm10"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 48
    kernel_size: 7
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh1"
  type: "TanH"
  bottom: "conv1"
  top: "tanh1"
}
```

```
layer {
  name: "norm1"
  type: "LRN"
  bottom: "tanh1"
  top: "norm1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "norm1"
  top: "pool1"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
  }
}

# STRATUM 2
layer {
  name: "conv20"
  type: "Convolution"
  bottom: "pool1"
  top: "conv20"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 60
    kernel_size: 3
    stride: 1
    weight_filler {
```

```
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "tanh20"
  type: "TanH"
  bottom: "conv20"
  top: "tanh20"
}
layer {
  name: "norm20"
  type: "LRN"
  bottom: "tanh20"
  top: "norm20"
}

layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm20"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 72
    kernel_size: 5
```

```
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "tanh2"
  type: "TanH"
  bottom: "conv2"
  top: "tanh2"
}
layer {
  name: "norm2"
  type: "LRN"
  bottom: "tanh2"
  top: "norm2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "norm2"
  top: "pool2"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
  }
}

# STRATUM 3
```

```
layer {
  name: "conv30"
  type: "Convolution"
  bottom: "pool2"
  top: "conv30"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 84
    kernel_size: 3
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh30"
  type: "TanH"
  bottom: "conv30"
  top: "tanh30"
}
layer {
  name: "norm30"
  type: "LRN"
  bottom: "tanh30"
  top: "norm30"
```

```
}

layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm30"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 96
    kernel_size: 7
    stride: 1
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}
layer {
  name: "tanh3"
  type: "TanH"
  bottom: "conv3"
  top: "tanh3"
}
layer {
  name: "norm3"
  type: "LRN"
```

```
  bottom: "tanh3"
  top: "norm3"
}

# STRATUM 4
layer {
  name: "fc4"
  type: "InnerProduct"
  bottom: "norm3"
  top: "fc4"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 6
    weight_filler {
        type: "xavier"
    }
    bias_filler {
        type: "constant"
        value: 0
    }
  }
}

layer {
  name: "loss"
  type: "Softmax"
  bottom: "fc4"
  top: "loss"
}
```

Приложение Б. Скрипт для тестирования сверточной нейронной сети по распознаванию различных типов эмоций

test_cnn.py

```
import sys
import os
import numpy as np
import caffe
import matplotlib.pyplot as plt
from scipy.misc import imsave
from sklearn import metrics

def getCountClases(mode):
    if mode == 1:
        return 6
    if mode == 2:
        return 2
    return 0


def getClassNumber(prototxt_file, caffemodel_file, picture_file, real_class, y_true, y_score):
    image = caffe.io.load_image(picture_file, color=False)
    transformed_image = transformer.preprocess('data', image)
    # copy the image data into the memory allocated for the net
    mean_image = np.zeros([118, 118])
    mean_image.fill(0.5)
    net.blobs['data'].data[...] = transformed_image - mean_image
    # perform classification
    output = net.forward(end='loss')
    output_prob = output['loss'] # the output probability vector for the first image in the batch
    np.set_printoptions(formatter={'float': '{: .6f}'.format})
    print(output_prob)
    y_true.append(real_class)
```

```
    y_score.append(output_prob[0, 1])
    print(output_prob[0, 1])
    predicted_class = output_prob.argmax()
    print(predicted_class, real_class)
    return predicted_class

def getSelectionAllocation(selection_file):
    res = dict()
    with open(selection_file, 'r') as f:
        while 1:
            fileline = f.readline()
            if not fileline: # EOF
                break
            dataline = fileline.split() # picture class_number
            res[dataline[0]] = int(dataline[1])
    return res

def visualizeFilters(picture_file, layer_name):
    print(picture_file)
    image = caffe.io.load_image(picture_file, color=False)
    transformed_image = transformer.preprocess('data', image)
    # copy the image data into the memory allocated for the net
    mean_image = np.zeros([118, 118])
    mean_image.fill(0.5)
    net.blobs['data'].data[...] = transformed_image - mean_image
    output = net.forward(end=layer_name)
    data = net.blobs[layer_name].data
    K = data.shape[1] / 5
    M = 5 * data.shape[2]
    N = K * data.shape[2]
    map_features = np.zeros((M, N))
    for i in range(data.shape[1]):
            st_j = (i/5)*data.shape[2]
            st_i = (i%5)*data.shape[2]
            print(st_i, st_j)
```

```
            for x in range(data.shape[2]):
                    for y in range(data.shape[2]):
                            map_features[st_i+x, st_j+y] = int(255.0*data[0, i, x, y])
    imsave(layer_name+'.png', map_features)

def buildConfusionMatrix(count_classes, selection_path, selection_file, prototxt_file, caffemodel_file):
    conf_matr = np.zeros((count_classes, count_classes))
    list_pictures = os.listdir(selection_path)
    selection_allocation = getSelectionAllocation(selection_file)
    y_true = []
    y_score = []
    for pict in list_pictures:
            print(pict)
        real_class = selection_allocation[pict]
        picture_file = selection_path + '/' + pict
        predicted_class = int(getClassNumber(prototxt_file, caffemodel_file, picture_file, real_class, y_true, y_score))
        conf_matr[predicted_class, real_class] += 1
    np.set_printoptions(formatter={'int': '{}'.format})
    print(conf_matr)
    np.savetxt('y_true.txt', y_true, delimiter=' ')
    np.savetxt('y_score.txt', y_score, delimiter=' ')
    return conf_matr

def computeErrors(count_classes, conf_matr):
    for class_numb in range(count_classes):
        precision = conf_matr[class_numb, class_numb] / np.sum(conf_matr[class_numb, :])
        recall = conf_matr[class_numb, class_numb] / np.sum(conf_matr[:, class_numb])
        f1 = 2.0*precision*recall / (precision+recall)
        print('Class ' + str(class_numb) + ': precision=' + str(precision) + ', recall=' + str(recall) + ', f1=' + str(f1))
    acc = np.trace(conf_matr)/np.sum(conf_matr)
    print('Total accuracy: ' + str(acc))
```

```
def getCNNSquad(net):
    # for each layer, show the output shape
    for layer_name, blob in net.blobs.iteritems():
        print(layer_name + '\t' + str(blob.data.shape))


if len(sys.argv) != 6:
    print('Script must be run in the format:\n python test_cnn.py mode selection_path selection_file prototxt caffemodel\n')
    print('Modes of script:')
    print('     1 - for emotion recognition')
    print('     2 - for smile detection')
else:
    mode = int(sys.argv[1])
    selection_path = str(sys.argv[2])
    selection_file = str(sys.argv[3])
    prototxt_file = str(sys.argv[4])
    caffemodel_file = str(sys.argv[5])
    count_classes = getCountClases(mode)

    caffe.set_device(0)
    caffe.set_mode_gpu()
    net = caffe.Net(prototxt_file,  # deploy prototxt model
                    caffemodel_file,  # trained weights
                    caffe.TEST)  # use test mode (e.g., don't perform dropout)
    #  create transformer for the input called 'data'
    transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
    transformer.set_transpose('data', (2, 0, 1))  # move image channels to outermost dimension
    net.blobs['data'].reshape(1,  # batch size
                              1,  # 3-channel (BGR) images
                              118, 118)
    conf_matr = buildConfusionMatrix(count_classes, selection_path, selection_file, prototxt_file, caffemodel_file)
    computeErrors(count_classes, conf_matr)
    getCNNSquad(net)
```