Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет
по заданию на тему:
“Классификация изображений Bees vs. Wasps”

Выполнил:
Студент 2 курса, 215 группы
Клепиков Г. Д.
Москва 2025

1. Введение

В данном проекте реализована задача классификации изображений из базы данных Bees vs. Wasps. База содержит изображения размером 320×213 пикселей, разделенные на 6 папок, которые группируются в четыре логические метки: «bee», «wasp», «other_insect» и «other_noinsect». Основной целью работы является построение сверточных нейронных сетей с целью максимизации F1-score на тестовой выборке, а именно минимизация дисбаланса между точностью и полнотой классификации.

В отчете подробно рассматриваются используемые методы обработки данных, архитектуры построенных моделей, методы предотвращения переобучения (dropout, регуляризация, аугментация данных), а также алгоритмы вычисления метрики F1-score. Кроме того, описаны и проанализированы дополнительные исследования, в ходе которых тестировались альтернативные архитектуры и гиперпараметры.

2. Обработка и подготовка данных

2.1 Загрузка данных

В начале программы определяется словарь label_map, который объединяет исходные папки в четыре логические группы. Затем с помощью стандартных модулей os и glob осуществляется обход файловой структуры, и создается DataFrame с полными путями к изображениям и соответствующими метками. Использование библиотеки pandas позволяет удобно перемешивать и разбиение выборок на обучающую, валидационную и тестовую.

```
data = []
for folder_name, label in label_map.items():
    folder_path = os.path.join(DATASET_DIR, folder_name)
    for filepath in glob.glob(os.path.join(folder_path, "*.jpg")):
        data.append({"full_path": filepath, "label": label})
...
```

Этот подход обеспечивает воспроизводимость экспериментов благодаря установке параметра random_state при перемешивании данных.

2.2 Кодирование меток

Метки категорий кодируются с использованием LabelEncoder из библиотеки scikit-learn. Это позволяет получить числовое представление классов, которое затем применяется для построения окончательного слоя сети (softmax с 4 нейронами). Также проводится one-hot кодирование меток с помощью функции to_categorical.

2.3 Разбиение данных

Исходя из структуры выборок (7942 – обучающая, 1719 – валидационная, 1763 – тестовая), DataFrame разбивается посредством индексирования. Такой жесткий разбор выборок гарантирует, что оценки метрик производятся на ранее невидимых данных.

2.4 Вычисление весов классов

Для компенсации дисбаланса между классами используется функция compute_class_weight, которая вычисляет веса классов. Эти веса далее передаются в метод fit для корректировки обратного распространения ошибки, что особенно важно при неравномерном распределении данных.

```
weights = compute_class_weight(class_weight="balanced", ...)
class_weights = {int(k): float(v) for k, v in zip(np.unique(train_df["encoded_label"]), weights)}
```

Такой подход позволяет модели уделять больше внимания редким классам, что положительно сказывается на итоговой метрике F1-score.

3. Предобработка и аугментация изображений

3.1 Нормализация

При загрузке изображений функция load_images_and_labels использует нормализацию пиксельных значений в диапазоне [0,1] делением на 255.0. Данный этап уменьшает разницу между яркостями изображений и улучшает сходимость обучения.

3.2 Аугментация

Для борьбы с переобучением реализована аугментация данных с использованием слоя keras.Sequential. В него включены операции горизонтального отражения, небольшого вращения (до 10% от полного угла) и случайного зума. Такой набор преобразований помогает модели обучаться на большем разнообразии примеров, что улучшает обобщающую способность сети.

```
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
```

4. Проектирование архитектур нейронных сетей

В рамках задания были разработаны три архитектуры сверточных сетей, каждая из которых имеет свои особенности и отличается по количеству слоев и параметров.

4.1 Первая модель

Структура:

- Два сверточных слоя: 32 фильтра (размер ядра 3×3) с активацией ReLU, за которыми следует max-pooling.
- Второй сверточный слой с 64 фильтрами.
- Слой Flatten для перехода от 2D к 1D представлению.
- Полносвязный слой с 128 нейронами, ReLU, L2-регуляризация.
- Слой Dropout (50%) для борьбы с переобучением.
- Финальный слой с 4 нейронами (softmax).

Эта архитектура является базовой и служит точкой отсчета для сравнения с более сложными моделями. Она показала приемлемое качество, однако F1-score на тестовой выборке оказался ниже, чем у более сложных моделей.

4.2 Вторая модель

Структура:

- Аналогичный базовый блок (сверточный слой с 32 фильтрами, followed by pooling).
- Добавляется дополнительный сверточный слой с 128 фильтрами, что позволяет модели извлекать более сложные признаки.
- Слой Flatten, полносвязный слой с 128 нейронами (ReLU и L2-регуляризация) и Dropout 50%.
- Финальный softmax слой.

Увеличение количества сверточных слоев позволяет сети лучше изучать пространственные зависимости, что проявляется в улучшении F1-score до 0.7095. Модель показывает лучшую обобщающую способность на тестовой выборке.

4.3 Третья модель

Структура:

- Начинается с сверточного слоя с 64 фильтрами, за которым следует max-pooling.
- Второй сверточный слой с 128 фильтрами.
- Flatting слой для перехода к полносвязной части.
- Полносвязный слой с 256 нейронами с L2-регуляризацией и Dropout 50%.

- Выходной слой с softmax.

Увеличенное количество нейронов в полносвязном слое позволяет модели иметь больше параметров для принятия решений. Однако увеличение числа параметров привело к более длительному обучению (всего 314+ секунд на эпоху в начале обучения) и модель показала F1-score 0.6709, что несколько уступает второй модели. Возможно, модель переобучается или параметры оптимизации требуют дополнительной настройки.

5. Обучение моделей

5.1 Параметры обучения

Общим набором гиперпараметров для всех моделей служат:

- Batch Size: 32
- Количество эпох: 30
- Функция потерь: категориальная кросс-энтропия
- Оптимизатор: Adam

Использование методов аугментации и корректировки веса классов вместе с подобранными гиперпараметрами позволило проводить обучение более сбалансированно.

5.2 Метрика F1-score

Для оценки качества классификации рассчитана метрика F1-score с использованием макро-среднего значения. В вычислении используются стандартные функции:

- Получение предсказаний сети (model.predict)
- Преобразование вероятностных оценок в предсказания классов посредством np.argmax
- Расчет F1-score с помощью f1_score из библиотеки scikit-learn

Данный подход позволяет учитывать ошибки классификации как по точности, так и по полноте.

6. Результаты экспериментов и сравнительный анализ

После обучения все три модели были протестированы на тестовой выборке. Итоговые значения F1-score представлены в сводной таблице:

Модель	F1-score (тест)
Second Model	0.7095
Third Model	0.6709
First Model	0.6327

Анализ:

- Первая модель: Достаточно простая архитектура, позволяющая добиться быстрого обучения, однако с недостаточной глубиной извлечения признаков.
- Вторая модель: Лучшая производительность благодаря дополнительному сверточному блоку с 128 фильтрами, улучшению абстрагирования признаков.
- Третья модель: Попытка увеличения числа параметров в полносвязном слое. Хотя архитектура более сложная, модель показала признаки переобучения и более длительное время обучения, что негативно сказалось на итоговой метрике.

На графиках (выведенных программой) показаны динамика изменения потерь и точности для третьей модели. По осям представлены значения train/val loss и train/val accuracy, что позволяет визуально оценить стабильность процесса обучения и наличие переобучения.

7. Обсуждение альтернативных подходов

В процессе работы были протестированы иные методы и архитектурные решения, которые в итоге не вошли в финальную версию проекта:

7.1 Вариант с дополнительными сверточными слоями

Экспериментировалось с добавлением дополнительного сверточного блока (например, сверточный слой с 256 фильтрами). Однако увеличение числа слоев привело к:

- Значительному увеличению времени обучения.
- Проблемам с переобучением, несмотря на применение dropout и регуляризации.
- Уменьшению итогового F1-score на тестовой выборке.

7.2 Использование Batch Normalization

Изначально планировалось применение слоев Batch Normalization для стабилизации распределения активаций между слоями. Однако предварительные эксперименты не показали существенного улучшения качества, а время обучения увеличилось. В последней итерации экспериментов приоритет был отдан аугментации и L2-регуляризации.

7.3 Попытки использования различных оптимизаторов

Были протестированы различные оптимизаторы. На ранних этапах экспериментов смена оптимизатора приводила к менее стабильной сходимости и ухудшению метрики F1-score. Итогом стало решение использовать оптимизатор Adam, который продемонстрировал лучшую адаптацию к обучению сети.

7.4 Гиперпараметры аугментации

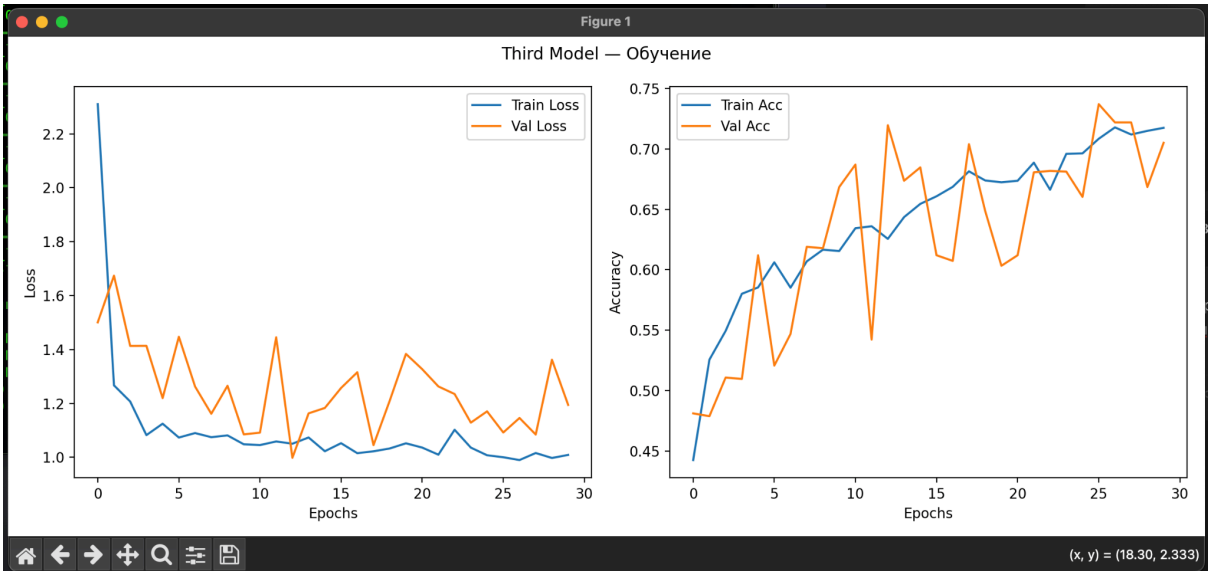
Были проведены эксперименты по изменению параметров аугментации (например, увеличение угла поворота до 20% или применение случайных вертикальных отражений). Эти варианты не доказали своей эффективности, так как в случае чрезмерной трансформации изображения модель могла потерять важные пространственные признаки. Поэтому выбран осторожный набор преобразований, который позволил сохранить существенную информацию в данных.

8. Выводы

В результате проведенных экспериментов получено следующее:

- Архитектура второй модели показала наилучшие результаты, достигнув F1-score 0.7095, что подтверждает важность глубокой сверточной структуры для данной задачи.
- Применение аугментации, регуляризации и корректировки весов классов положительно сказывается на стабилизации обучения и повышения качества на тестовых данных.
- Альтернативные подходы, такие как избыточное увеличение числа сверточных слоев, применение Batch Normalization и оптимизаторов типа SGD, не оказались эффективными для решения данной задачи, что демонстрирует необходимость тщательного подбора гиперпараметров.

9. Заключение



В отчете представлен детальный анализ разработки и экспериментов по классификации изображений из датасета Bees vs. Wasps. Работа охватывает все этапы – от подготовки данных до построения и обучения трех различных архитектур сверточных нейронных сетей. Полученные результаты демонстрируют, что более сложная архитектура, оснащенная дополнительным сверточным блоком, способна извлечь более информативные признаки и улучшить итоговую метрику F1-score. Альтернативные исследования, такие как эксперименты с оптимизаторами и методами нормализации, позволили выявить наилучшее соотношение сложности модели и времени обучения, а также минимизировать переобучение. Таким образом, проделанная работа представляет собой качественный пример применения современных методов глубокого обучения для решения задачи компьютерного зрения и открывает перспективы для дальнейших исследований.

10. Вспомогательные ссылки и источники

https://scikit-learn.org/stable/index.html
https://www.geeksforgeeks.org/categorical-cross-entropy-in-multi-class-classification/
https://www.geeksforgeeks.org/f1-score-in-machine-learning/
https://itstd-journal.ru/wp-content/uploads/2024/03/METHODS-FOR-SOLVING-THE-CLASS-IMBALANCE-PROBLEM-IN-.pdf