МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению лабораторных работ

по дисциплине «Технологии компьютерного зрения»

для студентов направления подготовки
09.03.02 «Информационные системы и технологии»
Профиль «Прикладное программирование в интеллектуальных информационных системах»

Ставрополь, 2026

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА № 1. УСТАНОВКА БИБЛИОТЕК КОМПЬЮТЕРНОГО ЗРЕНИЯ

Цель: получение навыков работы с библиотеками компьютерного зрения OpenCV.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Для установки OpenCV в Windows необходимо с официального сайта https://opencv.org/releases/ скачать версию OpenCV для Windows (рисунок 1.4).

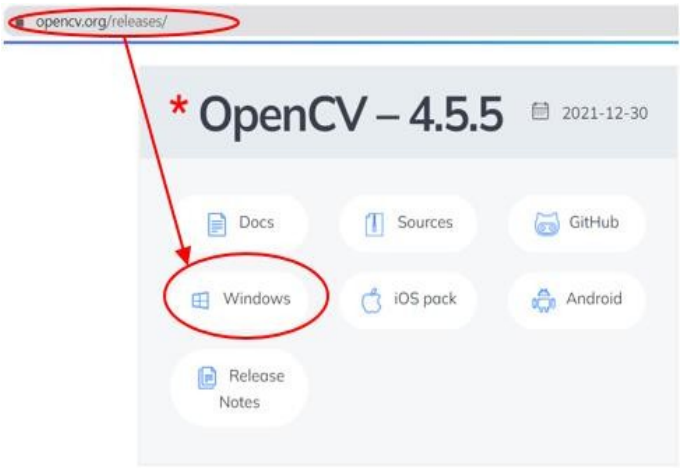


Рисунок 1 – Расположение OpenCV на официальном ресурсе

Распаковывать архив в C:\opencv_4.5.5\build и установить значение системной переменной, как показано на рисунке 2

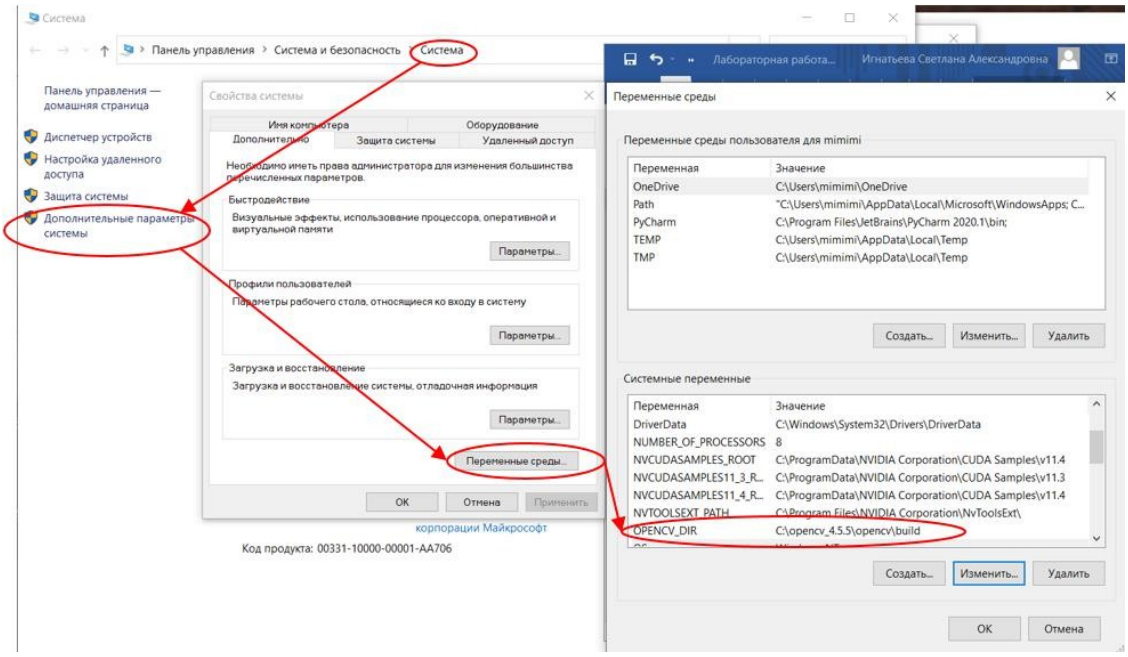


Рисунок 2 – Установка OpenCV на персональный компьютер

Установка OpenCV в среду Python:

```
pip install opencv-python
```

Установка OpenCV-contrib в среду Python:

```
pip install opencv-contrib-python
```

Для установки библиотеки dlib необходимо использовать кроссплатформенную утилиту CMake, которая обладает возможностями автоматизации сборки программного обеспечения из исходного кода. Для этого необходимо загрузить установочный файл CMake, доступный по ссылке https://cmake.org/download/ (рисунок 3).

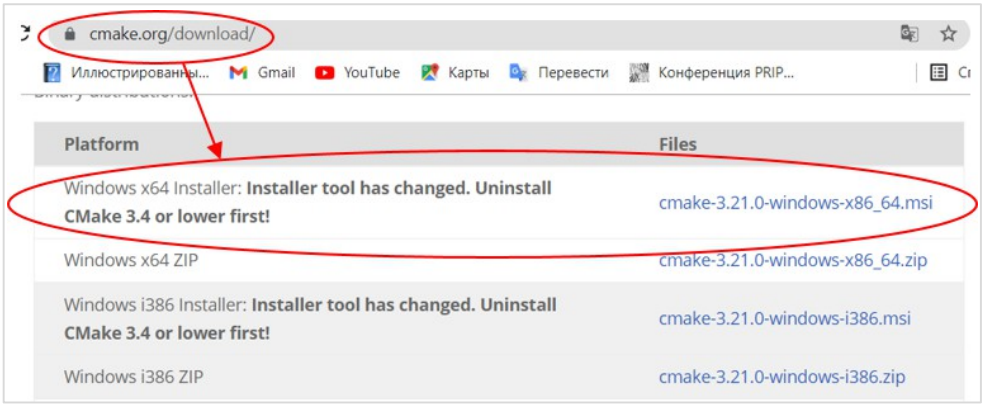


Рисунок 3 – Доступ к CMake на веб-ресурсе

При установке рекомендуется выбрать опцию «Add CMake to the sys- tem PATH for all users» («Добавить пути в переменные среды для всех пользователей»).

Dlib разработана на основе языка программирования C, следовательно, необходим компилятор. Автономный компилятор MS Visual Studio можно скачать по ссылке: https://visualstudio.microsoft.com/ru/visual-cpp-build-tools/. После завершения установки необходимо установить дополнительные пакеты для программирования C/C++, т. е. Packages CMake tools для Windows.

Чтобы установить dlib в среду Python требуется инсталлировать cmake: `pip install cmake`, а затем dlib: `pip install dlib`.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Функции библиотеки OpenCV

`cv2.imread ()` – чтение изображений в Python с помощью OpenCV. Возвращает 2D- или 3D-матрицу в зависимости от количества цветных каналов, присутствующих на изображении.

Синтаксис: `cv2.imread('Path/filename', flag)`.

`flag` является необязательным и может принимать одно из следую- щих значений: `cv2.imread_color` (считывает изображение с цветами RGB, но без канала прозрачности. Это значение по умолчанию для флага, когда значение не указано в качестве второго аргумента для `cv2.imread()`);

`cv2.imread_grayscale` (читает изображение как серое. Если исходное изоб- ражение является цветным, значение серого для каждого пикселя вычисляется путем получения среднего значения цветовых каналов и считывается в массив); `cv2.imread_unchanged` (читает изображение как есть из источника).

`cv2.imwrite ()` – запись изображения в Python с помощью OpenCV. Синтаксис: `cv2.imwrite ('Path/filename', image)` `cv.imshow()` – вывод изображения на экран.

Синтаксис: `cv.imshow("filename", image)`

`cv.waitKey()` – задержка отображения окна в течении заданных миллисекунд, или до нажатия какой-либо клавиши на клавиатуре. Если в качестве параметра принято число, то время ожидания до закрытия окна ограничено указанным числом в миллисекундах. Если указан 0 или параметр не указан, окно закроется по нажатию любой клавиши клавиатуры.

`cv2.resize` – изменение размера изображения. Синтаксис:

`cv2.resize(image, new_size, interpolation=cv2. INTER_XXX)`

`interpolation=cv2.INTER_XXX` – алгоритм интерполяции, может принимать значения: cv2.INTER_NEAREST (интерполяция ближайшего соседа); cv2.INTER_LINEAR (билинейная интерполяция); cv2.INTER_CU- BIC (бикубическая интерполяция); cv2.INTER_AREA (пересчет с использованием отношения площадей пикселей); cv2.INTER_LANCZOS4 (интерполяция Lanczoc по окресности 8×8).

`cv2.flip()` – переворачивает 2D-массив вокруг вертикальной, горизонтальной или обеих осей. Может принимать значения: 1 – переворот вокруг оси y, 0 – переворот вокруг оси z, –1 – переворот по обеим осям.

`cv2.split()` – разделить изображение по каналам. Синтаксис:

```
b, g, r = cv2.split(image)
```

`cv2.merge()` – объединить каналы изображения. Синтаксис:

```
cv2.split([b, g, r])
```

`cv2.blur()` – размытие изображения на основе свертки с ядром $n \times n$. Пиксель в центре матрицы устанавливается равным среднему значению всех остальных пикселей. Чем больше ядро свертки (т. е. n), тем больше размытие.

Синтаксис:

`cv2.blur(image, (n, n))`

`cv2.GaussianBlur()` – размытие по Гауссу. Похоже на предыдущее размытие, однако вместо простого среднего используется взвешенное, т. е.

чем ближе пиксели к центральному, тем большее влияние они оказывают на среднее значение.

Синтаксис:

`cv2.GaussianBlur(image, (n, n), s)`, где n – ядро свертки, s – стандартное отклонение ядра Гаусса. Установив значение 0, оно будет вычисляться автоматически.

`cv2.putText()` – наложение текста на изображение. Синтаксис:

`cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType]])`, где `image` – изображение; `text` – строка текста; `org` – координаты нижнего левого угла текстовой строки x и y; `font` – шрифт, например `FONT_HERSHEY_SIMPLEX`, `FONT_HERSEY_PLAIN` и т. д.; `fontScale` – масштабный коэффициент шрифта, который умножается на базовый размер; `color` – цвет строки, `thickness` – толщина; `lineType` – необязательный параметр, который определяет тип линии.

`cv2.line()` – начертить линию на изображении. Синтаксис:

`cv2.line(image, org, color[, thickness[, lineType[, shift]]])`, где `image` – изображение; `text` – строка текста; `org` – координаты; `color` – цвет; `thickness` – толщина; `lineType` – необязательный параметр, который определяет тип линии; `shift` – цвет.

`cv2.rectangle()` начертить прямоугольник на изображении.

Синтаксис: `cv2.line(image, org1, org2 color[, thickness[,`

lineType[, shift]]]), где image – изображение; text – строка текста; org1 – координаты верхнего левого угла; org2 – координаты нижнего правого угла; color – цвет; thickness – толщина; lineType – необязательный параметр, который определяет тип линии; shift – цвет.

cv2.cvtColor() – преобразование из одного цветового простран- ства в другое.

Синтаксис: cv2.cvtColor(image, code[, outImg[, CoutImg]]), где image – изображение, code – код цветового пространства; outImg – необязательный параметр, выходное изображение того же размера и глубины, что и исходное; CoutImg – количество каналов в целевом изображении. Если равен 0, то количество каналов определяется автоматически, на основе исходного изображения. Является необязательным параметром.

cv2.threshold() – функция бинаризации. Для каждого пикселя применяется одно и то же пороговое значение. Если значение пикселя меньше

порогового значения, оно устанавливается равным 0, в противном случае оно устанавливается на максимальное значение.

Синтаксис: cv2. threshold (image, thr, max, TypeThr), где image – исходное изображение, которое должно быть изображением в градациях серого; thr – это пороговое значение, которое используется для классификации значений пикселей; max – это максимальное значение, которое присваивается значениям пикселей, превышающим пороговое значение. OpenCV предоставляет различные типы пороговых значений, которые задаются четвертым параметром функции: cv2.THRESH_BINARY, cv2.THRESH_BINARY_INV, cv2.THRESH_TRUNC, cv2.THRESH_TOZERO, cv2.THRESH_TOZERO_INV.

cv2.adaptiveThreshold() – функция адаптивной бинаризации, устанавливает порог для пикселя на основе небольшой области вокруг него. Таким образом, для разных областей изображения используется разный порог.

Синтаксис: cv2.adaptiveThreshold(image, maxVal, adaptive-Method, TypeThr, blockSize, const), где image – исходное одноканальное изображение; maxVal – максимальное значение, которое может быть присвоено; adaptiveMethod – адаптивный метод, который определяет как рассчитывается пороговое значение. Может принимать значение cv2.ADAPTIVE_THRESH_MEAN_C(), cv2.ADAPTIVE_THRESH_GAUSSIAN_C().

cv2.findContours() – функция выделения контуров на изображении. Синтаксис: cv2.findContours(image, mode, method), где image – исходное изображение; mode – один из четырех режимов группировки найденных контуров: CV_RETR_LIST – выдаёт все контуры без группировки; CV_RETR_EXTERNAL – выдаёт только крайние внешние контуры; CV_RETR_CCOMP – группирует контуры в двухуровневую иерархию. На верхнем уровне – внешние контуры объекта. На втором уровне – контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень; CV_RETR_TREE – группирует контуры в многоуровневую иерархию; method – один из трёх методов упаковки контуров: CV_CHAIN_APPROX_NONE – упаковка отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей; CV_CHAIN_APPROX_SIMPLE – склеивает все горизонтальные, вертикальные и диагональные контуры; CV_CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS – применяет к контурам метод упаковки (аппроксимации) Teh-Chin.

cv2.filter2D() – может применяться для сглаживания или увеличения резкости изображений в зависимости от ядра свертки и выполнять другие преобразования изображений.

Синтаксис: cv2.Filter2D(src, ddepth, kernel), где src – исходное изображение; ddepth – глубина, значение –1 означает, что результирующее изображение будет иметь ту же глубину, что и исходное; kernel – матрица фильтра, применяемая к изображению.

Для повышения резкости может использоваться одна из масок H3–H6, для получения эффекта рельефа – ядро

$$kernel = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}.$$

`cv2.convertTo()` – выполняет преобразование изображения для изменения контраста и яркости new_img = α·img +β, где new_img – преобразованное изображение, α и β – параметры, для управления яркостью и контрастом.

Синтаксис: `img.convertTo(new_img, ddepth, alfa, beta)`, где `img` – исходное изображение; `new_img` – изображение с преобразо- ванием; `ddepth` – глубина; `alfa` и `beta` – параметры для управления ярко- стью и контрастом.

`cv2.createCLAHE()` – алгоритм адаптивной коррекции гистограмм с ограниченным контрастом. Изображение разделяется на небольшие блоки, и гистограмма выравнивается для каждого из них, после чего для объедине- ния каждого блока применяется билинейная интерполяция для устранения искусственных границ. Применяется для улучшения контрастности изображений. Имеет два параметра: `clipLimit` (устанавливает порог для ограничения контраста, по умолчанию 40) и `tileGridSize` (устанавливает количество плиток в строке и в столбце, по умолчанию 8×8).

Функции библиотеки dlib

Для выполнения лабораторной работы необходимы следующие инструменты из библиотеки dlib:

– `get_frontal_face_detector()` – функция для детектирования лиц. Для обнаружения лиц dlib использует методы HOG (Histogram of Oriented Gradients) и SVM (Support Vector Mashines).

– `shape_predictor()` – представляет собой инструмент, который

выбирает область изображения, содержащую некоторый объект, и выводит набор местоположений точек, определяющих положение объекта. В качестве аргумента необходимо указать используемую обученную модель shape_predictor_68_face_landmarks.dat.

Пример 68-точечной модели определения особых точек лица в dlib показан на рисунке 4

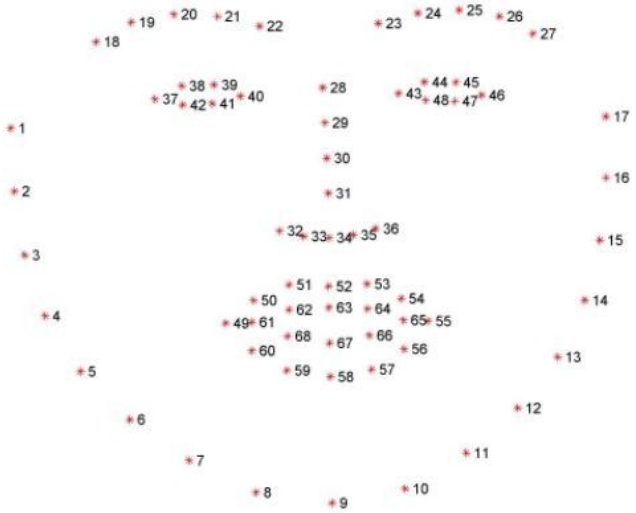


Рисунок 4 – Расположение ключевых точек лица в модели dlib

Примеры реализации базовых процедур обработки изображений на языке программирования Python с использованием OpenCV

Преобразование цветного изображения в градации серого

```
import os, cv2 as cv

image = cv.imread("./img2.jpg")
cv.imshow("original", image)

gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
cv.imshow("gray_image", gray_image)

cv.waitKey()
```

Добавление тестовой надписи на изображение

```
import os, cv2 as cv

image = cv.imread("./img2.jpg")
output = image.copy()

cv.putText(output, "Image with text", (50, 50),
```

```
  cv.FONT_HERSHEY_SIMPLEX, 2, (250, 80, 50), 3)
cv.imshow("Изображение с текстом", output)
cv.waitKey()
```

Пример фрагмента кода python для детектирования особых точек лица с использованием библиотеки dlib

Выбор детекторов:

```
# детектор лица на изображении
detector = dlib.get_frontal_face_detector()
# детектор контура лица (68 точек)
predictor = dlib.shape_predictor('shape_predic'
       'tor_68_face_landmarks.dat')
for file in face_images:
    image = cv2.imread(file)
```

Поиск и отображение на изображение особых точек лица:

```
rects = detector(gray, 1)
for rect in rects:
# нахождение 68 точек для каждого найденного лица
    shape = predictor(gray, rect)
for i in range(0, 68):
    # отображение точек
    cv2.circle(image, (shape.part(i).x, shape.part(i).y),
         1, (0, 0, 255), -1)
cv2.imwrite(f'./output/{os.path.split(file)[-1]}', image)
```

1. Изучить теоретический материал, представленный в теоретической части.
2. Установить библиотеку.
3. Реализовать все примеры.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. Что понимают под цифровым изображением?
2. Сформулируйте различия между обработкой изображений, компьютерной графикой и распознаванием образов.
3. Какова основная особенность градационных преобразований?
4. Сформулируйте процедуру преобразования цветного изображения в полутоновое (полутонового в бинарное).
5. В чем заключается сущность масочной обработки изображений и ее отличие от градационных преобразований?

ЛАБОРАТОРНАЯ РАБОТА №2. ОБРАБОТКА ИЗОБРАЖЕНИЙ КЛАССИЧЕСКИМИ АЛГОРИТМАМИ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК КОМПЬЮТЕРНОГО ЗРЕНИЯ

Цель: получение навыков работы с библиотеками компьютерного зрения OpenCV и dlib для базовых задач обработки изображений.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Обработка изображений, компьютерное зрение, распознавание образов и машинное обучения являются тесно связанными областями и, как правило, при решении сложных задач практически все они реализуются в прикладных системах.

Обработка изображений (в узком смысле) решает задачи преобразования изображений, поэтому исходными и результирующими данными этого этапа являются цифровые изображения. В прикладном плане – это построение некоторой системы (или программного комплекса), которая решает определенный класс задач от получения цифрового изображения до интерпретации его содержания и извлечения некоторой информации.

Компьютерное зрение (Computer (Machine) Vision) занимается анализом образов. Для некоторой сцены (аудитория, комната и др.) компьютер должен дать ее описание (есть ли в ней предметы, какая освещенность и т. д.). Таким образом, компьютерное «зрение» переводит изображение в описание.

Распознавание – это процесс отнесения изображения к некоторому определенному классу. В широком смысле допустимо трактовать распознавание как построение полной модели изображения, которую можно воспроизвести посредством машинной графики.

Изображения представляют собой специфический вид информации и их специфика еще не полностью изучена ни в исследованиях, посвященных

изучению зрительного восприятия человека, ни в информатике. Одной из особенностей является избыточность практически любого изображения. Причем изображение одной и той же сцены реального мира несет разную информацию разным пользователям, что не позволяет значительно уменьшить объем данных, содержащихся, например, в космическом снимке, хранящемся в базе данных изображений.

Цифровым изображением называются данные, организованные в виде числового массива, воспроизводящего свойства некоторой реальной сцены или синтезированного человеком объекта (чертежа, карты) и тесно связанного со способом получения этих данных.

С формальной точки зрения цифровое изображение представляется в виде матрицы действительных чисел F

$$F = \begin{bmatrix} f_{00} & f_{01} & \cdots & f_{0,N-1} \\ f_{10} & f_{11} & \cdots & f_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ f_{M-1,0} & f_{M-1,1} & \cdots & f_{M-1,N-1} \end{bmatrix} = \left[f_{ij} \right].$$

с элементами f_{ij}

(каждому элементу ставится в соответствие некоторое число – код яркости, обычно от 0 до 255), где *ij* – целочисленные координаты. Причем принято за начало координат принимать левый верхний угол изображения, где $i = 0,\ j = 0.$

С использованием введенных обозначений можно компактно записать полное цифровое изображение размерами $M \times N$

Каждый элемент этой матрицы называется элементом изображения или пикселем (pixel от английского picture element). Каждый пиксель цветного изображения в системах компьютерного зрения описывается тремя уровнями яркости – красным (R), зеленым (G) и синим (B) в диапазоне от 0 до 255.

В качестве примеров систем, использующих информацию из изображений и видео, требующих применения методов обработки изображений, компьютерного зрения и распознавания образов можно привести системы

управления процессами (промышленные роботы, автономные транспортные средства); системы видеонаблюдения с автоматизированным или автоматическим анализом видеоданных; системы организации информации (например, индексация баз данных изображений); системы моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование); системы человеко-машинного взаимодействия; системы дополненной реальности и др.

Множество методов обработки изображений делится на методы обработки в частотной и пространственной областях. Термин пространственная область относится к плоскости изображения как таковой, и дан- ная категория объединяет подходы, основанные на прямом манипулировании пикселями изображения. Способы обработки изображений в частотной области (после применения дискретного преобразования Фурье) достаточно развиты, но требуют значительно больших вычислительных затрат и для решения практических задач применяются реже.

Слабый контраст – один из наиболее распространенных дефектов фотографических и сканированных изображений, обусловленный ограниченностью диапазона воспроизводимых яркостей. Путем цифровой обработки контраст можно повысить, изменяя яркость каждого элемента изображения и увеличивая диапазон яркостей. Традиционно для цифрового представления каждого отсчета изображения в компьютере отводится 1 байт запоминающего устройства. Следовательно, входной или выходной сигналы могут принимать одно из 256 значений в диапазоне 0 – 255. Предположим, что минимальная и максимальная яркости исходного изображения равны f min и f max соответственно. Если эти параметры или один из них существенно отличаются от граничных значений яркостного диапазона, то изображение выглядит как неконтрастное. В реальных изображениях обычно существует перекос в сторону малых уровней и яркость большинства элементов изображения ниже средней. На темных участках подобных изображений детали часто оказываются неразличимыми. Одним из методов улучшения

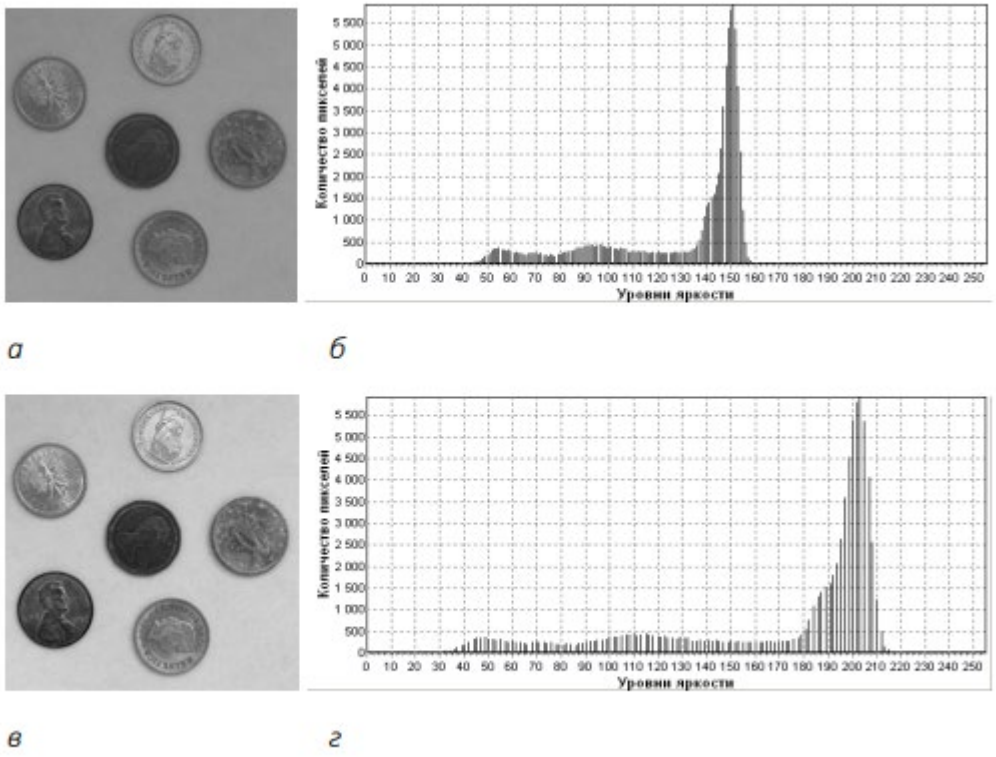
таких изображений является видоизменение гистограммы. Этот метод предусматривает преобразование яркостей исходного изображения с тем, чтобы гистограмма распределения яркостей обработанного изображения приняла желаемую форму. Одним из методов улучшения контраста является линейная растяжка гистограммы, когда уровням исходного изображения, лежащим в интервале $[f_{\min}, f_{\max}]$, присваиваются новые значения с тем, чтобы охватить весь возможный интервал изменения яркости, в данном случае [0, 255]. При линейном контрастировании используется линейное поэлементное преобразование вида

$$g_{ij} = \frac{f_{ij} - f_{\min}}{f_{\max} - f_{\min}}(g_{\max} - g_{\min}) + g_{\min}.$$

Такой алгоритм заложен во многих программных продуктах, реализующих функцию «Autocontrast». При нормализации гистограммы на весь максимальный интервал уровней яркости [0, 255] растягивается не вся гистограмма, лежащая в пределах $[f_{\min}, f_{\max}]$, а ее наиболее интенсивный участок $[f'_{\min}, f'_{\max}]$, из рассмотрения исключаются малоинформативные начальный и конечный участки. Например, для рассматриваемого изображения на основе анализа гистограммы исходного изображения примем $f'_{\min} = 45$ и $f'_{\max} = 160$.

Целью выравнивания гистограммы (линеаризации, эквализации) является такое преобразование, чтобы в идеале все уровни яркости приобрели бы одинаковую частоту, а гистограмма яркостей отвечала бы равномерному закону распределения. Кроме этого, существует метод видоизменения гистограмм, который обеспечивает экспоненциальную или гиперболическую форму распределения яркостей улучшенного изображения. Применяются также «локальные» методы улучшения на основе обработки части изображения (рисунок 1). Другой подход заключается в применении методов, в результате которых на выходное значение пикселя оказывают влияние текущий пиксель и его соседи, т. е. увеличивается окрестность вокруг пикселя

и это приводит к значительно большей гибкости при обработке изображений. При этом используется маска (апертура, окно, ядро, окрестность), которая представляет собой двумерный массив заданного размера, чаще всего размером 3×3. Такие методы называют обработкой или фильтрацией на основе маски.



а – исходное изображение; б – гистограмма исходного изображения; в – результат улучшения контраста; г – гистограмма после линейной растяжки

Рисунок 1 – Результат улучшения контраста

В качестве маски используется множество весовых коэффициентов, заданных во всех точках окрестности, обычно симметрично окружающих рабочую точку кадра. Распространенным видом окрестности, часто применяемым на практике, является квадрат 3×3 с рабочим элементом в центре (рисунок 2). На рисунке 3 представлены элементы области и изображения под маской. При цифровой обработке изображений используется декартова система координат с началом в левом верхнем углу кадра и с положительными направлениями из этой точки вниз и вправо.

$w_{(-1,-1)}$	$w_{(-1,0)}$	$w_{(-1,1)}$
$w_{(0,-1)}$	$w_{(0,0)}$	$w_{(0,1)}$
$w_{(1,-1)}$	$w_{(1,0)}$	$w_{(1,1)}$

Рисунок 2 – Коэффициенты маски с относительными значениями координат

$f_{(i-1,j-1)}$	$f_{(i-1,j)}$	$f_{(i-1,j+1)}$
$f_{(i,j-1)}$	$f_{(i,j)}$	$f_{(i,j+1)}$
$f_{(i+1,j-1)}$	$f_{(i+1,j)}$	$f_{(i+1,j+1)}$

Рисунок 3 – – Элементы области изображения под маской

Сущность процедуры фильтрации заключается в смещении маски фильтра по изображению с шагом один пиксель слева направо, сверху вниз

При этом отклик задается суммой произведений коэффициентов фильтра на соответствующие значения пикселей в области, покрытой маской фильтра. Для маски размером 3×3 отклик r определяется следующим образом:

$$r = w_{(-1,-1)}f_{(i-1,j-1)} + w_{(-1,0)}f_{(i-1,j)} + w_{(-1,1)}f_{(i-1,j+1)} +$$
$$+w_{(0,-1)}f_{(i,j-1)} + w_{(0,0)}f_{(i,j)} + w_{(0,1)}f_{(i,j+1)} + w_{(1,-1)}f_{(i+1,j-1)} +$$
$$+w_{(1,0)}f_{(i+1,j)} + w_{(1,1)}f_{(i+1,j+1)}.$$

В общем случае фильтрация изображения f_{ij} размером M×N с использованием маски размером m n × описывается выражением

$$g_{ij} = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w_{st} f_{i+s,j+t},$$

где $a=(m-1)/2,\ b=(n-1)/2.$

Таким образом, можно использовать маски размером не только 3×3, но и более, например, 5×5, 7×7 и т. п. Если центр маски находится на границе изображения, то ее края выходят за его пределы. Для обработки краев изображений используются следующие приемы: дополнение средним значением, доопределение повторением крайних отсчетов

последовательности, экстраполяция более высокого порядка. Усредняющая или сглаживающая пространственная фильтрация может служить эффективным средством подавления высокочастотных шумов. Сглаживающие маски, часто называемые шумоподавляющими, имеют вид:

$$H1=\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix},\quad H2=\frac{1}{10}\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix},\quad H3=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}.$$

Повышение резкости изображений может быть достигнуто путем численного дифференцирования. С принципиальной точки зрения величина отклика оператора производной в точке изображения пропорциональна степени разрыва изображения в данной точке. Таким образом, дифференцирование изображения позволяет усилить перепады яркости и другие разрывы (шумы, помехи) и не подчеркивать области с медленными изменениями яркостей.

Для вычисления двумерной второй производной и наложения результата на изображение (высокочастотная фильтрация) используются маски:

$$H4=\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix}\quad H5=\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}\quad H6=\begin{bmatrix}1&-2&1\\-2&5&-2\\1&-2&1\end{bmatrix}.$$

Следует отметить, что для сохранения постоянного уровня яркости изображения коэффициент передачи маски, определяемый как сумма всех элементов маски, должен быть равен единице. Это справедливо для представленных масок H1–H6. Для выделения контуров изображений можно использовать маски H1–H6 с уменьшенным на единицу центральным элементом. В настоящее время рассмотренные алгоритмы и множество других, применяемых для решения задач обработки изображений и компьютерного зрения, реализованы в различных библиотеках компьютерного зрения, среди которых можно выделить OpenCV [1], dlib,

PCL, VXL, AForge.Net, LTI и др.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

OpenCV (Open Source Computer Vision, https://opencv.org/) [1] – библиотека компьютерного зрения с открытым исходным кодом и широким спектром возможностей. Функциональность OpenCV доступна на языках C, C++, Python, Java, Ruby. Поддерживаются основные операционные системы: MS Windows, Linux, Mac, Android, iOS.

Библиотека включает большое количество оптимизированных классических и современных алгоритмов обработки изображений, распознавания образов, компьютерного зрения и машинного обучения, которые могут быть использованы для обнаружения и распознавания лиц, идентификации раз- ных классов объектов, анализа действий человека в видео, отслеживания движений камеры, сопровождения движущихся объектов на видео и других.

OpenCV имеет модульную структуру, а это значит, что пакет включает в себя несколько общих или статических библиотек. Библиотека непрерывно обновляется, в версии OpenCV 4.5.5 доступны:

1) модуль *core* определяет основные структуры данных, используемые всеми остальными модулями, такие как:

- операции с массивами;
- сохранение XML/YAML/JSON;
- кластеризация;

- утилиты, системные функции и макросы;
- совместимость с OpenGL;
- алгоритмы оптимизации;
- параллельная обработка;
- аппаратное ускорение;

2) модуль *imgproc* предназначен для обработки изображений и включает:
   - фильтрацию изображений;
   - геометрические преобразования;
   - функции рисования;
   - функции преобразования цветного пространства;
   - анализ гистограмм;
   - обнаружение признаков;
   - сегментацию;
   - обнаружение объектов;
3) модуль *imgcodecs* используется для чтения и записи изображений;
4) модуль *videoio* предназначен для чтения и записи видео или последовательности изображений;
5) модуль *highgui* – графический интерфейс высокого уровня. Позволяет создавать и управлять окнами, которые могут отображать изображения и «запоминать» их содержимое;
6) модуль *video*, предназначенный для анализа видео, включающий алгоритмы оценки движения, вычитания фона и отслеживания объектов;
7) модуль *calib3d* – калибровка камеры и 3D-реконструкция. Включает основные алгоритмы геометрии с несколькими видами, калибровку одиноч- ной и стереокамеры, оценку позы объекта, алгоритмы стереосоответствия и элементы 3D-реконструкции;
8) модуль *features2d* включает детекторы признаков, дескрипторы и средства сопоставления дескрипторов:

– обнаружение и описание признаков;

– сопоставление дескрипторов;

– рисование ключевых точек и совпадений;

– классификация объектов;

9) модуль *objdetect*, предназначен для обнаружения объектов и экземпляров предопределенных классов (например: лица, глаза, люди, автомобили и т. д.);

10) модуль *dnn*, предназначен для работы с глубокими нейронными сетями:

– инструментарий для загрузки моделей сетей из разных фреймворков;

– реализации готовых слоев нейронных сетей;

– утилиты, для создания новых слоев;

11) модуль *ml* – модуль для машинного обучения. Представляет собой набор классов и функций для статической классификации, регрессии и кластеризации данных;

12) модуль *flann* – модуль, содержащий коллекцию алгоритмов, оптимизированных для быстрого поиска ближайших соседей в больших наборах данных;

13) модуль *photo* – модуль для обработки фотографий:

– шумоподавление;

– HDR-визуализация (High Dynamic Range Rendering);

– обесцвечивание изображений с сохранением контраста;

– бесшовное клонирование, которое позволяет скопировать объект с одного изображения на другое таким образом, чтобы комбинация выглядела бесшовной и естественной;

14) модуль *stitching* используется для сшивания изображений:

– поиск признаков и сопоставление изображений;

– оценка вращения;

– автокалибровка (пытается оценить фокусные расстояния по заданной гомографии исходя из предположения, что камера вращается только вокруг своего центра, и может использоваться для построения панорамных мозаик);

– деформация изображений;

– оценка швов склеенных изображений;

– компенсация экспозиции;

15) модуль *gapi* – модуль, предназначенный для того, чтобы сделать обычную обработку изображений быстрой и переносимой путем внедрения новой графовой модели, суть которой заключается в построении графа на основе используемых операций к объектам данных. Графы представляются неявной структурой, для описания которой используются не «узлы» и «ребра», а операции и объекты данных. Объекты данных при этом содержат не фактические значения, а описывают зависимости. Модуль выступает в качестве фреймворка.

Opencv-contrib – отдельный репозиторий, содержащий новые расширяющие функциональность модули, которые еще не включены в основную часть. Это связано с тем, что новые модули часто не имеют стабильного интерфейса и плохо протестированы, поэтому их и не включают в официальный дистрибутив OpenCV. *Opencv-contrib 4.5.5* содержит ряд модулей, среди них можно выделить основные:

– *alphamat* (альфа-матирование) предназначен для отделения объекта на переднем плане от фона обеспечивая нерезкие границы. Его можно использовать для дальнейших операций;

– *aruco* предоставляет возможности обнаружения двоичных квадратных реперных маркеров и инструменты для их использования при оценке позы человека и калибровке камеры;

– *barcode* служит для обнаружения и декодирования штрих-кодов;

– *bgsegm* применяется для вычитания фона при обработке видео;

– *bioinspired* позволяет обрабатывать как изображения, вызывающие оптические иллюзии, так и обычные;

– *ccalib* предназначен для калибровки одной камеры, стереопары и многокамерных систем. Позволяет удалять большие искажения на изображениях, а также реконструировать 3D-сцены на основе двух стереоизображений;

– *cvv* может быть использован для отладки приложений с использованием визуального пользовательского интерфейса;

– *dnn_objdetect* служит для обнаружения объектов с использованием глубоких сверточных нейронных сетей;

– *dnn_superres* позволяет обрабатывать изображения сверхвысокого разрешения, также включает функции масштабирования фото и видео;

– *face* предназначен для обнаружения и распознавания лиц на изображениях;

– *fuzzy*. Модуль реализует алгоритмы обработки изображений, основанные на нечеткой математике;

– *mcc* применяется для цветокоррекции с использованием Color correction Model;

– *tracking* используется для сопровождения различных объектов на видео;

– *ximgproc* предназначен для фильтрации несоответствий на стереоизображениях, может быть использован для быстрого обнаружения границ с использованием алгоритма Structured Forest;

– *xphoto* применяется для восстановления поврежденных или отсутствующих частей изображения [3].

Библиотека компьютерного зрения dlib

Библиотека dlib включает реализацию множества традиционных ма-

тематических функций, операций линейной алгебры, а также алгоритмов сжатия данных, классической обработки изображений и алгоритмов на основе машинного обучения, реализованных в основном на языке C++, однако ряд инструментов dlib возможно использовать и на Python.

Лицензирование dlib с открытым исходным кодом позволяет использовать ее бесплатно [2]. Следует выделить следующие основные инструменты в dlib версии 19.23:

1) *для обработки изображений* – функции для работы с различными типами пикселей; чтение и запись различных форматов изображений, таких как BMP, PNG, JPEG, GIF и формат dlib файла DNG; автоматическое преобразование цветового пространства; алгоритмы поиска объектов на изображениях; традиционные операции с изображениями, такие как поиск краев и морфологические операции и др.; алгоритмы извлечения признаков SURF (Speeded Up Robust Features, Ускоренные надежные признаки), HOG (Histogram of Oriented Gradients, Гистограмма направленных градиентов) и FHOG (Felzenszwalb Histogram of Oriented Gradients, Гистограмма направленных градиентов Фельзеншвальба);

2) *для машинного обучения* – алгоритмы бинарной и многоклассовой классификации, регрессионного и структурного анализа, кластеризации, глу- бокого обучения, обучения без учителя, обучения с подкреплением;

3) *для метапрограммирования* – предоставляются функциональные возможности для отладки, выполнения различных операций с шаблонами. Включает большое число макрофункций;

4) *для вычислений:*

- быстрые алгоритмы обработки матричных объектов;
- многочисленные функции линейной алгебры и

математические операции;

5) *для оптимизации* dlib включает нелинейные оптимизаторы общего назначения;

6) *для работы с сетью* в dlib имеется набор объектов для создания

многопоточных программ, для обеспечения соединения между сетевыми приложениями, для создания сервера и т. д.

7) *для работы с текстом* имеются инструменты для анализа и различных манипуляций: преобразования кодировок, работы со строками, анализа параметров аргументов командной строки и др.

1. Изучить теоретический материал, представленный в теоретической части.
2. При необходимости установить OpenCV и dlib.

1. Загрузить полноцветное изображение с использованием OpenCV и выполнить преобразования согласно своему варианту. Исследовать влияние параметров функций, необходимых для выполнения преобразования. Подписать на изображении выпол- ненное преобразование и свою фамилию.

Реализовать следующие операции обработки изображений:

Вариант 1: Кадрирование, перевод изображения в полутоновое (в градациях серого).

Вариант 2: Поворот, перевод изображения в бинарное. *Вариант 3:* Масштабирование, размытие изображение. *Вариант 4:* Кадрирование, увеличение четкости.

Вариант 5: Поворот, линейное улучшение контраста.

Вариант 6: Масштабирование, улучшение контраста на основе эквализации гистограммы (clahe).

Вариант 7: Кадрирование, выделение контуров.

Вариант 8: Поворот, эффект рельефа.

Вариант 9: Масштабирование, увеличение четкости.

Вариант 10: Кадрирование, линейное улучшение контраста.

Вариант 11: Поворот, улучшение контраста на основе эквализации гистограммы (clahe).

Вариант 12: Масштабирование, размытие изображения.

Вариант 13: Кадрирование, эффект рельефа.

Вариант 14: Поворот, выделение контуров.

Вариант 15: Масштабирование, адаптивная бинаризация.

2. Загрузить полноцветное изображение лица человека с использованием OpenCV. Применить функцию выделения особых точек лица библиотеки dlib для изображений одного и тоже человека, полученных в различных условиях сьемки. Сделать выводы о корректности нахожде- ния точек лица на изображениях.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. Приведите примеры масок для ВЧ-фильтрации.
2. В каких областях используется цифровая обработка изображений? Приведите примеры.
3. Какие утилиты используются для установки OpenCV и dlib?
4. Перечислите основные модули, входящие в состав OpenCV.
5. Какими основными функциональными возможностями обладает библиотека dlib?

ЛАБОРАТОРНАЯ РАБОТА №3. КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ.

Цель: научиться обучению модели нейронной сети для классификации изображений одежды, на примере набора данных Fashion MNIST.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Одним из наиболее распространенных применений TensorFlow и Keras является распознавание и классификация изображений.

TensorFlow - это библиотека с открытым исходным кодом, созданная для Python командой Google Brain. TensorFlow компилирует множество различных алгоритмов и моделей, позволяя пользователю реализовать глубокие нейронные сети для использования в таких задачах, как распознавание и классификация изображений, а также обработка естественного языка. TensorFlow - это мощный фреймворк, который функционирует путем реализации ряда узлов обработки, каждый из которых представляет математическую операцию, а весь ряд узлов называется «графом».

Говоря о Keras, это высокоуровневый API (интерфейс прикладного программирования), который может использовать функции TensorFlow (а также другие библиотеки ML, такие, как Theano). Keras был разработан с удобством и модульностью в качестве руководящих принципов. С практической точки зрения Keras позволяет реализовать множество мощных, но зачастую сложных функций TensorFlow максимально просто, к тому же он настроен для работы с Python без каких-либо серьезных изменений или настроек.

Распознавание изображения относится к задаче ввода изображения в нейронную сеть и присвоения какой-либо метки для этого изображения. Метка, которую выводит сеть, будет соответствовать заранее определенному классу. Может быть присвоено как сразу несколько классов, так и только один.

Если существует всего только один класс, обычно применяется термин «распознавание», тогда как задача распознавания нескольких классов часто называется «классификацией».

Подмножество классификаций изображений - является уже определением объектов, когда определенные экземпляры объектов идентифицируются как принадлежащие к определенному классу, например, животные, автомобили или люди.

Ярким примером такой классификации является решение самой распространённой капчи — ReCaptcha v2 от Google, где из набора картинок необходимо выбрать только те, которые принадлежат к указанному в описании классу.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Для работы предлагается использовать Google Colaboratory — это бесплатная интерактивная облачная среда для работы с кодом от Google. Сервис нужен, чтобы писать код в jupyter notebook. Он функционирует по принципу облака, что позволяет работать над одним проектом целой командой.

Программа предоставляет доступ к графическим процессорам GPU и TPU, благодаря которым можно развивать приложения на основе нейросетей.

Как начать работу в Google Colab.

Для начала убедитесь, что вы вошли в аккаунт, а затем создайте отдельную папку под готовые проекты. После запуска блокнота нажмите

«Создать» → «Ещё» → Google Colaboratory. Для переименования блокнота щёлкните на имя файла.

Далее определитесь с процессором. Если будете работать на слишком мощном процессоре, то вы можете вылететь из блокнота. Чтобы попасть в настройки процессора, выберите вкладку «Среда выполнения» и команду «Сменить среду управления».

В Colab много встроенных библиотек Python: Pandas, NumPy, Scikit-learn. Для просмотра полного списка, введите команду !pip list.

Библиотека Pandas применяется для анализа и обработки табличных данных. То есть она как Excel, но работает с большей мощностью. Pandas позволяет работать с данными объемом в миллионы строк.

Чтобы импортировать сторонние библиотеки, используйте команды:

!pip install имя_библиотеки
import имя_библиотеки.

Чтобы очистить строку, щелкните на значок крестика в левой части окна.

Также при желании можно загрузить tensorflow на свой компьютер, делается это с помощью команды pip install tensorflow в командной строке анаконды. Но с большой вероятностью у вас могут возникнуть трудности с дальнейшим импортированием этих модулей в ваш проект. И если такие ошибки возникнут, вот по этой ссылке вы можете посмотреть самые типичные из них и как можно их устранить https://www.tensorflow.org/install/errors?hl=ru

Поэтому чтобы избежать этих трудностей с установкой на локальном компьютере, будем работать в Google Colaboratory, тут не должно возникнуть ни каких проблем, кроме того эта облачная среда дает вам возможность использовать больше оперативной памяти, чем возможно имеется на вашем компьютере.

Для начала, необходимо подключить нужные библиотеки и модули. Подключим numpy и np для работы с массивами и визуализацией рисунков. Далее импортируем модуль sequential это модель нейронных сетей, где слои идут друг за другом. После подключаем тип слоев Dense, который означает,

что слои будут полносвязными (Полносвязный слой — слой, выходные нейроны которого связаны со всеми входными нейронами. Нейроном в данном случае называется математическая модель искуственного нейрона, основанная на представлении о биологическом нейроне, однако в действительности не имеющая с ним практически ничего общего). А также подключаем различные утилиты, которые помогут перевести данные в подходящий для keras формат.

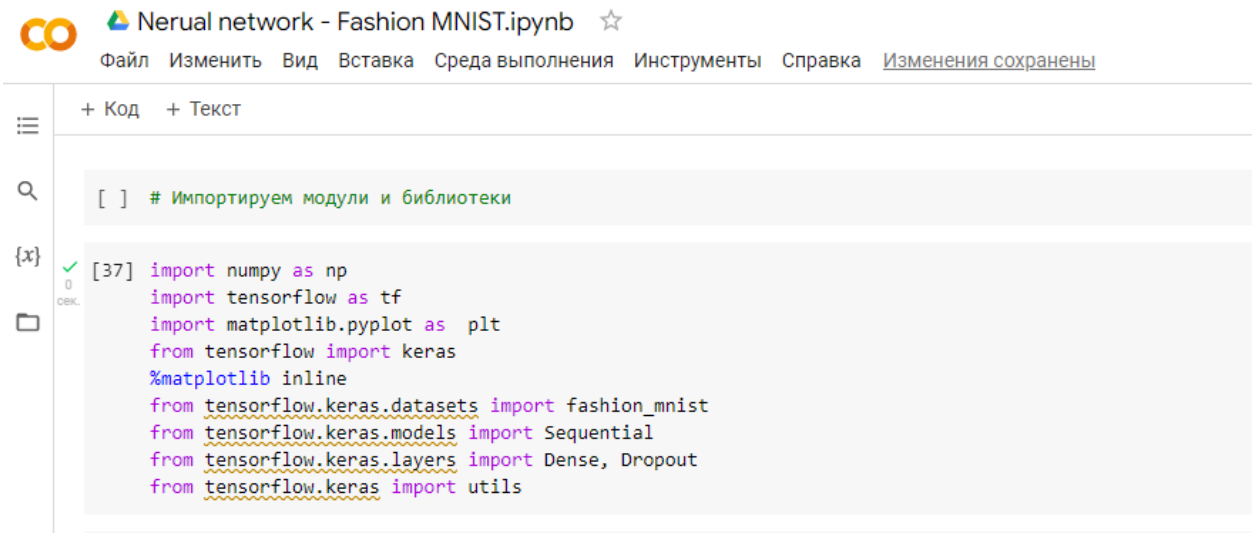


Рисунок 1 – Импорт модулей и библиотек

Модуль для работы fashion mnist уже присутствует keras, т.к он является одним из таких популярных наборов данных, на которых обучаются нейронным сетям все датасайнтисты.

Набор данных Fashion MNIST, который содержит 70 000 изображений в 10 категориях. На изображениях показаны отдельные предметы одежды с низким разрешением (28 на 28 пикселей):

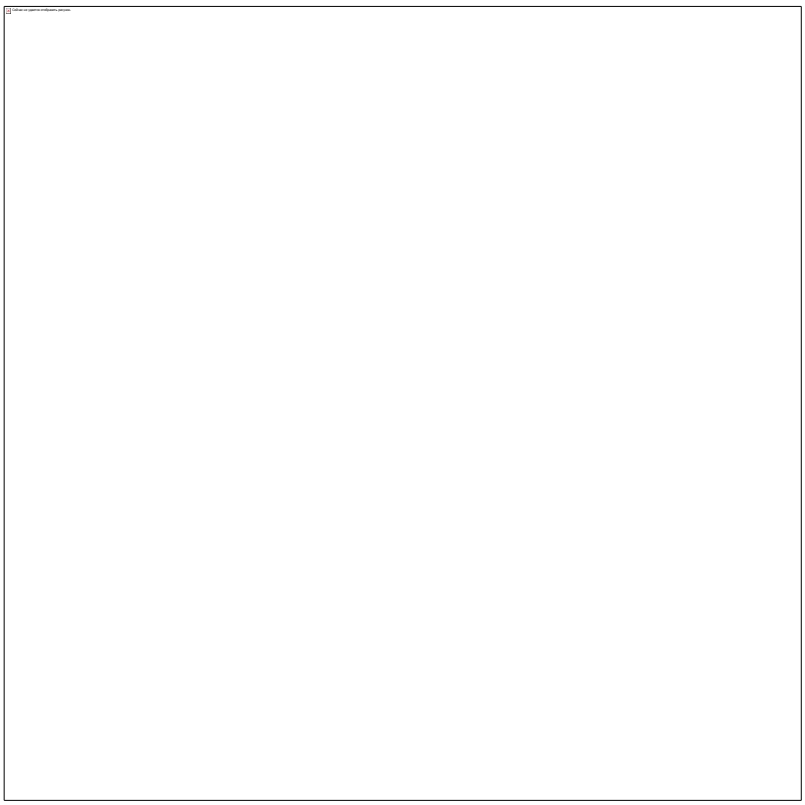


Рисунок 2 – Набор данных Fashion MNIST

Будем использовать 60 000 изображений для обучения сети и 10 000 изображений, чтобы оценить, насколько точно сеть научилась классифицировать изображения. Вы можете получить доступ к Fashion MNIST непосредственно из TensorFlow, просто импортировав и загрузив данные:

```
[13] # делим наш датасет  на обучающую и тестовую выборку

[14] (x_train, y_train),(x_test, y_test) = fashion_mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

Рисунок 3 – Загрузка набора данных

Загрузка набора данных возвращает четыре массива:

Массивы x_train и y_train — это данные, которые использует модель для обучения

Массивы x_test и y_test используются для тестирования модели

Часть x это сами изображения, а часть y это ответы или метки.

Изображения представляют собой NumPy массивы 28x28, значения пикселей которых находятся в диапазоне от 0 до 255. Метки (labels) представляют собой массив целых чисел от 0 до 9. Они соответствуют классу одежды:

Метка	Класс
0	T-shirt (Футболка)
1	Trouser (Брюки)
2	Pullover (Свитер)
3	Dress (Платье)
4	Coat (Пальто)
5	Sandal (Сандали)
6	Shirt (Рубашка)
7	Sneaker (Кроссовки)
8	Bag (Сумка)
9	Ankle boot (Ботильоны)

Имена классов не включены в набор данных, поэтому прописываем самостоятельно:

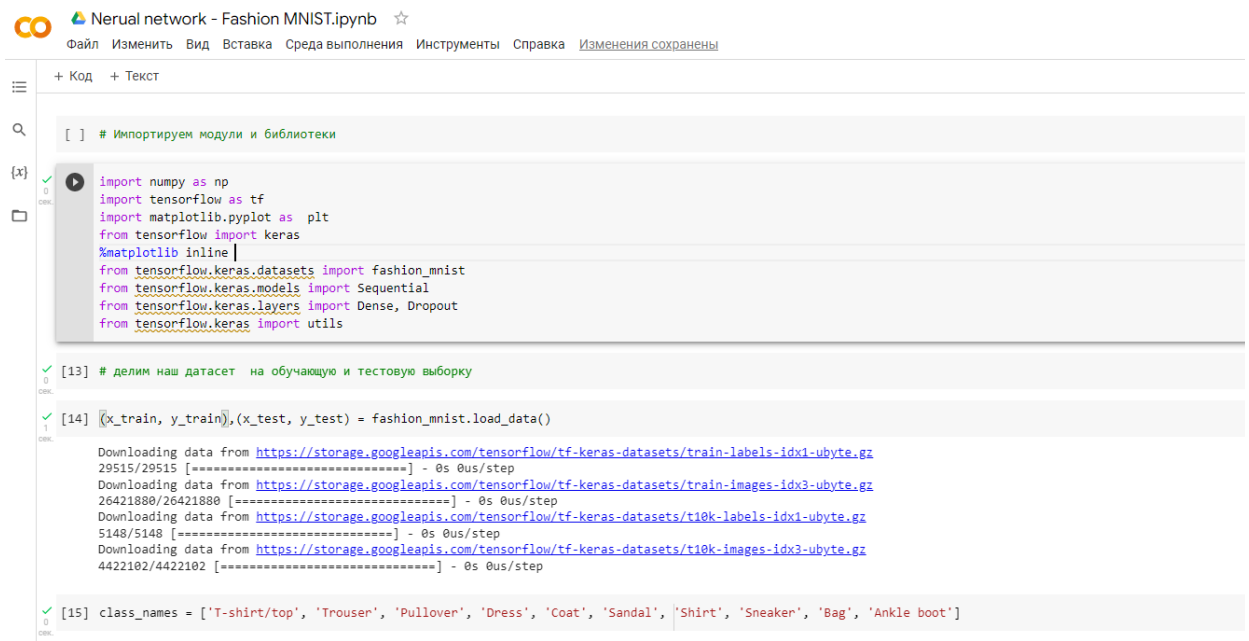


Рисунок 4 – Добавление имен классов

Далее нужно произвести предварительную обработку данных перед тем как создать нейронную сеть. Но сначала посмотрим, как выглядят изображения.

```
[16] # Предварительная обработк данных
```

```
[17] # посмотрим как выглядят изображения
```

```
[18] plt.figure()
     plt.imshow(x_train[25])
     plt.colorbar()
     plt.grid(False)
```

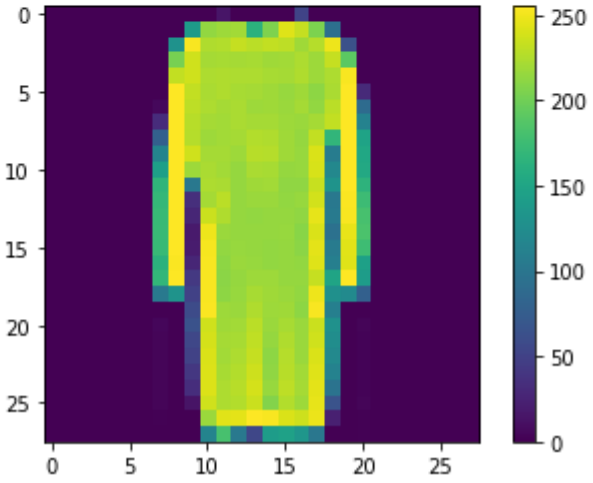


Рисунок 5 – Предварительная обработка данных

В квадратные скобки вставляем индекс от 0 до 59999, т.к в обучающей выборке 60000 рисунков. Сбоку на рисунке можно увидеть значения интенсивностей пикселей от 0 до 255, т.е если пиксель максимально темный, он имеет значение 0, а если максимально светлый значение 255.

Далее выполним нормализацию данных. Для улучшения алгоритма оптимизации, который используется в обучении нейронных сетей, делим интенсивность каждого пикселя изображения на 255, чтобы данные на входе в нейронную сеть находились в диапазоне от 0 до 1.

```
[19] # нормализация данных
```

```
[20] x_train = x_train / 255
     x_test = x_test / 255
```

```
plt.figure()
plt.imshow(x_train[25])
plt.colorbar()
plt.grid(False)
```

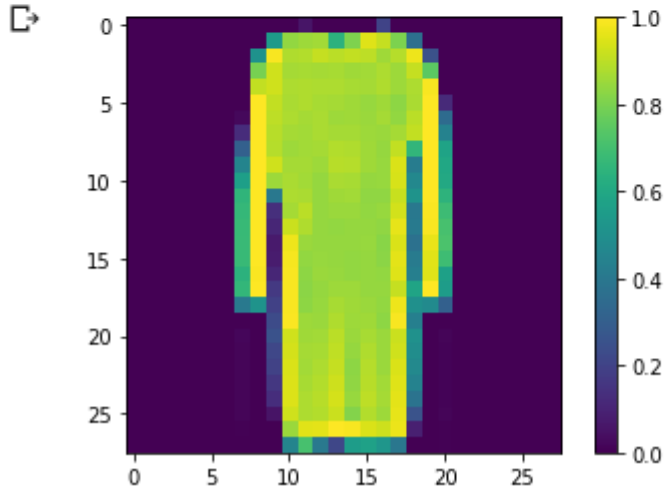


Рисунок 6 – Нормализация данных

Просмотрим несколько изображений на одном экране (25 рисунков по 5 в каждой строке и имя класса), для этого:

```
[22] # посмотрим несколько изображений
```

```
[23] plt.figure(figsize=(10,10))
     for i in range (25):
       plt.subplot(5,5,i+1)
       plt.xticks([])
       plt.yticks([])
       plt.imshow(x_train[i])
       plt.xlabel(class_names[y_train[i]])
```

Рисунок 7 – Просмотр нескольких изображений

Данные подготовлены и далее создаем модель нейронной сети. В нейронных сетях основным строительным блоком является слой, и основная часть глубокого обучения состоит в объединении простых слоев. В keras нейронные сети называются моделями. Будем использовать модель типа sequential, это означает, что слои будут идти последовательно. Создаем

последовательную модель. Первый слой сети Flatten преобразует формат изображений из 2-мерного массива в 1-мерный массив, т.е теперь изображение будет поступать в нейрон как строка в 784 пикселя.

После идут два полноценных слоя: первый слой входной полносвязный, здесь определяем сколько нейронов будет в этом слое (в ходе экспериментов установлено, что одним из самых удачных по предсказаниям является 128 нейронов, но можно указать также и 512 нейронов или 800). В каждый нейрон будут поступать все 784 пикселя изображения.

Указываем функцию активации, в нашем случае для входного слоя указываем ‘relu’, она показывает хорошую эффективность в подобных нейронных сетях.

Далее 3-й слой выходной полносвязный, в нем будет 10 нейронов по количеству классов, в качестве функции активации будет функция "softmax", благодаря которой 2-й слой будет возвращать массив из 10 вероятностных оценок сумма которых равна 1.

```
[24] # Создание модели нейронной сети
```

```
[32] from keras import layers
     from keras.layers.attention.multi_head_attention import activation
     model = keras.Sequential([
         keras.layers.Flatten(input_shape=(28,28)),
         keras.layers.Dense(128, activation="relu"),
         keras.layers.Dense(10, activation= "softmax")
     ])
```

Рисунок 8 – Создание модели нейронной сети

Оптимизатор в нашем случае используем 'adam', также можно использовать также SGD (Стохастический градиентный спуск-оптимизационный алгоритм, отличающийся от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как

сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно выбранного элемента)

Прежде чем модель будет готова к обучению, ей потребуется еще несколько настроек. Они добавляются во время этапа компиляции модели:

Категориальная перекрестная энтропия (В теории информации перекрёстная энтропия между двумя распределениями вероятностей измеряет среднее число бит, необходимых для опознания события из набора возможностей, если используемая схема кодирования базируется на заданном распределении вероятностей, вместо «истинного» распределения)

Доля правильных ответов

Loss function (функция ошибки) — измеряет насколько точная модель во время обучения.

Optimizer (оптимизатор) — это то, как модель обновляется на основе данных, которые она видит, и функции потери.

Metrics (метрики) — используется для контроля за этапами обучения и тестирования.

```
[33] # Компиляция модели

     model.compile(optimizer = 'adam',
                   loss = 'sparse_categorical_crossentropy',
                   metrics=['accuracy'])

[43] model.summary()

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                 Output Shape              Param #
     =================================================================
      flatten (Flatten)            (None, 784)               0

      dense (Dense)                (None, 128)               100480

      dense_1 (Dense)              (None, 10)                1290

     =================================================================
     Total params: 101,770
     Trainable params: 101,770
     Non-trainable params: 0
     _________________________________________________________________
```

Рисунок 9 – Настройка параметров

Далее приступим к обучению нейронной сети. Обучение выполняется с помощью функции fit, т.к в рамках данной задачи выполняется обучение с учителем, передаем значения с x_train, y_train. Далее указываем параметры – количество эпох. Одна эпоха — это когда весь набор проходит через нейронную сеть один раз. Количество эпох будет разниться в зависимости от наборов данных. В конце каждой строки эпохи указывается функция ошибки.

```
[40] # обучение модели

model.fit(x_train, y_train, epochs=10)

Epoch 1/10
1875/1875 [==============================] - 9s 4ms/step - loss: 0.5008 - accuracy: 0.8232
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3736 - accuracy: 0.8647
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3374 - accuracy: 0.8772
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3127 - accuracy: 0.8857
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2954 - accuracy: 0.8913
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2816 - accuracy: 0.8958
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2696 - accuracy: 0.9002
Epoch 8/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2571 - accuracy: 0.9049
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2464 - accuracy: 0.9074
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2401 - accuracy: 0.9096
<keras.callbacks.History at 0x7fb448db7280>
```

Рисунок 10 – Обучение модели

По мере обучения нейронной сети, т.е с каждой эпохой значение ошибки снижается, а точность повышается. При моделировании модели отображаются показатели потерь (loss) и точности (acc). Эта модель достигает точности около 0,9 (или 90%) по данным обучения.

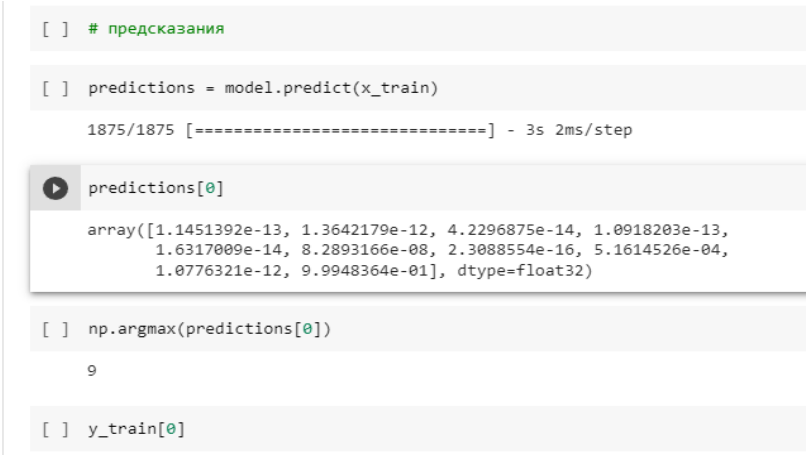
Далее проверим точность на тестовой выборке, это новые для нейронной сети изображения (10000).

```
[48] # проверка точности предсказания

[49] test_loss, test_acc = model.evaluate(x_test, y_test)
     print('Test accuracy:', test_acc)

313/313 [==============================] - 1s 2ms/step - loss: 0.3254 - accuracy: 0.8817
Test accuracy: 0.8816999793052673
```

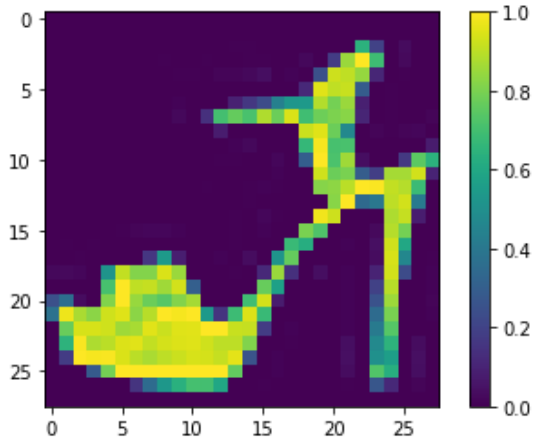
Рисунок 11 – Проверка точности предсказания

Качество предсказания немного ниже, но в целом достаточно высокое.

После обучения, будем предсказывать, что изображено на изображениях, для этого используем модель predict и изображения на которых модель обучалась. В квадратных скобках указываем индекс изображения. Выводится 10 значений по количеству нейронов и по количеству классов, данные значения в минусовой степени, а значит вероятность близка к 0, только одно число со значение в -1 степени, это значит примерно 0,99, а единица это 100% вероятность. Таким образом модель предсказывает, что изображение соответствует последнему классу. Далее проверим и выведем правильный ответ из меток.



```
[52] predictions[0]

     array([1.1451392e-13, 1.3642179e-12, 4.2296875e-14, 1.0918203e-13,
            1.6317009e-14, 8.2893166e-08, 2.3088554e-16, 5.1614526e-04,
            1.0776321e-12, 9.9948364e-01], dtype=float32)

[53] np.argmax(predictions[0])

     9

[54] y_train[0]

     9

[55] plt.figure()
     plt.imshow(x_train[9])
     plt.colorbar()
     plt.grid(False)
```




```
[56] class_names[np.argmax(predictions[9])]

     'Sandal'
```

Рисунок 12 – Результат работы модели

В итоге был произведен импорт изображений, далее разделили их на обучающую и тестовую выборку, оптимизировали изображения, после создали архитектуру нейронной сети из 3-х слоев, скомпилировали, обучили нейронную сеть и протестировали.

1. Изучить теоретический материал, представленный в теоретической

части.

2. Повторить все представленные выше шаги и создать нейронную сеть.
3. Протестировать нейронную сеть, написать выводы.
4. Создать аналогичную нейронную сеть, но с другим набором данных.

Содержание отчета и его форма

1.Отчет по лабораторной работе должен содержать:

2.Номер и название лабораторной работы; задачи лабораторной работы.

3.Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

4.Ответы на контрольные вопросы.

5.Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. Какие цветовые модели вы знаете, и для каких задач они могут использоваться?
2. Что такое фильтры обработки изображений и какие основные типы фильтров существуют?
3. Какие методы используются для улучшения качества изображений и подавления шума?
4. Что такое классификация изображений
5. Какие методы применяются для сегментации изображений? Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 4. ОБНАРУЖЕНИЕ ОБЪЕКТОВ ЗАДАННЫХ КЛАССОВ НА ИЗОБРАЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Цель: изучить принципы и инструменты программной реализации обнаружения объектов заданных классов на изображениях с использованием существующих фреймворков, реализующих сверточные нейронные сети.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Обнаружение объекта – это определение местоположения заданного объекта на изображении, при этом его размеры меньше размеров изображения, а количество объектов на изображении заведомо неизвестно.

Объекты на изображении задаются значениями некоторых признаков, которые представляются в виде вектора конечной размерности. Наборы признаков, т. е. длина векторов, являются одинаковыми для всех объектов. Совокупность признаков объекта определяется правилами его описания (например, описание полутонового изображения уровнями яркостей всех пикселей изображения или описание этого изображения его яркостной гистограммой). Соответственно, признаки могут выражаться в различными числовыми значениями.

На всём множестве объектов, которые могут присутствовать на изображениях, существует разбиение на подмножества, т. е. классы объектов. Объекты в пределах одного класса описываются наборами признаков, по которым они более схожи внутри этого класса и значительно отличаются от объектов других классов. Кроме этого, они имеют одно имя класса, по которому неразличимы. Например, класс «люди» и класс «автомобили».

Класс задается указанием некоторых признаков, присущих всем его членам. Соответственно, основными задачами являются выбор наиболее эффективного набора признаков для объектов и метода сравнения этих

признаков для принятия решения об отношении объекта к определенному классу.

При обнаружении объектов на изображении, как правило, неизвестно количество искомых объектов, их расположение и размеры, что значительно усложняет задачу. Одним из первых методов обнаружения был подход на основе сопоставления с шаблоном, т. е. искомым объектов. Несмотря на то, что такой подход может обеспечить максимальную точность, он неустойчив даже к незначительным трансформациям искомого объекта на изображении относительно исходного. Поэтому для описания объектов в дальнейшем использовали такие наборы признаков, как гистограммные признаки, ключевые точки объекта, признаки на основе спектральных преобразований (например, Фурье, Хаара, Адамара, вейвлет и др.). Такие подходы позволяют повысить возможность обнаружения отличных по масштабу и повороту объектов, однако все же не обеспечивают максимально возможную точность. Кроме этого, неясно как получить наиболее эффективный набор признаков.

Благодаря стремительному развитию вычислительной мощности компьютерной техники обнаружение объектов на изображениях все чаще выполняется с применением предварительно обученных сверточных нейронных сетей (СНС) на больших базах данных изображений, которые позволяют формировать эффективные наборы признаков объектов и их классифицировать.

СНС в настоящее время обладают наилучшей обобщающей способностью. СНС является одним из видов искусственных нейронных сетей (ИНС). ИНС – это существенно параллельно распределенный процессор, который обладает способностью к сохранению и репрезентации опытного знания. Она схожа с мозгом в двух аспектах: знание приобретается сетью в процессе обучения; для сохранения знания используются межнейронные соединения (синаптические соединения).

Работа ИНС состоит в преобразовании входных данных во времени, в результате чего меняется внутреннее состояние сети и формируются

выходные воздействия. Обычно ИНС оперирует цифровыми, а не символьными величинами. Большинство моделей ИНС требуют обучения. В общем случае обучение – это такой выбор параметров сети, при котором сеть лучше всего справляется с поставленной проблемой.

Основным элементом нейронной сети является нейрон, который осуществляет операцию нелинейного преобразования суммы произведений входных сигналов на весовые коэффициенты:

$$y = \varphi(b_0 + \sum_{k,j} w_k x_j),$$

где $X = (x_1, x_2, \ldots, x_n)$ – вектор входного сигнала;
$W = (w_1, w_2, \ldots, w_n)$ – весовой вектор;
φ – функция активации;
b_0 – смещение.

Модель нейронного элемента изображена на рисунке 1. Каждому входу нейрона соответствует весовой коэффициент (синапс), который характеризует силу синаптической связи по аналогии с биологическим нейроном.

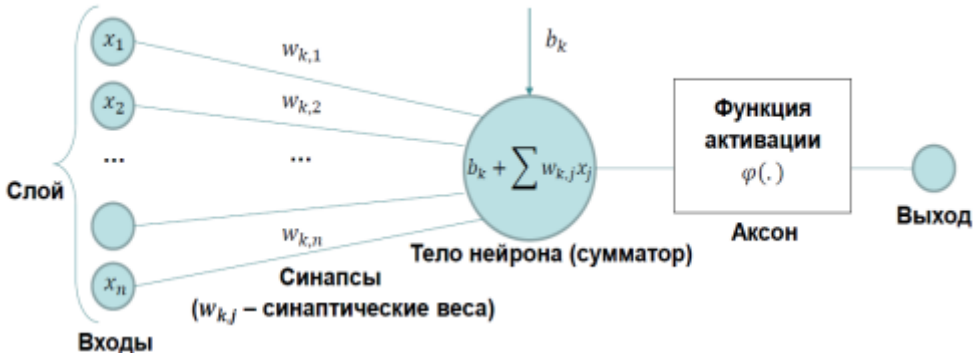


Рисунок 1 – Модель искусственного нейрона

Весовой вектор и функция активации определяют поведение любого нейрона, т. е. его реакцию на входные данные. Величина веса wi определяет степень влияния входа i на выход нейрона, а знак – характер влияния. Положительные веса характерны для возбуждающих связей, способствующих повышению активности нейрона. Отрицательные, как правило, соответствуют тормозящим связям. Каждый из входов связан с некоторым источником ин-

формации (рецептор, формирующий признаки, распределительная ячейка, выход другого нейрона). Коэффициент b0 является характеристикой, задаю щей начальный уровень активности (при нулевом входе). Изменение его эквивалентно смещению пороговой функции по оси абсцисс.

Функция активации используется для ограничения выхода нейрона в заданном диапазоне и для нелинейного преобразования взвешенной суммы. Последнее позволяет нейронному классификатору аппроксимировать любую нелинейную границу между классами в пространстве образов. Функция активации выбирается для конкретной задачи и является неизменной.

Веса конкретного нейрона являются настраиваемыми параметрами. Они содержат знания нейрона, определяющие его поведение. Процесс настройки этих знаний с целью получения нужного поведения называется обучением.

Слоем нейронной сети называется множество нейронных элементов, на которые в каждый такт времени параллельно поступает информация от других нейронных элементов сети. Помимо слоев нейронов часто используется понятие входного распределительного слоя. Распределительный слой передает входные сигналы на первый обрабатывающий слой нейронных элементов.

Сверточные нейронные сети (СНС) – это вид нейронных сетей, которые хотя бы в одном из своих слоев в качестве преобразования используют операцию свертки. Типичная архитектура СНС представляет собой многоступенчатый каскад сетей прямого распространения, т. е. является однонаправленной, и включает сверточные слои, слои подвыборки (субдискретизации, пуллинга) и полносвязные слои. При этом используется преимущество двумерной структуры изображений и с помощью метода локальной связности, ограничивая количество связей между нейронами скрытого сверточного слоя и входными данными, т. е. каждый нейрон скрытого слоя связан только с локальным участком изображения. Сверточные слои основаны на математической операции свертки входного изображения с набором фильтров, которые описываются весами нейронов скрытого слоя. Данная операция в общем виде

представляется как

$$y_{r,c,k} = \sum_{i=-H/2}^{H/2} \sum_{j=-W/2}^{W/2} \sum_{d=1}^{D} x_{i+r,j+c,d} \cdot w_{i,j,d,k} + b_k,$$

где (r,c) – положение фильтра на предыдущем слое;

x – выход нейрона предыдущего слоя;

H, W, D – высота, ширина и глубина k-го фильтра;

$w_{i,j,k,d}$ – веса фильтра;

b_k – вес связи с постоянным значением (смещение).

Таким образом, можно сказать, что сверточный слой – это набор карт (карты признаков – матрицы), сформированный на основе синаптического ядра (сканирующее ядро, фильтр), которое скользит по всей области предыдущей карты и находит определенные признаки объектов. Шаг свертки – это количество пикселей, на которые перемещается матрица фильтра по входному изображению. Если шаг равен единице, то ядро свертки смещается каждый раз на один пиксель, если шаг равен двум, то происходит- смещение на два пикселя и т. п. Чем больше шаг, тем меньшего размера карты признаков получаются на выходе. Количество карт определяется требованиями к задаче: большое количество карт повышает качество распознавания, но увеличивает вычислительную сложность. Традиционно используется соотношение «один к двум», т. е. каждая карта предыдущего слоя связана с двумя картами сверточного слоя.

Размер у всех карт сверточного слоя одинаковый и вычисляется как

$$(w,h) = (mW - kW + 1, mH - kH + 1),$$

где mW, mH – ширина и высота предыдущей карты соответственно;

kW, kH – ширина и высота ядра соответственно.

Каждое ядро находит определенные признаки объектов, например, это могут быть вертикальные или горизонтальные линии, или же линии, расположенные под определёнными углами. Ядро представляет собой систему разделяемых весов или синапсов и является одной из главных осо-

бенностей СНС. В обычной многослойной ИНС много связей между нейронами, что замедляет процесс обнаружения. В СНС общие веса позволяют сократить число связей и находить один и тот же признак по всей области изображения.

Размер ядра нечетный от 3×3 до 7×7, так как обработка ведется относительно центрального элемента ядра. Необходимо чтобы размер картсвер- точного слоя был четным для исключения потери информации при уменьшении размерности в подвыборочном слое. Малый размер ядра не сможет выделить какие-либо признаки большего размера, а при большом размере ядра увеличивается количество связей между нейронами и, соответственно, время обучения и работы СНС.

Подвыборочный слой также имеет карты, их количество совпадает с количеством карт предыдущего сверточного слоя. Подвыборочный слой служит для уменьшения размерности карт предыдущего слоя. Такой подход позволяет снизить вариативность данных, повысить устойчивость СНС к небольшим изменениям изображений в пределах локальной области.

В процессе сканирования карты предыдущего сверточного слоя ядром подвыборочного слоя сканирующее ядро на нем не пересекается, что является отличием от организации свертки. Обычно каждая карта имеет ядро размером 2×2, и это позволяет уменьшить предыдущие карты сверточного слоя в два раза.

Для этого вся карта признаков разделяется на ячейки 2×2 элемента, из которых выбираются определенные значения. Наиболее используемые два типа объединения: максимальное (max pooling) и среднее (average pool- ing) (рисунок 2). Первое возвращает максимальное значение из покрытой ядром части изображения, второе возвращает среднее значение из всех значений покрытой ядром части.

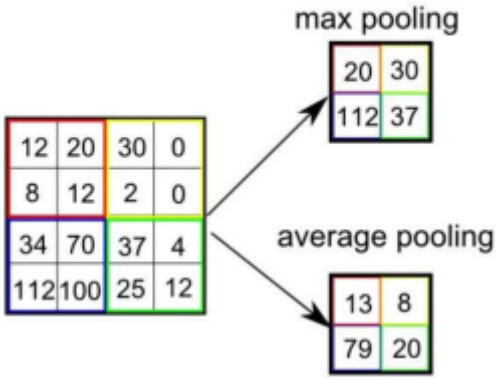


Рисунок 2 – Примеры типов объединений слоя подвыборки

Результат свертки после слоя подвыборки поступает на функцию активации, которая определяет выходной сигнал и является нелинейной, т. е. позволяет оставлять для дальнейшего анализа только значимые ней- роны, в качестве которых желательно использовать нелинейные активаци- онные функции, например, сигмоидального вида. Следует отметить, что ши- роко используются активационные функции, состоящие из линейных участ- ков, например, RELU (rectified linear unit) и PRELU, которая использует наклон отрицательной части в качестве параметра в процессе обучения с использованием алгоритма обратного распространения ошибки, что позволяет увеличить точность работы СНС.

Формально данный этап может быть описан выражением

$$y_p^l = \varphi\left(a^l \cdot \beta\left(x^{l-1}\right) + b^l\right),$$

где y_p^l – выход текущего подвыборочного слоя;

φ – функция активации;

a^l, b^l – коэффициенты сдвига;

β – функция выборки значений из локальной области.

Слои свёртки и подвыборки вместе образуют i-й слой свёрточной нейронной сети. За счет использования нескольких попеременных слоев свертки и подвыборки СНС позволяет получать представления, независимые от конкретного расположения локального признака в изображении, и

одинаковым образом реагировать на интересующие объекты, т. е. обнаруживать объекты на любом участке изображения. Количество таких слоёв может варьировать в зависимости от сложности изображений, чтобы точнее выделять признаки, однако увеличение слоев требует увеличения вычислительной мощности и большего объема памяти

СНС обеспечивает инвариантность по отношению к сдвигу, однако не предусматривает устойчивости к другим аффинным преобразованиям, например, вращению или зеркальному отражению. Для решения этой проблемы, как правило, используются эвристические методы, такие как выравнивание изображения по линии горизонта, использование многомасштабного представления и различных отражённых копий. Последний блок СНС является многослойным персептроном, машиной опорных векторов или другим классификатором. Он служит для задачи классификации, моделирует сложную нелинейную функцию, оптимизацией которой улучшается качество распознавания. Нейроны каждой карты предыдущего подвыборочного слоя связаны с одним нейроном скрытого слоя. Связи могут быть разными, например: только часть нейронов какой-либо из карт подвыборочного слоя быть связана с первым нейроном скрытого слоя, а оставшаяся часть со вторым, либо все нейроны первой карты связаны с нейронами первого и второго скрытого слоя. Пример топологии СНС для многоклассовой классификации показан на рисунке 3.

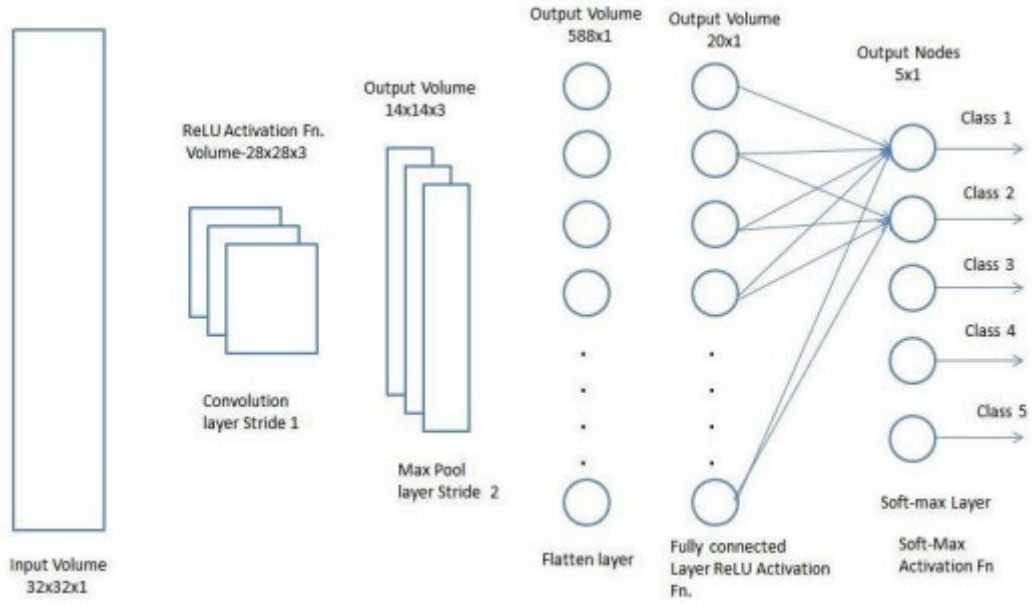


Рисунок 3 – Топология СНС для многоклассовой классификации

Среди известных моделей СНС, отличающихся архитектурой, точностью и быстродействием, для обнаружения объектов на изображениях и видеопоследовательностях следует выделить двухэтапные СНС, которые предполагают на первом этапе локализацию регионов-претендентов, а на втором этапе выполняется их классификация. Однопроходные СНС позволяют осуществлять локализацию и классификацию в составе одной модели, и не требуют обработки одного и того же изображения повторно.

Среди двухпроходных следует выделить модификации R-CNN (Region-based Convolutional Neural Networks) и ResNet (Residual Networks). Первая версия модели R-CNN основана на предварительном выделении регионов на изображении, вычислении признаков с использованием СНС (например, AlexNet) и применении классификатора для идентификации объектов. Данная модель получила развитие в версиях Fast R-CNN и Faster R-CNN. Архитектура Fast R-CNN направлена на уменьшение использования числа регионов на изображении за счет применения слоя ROI, который действует по принципу слоев субдискретизации, позволяя получать области уменьшенного размера. В Faster R-CNN предполагается выполнение двух этапов. На первом из них определяются области на изображении, в которых предположительно могут

быть расположены объекты, с применением глубокой полносвязной сети (Region Proposal Network). На втором этапе используется детектор Fast R-CNN, который выполняет поиск объектов в предложенных регионах. Модель Faster R-CNN неустойчива к зашумленным изображениям и требует значительных вычислительных затрат.

Архитектура СНС ResNet использует пропускающие (residual) соединения, которые позволяют минимизировать ухудшение качества работы при увеличении количества слоев СНС, если на некотором слое сети достигнут предел точности. Коэффициент ошибок составляет 3,57% в метрике top 5. В работе представлена модель Inseption-ResNet как развитие модели Inseption путем ввода замыкающих соединений (shortcut connection), которые при необходимости позволяют пропускать слой и, соответственно, обнулять его влияние на результат работы детектора. Это даёт возможность изменять архитектуру сети так, чтобы конечное количество слоёв определялось для конкретной задачи в процессе обучения, что позволяет уменьшить коэффициент ошибки до 3,1% в метрике top 5. Вместе с тем представленные модели характеризуются высокими временными затратами. Поэтому для решения многих задач используются усечённые варианты архитектур с уменьшенным количеством слоев (например, ResNet-34, ResNet-50 и др.), которые, однако, не позволяют достичь точности базовой модели.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

1. Изучить теоретический материал, представленный в теоретической

части.

2. Повторить все примеры из теоретической части.

Вычислить точность и полноту детектора при обнаружении объектов на 10 изображениях из базы данных MS COCO для различных степеней уверенности детектора в заданном диапазоне и с определенным шагом, например, от 0,3 до 0,9 с шагом 0,2. Каждое изображение должно содержать не менее 3 объектов разного класса. Данные представить в виде таблицы. Сделать выводы.

Таблица. – Представление результатов обнаружения объектов

Уверенность СНС	1		2		...	10	
	p	*r*	*p*	*r*		*p*	*r*

Пример обнаружения объектов на изображении показан на рисунке 4. Согласно визуальной оценке на изображении, представленном на этом рисунке, верно детектировано девять объектов, ложного обнаружения не было, пропущен один объект, т. к. не обнаружен один из четырех автомобилей в мозаике.

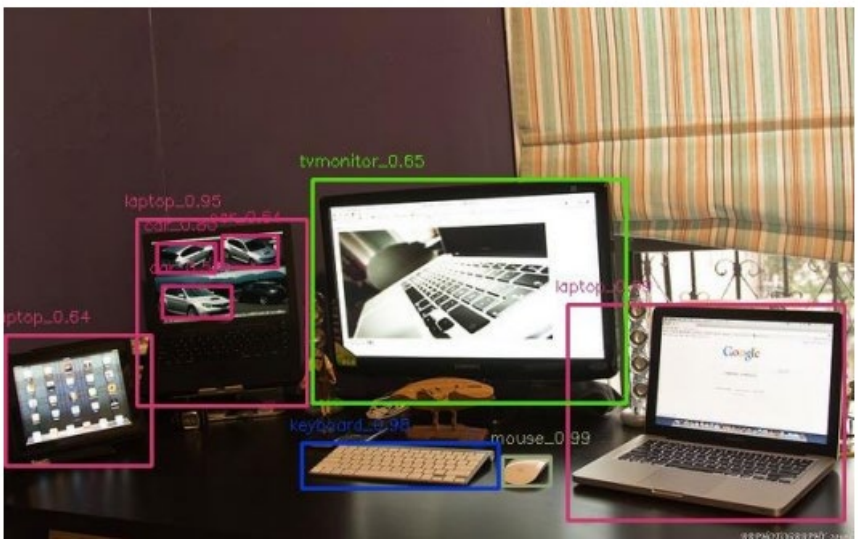


Рисунок 4 – Пример обнаружения объектов различных классов на изображении

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. Какие методы используются для выделения контуров и границ объектов на изображении?
2. В чем заключается применение компьютерного зрения в научных исследованиях? Приведите конкретные примеры.
3. Какие методы используются для распознавания образов и классификации на изображениях?
4. Какие аспекты следует учесть при применении нейронных сетей в компьютерном зрении?
5. Что такое аугментация данных, и как она может быть полезной при детекции объектов?

ЛАБОРАТОРНАЯ РАБОТА № 5. МОДЕЛИРОВАНИЕ ПРОСТЕЙШЕЙ НЕЙРОННОЙ СЕТИ

Цель: смоделировать простейшую нейронную сеть для распознавания рукописных цифр.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Распознавание рукописных цифр — это одна из классических задач в области машинного обучения и компьютерного зрения, которая имеет большое практическое значение. Эта задача часто используется как отправная точка для изучения более сложных методов обработки изображений и классификации данных. Рукописные цифры представляют собой пример данных, которые могут быть легко интерпретированы человеком, но для компьютера их распознавание является нетривиальной задачей из-за вариативности написания, шума и различий в стилях.

Распознавание рукописных цифр имеет множество практических применений, таких как:

1. Автоматическая обработка почтовых индексов: В почтовых службах для автоматической сортировки писем.
2. Банковская сфера: Для автоматического распознавания сумм на чеках и других финансовых документах.
3. Цифровизация документов: Для преобразования рукописных текстов в цифровой формат.
4. Образование: В системах автоматической проверки тестов и экзаменов.

Для решения задачи распознавания рукописных цифр часто используется набор данных MNIST (Modified National Institute of Standards and

Technology database). Этот набор данных содержит 60,000 тренировочных и 10,000 тестовых изображений рукописных цифр размером 28x28 пикселей. Каждое изображение сопровождается меткой, указывающей на соответствующую цифру (от 0 до 9).

Подходы к решению задачи

Существует несколько подходов к решению задачи распознавания рукописных цифр:

Традиционные методы машинного обучения: Использование алгоритмов, таких как k-ближайших соседей (k-NN), метод опорных векторов (SVM) и другие, с предварительной обработкой данных (например, выделение признаков).

Нейронные сети: Использование искусственных нейронных сетей, которые способны автоматически извлекать признаки из данных и обучаться на них. Простейшие нейронные сети, такие как многослойные перцептроны (MLP), могут достичь высокой точности на этой задаче.

Сверточные нейронные сети (CNN): Более сложные архитектуры, которые учитывают пространственную структуру изображений и показывают наилучшие результаты на задачах классификации изображений.

Целью данной работы является создание и обучение простейшей нейронной сети для распознавания рукописных цифр. Мы рассмотрим основные этапы построения нейронной сети, начиная с подготовки данных и заканчивая оценкой производительности модели. В процессе мы изучим ключевые концепции, такие как архитектура сети, функции активации, функция потерь, методы оптимизации и оценка качества модели.

Структура работы

Подготовка данных: Загрузка и предварительная обработка данных MNIST.

Построение модели: Определение архитектуры нейронной сети, выбор функций активации и инициализация весов.

Обучение модели: Описание процесса обучения, включая прямое и обратное распространение, а также обновление весов.

Оценка модели: Анализ производительности модели на тестовом наборе данных, использование метрик качества.

Этот теоретический обзор дает основу для понимания задачи распознавания рукописных цифр и подходов к ее решению с использованием нейронных сетей.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Для работы предлагается использовать Google Colaboratory — это бесплатная интерактивная облачная среда для работы с кодом от Google. Сервис нужен, чтобы писать код в jupyter notebook. Он функционирует по принципу облака, что позволяет работать над одним проектом целой командой.

Программа предоставляет доступ к графическим процессорам GPU и TPU, благодаря которым можно развивать приложения на основе нейросетей.

Как начать работу в Google Colab.

Для начала убедитесь, что вы вошли в аккаунт, а затем создайте отдельную папку под готовые проекты. После запуска блокнота нажмите «Создать» → «Ещё» → Google Colaboratory. Для переименования блокнота щёлкните на имя файла.

Задачи работы:

1Установить библиотеку matplotlib (pip install matplotlib)

3. Установить библиотеку машинного обучения tensorflow (pip install tensorflow)

4. Установить фреймворк Keras (pip install keras)

5. Аналогично практическому материалу лекции, построить нейронную сеть для распознавания рукописных цифр. Нейронная сеть должна быть обучена с помощью датасета MNIST, имеющемся во фреймворке Keras (можете воспользоваться готовым ipynb – файлом в доп материалах)

6. Загрузить тестовое изображение (сформировать самостоятельно)

7. Разработать алгоритм, находящий контуры цифр

8. Далее разработать алгоритм, который будет обрабатывать каждый найденный контур следующим образом: - вырезать прямоугольник с цифрой внутри - производить ресайз этого прямоугольника к разрешению 18*18 - добавлять черные полосы шириной 5 пикселей с каждой стороны прямоугольника, чтобы получить результирующий прямоугольник разрешением 28*28 пикселей Пример такого алгоритма:

```
preprocessed_digits = []
for c in contours:
    x,y,w,h = cv2.boundingRect(c)
    cv2.rectangle(img, (x,y), (x+w, y+h), color=(0, 255, 0), thickness=2)
    digit = thresh[y:y+h, x:x+w]
    resized_digit = cv2.resize(digit, (18,18))
    padded_digit = np.pad(resized_digit, ((5,5),(5,5)), "constant", constant_values=0)
    preprocessed_digits.append(padded_digit)

input_array = np.array(preprocessed_digits)
```

Таким образом, на выходе получаем массив изображений с цифрами.

9. Подаем полученный массив на вход нейронной сети (не забываем, что нужно преобразовать каждое изображение в вектор размерностью 28*28 = 784), выводим результат. Пример:

```
for digit in preprocessed_digits:
    prediction = model.predict(digit.reshape(1, 784))
    print ("Обработка изображения нейронной сетью \n\n")
    plt.imshow(digit.reshape(28, 28), cmap="gray")
    plt.show()
    print("\n\nРезультат: {}".format(np.argmax(prediction)))
```

Пример того, что должно получиться:

1. Изучить теоретический материал, представленный в теоретической части.
2. Повторить все примеры из теоретической части.
3. Нарисуйте от руки на белом листе цифры от 0 до 9 в любой последовательности. Отсканируйте / сфотографируйте его и повторите пункты с 7 по 9 для своего набора цифр. Сделайте выводы о точности обученной вами нейронной сети. Фото приложите в папку с лабораторной работой. Оцените результаты проделанной работы и напишите вывод.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. Что такое набор данных MNIST и почему он широко используется для задач распознавания рукописных цифр?
2. Какие основные этапы предварительной обработки данных необходимы перед обучением нейронной сети на наборе MNIST?
3. Какова роль входного слоя в нейронной сети для обработки изображений рукописных цифр?
4. Почему в качестве активационной функции в скрытых слоях часто используется функция ReLU (Rectified Linear Unit)?
5. Какие преимущества и недостатки имеет функция активации Softmax в выходном слое нейронной сети для задачи классификации?
6. Как работает процесс обратного распространения ошибки (backpropagation) в нейронной сети?
7. Какие метрики используются для оценки качества модели при распознавании рукописных цифр?
8. Почему метод оптимизации Adam часто используется для обучения нейронных сетей?
9. Какие проблемы могут возникнуть при обучении нейронной сети и как их можно устранить (например, переобучение, затухающие градиенты)?
10. В чем отличие простейшей полносвязной нейронной сети от сверточной нейронной сети (CNN) при решении задачи распознавания рукописных цифр?

ЛАБОРАТОРНАЯ РАБОТА № 6. АЛГОРИТМЫ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ.

Цель: изучить алгоритмы сегментации.

Формируемые компетенции: ПК-5, ПК-9

Теоретическая часть

Сегментация — это процесс разделение изображения на группы пикселей — сегменты. Также их называют суперпикселями. Данный процесс несёт в себе цель упросить последующий анализ изображения. Сегментация в результате выделяет отдельные объекты на изображении, что само по себе

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Для работы предлагается использовать Google Colaboratory — это бесплатная интерактивная облачная среда для работы с кодом от Google. Сервис нужен, чтобы писать код в jupyter notebook. Он функционирует по принципу облака, что позволяет работать над одним проектом целой командой.

Программа предоставляет доступ к графическим процессорам GPU и TPU, благодаря которым можно развивать приложения на основе нейросетей.

Как начать работу в Google Colab.

Для начала убедитесь, что вы вошли в аккаунт, а затем создайте отдельную папку под готовые проекты. После запуска блокнота нажмите «Создать» → «Ещё» → Google Colaboratory. Для переименования блокнота щёлкните на имя файла.

Далее определитесь с процессором. Если будете работать на слишком мощном процессоре, то вы можете вылететь из блокнота. Чтобы попасть в настройки процессора, выберите вкладку «Среда выполнения» и команду «Сменить среду управления».

Для начала загрузим изображение, а также преобразуем его, чтобы видеть только оттенки серого с помощью функции rgb2gray(). Для загрузки изображения был использован модуль io из scikit-image, а для последующего взаимодействия — класс Image библиотеки Pillow. Для вывода изображений на экран были использованы инструменты библиотеки matplotlib.

Рисунок 1 – Исходное изображение

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from skimage import io
from skimage.color import rgb2gray
```

```
from skimage.segmentation import morphological_geodesic_active_contour,
inverse_gaussian_gradient

from skimage.util import img_as_float

#необходимые библиотеки

image = io.imread('flowers.png')

draw = Image.fromarray(image)

image = np.array(image)

image = rgb2gray(image)

image = img_as_float(image)

plt.imshow(image, cmap='gray')
```

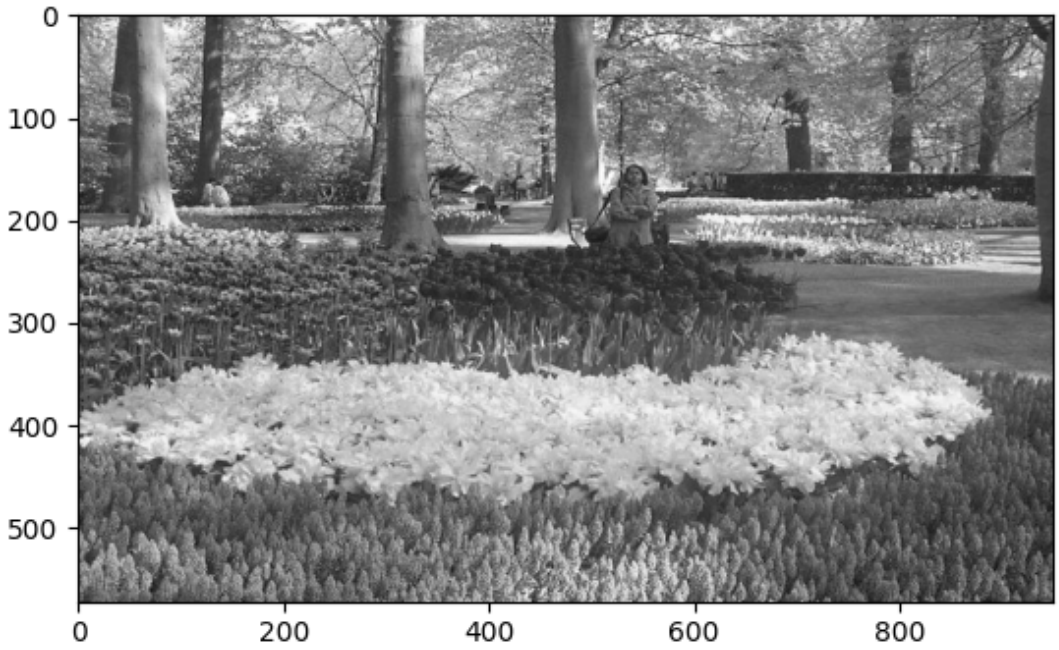


Рисунок 2 – Исходное изображение в серых оттенках

Далее выполним предварительную обработку изображения перед сегментацией. Используем функцию inverse_gaussian_gradient(). Она вычисляет величину градиента в изображении и инвертирует результат в диапазоне 0-1. Начальные области получат значение ближе к единице, а области близкие к границе — значения, близкие к 0.

```
gimage = inverse_gaussian_gradient(image)
```

Далее применим ещё одну модификацию, именуемую «Balloon Force». Существует проблема, выраженная в том, на плоских участках, где градиент равен 0, алгоритм не будет сходиться. Использование Balloon Force призвано решить эту проблему. Данная модификация основана на области (рамке), которая будет направлять контур в тех частях изображения, где информация, полученная с помощью градиента, не является достаточной (градиент слишком мал, чтобы «подтолкнуть» контур к границе).

Эта модификация уже встроена в метод morphological_geodesic_active_contour().

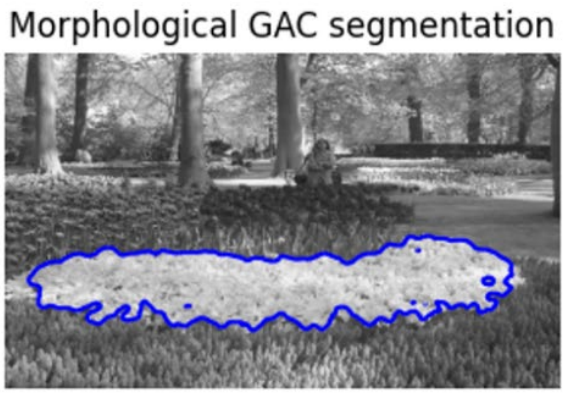
```
init_ls = np.zeros(image.shape, dtype=np.int8)  # создание

init_ls[350:-150, 100:-130] = 1                 # рамки

evolution = []   # для сохранения итераций

callback = store_evolution_in(evolution)

ls = morphological_geodesic_active_contour(gimage, 50, init_ls, smoothing=1,
balloon=1, iter_callback=callback)
```

Для наглядности я реализовал возможность сохранить результат итерации и показать его на экране (функция — store_evolution_in()). Вот как данный алгоритм справился с сегментацией цветочной клумбы.

```
fig, axes = plt.subplots(1, 2, figsize=(8, 8))

ax = axes.flatten()

ax[0].imshow(image, cmap="gray")

ax[0].set_axis_off()
```

```
ax[0].contour(ls, [0.5], colors='b')

ax[0].set_title("Morphological GAC segmentation", fontsize=12)

ax[1].imshow(gimage, cmap="gray")

ax[1].set_axis_off()

contour = ax[1].contour(evolution[0], [0.5], colors='r')

contour.collections[0].set_label("Starting Contour")

contour = ax[1].contour(evolution[5], [0.5], colors='g')

contour.collections[0].set_label("Iteration 5")

contour = ax[1].contour(evolution[-1], [0.5], colors='b')

contour.collections[0].set_label("Last Iteration")

ax[1].legend(loc="upper right")

title = "Morphological GAC Curve evolution"

ax[1].set_title(title, fontsize=12)

plt.show()
```

Данный код выводит 2 картинки. Слева — результат обработки. Справа можно отследить, как программа вела себя на 0-й, 5-й и последней итерации.



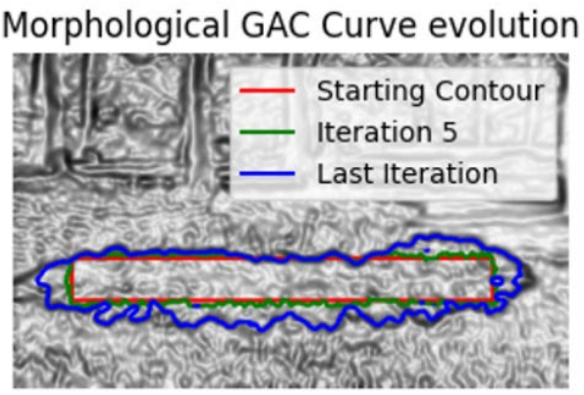


Рисунок 3 – Слева — результат обработки. Справа можно отследить, как программа вела себя на 0-й, 5-й и последней итерации

Стоит отметить, что при изменении размеров рамки Balloon Force можно добиться иных результатов.

SLIC

Данный алгоритм, в отличие от «Змеиного», основан на машинном обучении. Он генерирует на изображении суперпиксели с помощью кластеризации пикселей на основе их близости на изображении, а также цветового сходства. Упрощая, можно сказать, что суперпиксель — это группа пикселей, имеющих схожие характеристики, например, интенсивность пикселей.

```
from skimage.segmentation import slic

from skimage.segmentation import mark_boundaries

from skimage.util import img_as_float

from skimage import io

import matplotlib.pyplot as plt

import argparse
```

Используемые библиотеки

Итак, входные данные программы — это изображение. Я реализовал загрузку изображения в программу с помощью аргументов командной строки и параметра —image с помощью средств библиотеки argparse.

```
ap = argparse.ArgumentParser()

ap.add_argument("-i", "--image", required=True, help="Path to the image")

args = vars(ap.parse_args())
```

```
SLIC.py --image flowers.png
```

Код, реализующий загрузку файла через параметр командной строки при запуске программы и сама команда запуска

Затем преобразуем изображение так, чтобы оно описывалось не 8-битными целыми беззнаковыми числами, а float числами в промежутке 0-1.

```
image = img_as_float(io.imread(args["image"]))
```

Функция, которая выполняет сегментацию изображения с помощью данного алгоритма, называется slic(). Данная функция имеет лишь один обязательный параметр — само изображение. Я также указал количество суперпикселей, которое нужно сгенерировать — 100, 200 а затем 300. По умолчанию генерируется 100 сегментов.

```
for numSegments in (100, 200, 300):

   segments = slic(image, n_segments=numSegments)
```

После получения сегментов вывожу результаты сегментации с помощью средств библиотеки matplotlib, которая позволяет вывести изображение на экран. Также была использована функция mark_boundadies() для того, чтобы в изображении на экране подсветить выделенные сегменты.

```
for numSegments in (100, 200, 300):

   segments = slic(image, n_segments=numSegments)

   fig = plt.figure("Superpixels -- %d segments" % (numSegments))

   ax = fig.add_subplot(1, 1, 1)

   ax.imshow(mark_boundaries(image, segments))
```

```
    plt.axis("off")

plt.show()
```

Далее представлены результаты работы программы с количеством сегментов 100, 200 и 300 соответственно.

100 сегментов

200 сегментов

300 сегментов

В заключение стоит отметить, что оба алгоритма выполняются довольно быстро. Также приятно удивила точность сегментации при использовании SLIC.

4. Изучить теоретический материал, представленный в теоретической части.
5. Повторить все примеры из теоретической части.
6. Подобрать изображения, подходящие для сегментации. Повторить все представленные выше этапы. Оценить результат проделанной работы и написать вывод.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Методика и порядок выполнения работы» с приведением исходного кода программы.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Контрольные вопросы:

1. В чем заключается роль алгоритмов кластеризации при сегментации изображений?
2. Какие аспекты следует учитывать при применении компьютерного зрения в медицине?
3. Что такое ступенчатая диаграмма?
4. Какие технологические задачи решаются с помощью систем технического зрения в промышленности?
5. Какое значение имеет преобразование пространственных координат при анализе изображений?

Список основной литературы

1. Клетте, Р. Компьютерное зрение. Теория и алгоритмы: учебник / Р. Клетте; перевод с английского А. А. Слинкина. — Москва: ДМК Пресс, 2019. — 506 с. — ISBN 978-5-97060- 702-2. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/131691

2. Станкевич, Л.А. Интеллектуальные системы и технологии: учебник и практикум для вузов / Л.А. Станкевич. — 2-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 495 с. — (Высшее образование). — ISBN 978-5-534-16238-7. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/530657

3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для среднего профессионального образования / Д. Ю. Федоров. — 5-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 227 с. — (Профессиональное образование). — ISBN 978-5-534-17319-2. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/532858

Список дополнительной литературы

4. Селянкин, В.В. Компьютерное зрение. Анализ и обработка изображений: учебное пособие / В. В. Селянкин. — Санкт-Петербург: Лань, 2019. — 152 с. — ISBN 978-5-8114-3368- 1. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/113938

5. Лаврищева, Е.М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е.М.Лаврищева. — 2-е изд., испр. и доп.— Москва: Издательство Юрайт, 2023. — 432 с.— (Высшее образование). — ISBN 978-5-534-07604-2. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL:https://urait.ru/bcode/513067

6. Чернышев, С. А. Основы программирования на Python: учебное пособие для среднего профессионального образования / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 349 с. — (Профессиональное образование). — ISBN 978-5-534-17056-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/532292

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по организации самостоятельной работы обучающихся
по дисциплине **«Технологии компьютерного зрения»**
Направление подготовки 09.03.02 «Информационные системы и технологии»
Направленность (профиль) «Прикладное программирование в интеллектуальных информационных системах»

Ставрополь

2026

Введение

Подготовка бакалавра – сложный, комплексный процесс, в котором органически сочетается решение всех основных задач – образовательных, воспитательных и развивающих. Деятельность преподавателя в учебном процессе – ведущая. Однако, как бы ни были совершенны применяемые преподавателем в учебном процессе форм и методов обучения, без активной познавательной деятельности самих обучаемых они не могут давать желаемых результатов. Поэтому особое значение приобретает проблема сознательного отношения самих студентов к приобретению знаний, умений и навыков в результате самостоятельной работы.

Самостоятельная работа - это познавательная деятельность, выполняемая студентами самостоятельно, при частичном руководстве преподавателя. Самостоятельная работа предусмотрена учебной программой. Самостоятельная работа является управляемым процессом. Благодаря этому познавательно-практическая деятельность студентов становится целенаправленной, творчески осмысленной, содержательной.

Процесс управления самостоятельной работы служит главным целям обучения:

- усвоение, закрепление, совершенствование знаний в объеме вузовских программ;
- приобретение умений и навыков, составляющих содержание подготовки выпускника высшей школы;
- максимальная активность каждого обучающегося в овладении профессиональных компетенций (ПК-9, ПК-5), формируемых в результате освоения дисциплины.

1. Самостоятельная работа студентов на лабораторных и практических занятиях

1.1. Аудиторные занятия

Задачи:

- • расширение, углубление и детализация знаний, полученных на лекциях;
- • повышение уровня усвоения учебного материала (от уровня информации,
- полученной на лекциях, до уровня знаний за счет привития умений и навыков);
- • развитие научного мышления и речи студентов;
- • проверка и учет знаний;
- • развитие познавательной активности и привитие навыков самостоятельной
- работы, особенно с дополнительной и специальной литературой;
- • привитие навыков ведения коллективной беседы, участия в творческой
- дискуссии, умения аргументированно отстаивать свои взгляды.

Основное требование к подготовке и участию студентов в занятиях заключается в том, что студент должен самостоятельно готовиться к занятиям и творчески в них участвовать.

Этапы подготовки к занятиям включают:

- повторение уже имеющихся знаний по конспекту, а затем по учебнику;
- углубление знаний по теме с использованием рекомендованной
- литературы;
- выполнение конкретного задания (решение задач, составление отчетов и т.п.).

Особое место занимают лабораторные работы. Главная цель которых – быть связующим звеном теории изучаемой дисциплины с практической реализацией, что позволяет углублять и закреплять теоретические положения, получаемые студентами на лекции, проверять их применение в практике экспериментальным путем, знакомить студентов с оборудованием, приборами, вычислительной техникой, изучать на практике методы научных исследований.

Студенты обеспечиваются методическими указаниями к лабораторным работам, содержащими теоретическую информацию и конкретные практические задания.

Для студентов младших курсов указания даются подробно и детализировано. Для студентов старших курсов предусматривается большая самостоятельность при выполнении заданий. Т.е. они должны уже демонстрировать переход от уровня умений к уровню навыков.

Оформление лабораторных работ должно быть максимально приближено к уровню, на котором ведется экспериментальная научно-исследовательская работа в конкретной предметной области.

1.2. Самостоятельная работа студентов во внеаудиторное время.

Кроме наиболее распространенных в вузах форм самостоятельной работы во внеаудиторное время (написание рефератов, контрольных работ, курсовых работ/проектов и т.д.) учебные планы включают такие виды и соответственно такой компонент фонда оценочных средств, как собеседование (по конкретным вопросам) и выполнение предлагаемых тем для самостоятельной учебно-исследовательской работы.

2. Указания студентам по подготовке к собеседованию и материалов по темам самостоятельной работы

2.1. Указания представлены в виде рекомендаций студентам в ходе изучения вопросов, выносимых на собеседование и материалов тем самостоятельной работы.

Своевременное и качественное выполнение самостоятельной работы базируется на соблюдении настоящих рекомендаций и изучении рекомендованной литературы. Студент может дополнить список использованной литературы современными источниками, не представленными в списке рекомендованной литературы, и в дальнейшем использовать собственные подготовленные учебные материалы при написании рефератов, курсовых и выпускных квалификационных работ.

2.2. Рекомендации

2.2.1. Прежде чем приступить к изучению вопросов для собеседования и предложенной темы самостоятельной работы, необходимо.

Во-первых:

- уяснить суть вопроса (темы) на самостоятельную работу;
- провести подбор рекомендованной литературы;
- составить план работы, в котором определяются основные пункты предстоящей подготовки.

Составление плана дисциплинирует и повышает организованность в работе.

Во-вторых:

- начинать надо с изучения рекомендованной литературы.

Необходимо помнить, что на лекции обычно рассматривается не весь материал, а только его часть. Остальная его часть восполняется в процессе самостоятельной работы. В связи с этим работа с рекомендованной литературой обязательна (см. Приложение 1). Особое внимание при этом необходимо обратить на содержание основных положений и выводов, объяснение явлений и фактов, уяснение практического приложения рассматриваемых теоретических вопросов.

В процессе этой работы студент должен стремиться понять и запомнить основные положения рассматриваемого материала, примеры, поясняющие его, а также разобраться в иллюстративном материале.

Заканчивать подготовку следует составлением плана (оглавления) по изучаемому материалу. Это позволит дать целостный, сжатый ответ при собеседовании и представить целостную картину по предложенной тематике.

В процессе самостоятельной работы рекомендуется взаимное обсуждение материала (между студентами), во время которого закрепляются знания, а также приобретается практика в изложении и разъяснении полученных знаний, развивается речь.

При необходимости студент может обращаться за консультацией к преподавателю. Идя на консультацию, студенту необходимо хорошо продумать вопросы, которые требуют разъяснения.

При подготовке к собеседованию и выполнению самостоятельной работы по теме студент должен вести записи. В результате у него создается свой индивидуальный фонд дополнительных материалов для быстрого повторения прочитанного, для накопления знаний. Особенно важны и полезны записи тогда, когда в них находят отражение мысли, возникшие при самостоятельной работе.

Основные формы записи: план (простой и развернутый), выписки, тезисы.

Результаты этих записей могут быть представлены в различных формах.

План – это схема прочитанного материала, краткий (или подробный)

перечень вопросов, отражающих структуру и последовательность материала:

краткий

- воспроизведение наиболее важных положений и фактов источника;

подробный (развернутый)

- детализированный план, в котором достаточно подробные записи приводятся по тем пунктам плана, которые нуждаются в пояснении;
- это четко и кратко изложенные (сформулированные) основные положения в результате глубокого осмысливания материала;
- в нем могут присутствовать выписки, цитаты, тезисы.

Подробно составленный план позволяет эффективно использовать материал и получить желаемый результат.

2.2.2. При отчете студента о проделанной самостоятельной работе по предложенной теме, преподаватель может предложить студенту алгоритм действий, рекомендовать еще раз внимательно прочитать весь материал и все записи по теме самостоятельной работы, тщательно продумать свое устное выступление.

2.2.3. Устный отчет должен строиться свободно, убедительно и аргументированно.

2.2.4. Преподаватель следит, чтобы отчет не сводился к простому воспроизведению текста, не допускается и простое чтение. Необходимо, чтобы студент проявлял собственное отношение к тому, о чем он говорит, высказывал свое личное мнение, понимание, обосновывал его и мог сделать правильные выводы из сказанного. При этом студент может обращаться к записям собранного материала, использовать знания факты и наблюдения современной жизни и т. д.

2.2.5. В заключение преподаватель оценивает результаты работы студента по данной теме.

3. Критерий оценки самостоятельной работы студента

Оценка «зачтено» при ответах на вопросы при собеседовании и по итогам устной защиты выполненной самостоятельной работы выставляется студенту, если студент показал знание учебного материала. При этом возможно допустил неточные формулировки основных понятий и терминов или ряд незначительных неточностей, не исказивших существенно суть ответа.

Оценка «не зачтено» при ответах на вопросы при собеседовании и по итогам устной защиты выполненной самостоятельной работ выставляется студенту, если студент не знает значительной части учебного материала, допускает неправильную формулировку основных понятий и терминов, существенно исказившие суть вопроса.

Оценка «не зачтено» выставляется также, если студент не отвечает на вопросы при собеседовании, отсутствует план выполненной работы, нет объяснений основных понятий и терминов.

4. Заключение

Успешное освоение курса предполагает активное, творческое участие студента путем планомерной, повседневной работы как на лекциях, лабораторных (практических) занятиях, так и к подготовке ответов на вопросы к собеседованию и выполнению самостоятельной работы по предлагаемым темам.

Приложение 1

Рекомендации студентам по изучению рекомендованной литературы

Изучение дисциплины следует начинать с проработки рабочей программы по изучаемой дисциплине, особое внимание, уделяя целям и задачам, структуре и содержанию курса.

Студентам рекомендуется получить учебную литературу по дисциплине, необходимую для эффективной работы на всех видах аудиторных занятий, а также для самостоятельной работы по изучению дисциплины.

Приложение 2

Примерная тематика заданий для самостоятельной работы студентов

Студенты выполняют самостоятельную работу в соответствии с индивидуальным заданием, которое определяется примерными темами.

Примерные темы самостоятельной работы

1. Задачи компьютерного зрения
2. Дайте определение понятию пространственная реконструкция
3. Какие факторы помогают и мешают при распознавании изображения
4. Дайте определение цифровому и аналоговому изображению
5. Вычисление характерных признаков изображения

6. Какие типы цифровых изображений существуют?
7. Пиксели и окрестности пикселей. Маски
8. Пакет NumPy и SciPy назначение и отличие
9. Преобразование уровня яркости
10. Гистограмма, выравнивание гистограммы