Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет»

Кафедра информационных технологий и автоматизированных систем

**Лабораторные работы**

по дисциплине **«Современные проблемы информатики и вычислительной техники»**

**Выполнили** студенты гр. АСУ2-16-1м, АСУ3-16-1м

Климов Сергей Николаевич

Казанцев Александр Сергеевич

Федосеева Ольга Николаевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись студентов)*

**Проверил** доцент кафедры ИТАС

Курушин Даниил Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(отметка о зачете)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата, подпись преподавателя)*

Пермь 2016

**Содержание**

[Лабораторная работа №1 «Нейронные сети» 3](#_Toc462676618)

[1. Постановка задачи 3](#_Toc462676619)

[2. Ход решения 3](#_Toc462676620)

[3. Алгоритм программирования 3](#_Toc462676621)

[4. Результаты экспериментов 3](#_Toc462676622)

[5. Заключение 3](#_Toc462676623)

[6. Список используемых источников 3](#_Toc462676624)

# Лабораторная работа №1 «Нейронные сети»

1. **Постановка задачи**

Реализовать многослойный персептрон и определить количество скрытых нейронов в первом слое для следующей задачи: на вход нейронной сети подается изображение лица человека 20х20 пикселей, необходимо отличить изображения женщин от изображений мужчин.

1. **Ход решения**

Для реализации нейронной сети было установлено следующее программное обеспечение: Python3, OpenCV, Pybrain, Eric IDE.

Процесс решения поставленной задачи поделен на следующие этапы:

1. Установка ПО и подключение к Pytnon3 необходимых библиотек;
2. Подготовка обучающей выборки. Выборка содержит 213 изображений мужчин и 83 изображений женщин. Выборка получена из открытой базы данных FERET [3]. Полученные из базы изображения обработаны и скорректированы до размера 20х20 пикселей.
3. Создание нейронной сети. Многослойный персептрон состоит из 3 слоев, имеет 798000 входов (5\*400\*399), один выход (женский пол, мужской пол). Допустимое количество нейроподобных элементов в скрытом слое определенно по формуле:

,(1)

где Ny – размерность входного сигнала, Np – число элементов обучающей выборки,Nx – размерность входного сигнала,Nw – количество нейроподобных элементов в скрытом слое [1, 39].

Допустимое количество нейроподобных элементов в скрытом слое:

1. Написание программы для обучения выборки и ее тестирование (см. пункт 3. Алгоритм программирования);
2. Оценка полученных результатов.
3. **Алгоритм программирования**

С помощью библиотеки OpenCV с применением функции face\_cascade.detectMultiScale(img, 1.3, 5), которая использует признаки Хаара для нахождения лиц, на изображении распознается лицо. После чего изображение обрезается и ужимается до размера 20х20 пикселей с помощью функции cv2.imwrite('20\_' + sys.argv[1] , resized\_img). Полученное скорректированное изображение сохраняется в файл. Для всех фото с помощью shell выполнялся данный скрипт.

Для упрощения дальнейшего тестирования программы полученные изображения были вручную отсортированы и размещены в разных папках. Изображения мужского лица были поименованы с префиксом f, женские – m.

Входными данными для нейронной сети являются характеристики изображения, представляющие собой разность яркости каждой пары пикселей изображения. Данная разность сравнивается с некоторыми значениями, предлагаемыми авторами статей [2]:

│Pixeli – Pixelj │> 0

│Pixeli – Pixelj │ < 5

│Pixeli – Pixelj │ < 10

│Pixeli – Pixelj │ < 25

│Pixeli – Pixelj │ < 50

Таким образом, мы получаем вектор характеристик [0 0 0 1 1]. К этим пяти характеристикам добавляются обратные им характеристики – [1 1 1 0 0]. Объединяя эти векторы, получаем набор двоичных характеристик для пары пикселей. Получившиеся характеристики записываются в список, который далее будет передаваться нейронной сети как входные данные. Так как наши изображения имеют размер 20х20 пикселей, список имеет 399 х 400 х 10 (1 596 000) значений.

Работа с подготовленными ранее изображениями осуществляется следующим образом: каждый пиксель изображения загружается в градациях серого и помещается в список, который передается в функцию pixel\_camparision(img), которая возвращает список с готовыми характеристиками.

Получившийся список добавляется в структуру data set, необходимую для тренировки нейросети, что является первым параметром. Второй параметр – требуемый результат работы нейронной сети (женское лицо – 0, мужское – 1). Функция dataset.addSample(list, (1, )) применяется для всех изображений. В результате своей работы функция возвращает обучающую выборку в функцию main.

Главная функция представляет собой следующий набор действий: первым делом строится нейронная сеть с помощью net = buildNetwork(1596000, 3, 1), где первый параметр – число входов, второй – число скрытых слоев, третий – число выходов. Далее указываем, что сеть обучается «с учителем»: dataset = SupervisedDataSet(1596000, 1). После чего главная функция получает обучающую выборку из dataset = do\_train\_dataset(dataset). В следующем действии указываем, что сеть тренируется методом обратного распределения ошибки. После чего происходит непосредственно процесс обучения. Процесс обучения может происходить либо по указанному числу эпох, либо до того момента пока ошибка не сводится к нулю. В ходе тестов выяснилось, что при количестве эпох более 30 ошибка имеет значение около 0,1 и далее уменьшается на крайне низкое значение. На основе этого принято решение использовать 50 эпох. После проведения обучения программа выводит значение получившейся ошибки. Далее происходит тестирование работы с получившеюся нейронной сетью. На вход сети подается список характеристик изображений, не участвующих в обучении. После чего сеть выдает результат своей работы, что происходит это при помощи функции print(net.activate(list)). По получившимся данным оценивается результат работы нейронной сети. Для тестирования нейронной сети была взята выборка из 9 фото мужских и женских лиц. В результате чего нейросеть не выдала ни одного правильного ответа. Дополнительно были проведены тесты с 5, 10 характеристиками и без характеристик, используя яркость пикселей изображения. Тестирование не дало положительных результатов. Поэтому было принято решение использовать другой метод решения поставленной проблемы.

Основная идея второго метода – нахождение на лице глаз, носа, рта и определение расстояния между данными частями. Расстояния подавалось на вход нейросети. Для осуществления данных действий была использована библиотека Оpen CV. Для нахождения черт лица использовались те же методы что и при подготовке обучающей выборки фотографий, с разными каскадами Хаара. Данный способ дал положительные результаты с большими фотографиями (Рисунок 1).

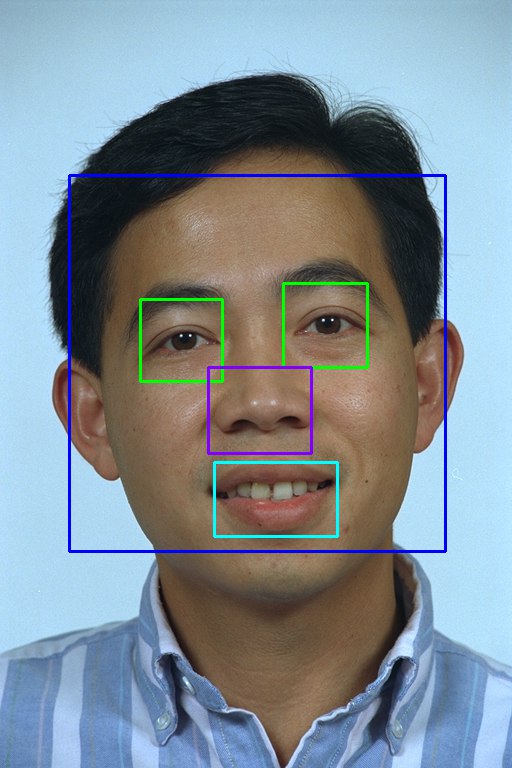


Рисунок 1 – результат распознавания на фото отдельных частей лица

Однако на фото размером 20х20 программа не смогла обнаружить части лица. Из-за чего данный способ был признан не рабочим.

1. **Результаты экспериментов**

В результате проделанной работы был сделан вывод, что при помощи OpenCV и Pybrain распознавание пола на фото размерам 20х20 пикселей не осуществимо. С работой можно ознакомиться в репозитории Lab1. https://github.com/KlimovSN-ASU2-16/Labs/tree/master/lab1

1. **Заключение**

Таким образом, в ходе выполнения лабораторной работы:

* создана такая модель нейронной сети как многослойный персептрон;
* подготовлена обучающая выборка, содержащая 296 изображений;
* обучен многослойный персептрон;
* проведено тестирование нейросети;
* выявлено неработоспособность нейронной сети;
* выявлены причины неработоспособности;
* предложен другой метод решения проблемы, также признанный неработоспособным.

1. **Список используемых источников**
2. Е.В. Долгова, Д.С. Курушин. Компьютерные нейросетевые технологии: учебное пособие.— Пермь: Изд-во Перм. гос. техн. ун-та, 2008 .— 87с.
3. Хабрахабр – Распознавание пола в изображениях и видео. URL: <https://habrahabr.ru/post/172463/>
4. Открытая база данных FERET. URL: Face-rec.org/databases
5. Face Detection using Haar Cascades. URL: http://docs.opencv.org/trunk/d7/d8b/tutorial\_py\_face\_detection.html