Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Изучение сверточных нейронных сетей для задач классификации изображений»

Студент РК6-76Б _____ **Онюшев А.А.**
(Подпись, дата) И.О. Фамилия

Руководитель _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой РК6
А.П. Карпенко

«_____» _____ 2023 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме: ___Изучение сверточных нейронных сетей для задач классификации изображений_____
_____

Студент группы __РК6-76Б___

_____Онюшев Артем Андреевич_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) _учебная___
Источник тематики (кафедра, предприятие, НИР) _кафедра_____

График выполнения НИР: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание: _1) Изучить устройство CNN_____
2) Изучить математическую составляющую_____
3) Рассмотреть варианты применения_____
4) Написать собственную НС с применением технологии CNN._____
5) Добиться приемлемых результатов классификации_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 17 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):
_____
_____
_____
_____

Дата выдачи задания «8» октября 2023 г.

Руководитель НИР _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

Студент _____ **Онюшев А.А.**
(Подпись, дата) И.О. Фамилия

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафеде

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе вопросы анализа и обработки изображений приобретают все большее значение, оказывая значительное влияние на различные области науки и технологий. Одним из ключевых направлений в этой области является классификация изображений с использованием сверточных нейронных сетей (CNN). Эта технология, основанная на механизмах восприятия визуальной информации человеческим глазом, является эффективным инструментом для автоматической обработки и анализа графических данных.

Цель настоящего исследования заключается в изучении и применении методов классификации изображений при помощи сверточных нейронных сетей. Развитие этой области имеет важное значение для ряда приложений, таких как распознавание объектов, медицинская диагностика, автономные транспортные средства и многие другие. Исследование фокусируется на анализе архитектурных особенностей CNN, изучении математической и биологической составляющих, методах предварительной обработки данных и подходах к оптимизации параметров модели с целью достижения более высокой точности классификации.

Акцентирование внимания на данной проблеме не только расширяет наши знания о возможностях сверточных нейронных сетей, но также может привести к разработке более эффективных и точных систем анализа и распознавания изображений, что имеет практическое значение для многих областей промышленности и науки.

1. Устройство CNN

Сверточная нейронная сеть (ConvNet/CNN) — это специальная архитектура искусственных нейронных сетей, которая может принимать входное изображение, присваивать важность (изучаемые веса и смещения) аспектам или объектам изображении и отличать одно от другого. При этом изображения в сравнении с другими архитектурами НС требуют гораздо меньше предварительной обработки. В примитивных методах фильтры разрабатываются вручную, но достаточно обученные сети CNN учатся применять эти фильтры/характеристики самостоятельно.

Архитектура CNN аналогична структуре связей нейронов в мозгу человека, учёные черпали вдохновение в организации зрительной коры головного мозга. Отдельные нейроны реагируют на стимулы только в некоторой области поля зрения, также известного как перцептивное поле. Множество перцептивных полей перекрывается, полностью покрывая поле зрения CNN.

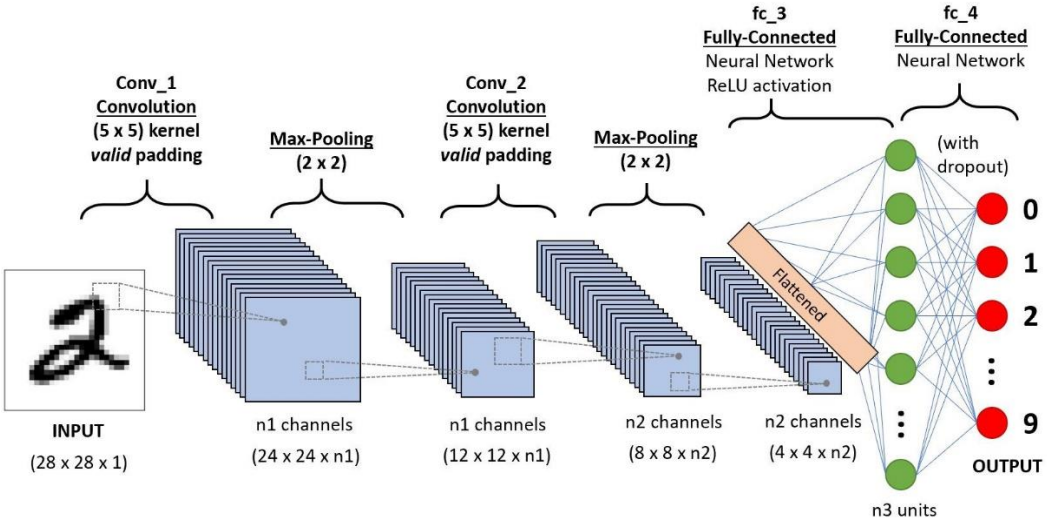


Рис. 1 Архитектура сверточной нейронной сети

Главной особенностью свёрточных сетей является то, что они обычно работают именно с изображениями, а потому можно выделить особенности, свойственные именно им. Многослойные персептроны работают с векторами, а потому для них нет никакой разницы, находятся ли какие-то точки рядом или на противоположных концах, так как все точки равнозначны и считаются совершенно одинаковым образом. Изображения же обладают локальной

связностью. Например, если речь идёт об изображениях человеческих лиц, то вполне логично ожидать, что точки основных частей лица будут рядом, а не разрозненно располагаться на изображении. Поэтому требовалось найти более эффективные алгоритмы для работы с изображениями и ими оказались свёрточные сети.

2. Изображение «в глазах» компьютера

Изучая CNN невольно появляется вопрос «Что такое свертка?». Перед тем, как ответить на него, нужно разобраться, как компьютер «видит» какое-то изображение.

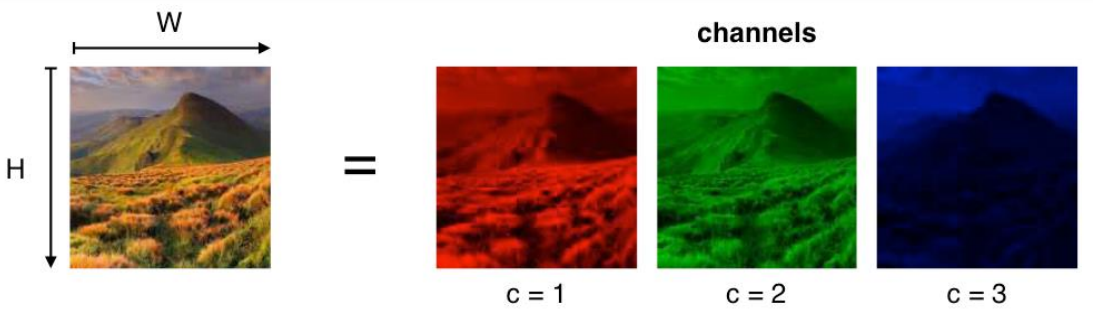


Рис. 2 Как компьютер видит изображение

Изображения в компьютере представляются в виде пикселей, а каждый пиксель – это значения интенсивности соответствующих цветовых каналов. При этом интенсивность каждого из каналов описывается целым числом от 0 до 255. Чаще всего используются цветные изображения, которые состоят из RGB пикселей – пикселей, содержащих яркости по трём каналам: красному, зелёному и синему. Различные комбинации этих цветов позволяют создать любой из цветов всего спектра. Чтобы хранить информацию о всех пикселях удобнее всего использовать тензор – 3D массив чисел, или, проще говоря, массив матриц чисел.

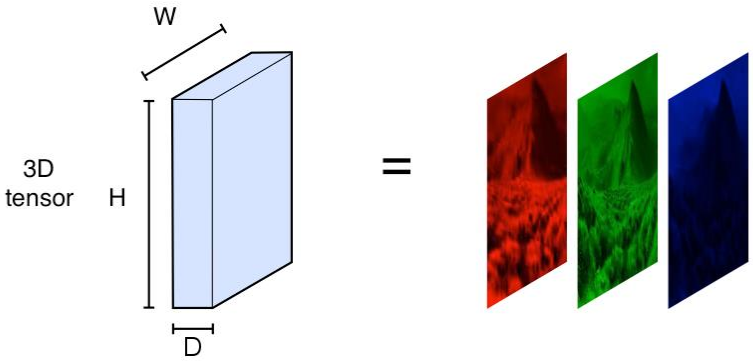
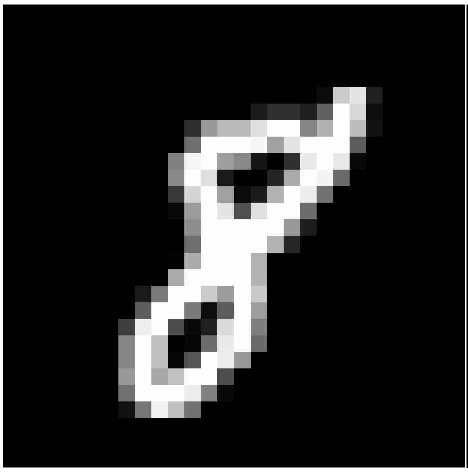
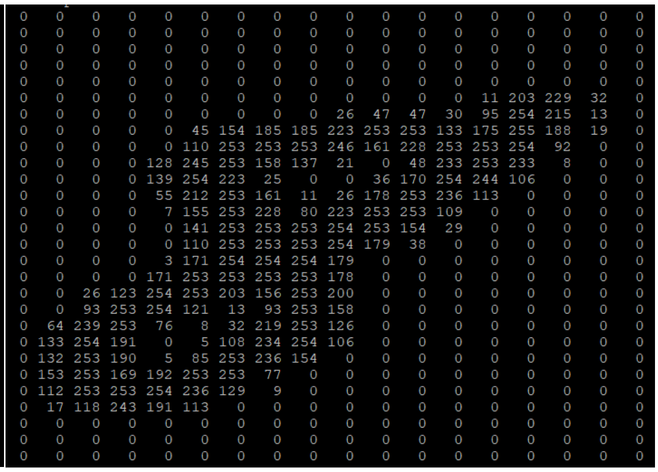


Рис. 3 Преобразование из изображения в тензор





0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	11	203	229	32	0
0	0	0	0	0	0	0	0	0	26	47	47	30	95	254	215	13	0
0	0	0	0	0	45	154	185	185	223	253	253	133	175	255	188	19	0
0	0	0	0	0	110	253	253	253	246	161	228	253	253	254	92	0	0
0	0	0	0	128	245	253	158	137	21	0	48	233	253	233	8	0	0
0	0	0	0	139	254	223	25	0	0	36	170	254	244	106	0	0	0
0	0	0	0	55	212	253	161	11	26	178	253	236	113	0	0	0	0
0	0	0	0	7	155	253	228	80	223	253	253	109	0	0	0	0	0
0	0	0	0	0	141	253	253	253	254	253	154	29	0	0	0	0	0
0	0	0	0	0	110	253	253	253	254	179	38	0	0	0	0	0	0
0	0	0	0	3	171	254	254	254	179	0	0	0	0	0	0	0	0
0	0	0	0	171	253	253	253	253	178	0	0	0	0	0	0	0	0
0	0	26	123	254	253	203	156	253	200	0	0	0	0	0	0	0	0
0	0	93	253	254	121	13	93	253	158	0	0	0	0	0	0	0	0
0	64	239	253	76	8	32	219	253	126	0	0	0	0	0	0	0	0
0	133	254	191	0	5	108	234	254	106	0	0	0	0	0	0	0	0
0	132	253	190	5	85	253	236	154	0	0	0	0	0	0	0	0	0
0	153	253	169	192	253	253	77	0	0	0	0	0	0	0	0	0	0
0	112	253	253	254	236	129	9	0	0	0	0	0	0	0	0	0	0
0	17	118	243	191	113	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рис 4. Изображение цифры и его представление в компьютере.

3. Свертка

Слой свёртки, как можно догадаться по названию типа нейронной сети, является самым главным слоем сети. Его основное назначение – выделить признаки на входном изображении и сформировать карту признаков. Карта признаков – это всего лишь очередной тензор (массив матриц), в котором каждая матрица отвечает за какой-нибудь выделенный признак.

Для того, чтобы слой мог выделять признаки, в нём имеются так называемые фильтры (или ядра). Фильтр – квадратная матрица небольшого размера (обычно 3x3 или 5x5), заполненная определенным образом. Чем больше будет таких фильтров – тем больше признаков удастся выделить и тем больше будет глубина каналов у выходного тензора слоя

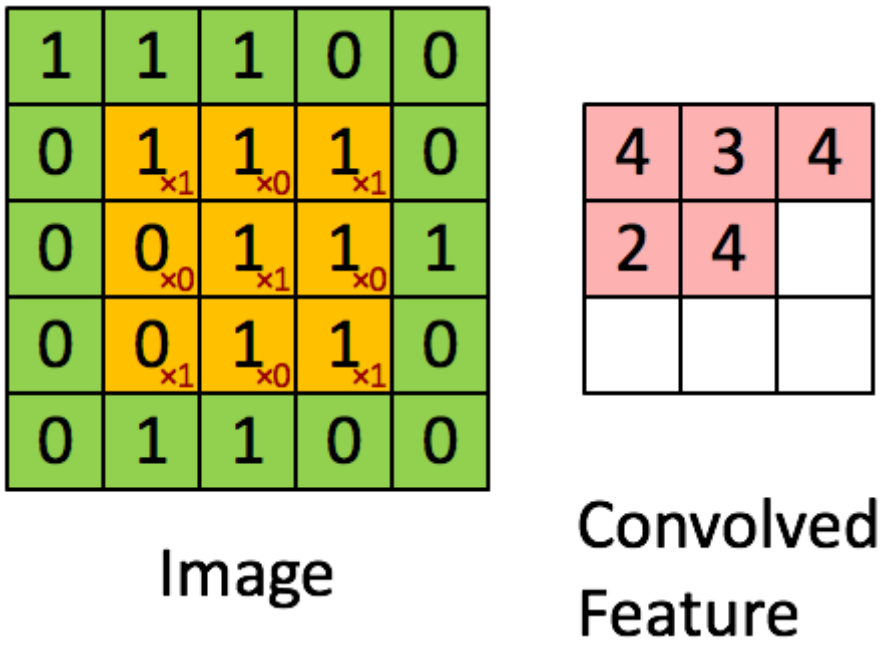


Рис. 5 Свертка изображения 5x5x1 с фильтром 3x3x1 для получения признака 3x3x1

Данный фильтр поочередно накладывается на изображение начиная с левого верхнего угла и пока не дойдет до правого нижнего угла. На каждом шаге числа в матрице фильтра перемножаются с соответствующими числами на изображении, полученные результаты складываются и записываются в матрицу признака. Лучше понять это можно, посмотрев на рисунок 5. На нем желтым цветом изображен фильтр размером 3x3x1, а красные цифры в правом углу –

числа матрицы фильтра. Зеленым цветом изображена матрица изображения 3x3x1, а красным полученный признак. Таким образом на 5-ом шаге итерации мы получим:

$$1 * 1 + 0 * 1 + 1 * 1 + 0 * 0 + 1 * 1 + 0 * 1 + 1 * 0 + 0 * 1 + 1 * 1 = 4$$

Полученные в итоге матрицы признаков могут иметь такой же размер, что и исходное изображение, либо размер меньше (как в нашем случае). Это зависит от заданных размера шага и начального заполнения. В этом репозитории можно найти множество разных GIF- файлов, которые помогут лучше понять, как заполнение и длина шага работают.

4. Слой подвыборки (пулинга)

Так как сверточные НС используют так же и простую сеть прямого распространения, нам необходимо уменьшить кол-во выходных параметров из слоя свертки и при этом не потерять важную информацию. Для этого используют слои подвыборки.

Данный слой позволяет уменьшить пространство признаков, сохраняя наиболее важную информацию. Существует несколько разных версий слоя пулинга, среди которых максимальный пулинг, средний пулинг и пулинг суммы. Наиболее часто используется именно слой макспулинга.

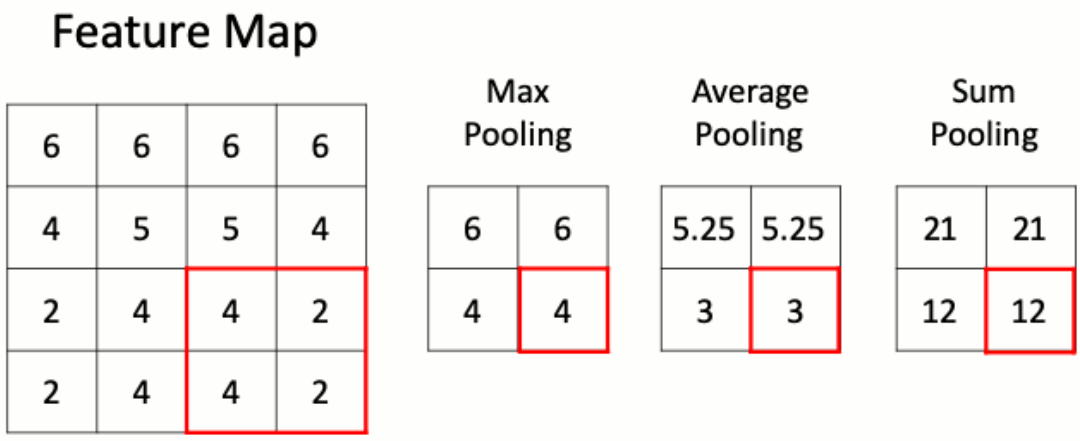


Рис. 6 Преобразования слоя подвыборки

Действия этого слоя идентичны действиям слоя свертки, только используются другие операции (для макспулинга – берется максимальное число, для среднего пулинга – берется среднее арифметическое от чисел).

5. Собственная CNN

Изучив, как устроена сверточная нейронная сеть, можно приступать к написанию собственной. Она будет классифицировать изображения на два класса: кошки и собаки.

Найти нужный датасет можно тут. Так как все фотографии имеют разный размер и требуют некоторых дополнений, приведем их все к одному размеру и виду, создав свой класс датасета. Код для создания собственного датасета смотреть в листинге 1.

Таким образом, после выполнения кода листинга 1, мы получим словарь, в котором будет храниться тензер изображения и требуемое значение.

Теперь требуется выбрать функцию оптимизации и функцию потерь. Хорошей функцией оптимизации считается функция Adam(), поэтому воспользуемся ей. Для задач классификации в качестве функции потерь рекомендуется использовать функцию CrossEntropyLoss(). Ей и воспользуемся. Полученный код представлен в листинге 2.

На этом моменте уже можно разрабатывать саму архитектуру CNN. У нас будет 4 слоя свертки. Между этими слоями будут слои подвыборки (макспулинга). В качестве функции активации для задач классификации рекомендуют использовать LeakyReLu(). На выходе с последнего слоя свертки установим НС прямого распространения состоящую из двух слоев. Код архитектуры НС представлен в листинге 3.

Нейронная сеть почти готова и осталось только обучить ее. Для этого воспользуемся листингом 4.

Наша нейронная сеть готова и её можно запускать!

6. Получение приемлемых результатов обучения

После того, как мы обучили нашу нейронную сеть, можем проестировать её. Используем для этого тестовый датасет, чтобы проверить НС на тех данных, которые она ещё не видела. Получим вот такие значения:

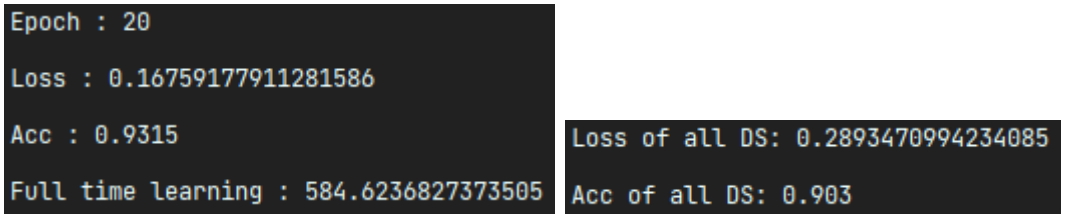


Рис. 7 Полученные значения точности и ошибки при обучении и при тесте, соответственно

ЗАКЛЮЧЕНИЕ

В заключение данного исследования можно утверждать, что сверточные нейронные сети (CNN) представляют собой мощный инструмент для задач классификации изображений. В процессе проведения исследования были изучены различные архитектуры CNN, методы предварительной обработки данных и оптимизации параметров модели. Результаты экспериментов подтверждают высокую эффективность сверточных нейронных сетей в решении задач классификации изображений по сравнению с традиционными методами.

Оптимизация параметров модели и правильный выбор архитектуры CNN существенно влияют на достижение высокой точности классификации. Также важным аспектом является качество предварительной обработки данных, включая масштабирование, аугментацию и нормализацию, что дополнительно улучшает способность модели к обобщению.

Полученные результаты предоставляют основу для дальнейших исследований и применения сверточных нейронных сетей в различных областях, таких как медицинская диагностика, автоматическое распознавание объектов и другие задачи, где анализ изображений играет важную роль. Развитие и усовершенствование подходов к классификации изображений с использованием CNN содействует продвижению технологий машинного зрения и их применению в реальных сценариях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Сверточная нейронная сеть с нуля» [Электронный ресурс] – URL: https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction (дата обращения: 20.10.2023 - 24.10.2023);
2. Блог о сверточной нейронной сети [Электронный ресурс] – URL: https://habr.com/ru/companies/skillfactory/articles/565232/ (дата обращения: 20.10.2023);
3. PyTorch Tutorials [Электронный ресурс] – URL: https://pytorch.org/tutorials/ (дата обращения: 20.10.2023 - 30.10.2023);
4. PyTorch Documentation [Электронный ресурс] – URL: https://pytorch.org/docs/stable/index.html (дата обращения: 20.10.2023 - 30.10.2023);
5. Сайт с датасетами [Электронный ресурс] – URL: https://www.kaggle.com/datasets/d4rklucif3r/cat-and-dogs?select=dataset (дата обращения: 21.10.2023).

Приложение

Листинг 1 – Класс датасета

```
class DataSet2Class(torch.utils.data.Dataset):
    def __init__(self, path_dir1: str, path_dir2: str):
        super().__init__()

        self.path_dir1 = path_dir1
        self.path_dir2 = path_dir2

        self.dir1_list = sorted(os.listdir(path_dir1))
        self.dir2_list = sorted(os.listdir(path_dir2))

    def __len__(self):
        return len(self.dir1_list) + len(self.dir2_list)

    def __getitem__(self, idx):

        if idx < len(self.dir1_list):
            class_id = 0
            img_path = os.path.join(self.path_dir1,
                                    self.dir1_list[idx])
        else:
            class_id = 1
            idx -= len(self.dir1_list)
            img_path = os.path.join(self.path_dir2,
                                    self.dir2_list[idx])

        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = img.astype(np.float32)
        img = img / 255.0

        img = cv2.resize(img, (128, 128),
                         interpolation=cv2.INTER_AREA)

        img = img.transpose((2, 0, 1))

        t_img = torch.from_numpy(img)
        t_class_id = torch.tensor(class_id)

        return {'img': t_img,
                'label': t_class_id}
```

Листинг 2 – Инициализация функции потерь и оптимизатора

```
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3,
                             betas=(0.9, 0.99))
loss_fn = nn.CrossEntropyLoss()
```

Листинг 3 – Класс нейронной сети

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
```

```
4
5           self.act = nn.LeakyReLU(0.18)
6           self.maxpool = nn.MaxPool2d(2, 2)
7           self.conv0 = nn.Conv2d(3, 128, 3, stride=1, padding=0)
8           self.conv1 = nn.Conv2d(128, 128, 3, stride=1, padding=0)
9           self.conv2 = nn.Conv2d(128, 128, 3, stride=1, padding=0)
10          self.conv3 = nn.Conv2d(128, 256, 3, stride=1, padding=0)
11
12          self.adaptivepool = nn.AdaptiveAvgPool2d((1, 1))
13          self.flatten = nn.Flatten()
14          self.linear1 = nn.Linear(256, 20)
15          self.linear2 = nn.Linear(20, 2)
16
17      def forward(self, x):
18          out = self.conv0(x)
19          out = self.act(out)
20          out = self.maxpool(out)
21          out = self.conv1(out)
22          out = self.act(out)
23          out = self.maxpool(out)
24          out = self.conv2(out)
25          out = self.act(out)
26          out = self.maxpool(out)
27          out = self.conv3(out)
28          out = self.act(out)
29
30          out = self.adaptivepool(out)
31          out = self.flatten(out)
32          out = self.linear1(out)
33          out = self.act(out)
34          out = self.linear2(out)
35          return out
```

Листинг 4 – Цикл обучения НС

```
1   for epoch in range(epochs):
2       loss_val = 0
3       acc_val = 0
4       for sample in train_loader:   # (pbar := tqdm(train_loader))
5           img, lbl = sample['img'], sample['label']
6           lbl = F.one_hot(lbl, 2).float()
7           img = img.to(device)
8           lbl = lbl.to(device)
9           optimizer.zero_grad()
10          with autocast(use_amp):
11              pred = CNNet(img)
12              loss = loss_fn(pred, lbl)
13          scaler.scale(loss).backward()
14          loss_item = loss.item()
15          loss_val += loss_item
16
17          scaler.step(optimizer)
18          scaler.update()
19
20          acc_current = accuracy(pred.cpu().float(),
21  lbl.cpu().float())
```

```
        acc_val += acc_current

    print(f"Epoch : {epoch+1}")
    print(f"Loss : {loss_val / len(train_loader)}")
    print(f"Acc : {acc_val / len(train_loader)}")
```