ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

2285

регистрационный номер

ФАКУЛЬТЕТ: ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ-7)

ПРОГРАММА ДЛЯ ПРОСМОТРА И РЕДАКТИРОВАНИЯ ИЗОБРАЖЕНИЙ

Автор: **ТЕРЕХОВ АЛЕКСАНДР ДМИТРИЕВИЧ**

ГБОУ города Москвы «Московская международная школа», класс 11 (Г)

Научный руководитель: **ХРЯКОВ ДМИТРИЙ АЛЕКСАНДРОВИЧ**

Group – IB, программист – разработчик

_____

подпись научного руководителя

Москва - 2019

Программа для просмотра и редактирования изображений

Аннотация

Целью данной работы является разработка кроссплатформенного программного обеспечения на языке программирования Python для просмотра изображений различных форматов с возможностью изменения размеров и поиска по ключевым параметрам.

В ходе работы был проведен обзор существующих библиотек языка Python 3 для работы с изображениями и построения графического интерфейса. Были выбраны библиотеки Pillow и Tkinter, как наиболее соответствующие поставленной цели. Было проведено обучение нейронной сети для решения задачи классификации изображений, для этого использовались библиотеки Tensorflow и Keras. Также были разработаны механизмы редактирования и фильтрации изображений.

С использованием выбранных библиотек и составленных алгоритмов была разработана программа, осуществляющая:

1. Просмотр изображений в выбранной директории.
2. Редактирование изображений.
3. Фильтрация изображений по размеру, расширению, имени.
4. Определение категории изображения на основе обученной сверточной нейронной сети.
5. Режим слайд-шоу.
6. Изменяемая цветовая схема интерфейса.
7. Возможность установить изображение в качестве заставки рабочего стола (в операционной системе Windows).
8. Просмотр свойств изображений.

Разработанная программа может быть использована как легковесный кроссплатформенный аналог популярных программ для просмотра изображений.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На современных компьютерах содержится множество снимков и рисунков, количество которых зачастую очень велико. Для удобной работы с изображениями необходимо иметь под рукой специальную программу, которая бы не только просматривала изображения, но и могла делать специфичные действия: удалять изображения, менять размер, расширение файла. Все эти функции можно найти в существующих программах. Но часто при работе с рисунками и фото может возникнуть потребность в их классификации по изображенным объектам. Эта функция может пригодиться пользователям, которые работают с большими объемами изображений и нуждаются в функции их классификации, имея при этом возможность простого редактирования нужных изображений.

Целью данной работы является разработка такой программы для просмотра изображений, которая имела бы возможность:

а) Просматривать изображения.

б) Редактировать изображения.

в) Классифицировать изображения по выбранным категориям:

1) Автомобили.

2) Кошки.

3) Цветы.

4) Люди.

Также необходимо выбрать и реализовать механизм для распознавания и классификации изображений, и внедрить его в разработанное приложение.

1. Нейронные сети

Задача распознавания графических образов имеет несколько решений. Для оптического распознавания образов можно применить метод перебора вида объекта под различными углами, масштабами, смещениями. Также можно найти контур объекта и исследовать его свойства (связность, наличие углов и так далее). Ещё один подход — использовать искусственные нейронные сети. Этот метод требует либо большого количества примеров задачи распознавания (с правильными ответами), либо специальной структуры нейронной сети, учитывающей специфику данной задачи.

Для решения нашей задачи подойдет метод распознавания изображения с помощью искусственной нейронной сети. В отличие от первых двух методов, он не требует большой алгоритмической подготовки, достаточно гибкий, имеет возможность реализации на большинстве популярных языков программирования. К тому же, возможно совершенствовать существующие модели нейронных сетей, дообучая их на своих данных.

1.1. Общее описание

Искусственная нейронная сеть — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

Нейронная сеть представляет собой систему соединённых и взаимодействующих между собой простых процессоров - искусственных нейронов (Рисунок 1). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый нейрон имеет дело только с сигналами, которые он периодически получает, и сигналами, которые периодически посылает другим процессорам.

Сигналы во время передачи от одного нейрона к другому могут быть усилены или ослаблены с помощью коэффициентов связи между нейронами – весов.

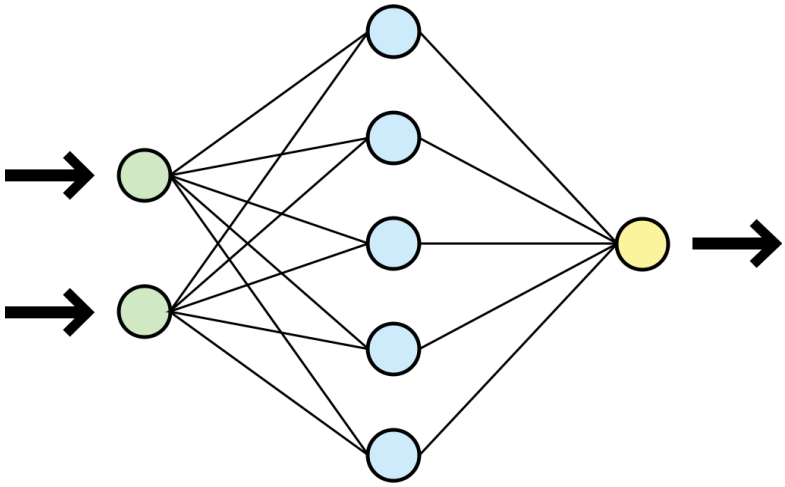

Рисунок 1 – Схема простой нейронной сети

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами.

Существует несколько типов архитектур искусственных нейронных сетей по типу обучения:

а) Обучение с учителем. Можно выделить два подтипа:

1) Полносвязная нейронная сеть (перцептрон)

2) Сверточная нейронная сеть

б) Обучение без учителя. Обычно такие нейронные сети используются для решения задач прогнозирования.

в) Смешанный тип.

В решении нашей задачи будет использована архитектура обучения с учителем, так как для распознавания определенных типов изображений будут

созданы наборы с эталонными данными. Также будет выбран тип свёрточной нейронной сети.

1.2. Сверточные нейронные сети

Для решения задачи распознавания изображений хорошо подходит сверточная нейронная сеть.

Свёрточная нейронная сеть (англ. convolutionalneuralnetwork, CNN) — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание изображений [1]. Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и субдискретизирующих слоёв (слоёв подвыборки). Такая структура представлена на рисунке 2. Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) может быть любой, по выбору исследователя. Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

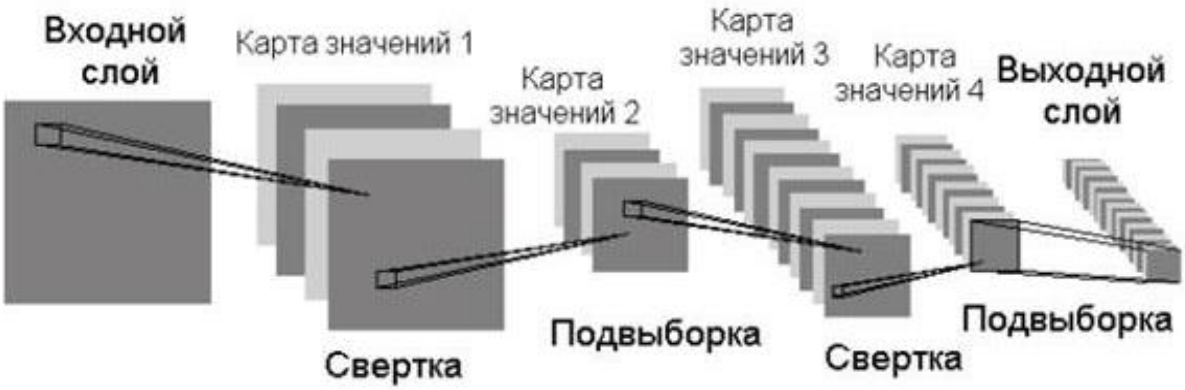

Рисунок 2 – Схема сверточной нейронной сети [2]

1.3. Обоснование методов

Для решения задачи была выбрана архитектура сверточной нейронной сети по нескольким причинам:

а) По сравнению с полносвязной нейронной сетью (типа перцептрона) — она имеет гораздо меньшее количество настраиваемых весов, так как при обучении нейросеть обобщает информацию об изображении.

б) Возможность распараллеливания вычислений.

Для практической реализации обучения и решения задачи классификации была выбрана связка из библиотек для машинного обучения Tensorflow и Keras. Такой выбор сделан по следующим причинам:

а) Для этих библиотек есть реализации на языке Python, который одинаково удобен и для решения задач машинного обучения, и для разработки утилитарного программного обеспечения.

б) Обе библиотеки обладают подробной документацией, в сети Интернет существует множество примеров работы этими библиотеками.

в) Существует реализация популярных архитектур нейронных сетей для этих библиотек, достаточно только дообучить модель для конкретной задачи.

Для обучения нейронной сети был использован сервис GoogleColabatory поскольку он позволяет обеспечить окружение, необходимое для быстрого обучения.

2. Методика обучения нейронной сети

Обучение свёрточной нейронной сети с использованием библиотек Tensorflow и Keras включает в себя следующие этапы:

а) составление нескольких выборок:

1) обучающей.

2) вадидации.

3) тестирования.

б) составление модели нейронной сети, учитывающей решаемую задачу.

в) обучение нейронной сети с заданными параметрами.

г) тестирование полученной модели на размеченных данных.

Также во время обучения будет происходить дополнительная проверка на размеченных данных в выборке валидации на каждой их эпох обучения.

2.1. Разметка данных.

Поставленная нами задачи – классифицировать изображения по нескольким категориям – требует заранее разметить набор используемых для обучения изображений. Надо разделить изображения по двум признакам:

а) По объекту, который находится на изображении.

б) По использованию на конкретном этапе обучения (принадлежность к выборке).

Таким образом, структура файловой системы в папке для обучения будет выглядеть следующим образом:

а) Изображения для тренировки.

б) Изображения для валидации на каждой эпохе обучения.

в) Изображения для тестирования модели.

Внутри каждой из выборок должно быть разделение изображений по классифицируемым категориям, в нашем случае это:

а) Люди.

б) Кошки.

в) Автомобили.

г) Цветы.

Количество изображений в выборке для тренировки должно быть больше,

чем в остальных выборках. Примерное соотношение 70, 15 и 15 процентов.

2.2. Выбор типа модели.

В этой работе было использовано два способа задания архитектуры модели:

а) Самостоятельное задание структуры модели.

б) Использование модели из примеров Keras, как части архитектуры.

В первом случае, создание архитектуры модели будет выглядеть так:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Для создания такой составной сети мы будем использовать последовательную (Sequential) модель Keras. Затем мы задаем сверточную часть нейронной сети (для краткости приведена только часть кода).

```
model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('sigmoid'))
```

После этого мы добавляем в модель слой Flatten, который преобразует двумерные вектора признаков, полученные от сверточной части сети VGG16, в

одномерный вектор. Полносвязные слои, используемые для классификации, не могут работать с двумерными данными на входе, им нужен одномерный вектор. Первый полносвязный (Dense) слой содержит 256 нейронов, функция активации полулинейная. Затем идет слой Dropout, который используется для снижения переобучения.

На выходном полносвязном слое четыре нейрона, что соответствует числу выбранных категорий. Функция активации выходного нейрона сигмоидальная. Эта функция плавно меняет свое значение от нуля до единицы, хорошо подходит для классификации.

Библиотека Keras позволяет использовать уже существующие модели нейронных сетей для дообучения. Несколько популярных моделей, применяющихся для распознавания изображений:

а) VGG16 – сеть VisualGeometryGroup из университета Оксфорда для распознавания объектов на изображениях, состоит из 16 слоев.

б) VGG19 – еще одна сеть VisualGeometryGroup для распознавания объектов, но содержит 19 слоев.

в) Inception v3 – нейронная сеть компании Google для распознавания объектов на изображениях.

г) ResNet50 – нейронная сеть компании Microsoft, использующая остаточное обучение (residuallearning). Применяется для распознавании объектов на изображениях.

д) Xception – модификация сети Inception от создателя KerasFrançoisChollet.

Мы выберем модель VGG16, так как она вполне удовлетворяет поставленным требованиям и обладает меньшим, по сравнению с подобными моделями, числом слоев, что уменьшит время на дообучение.

Существует возможность использовать модель с уже заполненными значениями весов для классификации, но мы будем обучать модель на своём наборе данных, поэтому используем модель только в качестве слоёв свертки. Таким образом, наша сеть может включать сверточную часть VGG16 и новый классификатор для распознавания изображений. Модель в этом случае задается следующим образом:

```
model = Sequential()

model.add(vgg16_net)
```

Далее код аналогичен первому случаю.

Перед обучением сеть необходимо скомпилировать. В качестве функции ошибки указываем categorical_crossentropy, также необходимо указать параметр скорости обучения при создании оптимизатора Adam в параметре lr (сокращение от learningrate, скорость обучения).

Обучать сеть мы будем при помощи генераторов изображений Keras. Текст программы для обучения находится в Приложении А.

2.3. Проверка обучения на размеченных данных.

После завершения обучения сети необходимо обязательно проверить качество ее работы на данных, которые сеть не видела в процессе обучения. Для этого необходимо использовать выборку из папки test.

3. Результаты обучения нейронной сети.

Во время обучения нейронной сети на каждой эпохе на экран выводились результаты проверки модели на выборке валидации. Мы исследуем параметры точность классификации и параметр ошибки.

Таблица 1. Результаты обучения нейронной сети с самостоятельным заданием структуры.

Эпоха	Точность классификации (acc)	Параметр ошибки (loss)
1	0.3074	1.3928
2	0.3074	1.3837
3	0.3595	1.3696
4	0.3638	1.3595
5	0.3953	1.3362
6	0.4339	1.3169
7	0.4046	1.2972
8	0.5375	1.2570
9	0.5275	1.2271
10	0.5454	1.1953
11	0.5640	1.1583
12	0.5847	1.1262
13	0.5933	1.0997
14	0.5826	1.0850
15	0.6026	1.0586

После обучения модели на 15 эпохах точность классификации на тестовых данных составляла 69.70%.

Таблица 2. Результаты обучения нейронной сети с использованием модели VGG16.

Эпоха	Точность классификации (acc)	Параметр ошибки (loss)
1	0.4132	1.3289
2	0.5818	1.2824
3	0.6969	1.2186
4	0.7641	1.1081
5	0.8242	0.9684
6	0.8513	0.8280
7	0.8663	0.7103
8	0.8799	0.6192
9	0.8813	0.5591
10	0.8863	0.5076
11	0.8906	0.4703
12	0.8942	0.4380
13	0.8935	0.4150
14	0.8964	0.3962
15	0.8971	0.3744

Можем заметить, что скорость повышения точности обучения падает начиная с восьмой эпохи.

После обучения модели на 10 эпохах точность классификации на тестовых данных составляла 94.32%.

После обучения модели на 15 эпохах точность классификации на тестовых данных составляла 94.57%.

В результате обучения мы получили файлы с описанием структуры нейронной сети и весами.

Эти файлы понадобятся для практической реализации программы просмотра и редактирования изображений.

4. Описание разработанной программы.

Перед запуском программы необходимо установить Python. Для того, чтобы поставить все необходимые библиотеки, надо выполнить следующую команду в корневом каталоге программы:

pip install –r requirements

Запускается программа командой:

python papers.py

4.1. Общие возможности

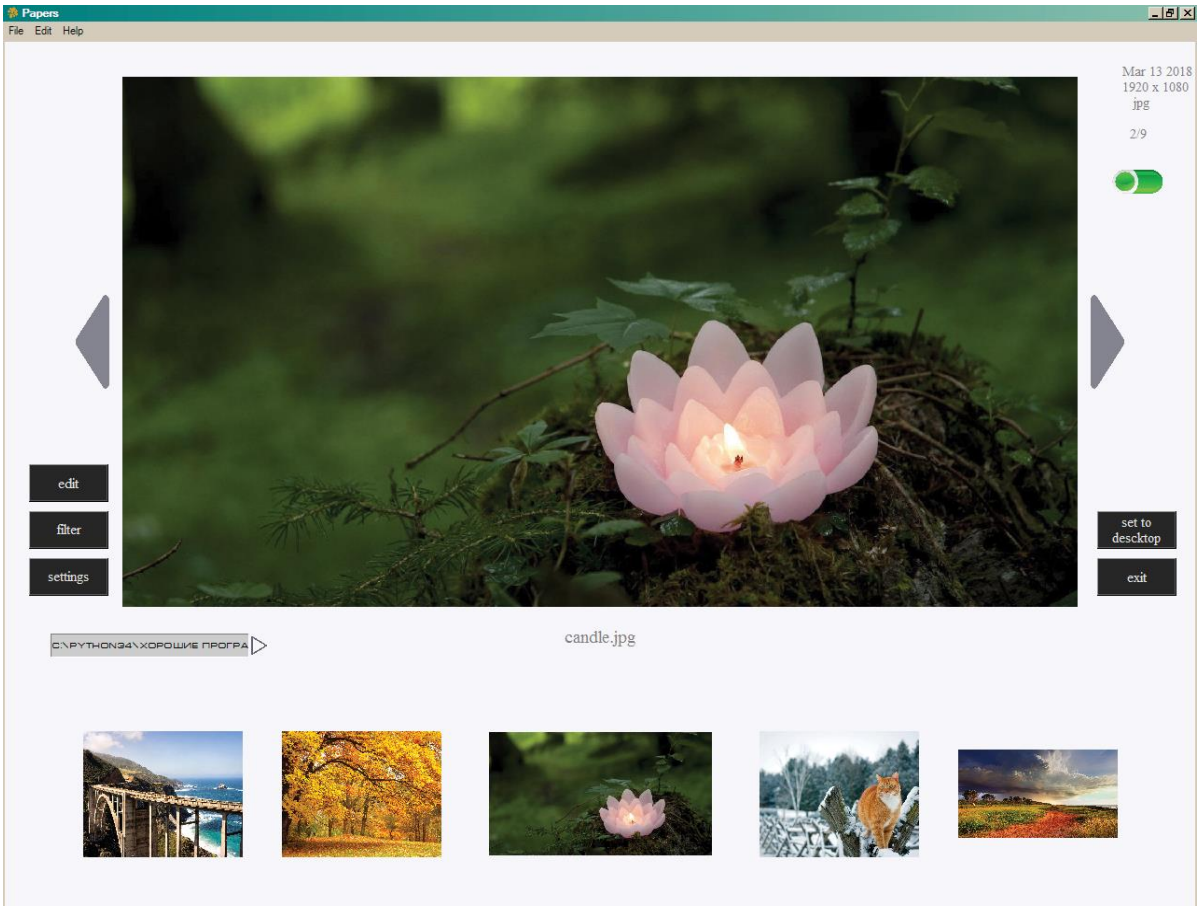

Рисунок 3 – Интерфейс главного окна программы

Разработанная программа имеет возможность:

а) Просматривать изображения в папке. Для этого из выбранной папки программа получает имена всех файлов изображений, затем фильтрует по списку допустимых для картинок расширений. Из получившегося списка выбирается первая картинка и выводится на главный экран. На экране прокрутки снизу отображаются соседние картинки.

б) Перелистывать изображения. Необходимо проитерироваться по списку изображений вперед или назад, новое изображение выводится на экран, полоса прокрутки также обновляется.

в) Фильтровать изображения по расширению, имени, размеру.

г) Фильтрация изображений по содержанию с помощью нейронной сети. Для этого каждое изображение в директории проверяется на принадлежность к какой-либо категории.

д) Режим полноэкранного просмотра изображения. По клику на отображаемое изображение, оно начинает отображаться в полноэкранном режиме. Для выхода из этого режима достаточно нажать на любую клавишу.

е) Режим слайд-шоу.

ж) Удалять, изменять размер, имя и тип изображения с помощью элементов интерфейса.

ж) Редактирование изображений.

з) Применение отображаемой картинки в качестве заставки рабочего стола (эта функция доступна только в Windows).

По нажатию на соответствующую клавишу программа переходит в режим редактирования.

4.2. Режим редактирования

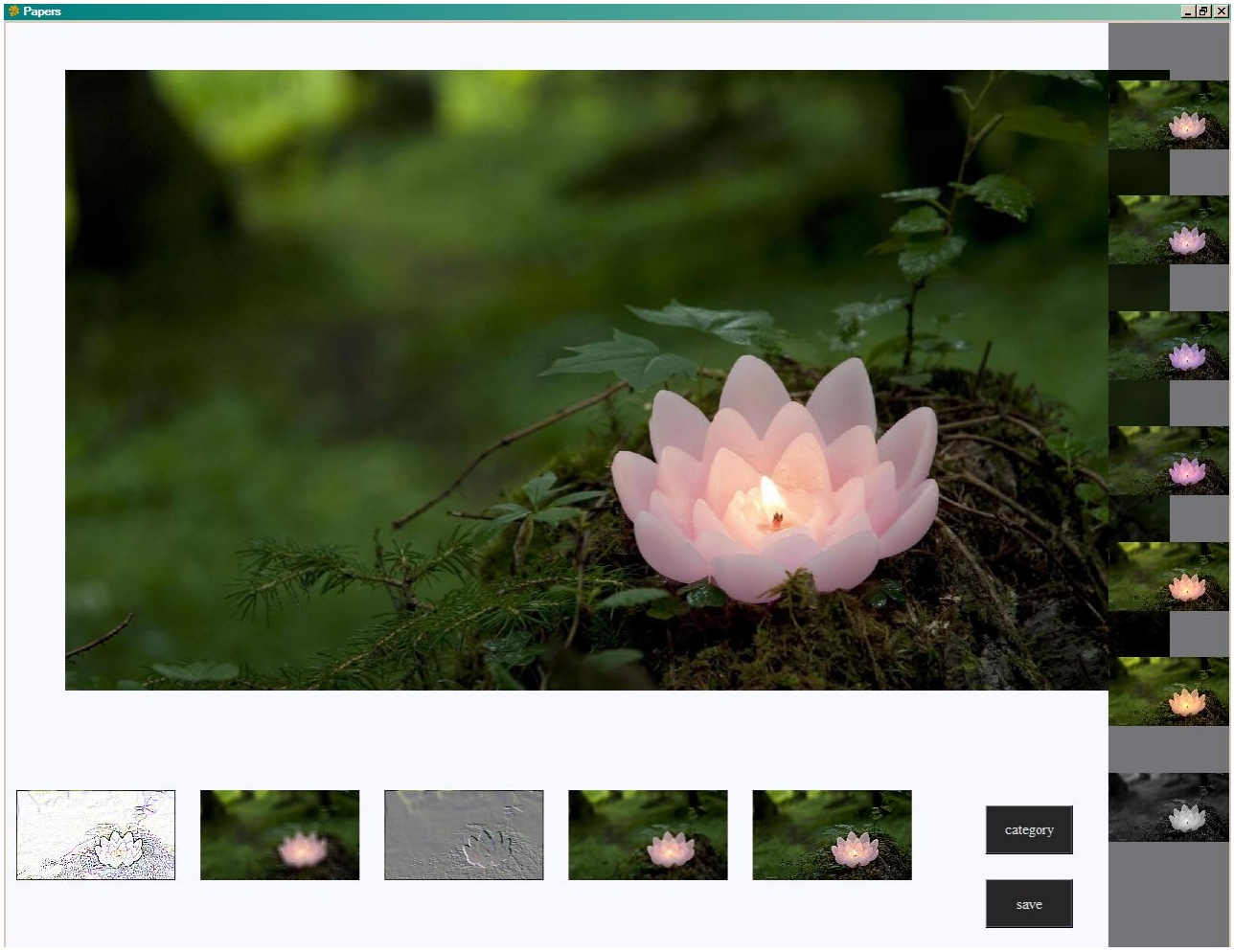

Рисунок 4 – Интерфейс окна редактирования

Режим редактирования имеет следующий интерфейс:

а) Главное изображение.

б) Вертикальная полоса цветовых фильтров.

в) Горизонтальная полоса эффектов (размытие, увеличение резкости).

г) Кнопка сохранения.

д) Кнопка определения категории.

При выборе какого-либо фильтра или эффекта главное изображение соответствующе меняется. Можно выбрать несколько эффектов, которые будут сочетаться между собой и выбранным цветовым фильтром.

Эффекты реализованы с помощью встроенного модуля библиотеки Pillow: ImageFilter.

Цветовые фильтры реализованы на основе пиксельных преобразований (функция convert) с использованием полученных опытным путем матриц перехода.

При нажатии на кнопку определения категории будет произведена попытка определить категорию изображения с помощью ранее обученной нейронной сети. В качестве результата будет продемонстрирована гистограмма, где наиболее подходящей категории будет соответствовать больший выступ.

5. Заключение

В результате работы была разработана программа для просмотра и редактирования изображений на языке Python. В процессе работы была создана и обучена свёрточная нейронная сеть, способная распознавать изображения нескольких категорий. Была определена связь между количеством эпох обучения и результатом тестирования нейронной сети на контрольной выборке, а также был определен оптимальный способ проектирования нейронных сетей. В дальнейшем можно улучшить качество получаемой модели путем совершенствования архитектуры нейронной сети.

Список литературы

1. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

2. Костецкая Г.Ю, Федяев О.И. Распознавание образов на основе свёрточных нейронных сетей

Приложение А. Текст программы для обучения нейронной сети

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Activation, Dropout, Flatten, Dense
from tensorflow.python.keras.applications import VGG16
from tensorflow.python.keras.optimizers import Adam
from tensorflow.python.keras.utils.np_utils import to_categorical
train_dir = my_path + 'train'
val_dir =  my_path + 'val'
test_dir =  my_path + 'test'
img_width, img_height = 150, 150
input_shape = (img_width, img_height, 3)
batch_size = 64
nb_train_samples = 600
nb_validation_samples = 100
nb_test_samples = 100
vgg16_net = VGG16(weights='imagenet', include_top=False,
                  input_shape=(150, 150, 3))
vgg16_net.trainable = False
model = Sequential()
model.add(vgg16_net)
```

```
model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('sigmoid'))
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=1e-5),
              metrics=['accuracy'])
datagen = ImageDataGenerator(rescale=1. / 255)
train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
test_generator = datagen.flow_from_directory(
```

```
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=15,
    validation_data=val_generator,
    validation_steps=nb_validation_samples // batch_size)
scores = model.evaluate_generator(test_generator, nb_test_samples // batch_size)
print("Результаты проверки на тестовой выборке: %.2f%%" % (scores[1]*100))
```