

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ  
компьютерной безопасности и  
криптографии

**Решение задачи распознавания зрительных образов нейронными сетями**

РЕФЕРАТ

студента 5 курса 531 группы  
направления 10.05.01—Компьютерная безопасность  
факультета КНиИТ

Бочкарева Матвея Сергеевича

Проверил

доцент

\_\_\_\_\_

подпись, дата

И. И. Слеповичев

Саратов 2024

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1 Задача распознавания зрительных образов нейронными сетями .....	5
1.1 Понятие распознавания зрительных образов.....	5
1.2 Проблемы, решение которых поможет усовершенствовать технологию распознавания образов .....	7
1.3 Нейросети для распознавания зрительных образов .....	8
1.3.1 CNN .....	8
1.3.2 Трансформеры .....	10
1.3.3 Автоэнкодеры .....	12
1.4 Обучение и тестирование нейронных сетей .....	14
1.4.1 Предварительная обработка изображений для обучения нейронных сетей .....	14
1.4.2 Данные для обучения .....	16
2 Демонстрация классификации изображений с помощью нейронной сети ..	18
ЗАКЛЮЧЕНИЕ .....	24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	25
ПРИЛОЖЕНИЕ А .....	27

## ВВЕДЕНИЕ

История развития методов распознавания зрительных образов берет свое начало в середине 20-го века, когда начались первые исследования и эксперименты в области компьютерного зрения и обработки изображений. С появлением микропроцессоров и возросшей вычислительной мощности исследователи начали разрабатывать алгоритмы для таких задач, как распознавание лиц. В начале 2000-х годов произошел значительный прорыв с внедрением методов глубокого обучения и нейронных сетей. Развитие сверточных нейронных сетей (CNNs) в последние годы значительно повысило точность и эффективность систем распознавания образов.

Ключевыми фигурами в этой области являются Джеффри Хинтон, чьи исследования внесли существенный вклад в развитие CNNs, и Ян Лекусн – основоположник глубокого обучения (deep learning) и создатель архитектуры CNN. Итальянский специалист в области компьютерной нейробиологии Томазо Поджо также внес важный вклад в понимание принципов процессов восприятия – как машин, так и людей.

Распознавание зрительных образов – это фундаментальная проблема компьютерного зрения, и нейронные сети зарекомендовали себя как мощный инструмент для решения этой проблемы. Распознавание зрительных образов включает в себя классификацию изображений по различным категориям в зависимости от их содержания. Важно отметить, что это действительно сложная задача вследствие высокой вариативности и сложности изображений. Значительный прогресс в распознавании зрительных образов за последнее десятилетие обусловлен развитием нейронных сетей и глубокого обучения. Эти методы помогли достигнуть поразительного уровня точности и надежности визуального восприятия.

Целью данной работы является изучение и систематизация методов распознавания зрительных образов с применением нейронных сетей.

Для достижения поставленной цели, необходимо сформулировать ряд задач:

- Описание основных принципов работы нейронных сетей, используемых в задачах распознавания зрительных образов;
- Изучение основных аспектов процесса обучения нейронных сетей для распознавания зрительных образов;
- Изучение процесса предобработки изображений;
- Рассмотрение основных наборов данных, используемых для обучения и тестирования нейронных сетей в задачах распознавания зрительных образов;
- Приведение демонстрации классификации изображений с помощью Python;
- Формулирование выводов о проделанной работе.

## **1 Задача распознавания зрительных образов нейронными сетями**

Человек не задумывается, когда пытается отличить лошадь от коровы или фуру от автобуса, – мозг делает это автоматически. Искусственному интеллекту же нужны для обучения математические формулы и технологии машинного обучения на основе датасетов.

В данном разделе будут рассмотрены базовые принципы работы нейронных сетей в рамках распознавания зрительных образов. Также проведено сравнение основных архитектур нейронных сетей, изучен процесс обучения нейронных сетей и рассмотрены основные датасеты, на которых и происходит обучение.

Также необходимо внести ясность, что задача распознавания зрительных образов и задача обработки изображений нейросетями – это не одно и то же, хотя они тесно связаны и часто используются вместе. Распознавание зрительных образов – это более широкий термин, охватывающий множество задач, связанных с анализом и интерпретацией визуальных данных. Обработка изображений – это более узкая и конкретная задача, которая является частью общего процесса распознавания зрительных образов. Обе задачи часто решаются с использованием нейронных сетей, но их цели и объем различны.

### **1.1 Понятие распознавания зрительных образов**

Распознавание образов – это процесс извлечения значимой информации из разнообразных данных и их классификации на основе общих признаков [1]. Эти мощные технологии сегодня широко применяются во многих сферах, в том числе:

- Военное дело: анализ и обработка разведывательной информации, распознавание целей на основе данных с дронов и спутников;

- Образование: автоматизированная проверка тестов и распознавание рукописного текста;
- Медицина: анализ медицинских изображений, таких как рентгеновские снимки, МРТ, КТ. Алгоритмы помогают выявлять патологии (опухоли и аномалии) с высокой точностью;
- Безопасность и видеонаблюдение: анализ видеопотока для обнаружения подозрительного поведения или несанкционированных действий и технологии распознавания лиц;
- Транспорт: анализ дорожной обстановки, распознавание пешеходов, знаков, препятствий и других автомобилей (важнейший элемент в разработке беспилотных автомобилей);
- Промышленность: мониторинг состояния оборудования для предотвращения поломок и контроль качества продукции на производственных линиях с использованием машинного зрения;
- Развлечения и медиа: классификация и поиск контента в больших медиаархивах; фильтры, изменяющие внешность, в приложениях;
- Финансы: анализ рукописных подписей и других биометрических данных и борьба с мошенничеством (распознавание подозрительных транзакций или поддельных документов);
- Интернет вещей (IoT): в «умных» устройствах и системах IoT-методы распознавания образов могут применяться для улучшения взаимодействия между устройствами, автоматического управления бытовыми системами и «умными» городами;
- Маркетинг и реклама: распознавание образов позволяет компаниям анализировать данные о потребителях, предоставляя персонализированные рекомендации и рекламные коммуникации, улучшая таким образом опыт пользователей.

## **1.2 Проблемы, решение которых поможет усовершенствовать технологию распознавания образов**

Нейронные сети произвели революцию в области компьютерного зрения, позволив машинам распознавать образы и анализировать изображения. Они становятся все более популярными благодаря своей способности изучать сложные паттерны и особенности. Особенно сверточные нейронные сети (CNN) – самый популярный тип нейронных сетей, используемых при обработке изображений.

Но в последнее время все большую популярность приобретают также визуальные преобразователи (ViT), благодаря прорывным достижениям генеративных предварительно обученных трансформеров (GPT) и других архитектур, основанных на преобразовании, в области обработки естественного языка.

Способ обработки зрительных образов зависит от архитектуры сети и задачи, которую мы должны решить. Некоторые из наиболее распространенных решений, которые должны усовершенствовать технологию распознавания образов, представлены ниже:

- Задача классификации – включает присвоение изображению метки или категории (например, содержит ли изображение кошку или собаку);
- Задача локализации – где на изображении находится интересующий объект (например, на фотографии леса выделить деревья или метку на стволе дерева);
- Задача детекции – сочетает локализацию и классификацию, когда машине нужно найти и идентифицировать несколько объектов на изображении одновременно (например, компьютер должен найти и определить разные машины на фотографии шоссе);
- Задача сегментации и другие [2].

### 1.3 Нейросети для распознавания зрительных образов

Нейросети, которые будут рассмотрены в данном подразделе направлены на извлечение основных характеристик или признаков исходных образов и изображений, а также классификацию этих образов и на решение задач оптимизации.

Применяются следующие архитектуры:

- Сверточные нейронные сети (convolutional neural networks — CNN);
- Трансформеры (visual transformers);
- Автоэнкодеры (autoencoders) и другие [1, 2].

Нейронные сети демонстрируют высокую точность при решении задач распознавания образов, особенно если доступны обширные обучающие данные. Эти методы подходят для различных задач, таких как классификация, сегментация и обнаружение объектов. Благодаря способности нейросетей автоматически выявлять сложные особенности изображений, они эффективно справляются с объектами, имеющими разнообразные текстуры и формы.

Однако у нейросетевых подходов есть и недостатки. Для достижения высокой точности требуется большое количество данных, что может стать проблемой при их ограниченном объеме. Кроме того, обучение и использование таких моделей требует значительных вычислительных мощностей и специализированного оборудования. Некачественный подбор данных может привести к переобучению или снижению эффективности модели [1].

#### 1.3.1 CNN

Сверточные нейронные сети (Convolutional Neural Networks, CNN) – это тип нейронных сетей, специально разработанный для анализа данных с пространственной структурой, таких как изображения и видеоролики. Данный подход был впервые предложен Яном Лекуном в 1988 году и относится к области глубокого обучения [1].



Для обучения CNN применяется алгоритм обратного распространения ошибки. По своей структуре они напоминают воронку: анализ начинается с общей информации и постепенно переходит к более детальным аспектам. Архитектура CNN представлена на рисунке 1.

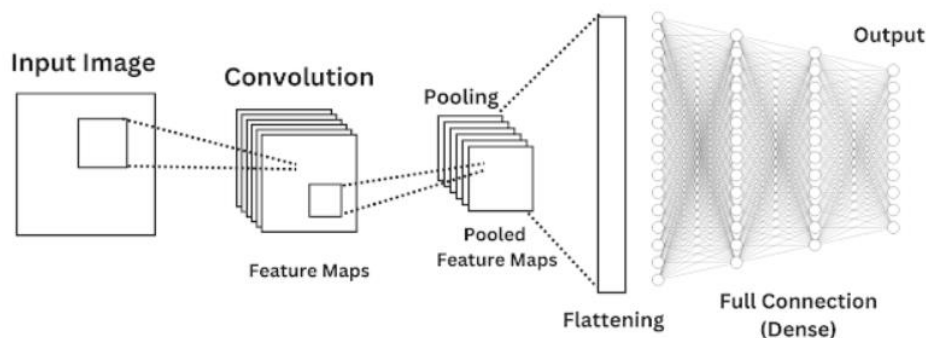


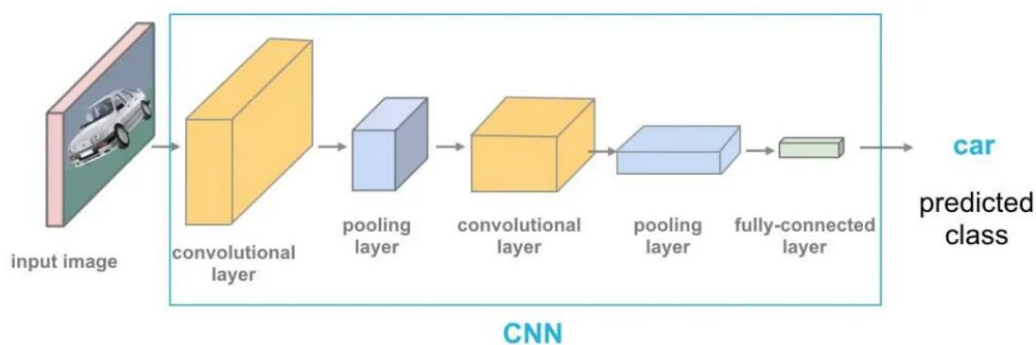
Рисунок 1 – Архитектура CNN [3]

Свертка позволяет сети игнорировать лишние данные и сосредотачиваться только на ключевых характеристиках объекта. На рисунке 2 показано, как выглядит изображение до и после свертки.



Рисунок 2 – Свертка [3]

Эти сети используются для решения задач распознавания и классификации, основываясь на принципах, аналогичных работе человеческого мозга (см. рис. 3). Одной из первых и наиболее известных сетей данного типа стала AlexNet, разработанная Алексеем Крижевским и Ильей Суцкевером, выпускниками Университета Торонто. Эта архитектура стала важным этапом в развитии машинного обучения.



3

Рисунок 3 – Этапы CNN [4]

Основные преимущества CNN включают их высокую точность при решении задач компьютерного зрения, таких как классификация, сегментация и обнаружение объектов. Эти сети способны автоматически извлекать важные признаки из входных данных, что упрощает обработку изображений и устраняет необходимость вручную создавать признаки. Гибкость архитектуры CNN позволяет адаптировать их для различных целей, например, для распознавания лиц или анализа медицинских снимков. Кроме того, существуют предобученные модели, которые можно настроить для конкретных задач [3].

Однако CNN имеют и недостатки. Они требуют значительных вычислительных ресурсов и большого объема данных для обучения. Их работа сложна для интерпретации, а некоторые модели склонны «запоминать» обучающие данные, что может снижать их способность к обобщению на новых наборах данных [3].

### 1.3.2 Трансформеры

Трансформеры — это современная архитектура нейронных сетей (см. рис 4), созданная для работы с последовательными данными, такими как текст или временные ряды. Они часто превосходят сверточные сети в ряде задач, показывая более высокую точность. Трансформеры применяются во многих областях, но пока не получили столь широкого распространения [5].

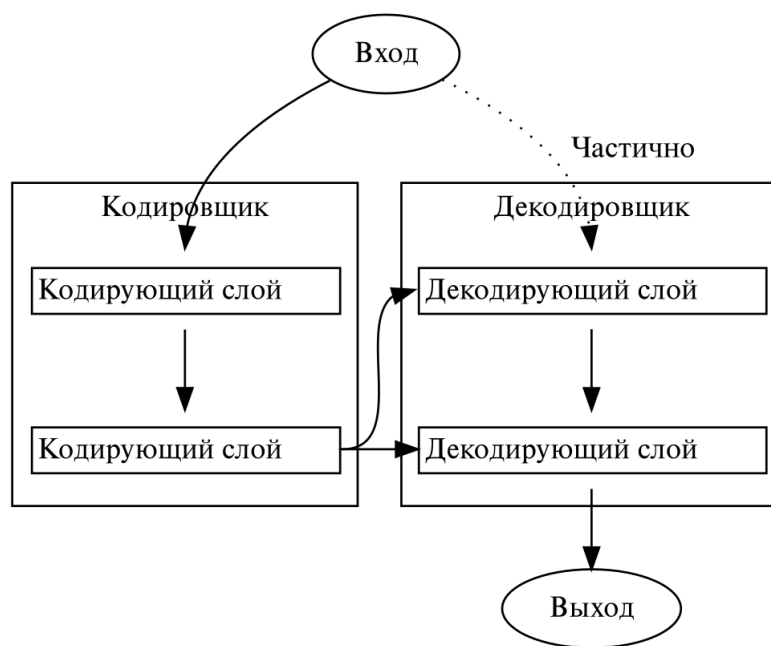


Рисунок 3 – Схема трансформера [5]

Одним из ключевых преимуществ трансформеров является их способность обрабатывать данные параллельно, что делает их особенно эффективными на многопроцессорных системах. Эти сети также хорошо справляются с анализом длинных текстов или последовательностей данных, не теряя при этом производительности. Их использование охватывает широкий спектр задач: от обработки изображений (например, для классификации, сегментации и обнаружения объектов) до машинного перевода, извлечения информации и других сложных задач. Центральным элементом трансформеров является механизм внимания, который позволяет сети сосредотачиваться на наиболее важных частях входных данных, повышая точность выполнения задач.

Однако трансформеры не лишены недостатков. Они требуют значительных вычислительных ресурсов и большого объема данных для обучения. Их архитектура может быть сложной для настройки, что требует глубоких знаний и опыта. Кроме того, результаты работы трансформеров порой сложно интерпретировать, что может создавать трудности в объяснении полученных решений [1].

### 1.3.3 Автоэнкодеры

Автоэнкодеры – это нейронные сети, которые могут научиться сжимать и реконструировать входные данные, такие как изображения, используя скрытый слой нейронов (см. рис. 4). Модель автоэнкодера состоит из трех частей: кодера, кода и декодера [6].

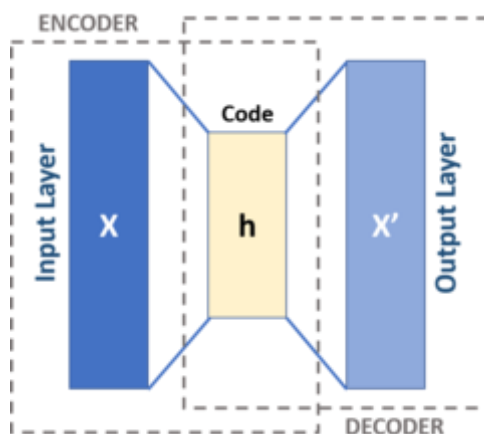


Рисунок 4 – Схема автоэнкодера [6]

Кодер – модуль, сжимающий входные данные в кодированное представление, которое обычно на несколько порядков меньше, чем входные данные. Кодер состоит из набора сверточных блоков, за которыми следуют объединенные модули или простые линейные слои, которые сжимают входные данные модели в компактную секцию, называемую «кодом».

Код – самая важная часть нейронной сети и, по иронии судьбы, самая маленькая. Код существует для того, чтобы ограничить поток информации, поступающей в декодер из кодера, таким образом, позволяя передавать только самую важную информацию. Код помогает нам сформировать представление знаний о входных данных.

Код, как сжатое представление входных данных, предотвращает запоминание нейронной сетью входных данных и переобучение модели. Чем меньше код, тем ниже риск переобучения. Этот слой обычно реализуется как линейный слой или как тензор, если используются свертки.

Декодер – компонент декодера в нейронной сети выполняет роль интерпретатора для кода.

Декодер помогает сети «декомпрессировать» представления знаний и восстанавливать данные из их закодированной формы. Затем выходные данные сравниваются с эталонными значениями (ground truth) (см. рис. 5). Обычно декодер реализуется с использованием транспонированных свёрток, если мы работаем с изображениями, или линейных слоёв.

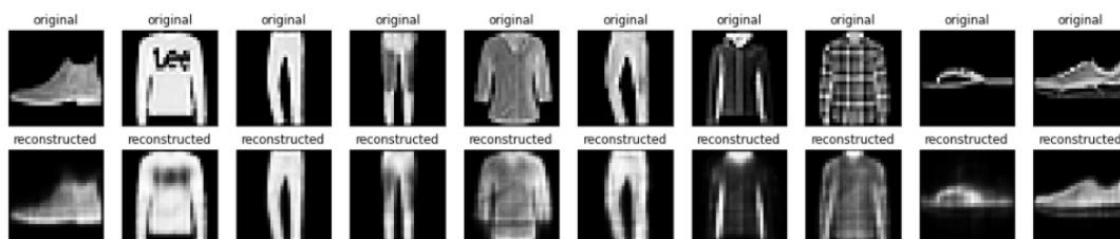


Рисунок 5 – Входные и выходные данные автоэнкодера [7]

Ранее описанная структура является общим обзором. На самом деле существует несколько типов автокодировщиков. Вот основные из них:

- Недоопределённые автокодировщики (Undercomplete Autoencoders) – восстанавливают изображение из сжатой области кода. Используются для уменьшения размерности данных, так как могут обучаться нелинейным зависимостям, в отличие от метода главных компонент;
- Шумоподавляющие автокодировщики (Denoising Autoencoders) – удаляют шум с изображений, обучаясь на зашумлённых версиях данных. Применяются, например, для удаления водяных знаков;
- Разреженные автокодировщики (Sparse Autoencoders) – ограничивают активации нейронов, чтобы предотвратить переобучение, активируя лишь небольшое число нейронов для каждого наблюдения;
- Контрастные автокодировщики (Contrastive Autoencoders) – обеспечивают стабильность кодировки при небольших изменениях входных данных, добавляя ограничение в функцию потерь;

- Вариационные автокодировщики (Variational Autoencoders, VAE) – могут генерировать данные, добавляя ограничение на латентное пространство, чтобы оно следовало нормальному распределению;
- Векторно-квантованные вариационные автокодировщики (VQ-VAE) – создают дискретное латентное представление, используя кодовую книгу для уменьшения сложности пространства [8].

## 1.4 Обучение и тестирование нейронных сетей

Данные для обучения нейронных сетей необходимо заранее подготавливать. В данном подразделе будет рассмотрено, какие этапы проходят, в частности, изображения, прежде чем они будут переданы нейронным сетям.

Кроме того, будут рассмотрены готовые популярные датасеты для тренировки нейронных сетей.

### 1.4.1 Предварительная обработка изображений для обучения нейронных сетей

Алгоритмы распознавания изображений нейронной сети зависят от качества набора данных – изображений, используемых для обучения и тестирования модели. Вот несколько важных параметров для подготовки данных изображений [9].

*Размер изображения* – более качественное изображение дает модели больше информации, но требует больше узлов нейронной сети и больше вычислительной мощности для обработки.

*Количество изображений* – чем больше данных вы подадите в модель, тем точнее она будет, но убедитесь, что обучающий набор представляет реальную популяцию.

*Количество каналов* – серые изображения имеют 2 канала (черный и белый), а цветные изображения обычно имеют 3 цветовых канала (красный, зеленый, синий / RGB), причем цвета представлены в диапазоне  $[0, 255]$ .

*Соотношение сторон* – как правило, модели нейронных сетей предполагают, что входное изображение имеет квадратную форму.

*Масштабирование изображения* – после того, как все изображения преобразованы в квадрат, можно масштабировать каждое изображение. Существует множество методов увеличения и уменьшения масштаба, которые доступны в виде функций в библиотеках глубокого обучения.

*Среднее значение* – можно посмотреть на «среднее изображение», вычислив средние значения для каждого пикселя во всех обучающих примерах, чтобы получить информацию о базовой структуре в изображениях.

*Нормализация входных данных изображения* – гарантирует, что все входные параметры (пиксели в данном случае) имеют равномерное распределение данных. Это ускоряет сходимость при обучении сети. Можно провести нормализацию данных, вычитая среднее значение из каждого пикселя, а затем разделив результат на стандартное отклонение.

*Сокращение размерности* – можно решить свернуть каналы RGB в канал оттенков серого.

*Аугментация данных* – включает в себя увеличение существующего набора данных путем добавления искаженных версий имеющихся изображений, включая масштабирование и поворот. Это делается для того, чтобы нейронная сеть столкнулась с разнообразием вариаций. Таким образом, нейронная сеть с меньшей вероятностью будет идентифицировать нежелательные характеристики в наборе данных. Пример аугментации данных представлен на рисунке 6.

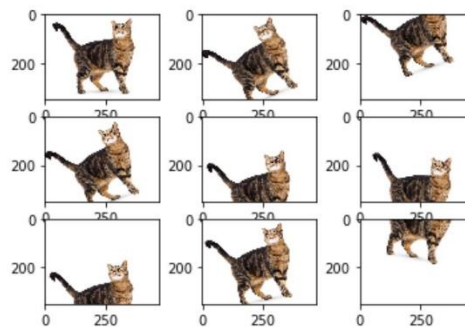


Рисунок 6 – Аугментация изображения [10]

#### 1.4.2 Данные для обучения

Для обучения компьютера распознаванию образов требуется множество размеченных данных, где объекты и классы уже известны. Это математическая задача по обучению модели на основе этих данных. При работе с большими объемами данных и сложными математическими моделями требуется оптимизация, чтобы вычисления были быстрыми и эффективными.

Ниже представлен перечень известных датасетов, которые могут использоваться для обучения нейронных сетей [11].

*COCO (Common Objects in Context)* – один из самых известных датасетов для задач компьютерного зрения, включая классификацию, сегментацию и определение объектов.

*ImageNet* – ключевой набор данных для тренировки и тестирования моделей классификации изображений.

*MNIST* и *Fashion-MNIST* – классические датасеты для задач классификации рукописных цифр и одежды соответственно.

*CIFAR-10* и *CIFAR-100* – небольшие датасеты для классификации изображений с 10 и 100 классами.

*PASCAL VOC* – набор данных для задач обнаружения объектов, сегментации и классификации.

*Open Images* – масштабный датасет изображений с аннотациями, охватывающий множество категорий объектов.



*Kaggle Datasets* – платформа предоставляет доступ к тысячам пользовательских наборов данных для различных задач.

*UCI Machine Learning Repository* – коллекция датасетов для анализа данных и машинного обучения, часто используемых в исследованиях.

*Google Dataset Search* – поисковый сервис для нахождения открытых датасетов по различным темам.

*KITTI* – набор данных для задач автономного вождения, включая данные с камер, лидара и GPS.

*LFW (Labeled Faces in the Wild)* – набор данных с лицами, используемый для тестирования моделей распознавания лиц.

*Cityscapes* – датасет для сегментации и анализа городских сцен.

*Medical Datasets (ChestX-ray (см. рис. 7), HAM10000)* – медицинские данные для анализа изображений, например, рентгеновских снимков.

*Human3.6M* – набор данных для анализа поз человека и компьютерного зрения.

Эти датасеты являются фундаментом для обучения моделей, тестирования алгоритмов и создания новых технологий.

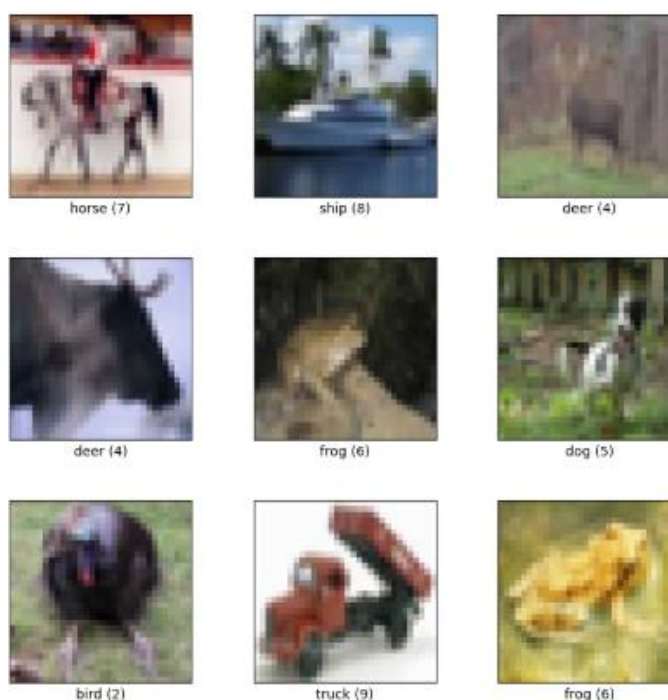


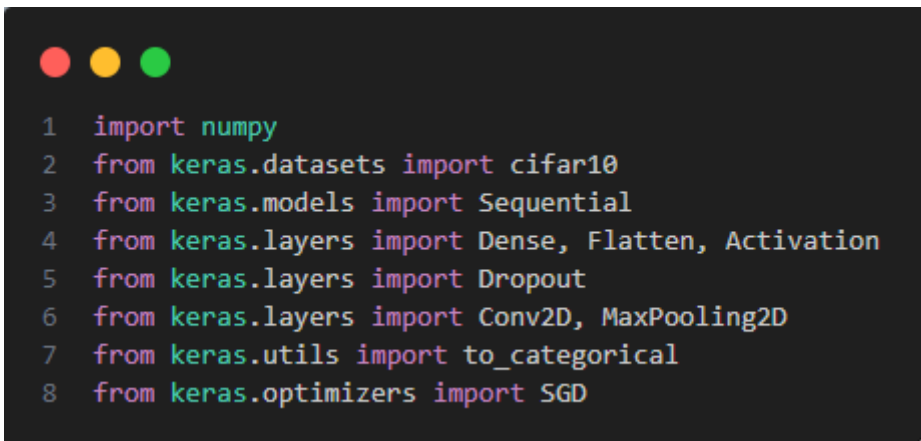
Рисунок 7 – Датасет CIFAR-10

## 2 Демонстрация классификации изображений с помощью нейронной сети

Для демонстрации классификации изображений реализуем программу на языке программирования Python, которая будет выполнять задачу классификации изображений из датасета CIFAR-10 с помощью сверточной нейронной сети. CIFAR-10 — это набор данных, содержащий 60 000 изображений размером 32×32 пикселя в цвете (3 канала). Данные уже разбиты на тренировочную (хранит 50 000 изображений) и тестовую (10 000 изображений) выборки. Метки классов представлены числами от 0 до 9, где каждая цифра соответствует одному из десяти классов, например: самолёты, автомобили, птицы.

В программе будет использована библиотека `keras`, предназначенная для глубокого машинного обучения.

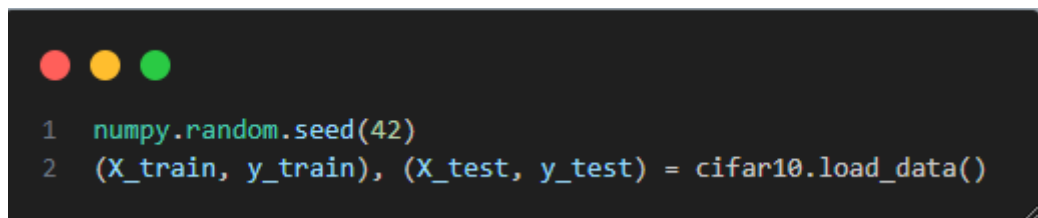
Для начала подключим все необходимые библиотеки: `numpy` для удобной работы с числовыми массивами, из вышеупомянутого `keras` получим доступ к заранее подготовленному датасету CIFAR-10, а также к методам для построения, обучения модели и настройке ее параметров (см. рис 8).



```
1 import numpy
2 from keras.datasets import cifar10
3 from keras.models import Sequential
4 from keras.layers import Dense, Flatten, Activation
5 from keras.layers import Dropout
6 from keras.layers import Conv2D, MaxPooling2D
7 from keras.utils import to_categorical
8 from keras.optimizers import SGD
```

Рисунок 8 – Подключаемые библиотеки

Для воспроизводимости данных установим фиксированный `seed`, который будет являться начальным значением для генератора случайных чисел, которое используется, чтобы обеспечить повторяемость результатов экспериментов. Также загрузим сами данные из CIFAR-10 (см. рис 9).



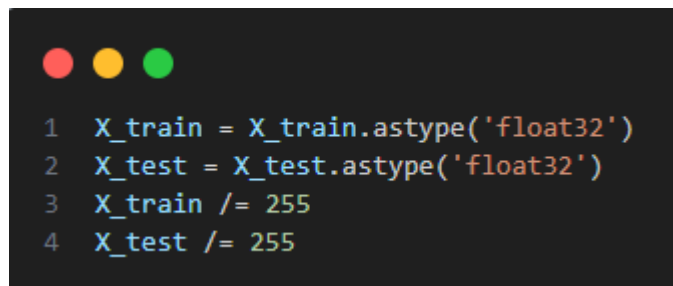
```

1  numpy.random.seed(42)
2  (X_train, y_train), (X_test, y_test) = cifar10.load_data()

```

Рисунок 9 – Установка seed и загрузка данных

Перед обучением модели подготовим данные, нормализуем их (см. рис 10). Значения пикселей преобразуются из диапазона  $[0, 255]$  в  $[0, 1]$ . Это важно, так как модели нейронных сетей обучаются быстрее и точнее при работе с нормализованными данными.



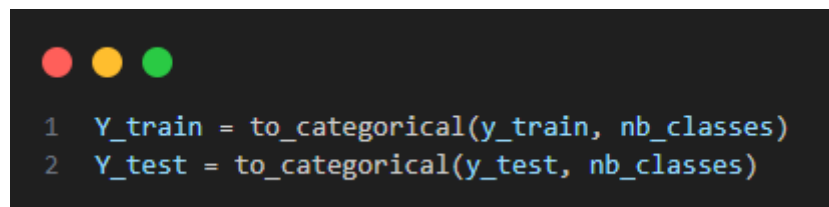
```

1  X_train = X_train.astype('float32')
2  X_test = X_test.astype('float32')
3  X_train /= 255
4  X_test /= 255

```

Рисунок 10 – Нормализация данных

Далее преобразуем метки классов в категориальный формат (см. рис 11). Это необходимо для корректной работы функции потерь, используемой при обучении.



```

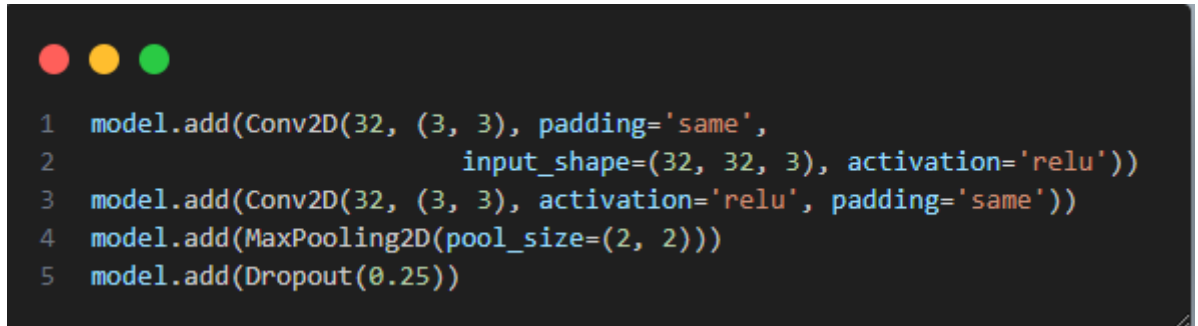
1  Y_train = to_categorical(y_train, nb_classes)
2  Y_test = to_categorical(y_test, nb_classes)

```

Рисунок 11 – Преобразование в категориальный формат

Будем использовать последовательную модель, добавляя слои последовательно, каждый из которых будет выполнять определенную функцию в обработке данных. На рисунке 12 представлено построение первого сверточного блока. В нем Conv2D применяет 32 фильтра размером  $3 \times 3$  к изображениям, извлекая признаки, например, края и текстуры. Параметр `padding = 'same'`, будет сохранять размерность данных после свертки, а параметр `activation = 'relu'`, устанавливает функцию активации ReLU, также известная как линейный выпрямитель, приравнивает значения, меньше нуля,

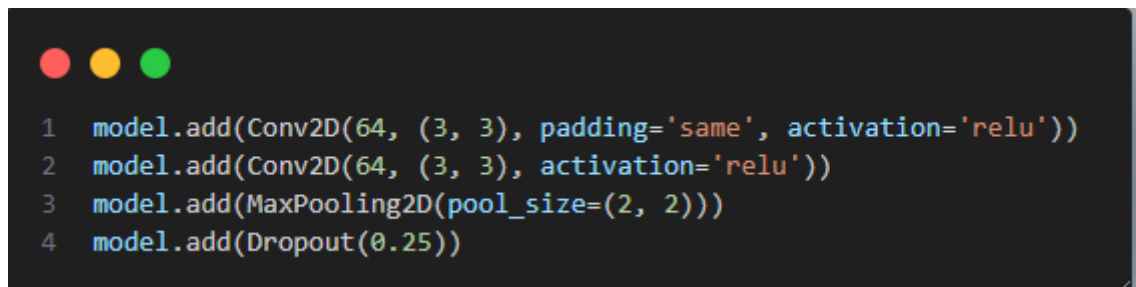
к нулю, и оставляет положительные значения неизменными. MaxPooling2D уменьшает размер данных, выбирая максимальное значение в каждом блоке размером  $2 \times 2$ , за счет чего сокращается объём вычислений. Функция Dropout случайным образом отключает 25% нейронов для повышения устойчивости модели.



```
1 model.add(Conv2D(32, (3, 3), padding='same',
2                 input_shape=(32, 32, 3), activation='relu'))
3 model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
4 model.add(MaxPooling2D(pool_size=(2, 2)))
5 model.add(Dropout(0.25))
```

Рисунок 12 – Построение первого сверточного блока

Затем происходит построение второго сверточного блока, однако в нем будет использоваться 64 фильтра для нахождения более сложных признаков (см. рис 13).



```
1 model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
2 model.add(Conv2D(64, (3, 3), activation='relu'))
3 model.add(MaxPooling2D(pool_size=(2, 2)))
4 model.add(Dropout(0.25))
```

Рисунок 13 – Построение второго сверточного блока

Создадим полносвязный слой для преобразования признаков, извлечённых сверточными слоями, в вероятности принадлежности к каждому классу. Метод Flatten преобразует данные из двумерного представления в одномерный массив, чтобы передать их в полносвязный слой. Dense создает слой с 512 нейронами и функцией активации ReLU. Затем отключаются 50% нейронов, чтобы снизить переобучение. Выходной слой Dense возвращает вероятности принадлежности к каждому из 10 классов с использованием функции активации Softmax (см. рис 14).

Функция Softmax принимает на вход вектор  $z$ , состоящий из  $K$  действительных чисел, и преобразует его в распределение вероятностей,

состоящее из  $K$  вероятностей, пропорциональных экспоненциальным значениям входных чисел. То есть до применения функции softmax некоторые компоненты вектора могут быть отрицательными или превышать 1, и их сумма может не равняться 1, но после применения функции softmax каждый компонент будет находиться в интервале  $(0, 1)$ , и их сумма будет равна 1, так что их можно интерпретировать как вероятности. Эта функция имеет вид:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{ для } i = 1, \dots, K$$

```

1 model.add(Flatten())
2 model.add(Dense(512, activation='relu'))
3 model.add(Dropout(0.5))
4 model.add(Dense(nb_classes, activation='softmax'))

```

Рисунок 14 – Построение полносвязного слоя

Далее происходит компиляция модели (см. рис 15). Используем метод SGD, который реализовывает стохастический градиентный спуск для обновления весов, он имеет следующую формулу:

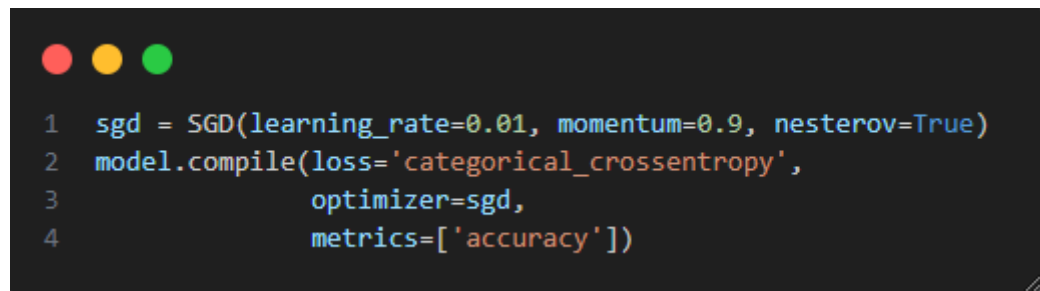
$$W_{t+1} = W_t - \eta \nabla L(W_t, x_i, y_i),$$

где  $W_t$  – текущие веса на шаге  $t$ ,  $\eta$  – скорость обучения,  $\nabla L(W_t, x_i, y_i)$  – градиент функции потерь  $L$ , вычисленный для случайного примера  $(x_i, y_i)$ .

Также в спуске будет использоваться инерция Нестерова, которая помогает стохастическому градиентному спуску быстрее сходиться и избегать «застреваний» в узких долинах функции потерь. Идея заключается в том, чтобы добавлять к текущему шагу часть предыдущего шага, как если бы параметры двигались по функции потерь с ускорением. Формула инерции Нестерова выглядит следующим образом:

$$v_{t+1} = \mu v_t - \eta \nabla L(W_t + \mu v_t, x_i, y_i), W_{t+1} = W_t + v_{t+1},$$

где  $v_{t+1}$  – скорость (накопленное изменение весов),  $\mu$  – коэффициент инерции, который контролирует вклад всех предыдущих шагов.



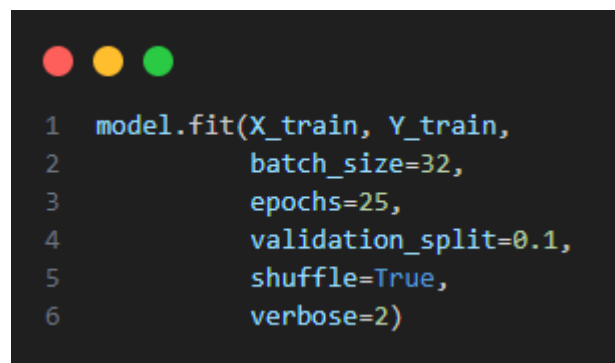
```

1  sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
2  model.compile(loss='categorical_crossentropy',
3               optimizer=sgd,
4               metrics=['accuracy'])

```

Рисунок 15 – Компиляция модели

Под конец происходит обучение модели: данные обучаются в мини-выборках по 32 изображения, задается 25 эпох и 10% данных выделяются для валидации (см. рис 16).



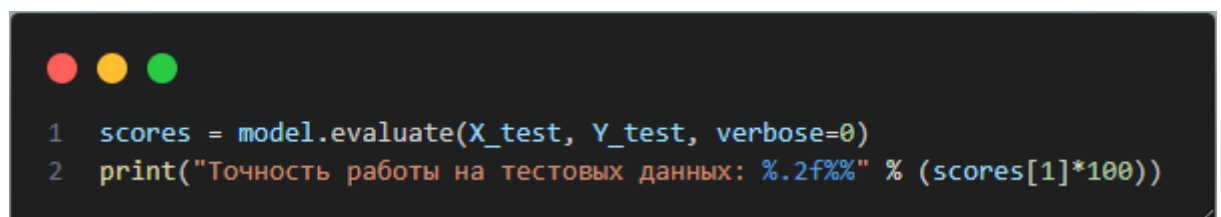
```

1  model.fit(X_train, Y_train,
2           batch_size=32,
3           epochs=25,
4           validation_split=0.1,
5           shuffle=True,
6           verbose=2)

```

Рисунок 16 – Обучение модели

Для получения результата работы производится оценка модели. Выборка используется для оценки точности модели, которая выводится в процентах (см. рис 17).



```

1  scores = model.evaluate(X_test, Y_test, verbose=0)
2  print("Точность работы на тестовых данных: %.2f%%" % (scores[1]*100))

```

Рисунок 17 – Оценка модели

Результат работы программы показывает увеличение точности распознавания зрительных образов и уменьшение ошибок от эпохи в эпоху (см. рис 18, 19). Итоговая точность равна 75,75%

```

Epoch 1/25
1407/1407 - 48s - 34ms/step - accuracy: 0.3507 - loss: 1.7583 - val_accuracy: 0.4800 - val_loss: 1.3811
Epoch 2/25
1407/1407 - 48s - 34ms/step - accuracy: 0.5178 - loss: 1.3291 - val_accuracy: 0.6026 - val_loss: 1.1184
Epoch 3/25
1407/1407 - 52s - 37ms/step - accuracy: 0.5956 - loss: 1.1347 - val_accuracy: 0.6596 - val_loss: 0.9472
Epoch 4/25
1407/1407 - 55s - 39ms/step - accuracy: 0.6416 - loss: 1.0150 - val_accuracy: 0.6982 - val_loss: 0.8738
Epoch 5/25
1407/1407 - 55s - 39ms/step - accuracy: 0.6710 - loss: 0.9299 - val_accuracy: 0.6650 - val_loss: 0.9936
Epoch 6/25
1407/1407 - 54s - 38ms/step - accuracy: 0.6921 - loss: 0.8766 - val_accuracy: 0.7454 - val_loss: 0.7487
Epoch 7/25
1407/1407 - 51s - 36ms/step - accuracy: 0.7081 - loss: 0.8341 - val_accuracy: 0.7350 - val_loss: 0.7869

```

Рисунок 18 – Результаты работы программы на эпохах 1 – 7

```

Epoch 20/25
1407/1407 - 49s - 35ms/step - accuracy: 0.7919 - loss: 0.6074 - val_accuracy: 0.7720 - val_loss: 0.7237
Epoch 21/25
1407/1407 - 49s - 35ms/step - accuracy: 0.7921 - loss: 0.5979 - val_accuracy: 0.7666 - val_loss: 0.7033
Epoch 22/25
1407/1407 - 49s - 34ms/step - accuracy: 0.7950 - loss: 0.5944 - val_accuracy: 0.7704 - val_loss: 0.7008
Epoch 23/25
1407/1407 - 52s - 37ms/step - accuracy: 0.8017 - loss: 0.5760 - val_accuracy: 0.7628 - val_loss: 0.7065
Epoch 24/25
1407/1407 - 56s - 40ms/step - accuracy: 0.8000 - loss: 0.5785 - val_accuracy: 0.7788 - val_loss: 0.6632
Epoch 25/25
1407/1407 - 55s - 39ms/step - accuracy: 0.8047 - loss: 0.5713 - val_accuracy: 0.7810 - val_loss: 0.7317
Точность работы на тестовых данных: 75.75%

```

Рисунок 19 – Результаты работы программы на эпохах 20 – 25

## ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены основные подходы и методы распознавания зрительных образов с применением нейронных сетей. Были проанализированы ключевые архитектуры, используемые для решения задач классификации изображений, а также основные принципы их работы и обучения. Отдельное внимание уделено процессам предобработки данных и использованию популярных наборов данных для обучения и тестирования нейронных сетей.

На основе полученных знаний можно сделать вывод, что нейронные сети, особенно глубокие сверточные нейронные сети (CNN), являются мощным инструментом для решения задач распознавания зрительных образов, обеспечивая высокую точность и эффективность. Современные методы позволяют значительно улучшить результаты по сравнению с традиционными подходами, однако для достижения ещё более высоких показателей необходимы дальнейшие исследования и совершенствование архитектур и алгоритмов обучения.

Значительный вклад в развитие распознавания образов внесли такие глобальные игроки, как Google, Microsoft, Amazon, IBM, NVIDIA, Qualcomm, Intel. Эти компании активно инвестируют в исследования и разработки, что способствует постоянному улучшению технологий.

Практическая демонстрация классификации изображений с использованием Python показала высокую применимость данных методов в реальных задачах. Нейронные сети остаются одним из самых перспективных направлений в области компьютерного зрения.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Методы распознавания образов: от простых до сложных [Электронный ресурс] // SimbirSoft [Электронный ресурс]. - URL: <https://www.simbirsoft.com/blog/metody-raspoznavaniya-obrazov-ot-prostykh-do-slozhnykh/> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. рус.
- 2 How Does a Neural Network Recognize Images? [Электронный ресурс] // Baeldung [Электронный ресурс]. - URL: <https://www.baeldung.com/cs/neural-networks-image-recognition> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 3 Convolutional Neural Network (CNN): Architecture Explained | Deep Learning [Электронный ресурс] // PyCodeMates [Электронный ресурс]. - URL: <https://www.pycodemates.com/2023/06/introduction-to-convolutional-neural-networks.html> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 4 Vision Transformers (ViT) in Image Recognition – 2024 Guide [Электронный ресурс] // viso.ai [Электронный ресурс]. - URL: <https://viso.ai/deep-learning/vision-transformer-vit/> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 5 Трансформер (модель машинного обучения) [Электронный ресурс] // Википедия [Электронный ресурс]. - URL: [https://ru.wikipedia.org/wiki/Трансформер\\_\(модель\\_машинного\\_обучения\)](https://ru.wikipedia.org/wiki/Трансформер_(модель_машинного_обучения)) (дата обращения: 28.12.24). - Загл. с экрана. - Яз. рус.
- 6 Autoencoder [Электронный ресурс] // Wikipedia [Электронный ресурс]. - URL: <https://en.wikipedia.org/wiki/Autoencoder#> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 7 Введение в автоэнкодеры [Электронный ресурс] // TensorFlow [Электронный ресурс]. - URL: <https://www.tensorflow.org/tutorials/generative/autoencoder?hl=ru#:~:te>

xt=An%20autoencoder%20is%20a%20special,while%20minimizing%20the%20reconstruction%20error (дата обращения: 28.12.24). - Загл. с экрана. - Яз. рус.

- 8 What is an Autoencoder? [Электронный ресурс] // roboflow [Электронный ресурс]. - URL: <https://blog.roboflow.com/what-is-an-autoencoder-computer-vision/> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 9 Neural Networks for Image Recognition: Methods, Best Practices, Applications [Электронный ресурс] // IndianTechWarrior [Электронный ресурс]. - URL: <https://indiantechwarrior.com/neural-networks-for-image-recognition-methods-best-practices-applications/> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 10 How Data Augmentation Impacts Performance Of Image Classification, With Codes [Электронный ресурс] // aim [Электронный ресурс]. - URL: <https://analyticsindiamag.com/ai-mysteries/image-data-augmentation-impacts-performance-of-image-classification-with-codes/> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. англ.
- 11 20+ популярных опенсорсных датасетов для Computer Vision [Электронный ресурс] // Training Data [Электронный ресурс]. - URL: <https://trainingdata.ru/journal2/tpost/x6tbn49sb1-20-populyarnih-opensorsnih-datasetov-dly> (дата обращения: 28.12.24). - Загл. с экрана. - Яз. рус.

## ПРИЛОЖЕНИЕ А

### Листинг программы snn.py

```
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Flatten, Activation
from keras.layers import Dropout
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras.optimizers import SGD

numpy.random.seed(42)
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
batch_size = 32
nb_classes = 10
nb_epoch = 25
img_rows, img_cols = 32, 32
img_channels = 3
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_train = to_categorical(y_train, nb_classes)
Y_test = to_categorical(y_test, nb_classes)

model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(32, 32, 3), activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes, activation='softmax'))
sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
model.fit(X_train, Y_train,
          batch_size=32,
          epochs=25,
          validation_split=0.1,
```

```
        shuffle=True,  
        verbose=2)  
  
scores = model.evaluate(X_test, Y_test, verbose=0)  
print("Точность работы на тестовых данных: %.2f%%" % (scores[1]*100))
```