УДК 004.93, 004.8

АВТОМАТИЗАЦИЯ ПРОЦЕССА ИДЕНТИФИКАИИ ЮВЕЛИРНЫХ ИЗДЕЛИЙ НА ОСНОВЕ СВЕРТОЧНЫХ НЕЙРОСЕТЕЙ

Самыгин А. В., студент гр. ИИби-231, III курс
Научный руководитель: Протодьяконов А. В., доцент (к.н.) кафедры ИиАПС
Кузбасский государственный технический университет им. Т. Ф. Горбачева

1. Введение

В рамках данной работы представлена система автоматической классификации изображений ювелирных изделий, построенная на основе сверточных нейронных сетей (CNN). Система объединяет методы машинного обучения, современные подходы к обработке изображений и графический интерфейс пользователя (GUI), обеспечивая полный цикл работы с данными: добавление новых изображений в датасет, обучение и переобучение модели, а также выполнение предсказаний на новых данных.

Ключевая цель разработки — создание инструмента, позволяющего быстро и с высокой точностью определять тип ювелирного изделия на основе загруженного пользователем изображения.

2. Постановка задачи

Задача работы заключается в создании информационной системы, которая обеспечивает:

- классификацию изображений ювелирных изделий по категориям;
- возможность добавления новых изображений в датасет;
- переобучение модели на обновлённом датасете через графический интерфейс;
- вывод статистики по загруженным изображениям;

Дополнительно система должна быть способна:

- работать с изображениями различных разрешений и форматов;
- обеспечивать аугментацию данных для повышения качества модели;
- поддерживать масштабирование на большие датасеты;
- предоставлять визуализацию распределения классов и метрик точности.

Таким образом, задача включает как разработку эффективной модели классификации, так и создание удобного интерфейса, через который пользователи смогут управлять данными и получать результаты без программирования.

3. Архитектура программного комплекса

Информационная система реализована на языке Python с использованием среды разработки PyCharm. Выбор Python обусловлен наличием экосистемы библиотек для машинного обучения (PyTorch, TensorFlow, Keras), обработки

изображений (OpenCV, Pillow) и разработки графических интерфейсов (PyQt, Tkinter).

Архитектура системы является модульной и включает в себя следующие ключевые компоненты:

- модуль загрузки и предобработки данных;
- модуль нейросетевой модели;
- модуль предсказания;
- модуль графического интерфейса;
- модуль обучения и переобучения.

Логика взаимодействия пользователя с системой отражена на диаграммах деятельности:

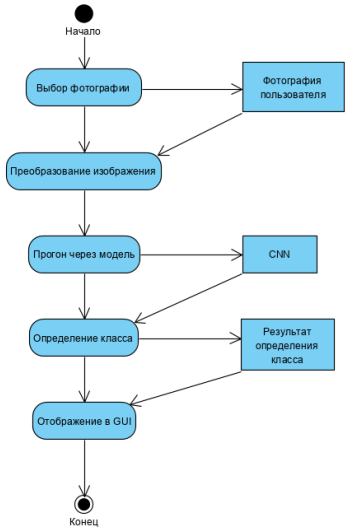

Рисунок 1 – Диаграмма деятельности процесса предсказания

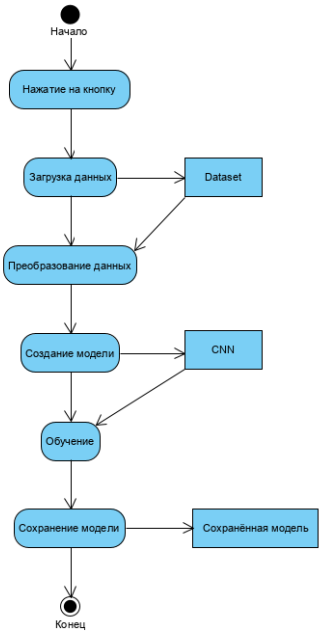

Рисунок 2 – Диаграмма деятельности процесса переобучения модели

3.1 Реализация ключевых функций

В данном разделе приведены фрагменты программного кода, реализующие основные функциональные возможности системы.

Функция предсказания. Данный фрагмент демонстрирует: Загрузку изображения и преобразование его в формат, подходящий для модели; Получение предсказания модели с вычислением вероятности; Возврат класса изделия и уверенности предсказания для отображения в графическом интерфейсе

```
def predict(image_path):  2 usages
    image = Image.open(image_path).convert("RGB")
    image = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        outputs = model(image)
        probs = torch.nn.functional.softmax(outputs, dim=1)
        conf, pred = torch.max(probs, 1)
    return classes[pred.item()], conf.item()
```

Рисунок 3 – Предсказание

Функция загрузки модели переобучения модели**.** Отвечает за инициализацию предварительно обученной сверточной нейросети

```
def load_model():  2 usages
    global model, classes
    import numpy  # для safe_globals
    with torch.serialization.safe_globals([numpy._core.multiarray._reconstruct]):
        checkpoint = torch.load( f: "jewelry_model.pth", map_location=device, weights_only=False)
    classes = checkpoint['label_encoder']
    model = SimpleJewelryCNN(num_classes=len(classes))
    model.load_state_dict(checkpoint['model_state_dict'])
    model.to(device)
    model.eval()

load_model()
```

Рисунок 4 – Загрузка модели

Функция переобучения модели. Отвечает за инициализацию предварительно обученной сверточной нейросети

```
def main():  1 usage
    print("Начинаем обучение модели...")
    dataset = JewelryDataset(CSV_FILE, DATASET_DIR, transform=transform)
    dataloader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=True)
    model = SimpleJewelryCNN(num_classes=len(dataset.classes)).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=LR)
    for epoch in range(EPOCHS):
        running_loss = 0.0
        for images, labels in dataloader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        print(f"Epoch {epoch+1}/{EPOCHS} - Loss: {running_loss/len(dataloader):.4f}")
    checkpoint = {
        'model_state_dict': model.state_dict(),
        'label_encoder': dataset.classes
    }
    torch.save(checkpoint,  f: "jewelry_model.pth")
    print("Модель сохранена как jewelry_model.pth")
```

Рисунок 5 – Переобучение модели

3.2 Пример работы приложения

На рисунке 6 представлен главный экран приложения.

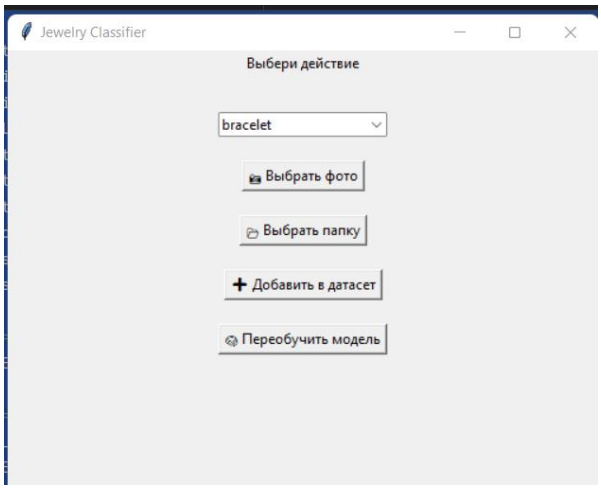

Рисунок 6 – Пример работы приложения

На рисунке 7 представлен результат после того, как мы выбираем фотографию.

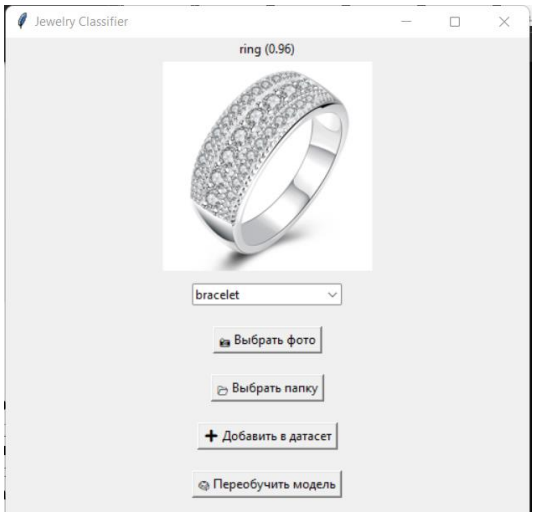

Рисунок 7 – Пример работы приложения

На рисунке 8 представлен результат после того, как мы выбираем папку, где хранятся фотографии.

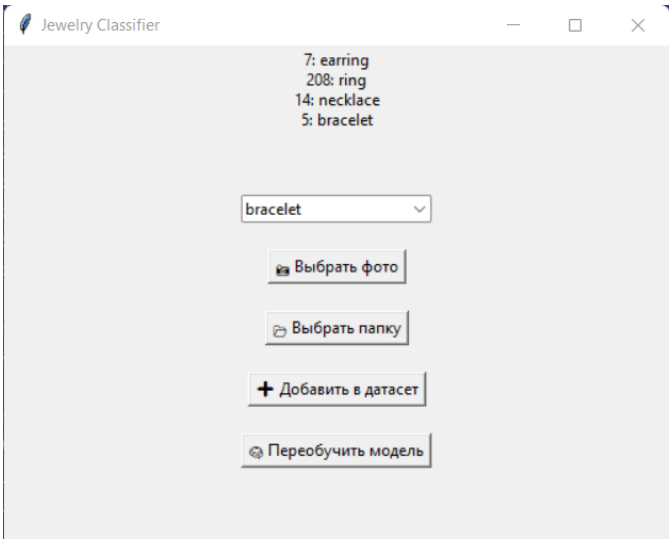

Рисунок 8 – Пример работы приложения

На рисунке 9 представлен результат после того, как мы выбираем категорию, нажимаем на кнопку “Добавить в датасет” и выбираем фотографию. Данная фотографию добавляется в датасет.

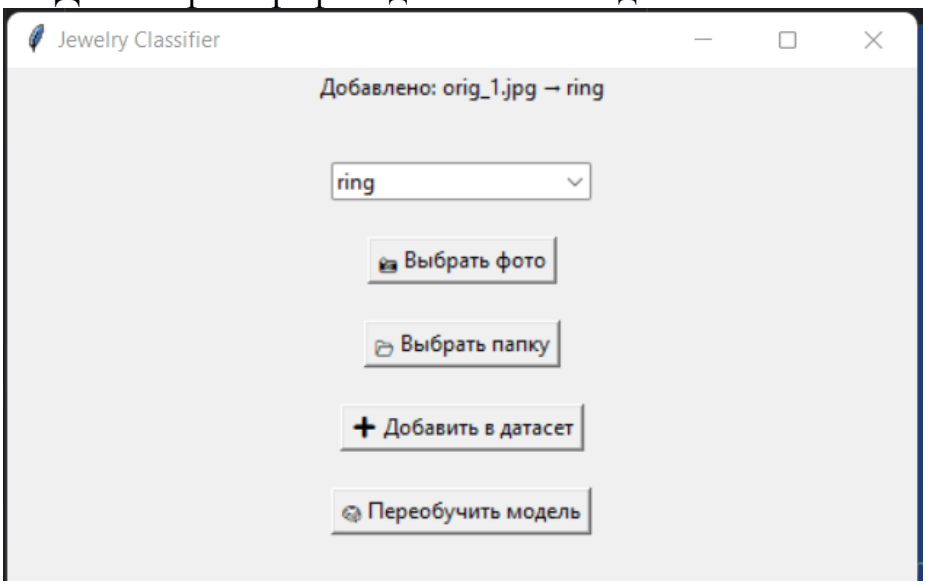

Рисунок 9 – Пример работы приложения

4. Результаты и тестирование

Для оценки качества работы системы был проведен эксперимент на выборке из 500 изображений, включающей несколько категорий ювелирных изделий.

Данные были разделены на:

- обучающую выборку (80%);
- тестовую выборку (20%).

В результате тестирования были получены следующие показатели:

- точность классификации: 93,5%;
- стабильность результатов при различных условиях освещения;
- высокая скорость обработки (менее 1 секунды на изображение).

Анализ ошибок показал, что наибольшие трудности возникают при классификации визуально схожих изделий (например, кольца и серьги с похожим дизайном).

5. Заключение

В ходе выполнения работы была разработана интеллектуальная система классификации ювелирных изделий, основанная на использовании сверточных нейронных сетей.

Система обеспечивает:

- автоматическую обработку изображений;
- высокую точность классификации;
- удобный пользовательский интерфейс;
- возможность динамического обновления модели.

Практическая значимость работы заключается в возможности применения разработанной системы в:

- интернет-магазинах;
- складских системах учета;

- автоматизированных каталогах продукции.

Дальнейшее развитие системы может включать:

- увеличение объема датасета;
- внедрение более сложных архитектур нейросетей;
- интеграцию с облачными сервисами;
- добавление функции распознавания дефектов.

6. Список литературы

1. Гудфеллоу, Я., Бенджио, И., Курвилль, А. "Глубокое обучение". — М.: ДМК Пресс, 2017. — 652 с.

2. Шолле, Ф. "Глубокое обучение на Python". — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-0770-4. — ББК 32.973.2-018.1. — УДК 004.43.