УДК 004.423

РАЗБИЕНИЕ ДАННЫХ НА ТРЕНИРОВОЧНЫЕ, ВАЛИДАЦИОННЫЕ И ТЕСТОВЫЕ ПРИ ОБУЧЕНИИ СВЕРТОЧНОЙ НЕЙРОСЕТИ

Кулинча П.В.,
студент факультета информатики и вычислительной техники,
Хакасский государственный университет имени Н.Ф. Катанова,
г. Абакан, Россия
Спирин Д.В.,
научный руководитель, доцент кафедры ЦТиД,
Хакасский государственный университет имени Н.Ф. Катанова,
г. Абакан, Россия

Аннотация: В данной статье рассматривается важность корректного разбиения датасета при обучении сверточных нейросетей. Представлена реализация на Python с использованием библиотеки scikit-learn для создания тренировочной, валидационной и тестовой выборок, с учётом структуры данных и принадлежности изображений одному человеку. Такое разбиение повышает достоверность оценки модели и предотвращает переобучение.

Ключевые слова: сверточные нейросети, машинное обучение, разбиение данных, валидация, тестирование, Keras, подготовка изображений.

DATA PARTITIONING INTO TRAINING, VALIDATION AND TEST WHEN TRAINING A CONVOLUTIONAL NEURAL NETWORK

Kulincha P.V.,
Student of the Faculty of Informatics and Computer Engineering,
Khakass State University named after N.F. Katanov,
Abakan, Russia

Spirin D.V.,
Scientific Supervisor, Associate Professor of the Department of Computer Science and Computer Engineering,
Khakass State University named after N.F. Katanov,
Abakan, Russia

Annotation: This article explores the importance of proper dataset splitting when training convolutional neural networks. It presents a Python-based implementation using the scikit-learn library to divide images into training, validation, and test sets, taking into account the grouping of data by individual identity. Such an approach improves model evaluation reliability and prevents overfitting.
Keywords: convolutional neural networks, machine learning, data splitting, validation, testing, Keras, image preprocessing.

Введение

Разбиение набора изображений на обучающую (training), валидационную (validation) и тестовую (test) выборки является важнейшим этапом в процессе построения и оценки нейросетевой модели. Данный подход позволяет эффективно обучить сверточную нейросеть (CNN), провести контроль за переобучением и объективно оценить обобщающую способность модели на новых, ранее не виденных данных.

Исторический аспект сверточных нейросетей

Сверточные нейросети (CNN, Convolutional Neural Networks) были впервые предложены в конце 1980-х годов, однако широкое признание они получили благодаря работе Яна ЛеКуна (Yann LeCun), который в 1998 году представил архитектуру LeNet-5. Эта модель была разработана для распознавания рукописных цифр и использовалась в системе автоматического считывания ZIP-кодов на письмах в США.

LeNet-5 стала первой успешной реализацией сверточной нейросети, включающей в себя чередование сверточных и субдискретизирующих (пулинговых) слоёв. Именно эта архитектура заложила основу для более современных моделей, таких как AlexNet, VGG, ResNet и других, которые сегодня широко применяются в задачах компьютерного зрения — от диагностики медицинских изображений до распознавания лиц и автономного вождения.

Теоретические основы

При обучении сверточной нейросети, обучающая выборка используется для обновления весов модели посредством градиентного спуска. Валидационная выборка позволяет отслеживать метрики качества модели на каждом этапе обучения и корректировать параметры, предотвращая переобучение. Тестовая выборка применяется исключительно на финальном этапе и служит независимой метрикой качества, которая демонстрирует, насколько хорошо модель может работать в реальных условиях.

Нарушение этого принципа, например, использование одних и тех же изображений в обучении и тестировании, приводит к завышенной оценке точности модели и неспособности адекватно справляться с новыми данными [1].

Описание используемого датасета

В рамках настоящего эксперимента используется датасет «**Selfies ID Images dataset»** [2], содержащий изображения людей, сгруппированных по уникальным идентификаторам **SetId**, которые соответствуют одному человеку. Каждая группа включает как селфи, так и официальные ID-фотографии.

Для целей обучения нейросети на задачу **идентификации личности по селфи**, из набора **исключаются ID-фотографии**, так как они, как правило, сделаны в других условиях и могут нарушить единообразие тренировочного набора. Оставшиеся изображения включают:

- разнообразные ракурсы;

- различные выражения лиц;
- различные условия освещённости и фона.

Набор представлен двумя подкаталогами:

- **18_sets_Caucasians** — изображения 18 человек европеоидной расы;
- **11_sets_Hispanics** — изображения 11 человек латиноамериканского происхождения.

Таким образом, итоговый датасет включает **29 классов (людей)**, каждый из которых представлен множеством изображений для обучения, валидации и тестирования. Это обеспечивает возможность построения устойчивой модели, способной различать лица даже в условиях визуальных искажений, с которыми сталкиваются реальные системы распознавания.

Практическая реализация

В данной работе в качестве примера рассмотрим реализацию разбиения данных в среде Google Colab. Используемый набор изображений содержит фото людей, сгруппированных по идентификаторам SetId. Для корректного разбиения и организации структуры папок была реализована следующая последовательность действий:

1. Импорт необходимых библиотек и задание путей к изображениям (рисунок 1).

```
import os
import shutil
from sklearn.model_selection import train_test_split
```

Рисунок 1 – Фрагмент кода программы [разработано автором]

На этом этапе подключаются необходимые модули:

- os и shutil — для работы с файловой системой;
- train_test_split из sklearn — для случайного разбиения списков изображений.

```
# Путь к папке с изображениями
images_base_path = "/content/drive/My Drive/DataFramesImages/Selfies ID Images dataset/"
# Подпапки с фото
subfolders = ["18_sets_Caucasians", "11_sets_Hispanics"]
```

Рисунок 2 – Фрагмент кода программы [разработано автором]

Указывается путь к папке с изображениями, а также список подпапок, в которых хранятся изображения разных этнических групп. Это позволяет гибко работать с источниками данных, даже если они структурированы по подкаталогам.

2. Удаление существующей структуры и создание новой иерархии директорий: train/, validation/ и test/.

```
# Создание папок для train, validation и test
base_dir = "/content/face_dataset/"
train_dir = os.path.join(base_dir, "train")
val_dir = os.path.join(base_dir, "validation")
test_dir = os.path.join(base_dir, "test")
```

Рисунок 3 – Фрагмент кода программы [разработано автором]

Создаются отдельные директории для трех типов выборок. Это важно, потому что ImageDataGenerator.flow_from_directory() ожидает такую структуру: /train/класс/изображения.jpg.

```
# Очищаем папки, если они уже существуют
if os.path.exists(base_dir):
    shutil.rmtree(base_dir)
# Создаём заново
os.makedirs(train_dir)
os.makedirs(val_dir)
os.makedirs(test_dir)
```

Рисунок 4 – Фрагмент кода программы [разработано автором]

Перед созданием новых папок код удаляет уже существующую структуру (если она есть), чтобы избежать конфликтов или дублирования данных.

3. Группировка изображений по SetId, исключение ID-фотографий (например, паспортных), так как они могут исказить процесс обучения.

```
# Группируем данные по SetId (каждый SetId – это отдельный человек)
grouped = df.groupby("SetId")
```

Рисунок 5 – Фрагмент кода программы [разработано автором]

Данные в DataFrame (предположительно df) группируются по идентификатору человека (SetId). Это позволяет разрабатывать логику разбиения индивидуально для каждого человека, гарантируя, что фотографии одного и того же лица попадут только в одну из выборок (что очень важно для задач классификации по лицам).

4. Разбиение оставшихся изображений по принципу: 80% на обучение, 15% на валидацию и 5% на тест.

```
# Разбиваем данные для каждого человека
for person_id, group in grouped:
    # Получаем имя человека по его SetId
    person_name = group["Name"].iloc[0]  # Берем имя из первой строки группы
    person_name = person_name.replace(" ", "_")  # Заменяем пробелы на "_", чтобы не было ошибок
```

Рисунок 6 – Фрагмент кода программы [разработано автором]

Цикл проходит по каждому человеку. Имя используется как имя класса (папки), но пробелы заменяются на _ — это важно, чтобы избежать ошибок при сохранении файлов в систему.

```
# Исключаем ID-фотографии (не берём фото с названием, содержащим "ID" или "id")
group = group[~group["FName"].str.contains("ID", case=False, na=False)]
# Если после фильтрации у человека не осталось фото, пропускаем его
if group.empty:
    continue
```

Рисунок 7 – Фрагмент кода программы [разработано автором]

Исключаются ID-фотографии, потому что они могут отличаться по качеству и формату. Если после фильтрации изображений не осталось — человек пропускается.

```
# Разделяем на train (80%) и temp (20%)
train_files, temp_files = train_test_split(group["FName"], test_size=0.2, random_state=42)
# Разделяем temp на validation (15%) и test (5%)
val_files, test_files = train_test_split(temp_files, test_size=0.25, random_state=42)  # 5% от общего объёма
```

Рисунок 8 – Фрагмент кода программы [разработано автором]

Разбиение:

- **80%** изображений идёт в train;
- оставшиеся **20%** разбиваются на:
  - 15% — validation;
  - 5% — test.

5. Создание папок по именам людей, куда копируются соответствующие изображения из подкаталогов.

```
# Создаём папки с именами людей вместо SetId
os.makedirs(os.path.join(train_dir, person_name), exist_ok=True)
os.makedirs(os.path.join(val_dir, person_name), exist_ok=True)
os.makedirs(os.path.join(test_dir, person_name), exist_ok=True)
```

Рисунок 9 – Фрагмент кода программы [разработано автором]

Для каждого человека создаются отдельные папки в каждой из трёх выборок. Это обязательно для flow_from_directory() — он ожидает структуру: /тип_выборки/класс/фото.jpg.

6. Копирование файлов происходит с проверкой их существования по заранее известным путям.

```
# Функция для копирования файлов
def copy_files(file_list, dest_folder):
    for file_name in file_list:
        url = group.loc[group["FName"] == file_name, "URL"].values[0]
        for subfolder in subfolders:
            file_path = os.path.join(images_base_path, subfolder, url)
            if os.path.exists(file_path):
                shutil.copy(file_path, os.path.join(dest_folder, person_name, file_name))
```

Рисунок 10 – Фрагмент кода программы [разработано автором]

Функция copy_files получает список файлов и копирует изображения из исходных папок (subfolders) в соответствующие директории. Используется путь URL из таблицы df, что особенно удобно при работе с метаинформацией в DataFrame.

```
    # Копируем изображения
    copy_files(train_files, train_dir)
    copy_files(val_files, val_dir)
    copy_files(test_files, test_dir)
print("Разбиение завершено ✅")
```

Рисунок 11 – Фрагмент кода программы [разработано автором]

Финальный шаг — распределение файлов по папкам train, validation, test.

Итого, этот код позволяет:

- системно разделить изображения по классам и выборкам;
- сохранить структуру, совместимую с Keras-генераторами;
- учитывать индивидуальные особенности каждой группы (человека);
- автоматизировать подготовку данных для обучения модели.

На рисунках 12-13 представлен результат выполнения программы

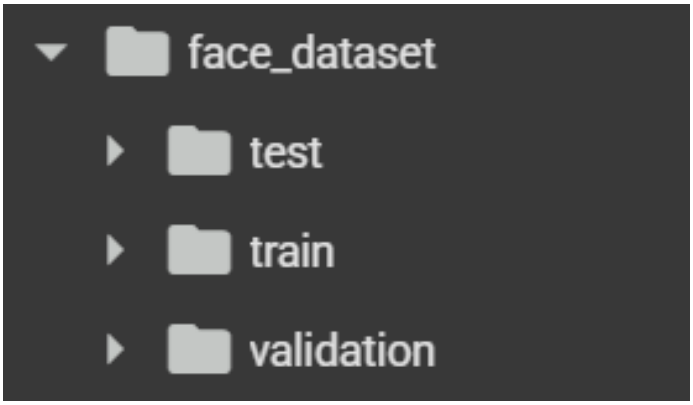

Рисунок 12 – Результат выполнения программы [разработано автором]

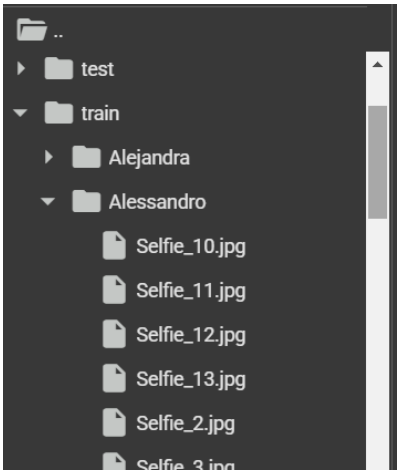

Рисунок 13 – Результат выполнения программы [разработано автором]

Заключение

Разбиение данных на тренировочную, валидационную и тестовую выборки является важнейшим этапом при обучении сверточных нейросетей. Оно позволяет корректно оценивать качество модели и предотвращать переобучение. В данной работе была реализована автоматизированная система распределения изображений по выборкам с учетом принадлежности к конкретным пользователям. Такой подход обеспечивает корректную генерацию выборок без пересечений и утечек данных между этапами обучения и тестирования. Применение данной методики в связке с библиотекой Keras и генераторами изображений облегчает последующий процесс обучения модели и делает его более воспроизводимым и масштабируемым.

Библиографический список:

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

2. Selfies & ID Images Dataset, 95,000 files [Электронный ресурс] – URL: https://www.kaggle.com/datasets/tapakah68/selfies-id-images-dataset

Оригинальность 75%