ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

**Программа подготовки бакалавров по направлению
Компьютерные науки и технологии**

Пестов Лев Евгеньевич

КУРСОВАЯ РАБОТА

Интерактивный конструктор моделей искусственного интеллекта с поддержкой мультимодальных задач.
Реализация пайплайна обучения глубоких нейросетей

Научный руководитель
старший преподаватель НИУ
ВШЭ - НН

Саратовцев Артём Романович

Нижний Новгород, 2025г.

Структура работы

1. Введение

Развитие искусственного интеллекта (ИИ) и его интеграция в различные сферы деятельности привело к значительному увеличению числа приложений, использующих глубокие нейронные сети. Однако процесс разработки таких моделей по-прежнему остается сложной и ресурсоемкой задачей, требующей от разработчика понимания не только архитектуры нейросетей, но и глубоких знаний в области обработки данных, настройки гиперпараметров и оптимизации вычислительных ресурсов. Это существенно ограничивает доступность технологий машинного обучения для широкой аудитории, в частности, для исследователей, предпринимателей и специалистов, не обладающих достаточным уровнем подготовки в данной области.

Данная работа направлена на создание интерактивного конструктора моделей искусственного интеллекта, который позволит пользователям, не имеющим значительного опыта в разработке ИИ, самостоятельно обучать нейросети на своих данных под определенные классы задач. В отличие от существующих решений, сервис ориентирован на адаптацию моделей к малым объемам данных, что особенно актуально в условиях, когда сбор и разметка больших датасетов являются трудоемкими и дорогостоящими.

Существенной частью системы является интеграция с облачными вычислительными мощностями Yandex Cloud, что позволяет пользователям запускать обучение моделей удалённо, без необходимости наличия высокопроизводительного оборудования. Впоследствии обученные модели могут быть загружены обратно в сервис для тестирования либо экспортированы в виде предобученных весов с преднаписанным кодом для внедрения в сторонние проекты.

Целью исследования является эмпирическое сравнение и выбор оптимальных архитектур и стратегий обучения глубоких нейронных сетей для задач *классификации изображений и звуков*, применимых к сценариям с малым объемом данных и ограниченными вычислительными ресурсами, для последующей интеграции в интерактивный конструктор.

Задачи проекта:

1. Провести аналитический обзор литературы и выбрать лучшие архитектуры моделей под разные домены машинного обучения, также рассмотреть способы аугментации данных.

2. Реализовать пайплайны обучения с использованием выбранных архитек-

тур и стратегий для задач классификации изображений и звуков.

3. На основе результатов сравнения сделать обоснованный выбор оптимальных архитектур и стратегий обучения для каждой из задач применительно к условиям работы сервиса.

2. Разработка пайплайна обучения нейросетей для задачи классификации изображений

2.1. Теоретические основы и архитектуры нейросетей для классификации изображений

Задача классификации изображений является одной из основных в компьютерном зрении и представляет собой процесс экстракции высокоуровневых и низкоуровневых признаков в один или несколько классов. Для решения таких задач в последние годы наиболее эффективными оказались глубокие нейронные сети, в частности, сверточные нейронные сети (CNN) и трансформеры (ViT).

Сверточные нейронные сети (Convolutional Neural Networks, CNN) - это класс глубоких нейронных сетей, специально разработанных для обработки изображений. Их ключевой особенностью является использование операции свертки вместо обычного матричного умножения хотя бы в одном из слоев. CNN произвели революцию в области компьютерного зрения, предоставив эффективный способ автоматического извлечения признаков из изображений.

Операция свертки - Фильтр перемножает числа своей матрицы и матрицы картинки, далее они суммируются в одно число, процесс итеративно продолжается, и тем самым получается новая матрица изображения.

Архитектура **AlexNet** — первая глубокая сверточная нейронная сеть, которая значительно превзошла предыдущие подходы к распознаванию изображений. Предложенная в 2012 году [1], она стала прорывом в задачах классификации изображений, впервые показав возможности глубоких нейронных сетей превзойти классические методы машинного обучения на соревновании ImageNet. AlexNet состоит из 5 сверточных слоёв и 3 полносвязаных слоёв.

VGGNet — продолжение развития сверточных нейросетей, архитектура сети предложенная Оксфордским университетом в 2014 году [2]. Основное различие между AlexNet и VGG заключается в размере фильтров, глубине сети и количестве параметров. AlexNet использует более крупные фильтры, например, 11×11 в первом сверточном слое, тогда как VGG применяет исключительно небольшие 3×3 фильтры, этот приём значительно увеличил глубину сети. Младшая модель VGG 11 имеет 9 сверточных и 2 полносвязаных слоя, и содержит в себе более 130 миллионов параметров, в то время как у AlexNet их всего 60. Это сделало VGGNet тяжеловесными и требовательными к вычислительным ресурсам.

Архитектура **Inception** (GoogLeNet) - её ключевой особенностью являются Inception-блоки, которые параллельно применяют сверточные операции с разными размерами фильтров (1×1, 3×3, 5×5) вместе с операциями max-pooling. Это позволяет сети извлекать признаки на разных уровнях детализации одновременно [3]. Архитектура активно использует сверточные слои 1×1 для уменьшения размерности данных, что помогло снизить вычислительную сложность и количество параметров.

Каждая новая архитектура увеличивалась в размере, и поэтому появилась проблема, что градиенты на последних слоях были совсем маленькими, и это не давало нормально обновить веса модели. **ResNet** (Residual Networks) решила проблему увеличения глубины сетей [4]. Авторы ResNet использовали "остаточные блоки"с соединениями быстрого доступа (skip connections), которые позволяют градиентам эффективно распространяться через многие слои. Входные данные передаются по дополнительному соединению в обход следующих слоев и добавляются к полученному результату, это соединение не добавляет дополнительные параметры в сеть, поэтому ее структура не усложняется.

Появилась необходимость использования нейронных сетей на мобильных устройствах, и в 2017 году исследователями из Google была представлена архитектура **MobileNet** [5]. Основным отличием MobileNet является использование глубоких отделимых сверточных слоев (depthwise separable convolutions), которые разделяют стандартную свертку на две операции - глубинную свертку (depthwise convolution), которая выполняет фильтрацию, применяя один фильтр для каждого входного канала, и поточечную свертку (pointwise convolution) с фильтром 1×1, которая выполняет комбинирование выходных каналов. Этот подход значительно снижает вычислительную сложность и количество параметров по сравнению с обычными сверточными слоями. MobileNetV1 содержит около 4,2 млн параметров и показывает 70,6% top-1 на соревновании ImageNet при значительно меньших вычислительных затратах по сравнению с предыдущими сетями.

Авторы обнаружили, что для достижения оптимальной производительности необходимо сбалансированное масштабирование 3 измерений сети: глубины (количества слоев), ширины (количества каналов), разрешения (размера входного изображения). В отличие от предыдущих подходов, которые обычно масштабировали только одно из этих измерений, авторы предлагают составное масштабирование (compound scaling). Из существующего метода под названием «Neural Architecture Search» [6] для автоматического создания новых сетей и своего собственного метода масштабирования авторы получают новый класс моделей под названием **EfficientNet** [7]. EfficientNet-B0 содержит около 5,3 млн параметров

и достигает точности 77,1% top-1 на ImageNet. Более крупные версии демонстрируют еще более высокую точность при контролируемом увеличении количества параметров: например, EfficientNet-B7 достигает 84,4% с более чем 60 млн параметров, что является State-Of-The-Art разработкой до сих пор.

Обращая внимание на архитектуры, основанные на трансформерах, стоит упомянуть **Vision Transformer** (ViT), предложенный в 2020 году [8], адаптирует архитектуру трансформера для изображений. ViT разбивает изображение на последовательность непересекающихся патчей, преобразует их в эмбеддинги и подает эту последовательность на вход стандартного трансформера. ViT демонстрирует впечатляющие результаты при обучении на больших наборах данных (ViT-L/16 получил 85.30% top-1 точности после предобучения на датасете JFT-300M), но уступает CNN на меньших датасетах (ViT-B/16 показал слабые результаты при обучении только на ImageNet-1k без предварительного обучения на больших наборах данных, 79,9% top-1 точности на ImageNet, что уступало современным CNNs) и требует значительных вычислительных ресурсов для обучения.

Также важно отметить семейство моделей **YOLO** (You Only Look Once), разработанных специально для детекции объектов в изображениях в режиме реального времени [9]. В отличие от обычных CNN, которые часто используются для классификации изображений или извлечения признаков, YOLO объединяет в себе процессы детекции объектов и их классификации, обрабатывая изображение целиком за один проход. Традиционно многие системы сначала выделяют регионы интереса, а затем отдельно классифицируют содержимое этих регионов. В случае YOLO всё это происходит за один шаг - сеть делит изображение на сетку и для каждой ячейки одновременно предсказывает координаты прямоугольника и вероятность того, что в нём находится объект определённого класса. Это объединение позволяет существенно ускорить задачу классификации. **YOLOv11** [10] является одной из последних моделей этой серии, конкурируя с классическими CNNs.

2.1.1. Обоснование выбора архитектур для работы

Для задач класса *Few-Shot Learning* - необходимо выбрать такие архитектуры, которые обеспечат баланс между точности, скорости обучения и возможности обучаться на небольших данных. На основе проведенного анализа современных архитектур, особенно учитывая требование адаптации к малым объемам данных, для экспериментов выбраны две архитектуры:

1. EfficientNet

- Эффективность использования параметров - отличные метрики при относительно небольшом количестве параметров
- Масштабируемость - семейство моделей от B0 до B7 позволяет выбрать оптимальный баланс между точностью и сложностью
- Меньшая требовательность к вычислительным ресурсам - можно эффективно использовать облачную инфраструктуру Yandex Cloud и быстрый локальный инференс
- Простая поддержка и создание пайплайна для обучения, так как EfficientNet уже есть в базовых классах PyTorch.

2. YOLO

- Универсальность - может использоваться как для классификации, так и для обнаружения объектов. (Удобно для развертывания в дальнейших задач проекта)
- Высокая скорость работы - однопроходная архитектура обеспечивает быстрый инференс, но и обучение.
- Точность — последние версии (YOLOv11) показывают конкурентные результаты по сравнению с другими современными моделями.
- Масштабируемость — доступны варианты разного размера (nano, small, medium, large, xlarge)

2.2. Аугментации данных и реализация процесса обучения моделей

Таблица 1: Сравнение характеристик выбранных архитектур для классификации (тестирование на ImageNet)

Модель	Параметры (млн)	Точность Top-1 (%)	Точность Top-5 (%)	FLOPs (B)
EfficientNet-B0	5.3	77.1	93.3	0.39
EfficientNet-B7	66.0	84.3	97.0	37.0
YOLOv11n-cls	1.6	70.0	89.4	0.5
YOLOv11s-cls	5.5	75.4	92.7	1.6

Примечание: Тестирование проводилось на наборе данных ImageNet [7] и [10].
Точность Top-1 (Top-1 Accuracy) — это процент правильно классифицированных изображений, где предсказанный класс с самой высокой вероятностью совпадает с истинным классом изображения.
Точность Top-5 (Top-5 Accuracy) — это процент правильно классифицированных изображений, где истинный класс изображения входит в пять наиболее вероятных классов, предсказанных моделью.
FLOPs (Floating Point Operations per Second) - метрика, показывающая количество арифметических операций с плавающей запятой, необходимых для выполнения одного прохода (инференса) модели.

Столь большая модель EfficentNet-B7 была взята лишь для исследования - "Насколько можно увеличить метрики на малом обьеме данных увеличив лишь кол-во обучаемых параметров?"

Так как важным фактором проекта является лучшая интерпретируемость результатов и предсказуемость поведения на небольших датасетах, то для этого помогут 2 важные стратегии - Аугментации данных и Трансферное обучение.

Аугментации данных (Data augmentation) - процесс синтетического «раздутия» тренировочных данных по специальному алгоритму для увеличения объема выборки. Аугментации направлены на увеличение разнообразия обучающих данных, что помогает модели лучше обобщать и справляться с новыми, ранее не виденными обьектами.

Трансферное обучение (Transfer Learning) - это методика машинного обучения, с помощью которой можно применять накопленные знания, полученные из одной задачи, к другой такой же задаче этого класса.

2.2.1. Аугментации данных для EfficentNet

Для моделей семейства EfficentNet была выбрана следующая стратегия аугментации данных:

1. **Изменение размера изображений (Resize):** Все изображения были приведены к размеру 224x224 (или 600x600 для B7) пикселя. Это необходимо, чтобы обеспечить совместимость входных данных с архитектурой модели.

2. **Случайные горизонтальные отражения (RandomHorizontalFlip):** Данная аугментация случайным образом отражает изображения по горизонтали, имитируя изменение точки обзора объекта. Это помогает модели стать более устойчивой к зеркальным отражениям.

3. **Случайные повороты (RandomRotation):** Изображения поворачиваются на случайный угол в диапазоне ± 15 градусов. Повышает способность модели игнорировать небольшие изменения ориентации объекта.

4. **Изменение цветовой гаммы (ColorJitter):** Яркость, контрастность и насыщенность изображений случайным образом изменяются. Обучает модель быть менее чувствительной к изменениям освещения и цветовых характеристик.

5. **Нормализация (Normalize):** Изображения нормализуются с использованием средних и стандартных отклонений, полученных из ImageNet. Значительное улучшение процесса обучения, приводя значения пикселей к диапазону, более удобному для нейросети.[7]

6. **Искусственное увеличение тренировчного датасета (Target Size):** Если пользователь в сумме по классу загружает < 50 сэмплов изображений, то тренировочный датасет увеличивается дубликатами до необходимого минимума.

Все указанные аугментации применялись только к обучающей выборке. Тестовая выборка использовалась для оценки производительности модели без дополнительных преобразований, для честной оценки обобщающей способности.

2.2.2. Пайплайн обучения (EfficentNet)

Примененные техники в процессе обучения:

1. **Загрузка предобученной модели:** EfficientNet B0 загружается с предобученными весами, полученными на датасете ImageNet, позволяя модели быстро освоить общие признаки изображений, что особенно важно при обучении на небольших датасетах.

2. **Заморозка слоев (RandomHorizontalFlip):** Чтобы предотвратить переобучение, все слои модели, кроме классификатора, замораживаются. Это означает, что их веса не будут изменяться в процессе обучения, для сохранения информации, полученной на предтрейне ImageNet. [11]

3. **Замена классификатора (Fine-Tuning):** Классификатор (последний полносвязный слой) заменяется на новый, с количеством выходных нейронов, соответствующим количеству классов в нашем датасете. Это тот самый слой, который будет обучаться.

4. **Оптимизаторы и настройка гиперпараметров:** Для оптимизации используется алгоритм Adam, который хорошо подходит для обучения глубоких нейросетей. В качестве функции потерь используется CrossEntropyLoss, стандартная функция для задач многоклассовой классификации. Значения learning rate - *0.0001* (скорость обучения), weight decay - *0.0001* (коэффициент L2 регуляризации), Batch size *(8)* и количество эпох *(10)* были выбраны на основе эмпирических наблюдений

5. **Обучение:** Модель обучается на аугментированных данных в течение заданного количества эпох. На каждой эпохе вычисляется функция потерь на обучающей выборке, и веса классификатора обновляются с использованием обратного распространения ошибки. Для контроля переобучения производилась валидация модели на тестовой выборке после каждой эпохи.

2.2.3. Аугментации данных и процесс обучения YOLOv11

Несмотря на потенциальные преимущества использования аугментации для повышения обобщающей способности, в ходе экспериментов с YOLOv11 было решено отказаться от их применения на данном этапе исследования. Это связано с несколькими причинами:

- **Сложность интеграции:** Ultralytics YOLOv11 использует собственную систему обучения и аугментации данных. Интеграция кастомных аугментаций потребовала бы значительного изменения кода, что выходило за рамки задач данного этапа проекта.
- **Необходимость адаптации параметров:** Аугментации, которые хорошо работают для CNN, могут быть неэффективны или даже вредны для YOLO, ведь изначально модель была разработана для детекции обьектов. Требуется тщательная настройка параметров аугментаций, чтобы избежать ухудшения производительности.

Также отметим, что в пайплайне обучения использовался стандартный *trainer* на 10 эпохах, с аналогичным EfficientNet *resize* изображений.

2.3. Тестирование и сравнение выбранных архитектур на различных данных

Наша цель - эмпирически проверить и сравнить производительность выбранных архитектур (EfficientNet B0, B7 и YOLOv11 n-cls, s-cls) в условиях, релевантных для задачи сервиса, а именно: на данных разного объема и новизны (известные/-неизвестные классы), чтобы окончательно подтвердить выбор основной модели.

2.3.1. Описание тестовых сценариев и метрик

Таблица 2: Описание наборов данных, использованных для тестирования моделей

Название датасета	Кол-во классов	Исходных изобр. (тренировка, на класс)	Тип классов (относит. ImageNet)
CatsDogs-50	2	50	Известные (Cat, Dog)
CatsDogs-15	2	15	Известные (Cat, Dog)
Celeb-15	2	15	Неизвестные (Celebrity Faces)

Примечание: "Известные"классы присутствовали в датасете ImageNet, на котором предобучались модели, в то время как "неизвестные"классы (лица знаменитостей) отсутствовали в ImageNet. Количество изображений указано до применения аугментаций (включая дублирование до `target_size=50` для EfficientNet). Ссылка для просмотра датасетов: Google Drive

Метрики оценки: Для количественной оценки моделей EfficientNet использовались стандартные метрики классификации, основанные на значениях True Positives (TP), True Negatives (TN), False Positives (FP) и False Negatives (FN) для каждого класса:

- **TP (True Positives):** Количество правильно классифицированных объектов данного класса.
- **TN (True Negatives):** Количество объектов других классов, правильно классифицированных как не относящиеся к данному классу.
- **FP (False Positives):** Количество объектов других классов, ошибочно классифицированных как данный класс (ошибка I рода).
- **FN (False Negatives):** Количество объектов данного класса, ошибочно классифицированных как объекты других классов (ошибка II рода).

На основе этих значений рассчитывались:

- **Accuracy (Точность):** Общая доля правильно классифицированных объектов.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$
- **Precision (Точность предсказания):** Доля объектов, действительно относящихся к классу, среди всех объектов, которые модель отнесла к этому классу.
$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

- **Recall (Полнота):** Доля объектов данного класса, которые модель смогла правильно обнаружить.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

- **F1-score (F-мера):** Среднее гармоническое Precision и Recall, используется как обобщенная метрика качества, особенно полезная при несбалансированных классах.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{4}$$

Матрица ошибок (Confusion Matrix): Для визуального анализа ошибок классификации по всем моделям (включая YOLOv11) использовалась матрица ошибок. Это таблица, где строки соответствуют истинным классам объектов, а столбцы — предсказанным моделью классам. Элементы на главной диагонали показывают количество правильно классифицированных объектов для каждого класса. Внедиагональные элементы показывают количество ошибок, то есть объекты одного класса, которые были ошибочно отнесены к другому классу. Нормализованная матрица ошибок показывает долю объектов (от 0 до 1) для каждой ячейки, что облегчает сравнение производительности по разным классам.

Нюанс с метриками YOLOv11: Стандартный модуль обучения классификации в библиотеке Ultralytics на момент проведения экспериментов не предоставлял прямого доступа к расчету метрик Precision, Recall и F1-score на валидационной выборке в том же формате, что и для EfficientNet. Поэтому для сравнения моделей YOLOv11 между собой и с EfficientNet использовался **визуальный анализ нормализованных матриц ошибок**, генерируемый библиотекой Ultralytics по окончании обучения.

Гиперпараметры обучения: Основные гиперпараметры, использованные при обучении моделей, были заданы в конфигурационном файле.

```
{
  "num_epochs": 10,
  "learning_rate": 0.0001,
  "batch_size": 8,
  "weight_decay": 0.0001
}
```

Листинг 1: Содержимое файла `hyperparams.json`

2.3.2. Эксперименты с EfficientNet B0

Для начала посмотрим на результаты обучения на датасете CatsDogs-50: Основные показатели на 10-й эпохе обучения:

- Train Loss: 0.3249, Train Accuracy: 0.9300
- Validation Loss: 0.3550, Validation Accuracy: 0.9150
- Validation F1-score (weighted): 0.9149
- Validation Precision (weighted): 0.9175
- Validation Recall (weighted): 0.9150

Затраченные ресурсы на обучение составили:

- Время обучения: ≈ 72.6 секунд (≈ 1.21 минуты).
- Пиковое потребление GPU памяти для обучения: ≈ 0.145 ГБ.

Модель показывает отличные результаты на данном датасете, с очень оптимальным использованием памяти, проведем сравнение с уменьшенной версией CatsDogs-15:

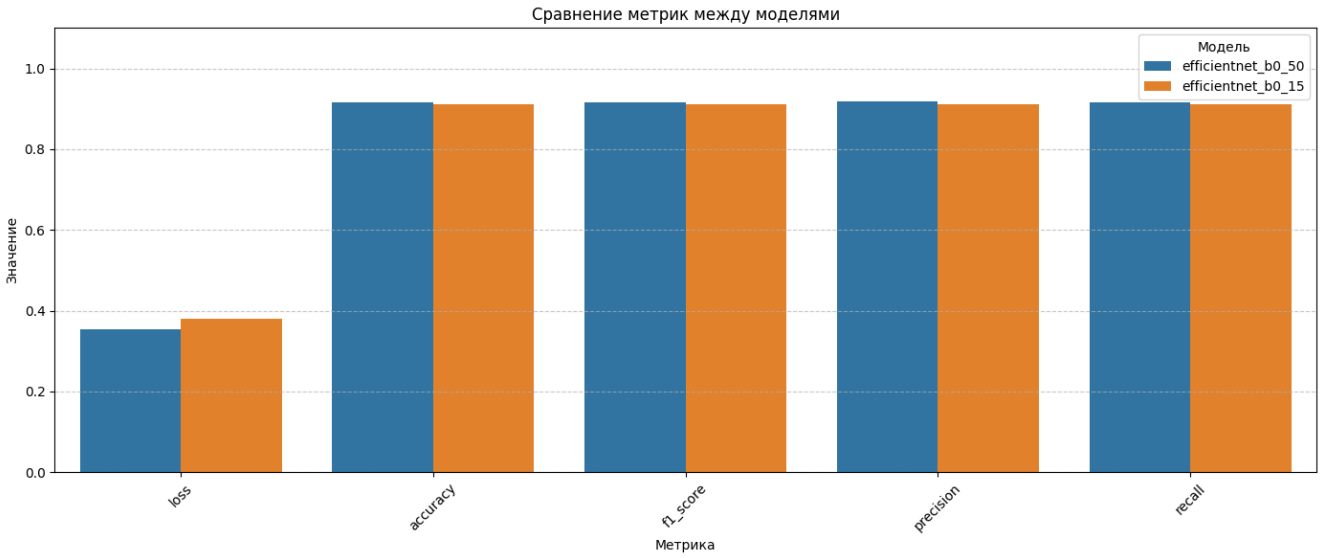
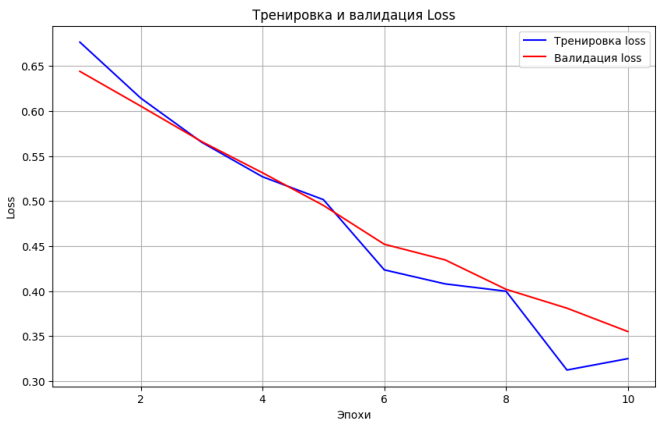
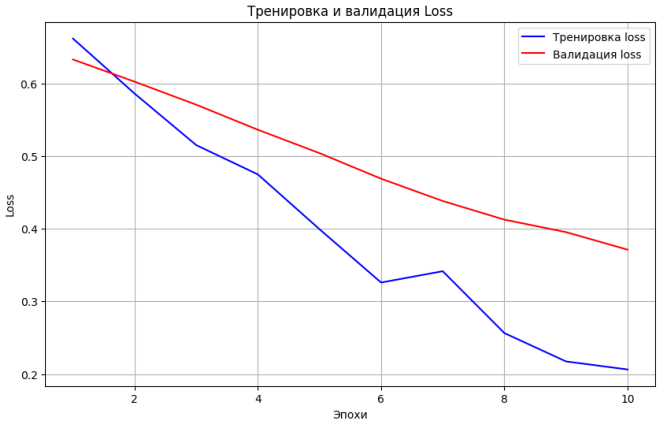


Рис. 1: Метрики efficentnet_b0_50 vs efficentnet_b0_15

Заметим, что метрики идентичны. Сравним графики лоссов:



(а) CatsDogs-50



(б) CatsDogs-15

Рис. 2: Сравнение графиков потерь (Loss) для EfficientNet B0 на обучающей (синий) и валидационной (красный) выборках при разном количестве исходных образцов.

В целом тенденция лоссов также схожа, единственное наблюдение – больший разрыв между тренировочной и валидационной выборкой, что, возможно, предшествует переобучению.

Теперь посмотрим на те же метрики и график лосса уже на другом тренировочном датасете Celeb-15:

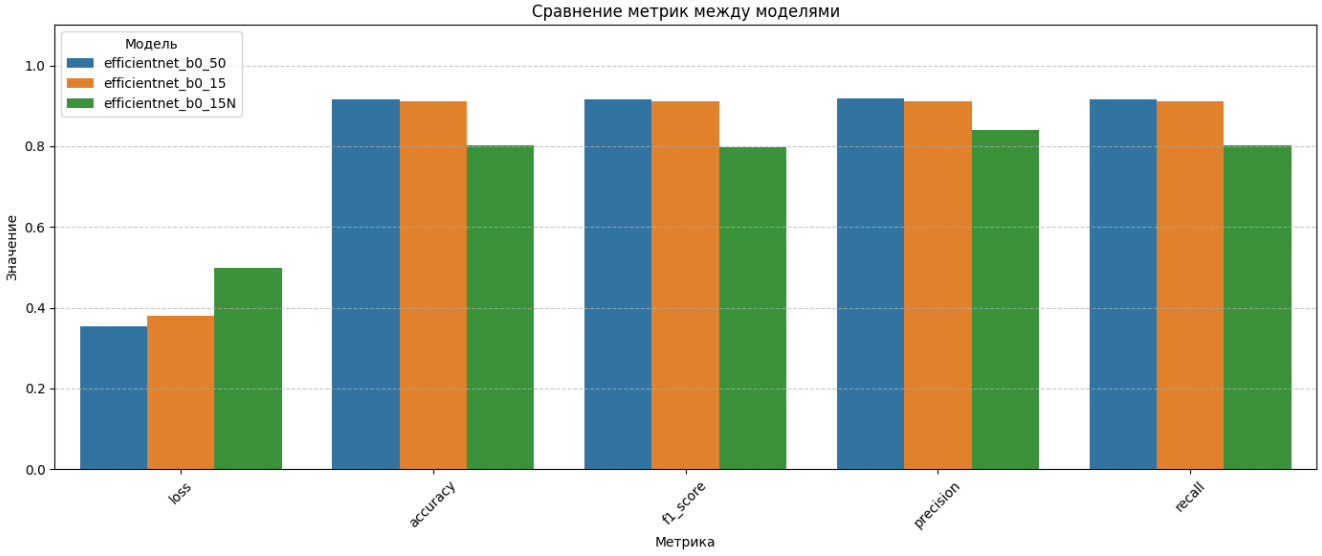


Рис. 3: Метрики efficentnet_b0_50 vs efficentnet_b0_15 vs efficentnet_b0_15N

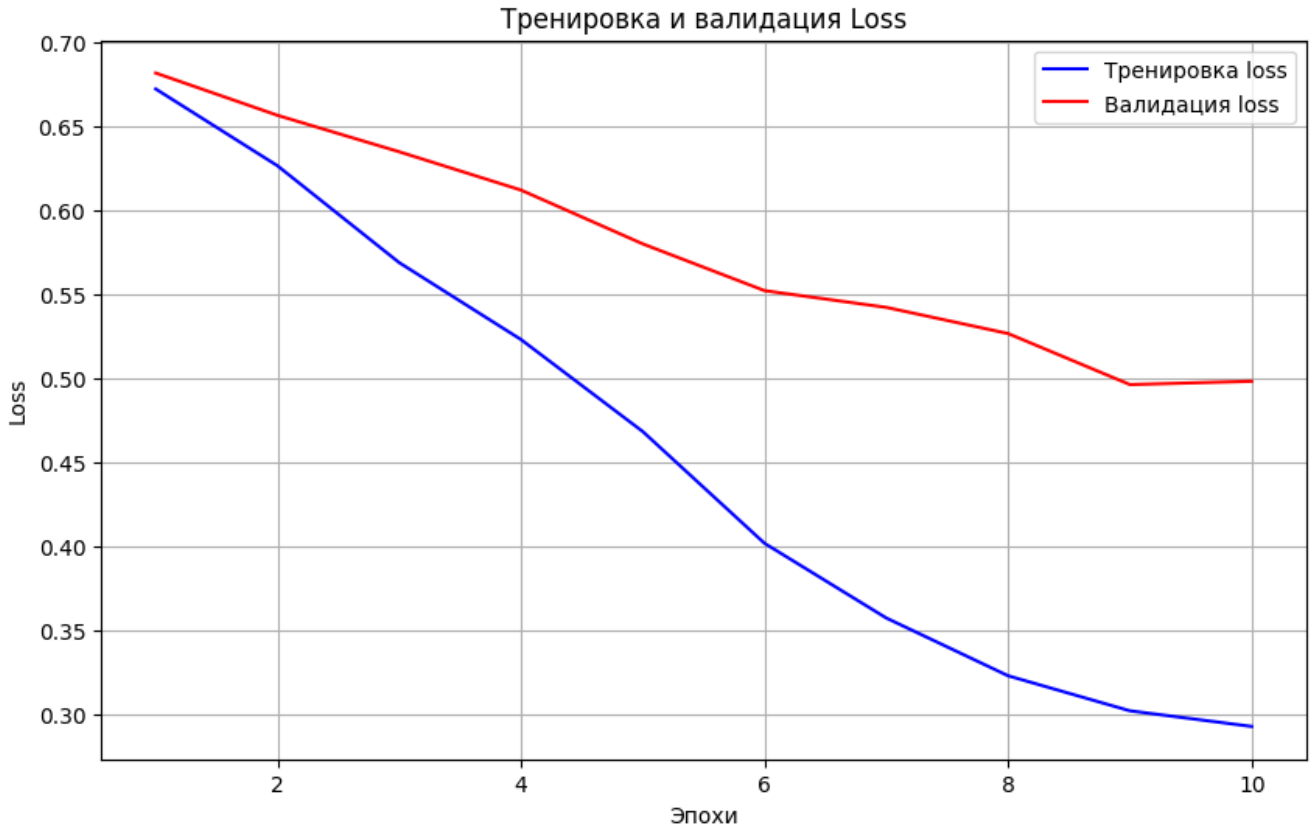


Рис. 4: Loss efficentnet_b0_15N

В среднем качество инференса упало на 10%, однако показатель Accuracy $\approx$ 0.8 остаётся отличным результатом на столь малом тренировочном датасете. По графику лосса также можно сделать вывод, что модель на 10-й эпохе была максимально близка к моменту переобучения, так как лосс на валидации перестал снижаться, и разрыв между лоссом при обучении стал максимальным.

2.3.3. Эксперименты с EfficientNet B7

Таблица 3: Сравнение вычислительных ресурсов при обучении EfficientNet B0 и B7 (датасет CatsDogs-50, 10 эпох)

Модель	Время обучения (минуты)	Пиковое потребление GPU памяти (ГБ)
EfficientNet-B0	1.21	0.145
EfficientNet-B7	14.41	2.096

Примечание: Данные измерены на Google Colab с GPU Tesla T4. Время обучения и потребление памяти могут варьироваться в зависимости от аппаратного обеспечения, размера батча и других факторов.

Время обучения и потребление видеопамяти $\geq$ в 10 раз больше, чем у младшей модели, посмотрим, смогли ли мы улучшить наши метрики, потратив столько ресурсов:

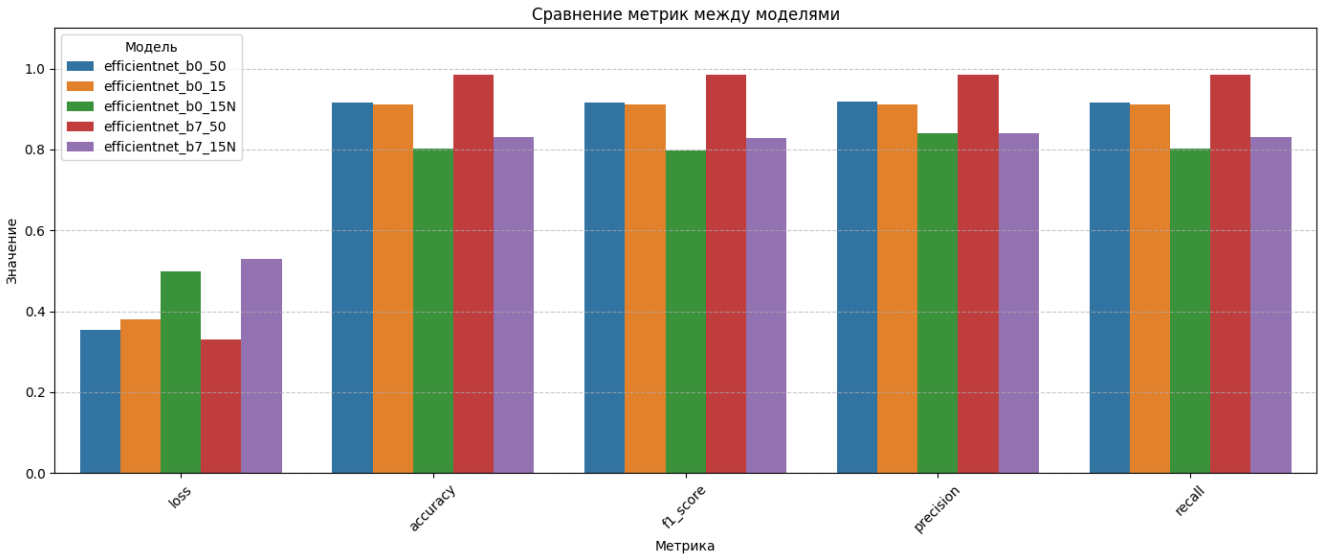


Рис. 5: Метрики моделей семейства EfficentNet

Примечание: В ходе тестирования было выявлено, что 10 эпох для обучения модели B7 недостаточно для нахождения экстремума функции потерь, Val Loss $\approx$ 0.5305, после дополнительных 10 эпох, Val Loss $\approx$ 0.3699, и метрики качества незначительно улучшились. Подробнее, см. ноутбук Google Drive

По диаграмме видно, что в эксперименте с датасетом CatsDogs-50 EfficientNet B7 показывает способность практически с 100% вероятностью предсказывать правильный класс (ведь изображений этих классов было гораздо больше на момент предтрейна на ImageNet), однако на более реальном пользовательском случае, когда классы неизвестны (Celeb-50), метрики лишь на 5% выше младшей модели. Это явно говорит, что количество параметров у модели B0 достаточно для запоминания основных признаков на столь малой выборке данных.

Рассмотрим также график лосса при обучении на датасете CatsDogs-50:

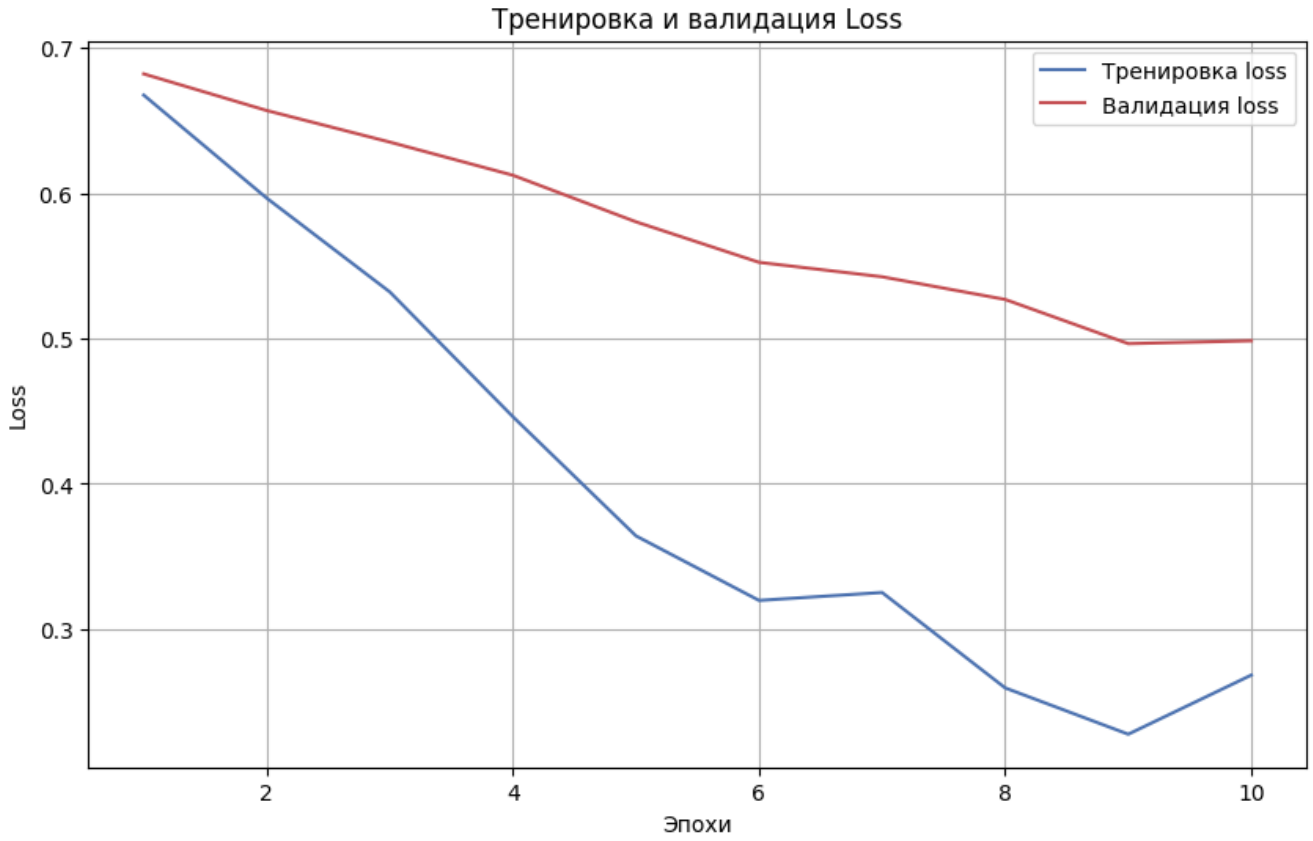
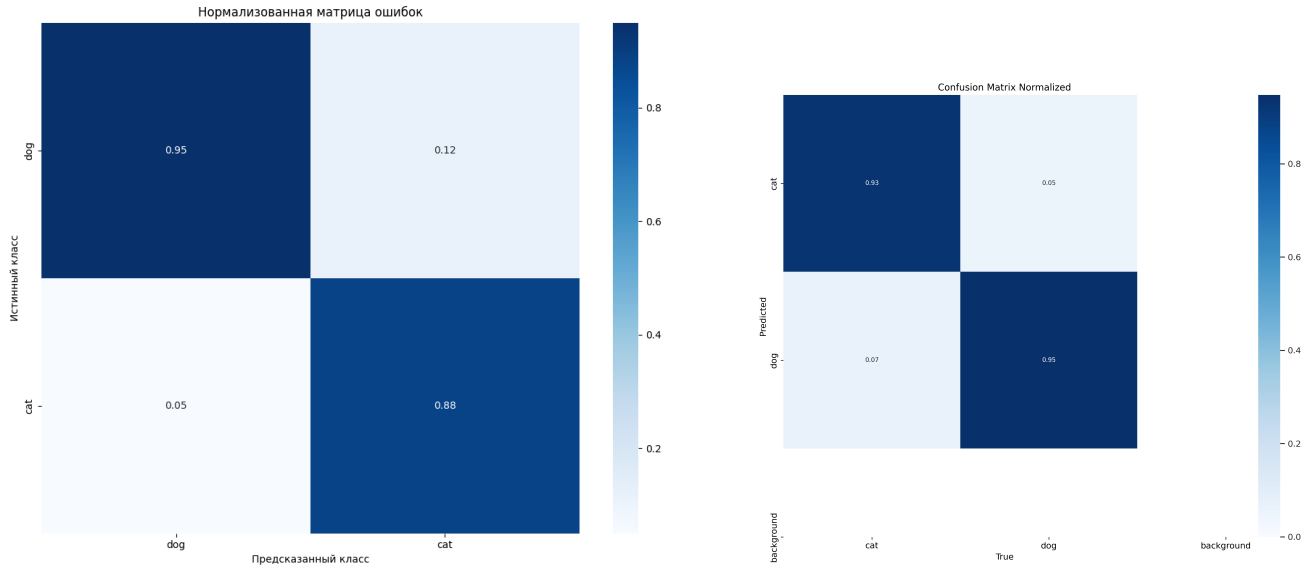


Рис. 6: Loss efficentnet_b7_50

Это подтверждает нашу гипотезу о том, что EfficientNet B7 практически не обучается на знакомых классах, так как падение Val Loss совсем незначительное, при том что метрики около 1.

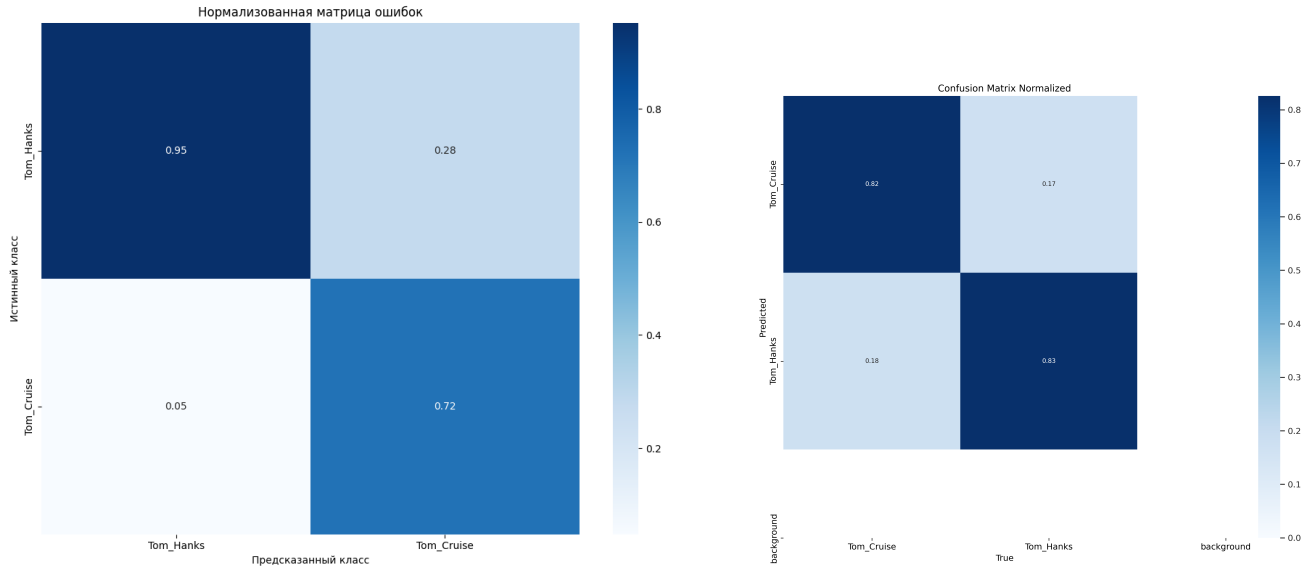
2.3.4. Эксперименты с YOLOv11

Будем сравнивать Confusion Matrix наших моделей, из-за технических ограничений в библиотеке Ultralytics. Рассмотрим самую малую модель - *yolo11n-cls* на датасете CatsDogs-50 без использования Transfer Learning:



(а) EfficientNet B0 (б) yolo11n-cls

Рис. 7: Сравнение Confusion Matrix для EfficientNet B0 и yolo11n-cls (Full Train) на датасете CatsDogs-50.



(а) EfficientNet B0 (б) yolo11n-cls

Рис. 8: Сравнение Confusion Matrix для EfficientNet B0 и yolo11n-cls (Full Train) на датасете Celeb-15.

Отметим, что EfficientNet B0 плохо справляется с детекцией одного из классов, в то время как YOLO показывает более ожидаемые результаты на этих тестах, однако в целом результаты сопоставимы. Рассмотрим также потраченные ресурсы при обучении *yolo11n-cls* на датасете CatsDogs-50(Full Train):

- Время обучения: ≈ 83.1 секунд (≈ 1.39 минуты).

- Пиковое потребление GPU памяти для обучения: ≈ 0.133 ГБ.

YOLO очень эффективно справляется с обучением всех весов модели, попробуем использовать Transer Learning *(Подробности см. в ноутбук)*:

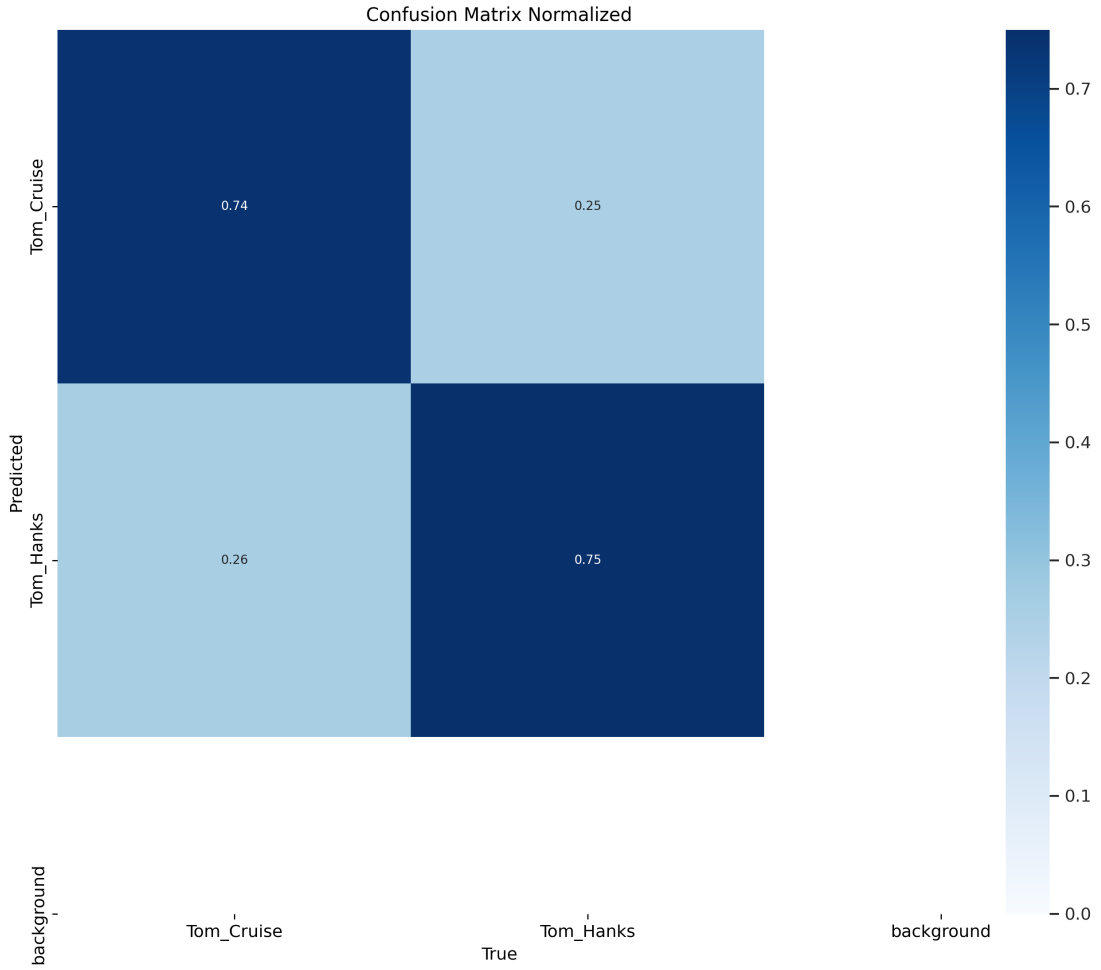
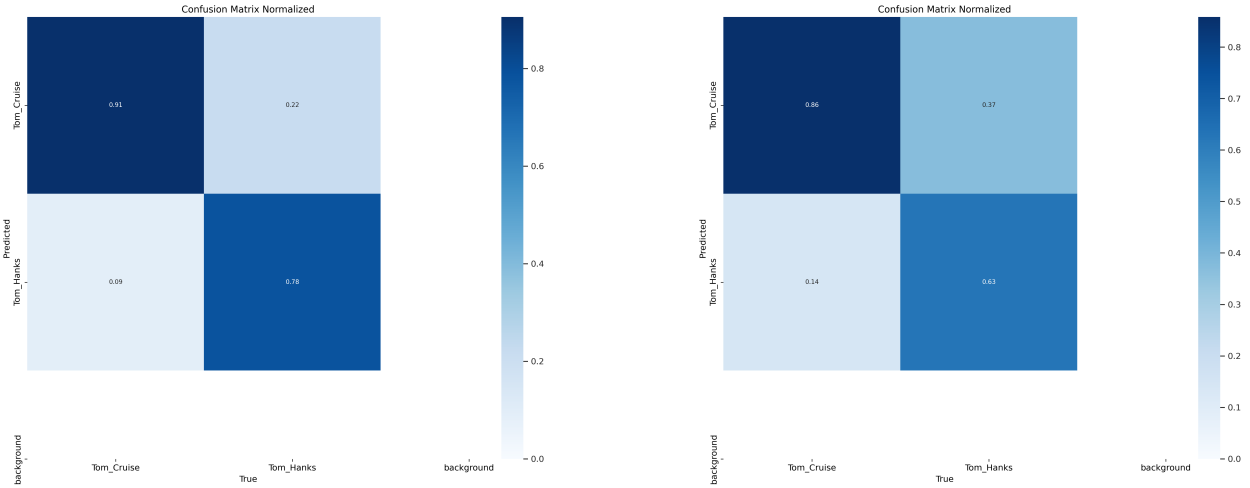


Рис. 9: Confusion Matrix для yolo11n-cls (Transfer Learning) на датасете Celeb-15

Результаты упали более чем на 10%, из-за сложностей архитектуры YOLO, Transfer Learning лишь помогает модели забыть уже полученные знания на этапе тренировки на ImageNet. Для чистоты эксперимента, посмотрим на результаты обучения старшей модели *yolo11s-cls* на датасете Celeb-15:



(а) Full Train (б) Transfer Learning

Рис. 10: Сравнение Confusion Matrix для yolo11s-cls с разными методиками обучения на датасете Celeb-15.

2.3.5. Сводное сравнение и выводы по тестированию

- **EfficientNet B0** показала стабильно хорошие результаты, особенно на малых и неизвестных данных, при низких затратах ресурсов. *Fine-tuning* работает эффективно.
- **EfficientNet B7** дает лучшие результаты на более объемных и известных выборках, но требует значительно больше ресурсов и не дает преимуществ на очень малых выборках.
- **YOLOv11n** показала себя конкурентоспособной (без заморозки), но стратегия заморозки слоев оказалась неудачной. Результаты YOLO были менее предсказуемыми. **YOLOv11s** в данных условиях показала себя хуже младшей модели.

На основе проведенных тестов, учитывая требования сервиса, выбор ***EfficientNet B0*** с применением *Fine-Tuning* и аугментаций данных подтверждается как наиболее сбалансированное решение.

3. Разработка пайплайна обучения нейросетей для задачи классификации звуков

3.1. Теоретические основы и архитектуры нейросетей для классификации звуков

Классификация звуков (Audio Classification) – это задача машинного обучения, направленная на категоризацию аудиозаписей по их содержимому. В отличие от классификации изображений, где входными данными являются статические многомерные массивы пикселей, звуковые сигналы имеют временную природу и могут быть представлены как одномерные временные ряды. Эффективное извлечение признаков из таких сигналов является ключевым аспектом задачи. Современные подходы к классификации звуков часто преобразуют временные ряды в двумерные спектральные представления (например, спектрограммы или их улучшенная версия - мел-спектрограммы), которые затем могут быть обработаны с использованием техник, аналогичных тем, что применяются в компьютерном зрении.

Спектральные представления, такие как **мел-спектрограммы**, отображают частотное содержимое звука во времени, делая видимыми паттерны, связанные с различными типами звуков (например, речь, музыка, звуки окружающей среды). Это дает возможность использовать мощные архитектуры, изначально разработанные для изображений, такие как сверточные нейронные сети (CNN), для анализа аудиоданных.

Однако, помимо адаптации моделей компьютерного зрения, активно развиваются архитектуры, специфичные для аудиозадач, а также подходы, основанные на извлечении предобученных эмбеддингов.

Для задачи классификации звуков существует множество подходов, которые можно разделить на несколько категорий:

1. **Подходы, основанные на спектральных представлениях + CNN:** Этот классический подход включает преобразование аудиосигнала в двумерное представление (например, мел-спектрограмму) и последующую подачу этого представления на вход стандартной CNN, такой как ResNet, VGG или EfficientNet. Предобученные на больших наборах изображений (например, ImageNet) CNN могут быть тонко настроены (fine-tuned) для аудиозадач, используя спектрограммы как "изображения". Это подход хорошо зарекомендовал себя благодаря способности CNN извлекать иерар-

хические пространственные признаки, которые имеют аналоги в частотно-временных паттернах спектрограмм.

2. **Подходы, основанные на предобученных аудио-эмбеддингах + простой классификатор:** Идея этого подхода заключается в использовании мощной, предварительно обученной на большой и разнообразной коллекции аудиоданных модели (например, AudioSet), которая способна извлекать высокоуровневые, семантически насыщенные признаки или эмбеддинги из аудио. Одним из ярких представителей таких моделей являются **PANNs** (Pre-trained Audio Neural Networks) [12]. PANNs, основанные на различных архитектурах CNN (ResNet, MobileNet), обучаются на задачах, требующих понимания содержимого аудио (Audio Tagging). Извлеченные из этих моделей эмбеддинги затем используются как вход для простого классификатора – многослойного перцептрона (MLP), который обучается решать специфическую целевую задачу классификации. Этот подход выгоден, когда объем данных для целевой задачи мал, так как большая часть "знаний"о звуках уже закодирована в предобученной модели.

3. **Подходы, основанные на трансформерах:** Архитектуры трансформеров, изначально показавшие выдающиеся результаты в обработке естественного языка, были успешно адаптированы для аудио. **Audio Spectrogram Transformer** (AST) [13] – это пример такой адаптации. AST обрабатывает мел-спектрограммы как последовательности "патчей"(подобно Vision Transformer) и применяет механизм самовнимания для улавливания дальних зависимостей во временной и частотной областях. Модели AST, предобученные на масштабных аудиодатасетах (AudioSet, SpeechCommands), демонстрируют передовые результаты в различных аудиозадачах, включая классификацию звуков.

3.1.1. Обоснование выбора архитектур для работы

Исходя из условий, в которых будет использоваться модель *(Few-Shot Learning)*, для эмпирического сравнения и выбора оптимального пайплайна были выбраны три метода:

1. **CNN на мел-спектрограммах с использованием ResNet18:** является легковесной, но достаточно мощной CNN. Ее предобучение на ImageNet и адаптация для спектрограмм (изменение первого сверточного слоя для

одного входного канала) – это стандартный и эффективный метод для задач, где данные могут быть представлены в виде изображений. Этот подход позволяет оценить базовую эффективность CNN на спектральных аудиопредставлениях при тонкой настройке на малом датасете.

2. **PANNs-эмбеддинги + MLP:** подход представляет категорию методов, основанных на использовании предобученных аудио-специфичных признаков. PANNs являются одними из лучших предобученных моделей для извлечения общих аудио признаков. Обучение простого MLP на этих эмбеддингах очень быстрое и требует минимального объема целевых данных для обучения самого классификатора.

3. **Audio Spectrogram Transformer (AST):** это представитель современных архитектур на базе трансформера, разработанных специально для аудио. Fine-tuning предобученной AST модели позволяет потенциально найти более сложные зависимости в данных. Этот подход выбран для оценки пиковой производительности, которую можно достичь с использованием State-Of-The-Art архитектур, и сравнения ее с более традиционными методами.

3.2. Подготовка данных и методы обучения для классификации звуков

Для проведения экспериментов был использован набор данных **ESC-50** [14], который содержит 2000 звуков окружающей среды из 50 категорий (по 40 записей на каждую категорию). Аудиозаписи имеют длительность 5 секунд.

3.2.1. Подготовка данных

Поскольку сервис ориентирован на небольшие пользовательские датасеты, из полного набора ESC-50 были выбраны небольшие подмножества данных для обучения, валидации и тестирования. Были использованы две пары классов для сравнения:

- ***dog* и *cat*:** Классы, которые присутствовали в больших предобучающих наборах данных (ImageNet и AudioSet), это позволяет нам оценить эффективность на знакомых данных.

- ***vacuum cleaner* и *airplane*:** Классы, которые представлены как более сложные/шумные аудио характеристики, приведены в роли реалистичного сценария пользовательских данных.

Для каждого класса в выбранной паре, аудиозаписи были разделены следующим образом:

- **Обучение (Train):** $NUM\_TRAIN = 15$ исходных записей на класс.
- **Валидация (Validation):** $NUM\_VAL = 5$ записей на класс.
- **Тестирование (Test):** Оставшиеся записи на класс (20 штук).

Подход гарантирует, что валидационный и тестовый наборы содержат полностью невидимые для модели в процессе обучения записи.

Все аудиозаписи были приведены к единой частоте дискретизации $SR = 16000$ Гц и фиксированной длительности $DURATION = 5$ секунд ($SAMPLES = SR \times DURATION = 80000$ сэмплов). Записи длиннее 5 секунд обрезались (в рамках дальнейшей имплементации в сервис), более короткие — дополнялись нулями до необходимой длины. Многоканальные записи преобразовывались в моно путём усреднения по каналам.

3.2.2. Аугментации данных

Для увеличения разнообразия обучающей выборки и улучшения обобщающей способности моделей применялись различные техники аугментации. Важно отметить, что набор аугментаций варьировался в зависимости от входного представления, используемого конкретным подходом:

1. **Аугментации для подходов на основе спектрограмм (ResNet18):** Для моделей, работающих с мел-спектрограммами, применялись как волновые (на сыром аудио), так и спектральные аугментации:
    - *Волновые аугментации:* Применялись к аудиосигналу до преобразования в спектрограмму. Это такие техники как (RandomVolume) и добавление Гауссовского шума (GaussianNoise). Цель — сделать модель устойчивой к изменениям уровня громкости и фоновому шуму.

- *Спектральные аугментации:* Применялись непосредственно к Мел-спектрограмме - маскирование по времени (TimeMasking) и маскирование по частоте (FrequencyMasking), эти методики помогают модели фокусироваться на различных участках спектрограммы и повышает устойчивость к пропускам данных или отдельным шумовым компонентам.

Тренировочный датасет дополнительно расширялся дублированием и аугментацией исходных 15 записей на класс до $TARGET\_SIZE = 50$ примеров на класс, чтобы предоставить модели больше вариативности. [1]

2. **Аугментации для подхода на основе PANNs-эмбеддингов:** Поскольку PANNs работают с сырым аудиосигналом для извлечения эмбеддингов, здесь применялись только *волновые аугментации* (RandomVolume, GaussianNoise) перед подачей аудио в PANNs модель. Использование спектральных аугментаций не имело смысла, так как PANNs обрабатывают аудио на более низком уровне.

3. **Аугментации для подхода на основе AST:** AST, являясь специализированной трансформерной моделью для аудио, часто использует свои собственные рекомендованные аугментации. В данном случае применялись аугментации из библиотеки `audiomentations` к сырому аудио перед его обработкой feature extractor'ом AST модели - *изменение скорости воспроизведения (TimeStretch), сдвиг тональности (PitchShift) и добавление Гауссовского шума (AddGaussianNoise)*. Эти аугментации направлены на имитацию вариаций в воспроизведении звука и условий записи.

[1] Аналогичная техника применена ко всем подходам

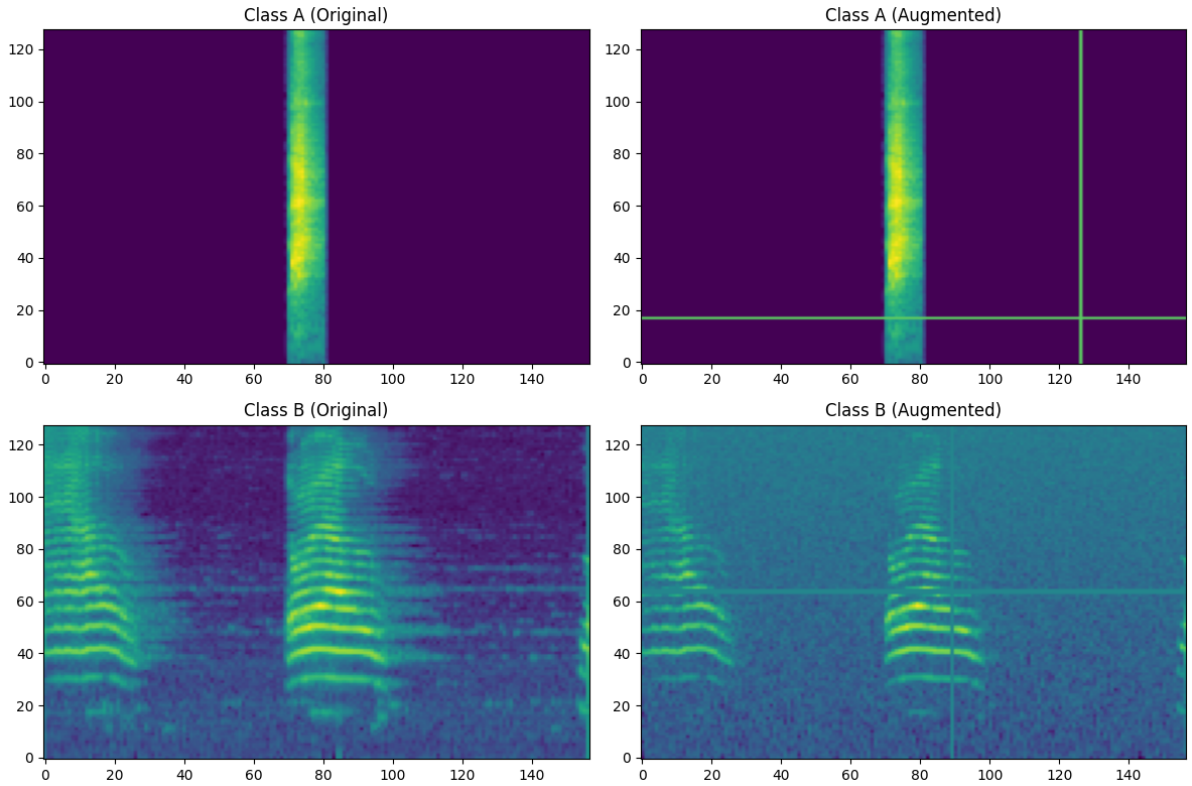


Рис. 11: Сравнение мел-спектрограмм для двух классов до и после применения аугментаций. Слева: оригинальные спектрограммы для класса A ('dog') и класса B ('cat'). Справа: те же спектрограммы после применения случайных волновых и спектральных аугментаций (TimeMasking, FrequencyMasking, RandomVolume, GaussianNoise).

Для валидационных и тестовых наборов аугментации не применялись, использовались только стандартные шаги подготовки аудио (ресэмплинг, обрезка/дополнение).

3.2.3. Методы обучения

Каждый из трех подходов требовал специфической настройки процесса обучения:

1. **CNN на мел-спектрограммах (ResNet18):** Использовалась предобученная на ImageNet модель ResNet18. Первый сверточный слой был модифицирован для приема одноканальной мел-спектрограммы (размер ядра, stride, padding оставались прежними, менялось только количество входных каналов с 3 на 1). Финальный полносвязный слой (классификатор) заменялся для соответствия количеству целевых классов. Обучение проводилось методом *Fine-Tuning*, при этом веса всех слоев, кроме модифи-

цированного первого сверточного и финального полносвязного, были заморожены (не обновлялись в процессе обучения). Использовался оптимизатор Adam и функция потерь CrossEntropyLoss с гиперпараметрами из `hyperparams.json` (см. выше).

2. **PANNs-эмбеддинги + MLP:** В этом подходе модель PANNs (CNN14) использовалась только для извлечения эмбеддингов. Аудиозаписи из тренировочной, валидационной и тестовой выборки (после волновых аугментаций для тренировки) пропускались через PANNs модель в режиме инференса для получения фиксированных эмбеддингов для каждого аудио. Затем обучался отдельный, относительно простой многослойный перцептрон (MLP) на этих предобученных эмбеддингах. MLP состоял из нескольких полносвязных слоев с активациями ReLU и Dropout для регуляризации, завершаясь финальным полносвязным слоем с числом выходов, равным количеству классов. MLP обучался с использованием оптимизатора Adam и функции потерь CrossEntropyLoss с гиперпараметрами из `hyperparams.json`.

3. **Audio Spectrogram Transformer (AST):** Использовалась предобученная на AudioSet модель AST (*MIT/ast-finetuned-audioset-10-10-0.4593*) из библиотеки Hugging Face Transformers [15]. Эта модель уже включает в себя feature extractor и классификационную голову. Для задачи тонкой настройки, количество выходных лейблов в классификационной голове модели изменялось для соответствия количеству целевых классов. Обучение проводилось методом *Fine-Tuning* всей модели (все веса обновлялись). Использовался оптимизатор AdamW (часто рекомендуемый для трансформерных моделей) и функция потерь CrossEntropyLoss. Гиперпараметры (LR, epochs) также соответствовали `hyperparams.json` где применимо, или использовались стандартные параметры для fine-tuning AST моделей.

Каждая модель обучалась в течение заданного количества эпох ($EPOCHS = 10$) на обучающей выборке, с регулярной оценкой производительности на валидационной выборке для мониторинга прогресса и выявления переобучения. Оценка на тестовой выборке проводилась один раз по окончании обучения.

3.3. Тестирование и сравнение выбранных подходов

3.3.1. Эксперименты с ResNet18 на спектрограммах

Первый подход основан на использовании предобученной модели ResNet18, адаптированной для классификации мел-спектрограмм. Модель обучалась с применением Fine-Tuning и комплекса волновых и спектральных аугментаций. Эксперименты проводились на двух парах классов из датасета ESC-50.

Результаты на классах *dog* и *cat*: На первой паре классов, которые хорошо представлены в предобучающем наборе (ImageNet), модель ResNet18 показала высокие результаты на тестовой выборке:

- Accuracy: 0.9500
- Precision (weighted): 0.9500
- Recall (weighted): 0.9500
- F1-score (weighted): 0.9500

Матрица ошибок подтверждает высокую точность модели на этих классах: 19 из 20 изображений каждого класса были классифицированы верно, с минимальным количеством ошибок перекрестной классификации (по 1 ошибке для каждого класса).

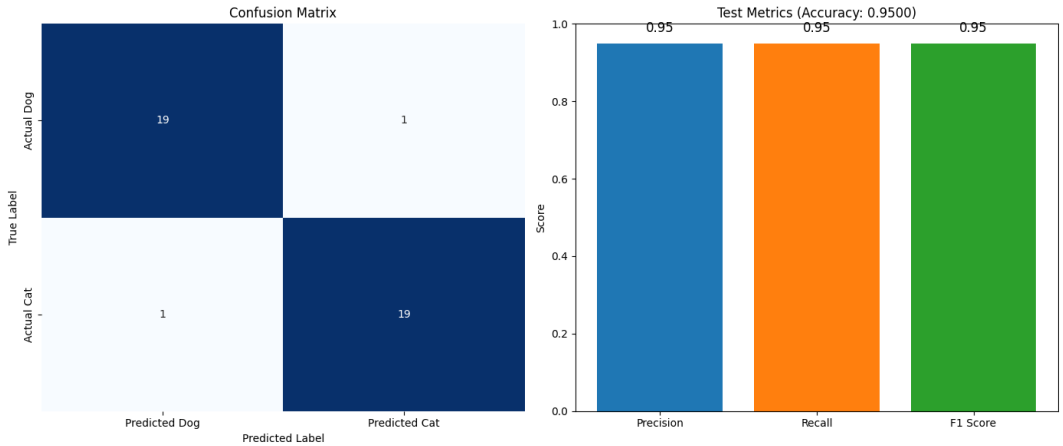


Рис. 12: Нормализованная матрица ошибок и метрики Precision, Recall, F1-score на тестовой выборке для ResNet18 на классах 'dog' и 'cat'.

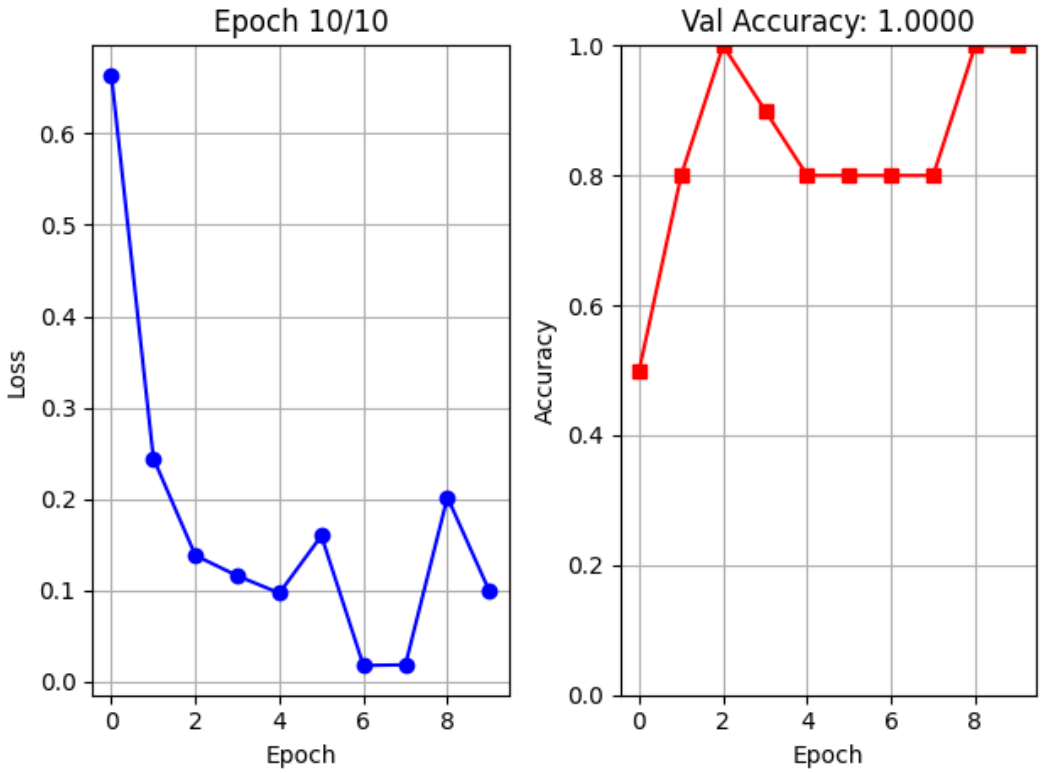


Рис. 13: История обучения (Loss) и валидационная точность (Accuracy) модели ResNet18 на классах 'dog' и 'cat'.

Потребление вычислительных ресурсов при обучении:

- Время обучения: около 27.3 s (0.46 min)
- Пиковое потребление GPU памяти: 0.219 GB (1.49 % от общей доступной памяти)

Результаты на классах *vacuum cleaner* и *airplane*: Далее модель была протестирована на более сложной паре классов ('vacuum cleaner', 'airplane'), мел-спектрограммы которых имеют менее выраженные и более схожие паттерны по сравнению с чёткими паттернами звуков животных. [2]

[2]Следующие подходы будут также обучаться на этой паре классов.

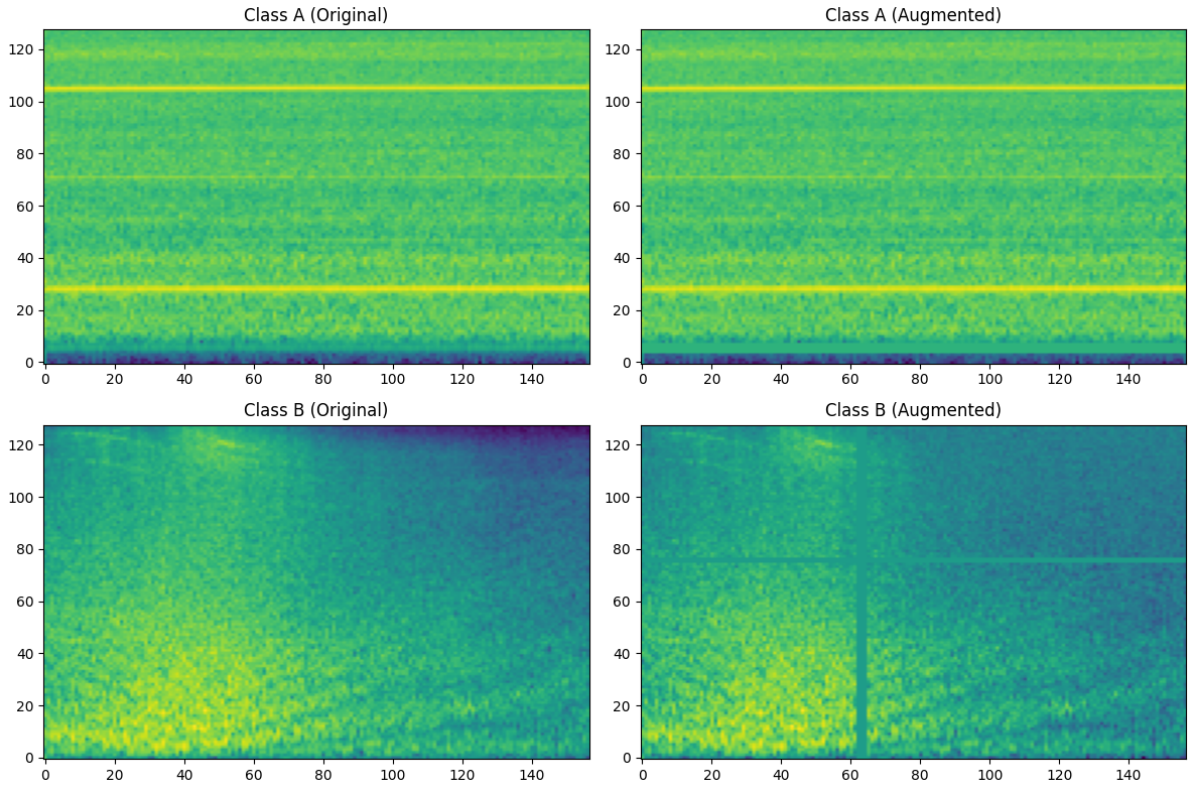


Рис. 14: Сравнение мел-спектрограмм для классов 'vacuum cleaner' (Class A) и 'airplane' (Class B) до (Original) и после (Augmented) применения аугментаций.

Результаты на тестовой выборке для этой пары составили:

- Accuracy: 0.9000
- Precision (weighted): 0.8636
- Recall (weighted): 0.9500
- F1-score (weighted): 0.9048

По матрице ошибок видно, что модель испытывала некоторые затруднения с различением этих классов, чаще ошибаясь при классификации 'vacuum cleaner' как 'airplane' (3 случая), в то время как 'airplane' почти всегда определялся верно (1 ошибка).

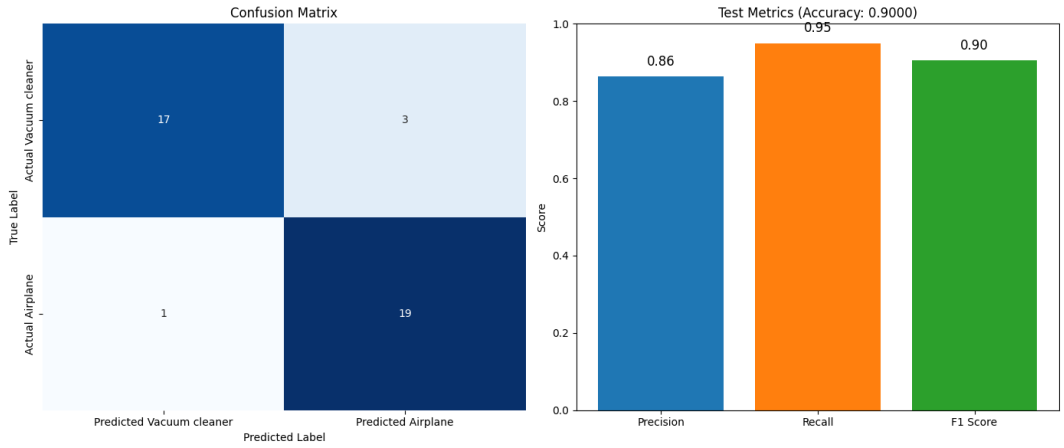


Рис. 15: Нормализованная матрица ошибок и метрики Precision, Recall, F1-score на тестовой выборке для ResNet18 на классах 'vacuum cleaner' и 'airplane'.

Несмотря на снижение метрик на более сложной паре классов, модель ResNet18, обученная на аугментированных мел-спектрограммах, продемонстрировала способность к классификации даже при небольшом объеме обучающих данных и на классах с менее выраженными спектральными признаками, при этом оставаясь эффективной по ресурсам и времени обучения.

3.3.2. Эксперименты с PANNs-эмбеддингами + MLP

Второй исследуемый подход основан на использовании предобученных аудио-эмбеддингов из модели PANNs в сочетании с простым Многослойным Перцептроном (MLP) для классификации. Этот метод оценивает эффективность использования высокоуровневых признаков, извлеченных из модели, предобученной на обширной коллекции аудиоданных (AudioSet), вместо обучения CNN непосредственно на спектрограммах.

Аудиозаписи обрабатывались волновыми аугментациями и подавались в модель PANNs для получения эмбеддингов. Далее MLP обучался на этих эмбеддингах. Результаты на тестовой выборке оказались впечатляющими:

- Accuracy: 1.0000
- Precision (weighted): 1.0000
- Recall (weighted): 1.0000
- F1-score (weighted): 1.0000

Матрица ошибок показывает идеальный результат: все 20 записей каждого класса были классифицированы верно.

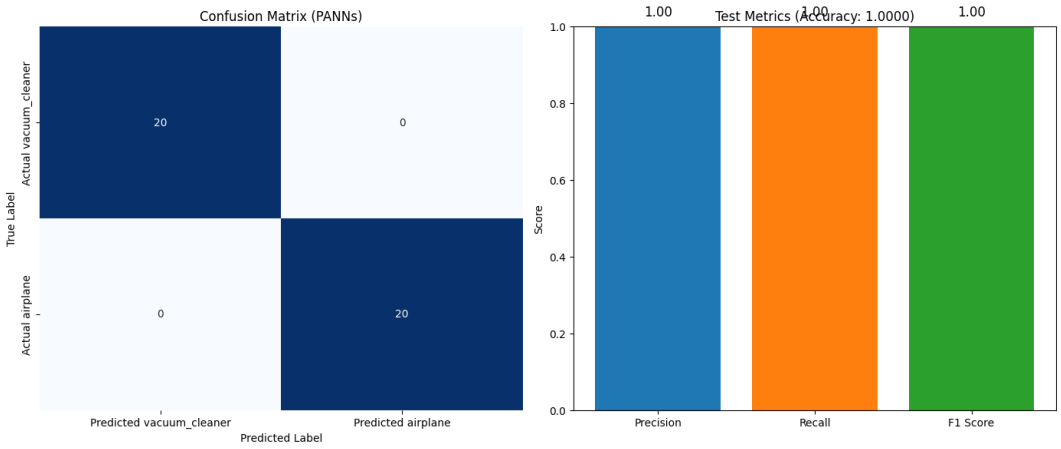


Рис. 16: Нормализованная матрица ошибок и метрики Precision, Recall, F1-score на тестовой выборке для подхода PANNs-эмбеддинги + MLP на классах 'vacuum cleaner' и 'airplane'.

Обучение самого MLP на извлеченных эмбеддингах было быстрым, заняв около 35.0 s (примерно 0.58 min). Однако важно учитывать, что время на извлечение самих эмбеддингов моделью PANNs также требует вычислительных затрат в зависимости от кол-ва классов.

Высокие метрики на данной паре классов свидетельствуют о том, что эмбеддинги, извлекаемые моделью PANNs, содержат достаточно семантической информации для эффективного различения даже сложных и акустически схожих звуков, даже при обучении на небольшом количестве примеров с использованием простого классификатора.

3.3.3. Эксперименты с Audio Spectrogram Transformer (AST)

Третий подход основан на тонкой настройке (Fine-Tuning) предобученной модели Audio Spectrogram Transformer (AST). AST — это современная архитектура на основе трансформеров, специально разработанная для обработки аудио спектрограмм. В этом эксперименте использовалась стратегия **полного Fine-Tuning**, при которой обучались все веса предобученной на AudioSet модели, а не только финальный классификатор.

Модель AST обрабатывает мел-спектрограммы, извлекая признаки с помощью механизма самовнимания. К сырому аудио применялись специфические для AST аугментации перед извлечением признаков. Результаты на тестовой выборке показали высокую производительность:

- Accuracy: 0.9750
- Precision (weighted): 1.0000
- Recall (weighted): 0.9500
- F1-score (weighted): 0.9744

Матрица ошибок демонстрирует всего одну ошибку: одна запись класса 'airplane' была ошибочно классифицирована как 'vacuum cleaner'. Класс 'vacuum cleaner' был распознан идеально.

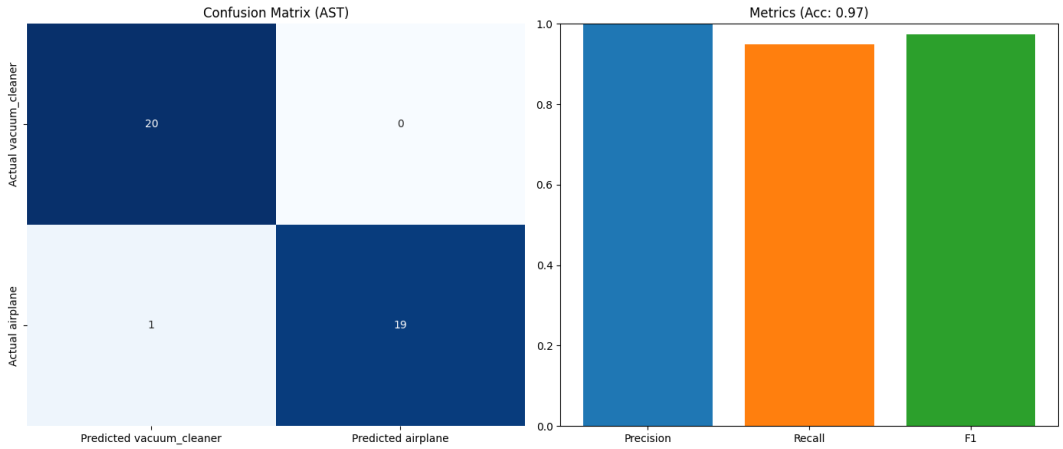


Рис. 17: Нормализованная матрица ошибок и метрики Precision, Recall, F1-score на тестовой выборке для подхода AST (Full Fine-Tuning) на классах 'vacuum cleaner' и 'airplane'.

Обучение модели AST методом полного Fine-Tuning заняло значительно больше времени по сравнению с предыдущими подходами — около 111.8 s (примерно 1.86 min). Это объясняется тем, что в процессе обучения обновлялось гораздо большее количество параметров (веса всей предобученной модели).

Высокая точность AST на сложной паре классов подтверждает потенциал современных трансформерных архитектур для аудиоклассификации, однако такие архитектуры громоздки и требуют большого количества данных для хорошего обучения.

3.3.4. Выбор оптимальной архитектуры для сервиса

Сравнительный анализ трех подходов для классификации звуков на малых данных выявил, что пайплайн на основе PANNs-эмбеддингов с последующей классификацией MLP продемонстрировал наилучшую производительность, достигнув идеальной точности на тестовой выборке для сложных классов. Эффек-

тивное использование предобученных признаков при быстром обучении финального классификатора делает этот метод оптимальным выбором для реализации в сервисе.

4. Заключение

Целью данной работы являлась разработка пайплайна обучения глубоких нейронных сетей и выбор оптимальных моделей для интерактивного конструктора ИИ, ориентированного на пользователей с ограниченными данными и вычислительными ресурсами.

В ходе работы был проведен анализ современных архитектур для классификации изображений (CNN, Transformer, YOLO) и звуков (CNN на спектрограммах, эмбеддинги + классификатор, Transformer), а также изучены техники аугментации данных и трансферного обучения.

Для задачи классификации изображений в качестве кандидатов были выбраны семейства EfficientNet и YOLO. Эксперименты на данных разного объема и новизны (известные/неизвестные классы относительно ImageNet) показали, что EfficientNet B0, использующая стратегию Fine-Tuning с применением аугментаций данных, продемонстрировала наилучший баланс между точностью на малых и неизвестных данных. В отличие от EfficientNet B7, не показавшей существенного прироста метрик на очень малых выборках при значительно более высоких требованиях к ресурсам, и моделей YOLO, чья производительность оказалась менее предсказуемой.

Для задачи классификации звуков были исследованы подходы, основанные на CNN+мел-спектрограммах, PANNs-эмбеддингах+MLPP и AST. Тестирование на сложной для различения паре классов показало, что подход с использованием предобученных PANNs-эмбеддингов достиг наивысшей точности на тестовой выборке, продемонстрировав идеальный результат. Этот метод эффективно использовал знания, закодированные в мощной предобученной аудио-модели, что позволило быстро обучить высокоточный классификатор даже на небольшом числе примеров. Несмотря на высокую точность AST при полном Fine-Tuning, требовал больше времени на обучение, а ResNet18 показал более низкие метрики на сложной задаче по сравнению с PANNs+MLP.

Таким образом, были выбраны следующие архитектуры:

- Для задачи классификации изображений: **EfficientNet B0** с использованием Fine-Tuning и аугментаций данных.
- Для задачи классификации звуков: Пайплайн на основе **PANNs-эмбеддингов** с последующим обучением **MLP**.

Выбранные модели наилучшим образом соответствуют заявленным целям сер-

виса, они показывают наилучшие метрики и удобнее всего могут быть встроены в конструктор ИИ моделей.

Список использованной литературы

[1] Alex Krizhevsky, Ilya Sutskever и Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". В: *Advances in neural information processing systems*. Т. 25. 2012. URL: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[2] Karen Simonyan и Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". В: *arXiv preprint arXiv:1409.1556* (2014). URL: https://arxiv.org/abs/1409.1556.

[3] Christian Szegedy и др. "Going deeper with convolutions". В: *arXiv preprint arXiv:1409.4842* (2014). URL: https://arxiv.org/abs/1409.4842.

[4] Kaiming He и др. "Deep residual learning for image recognition". В: *arXiv preprint arXiv:1512.03385* (2015). URL: https://arxiv.org/abs/1512.03385.

[5] Andrew G Howard и др. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". В: *arXiv preprint arXiv:1704.04861* (2017). URL: https://arxiv.org/abs/1704.04861.

[6] Barret Zoph и Quoc V Le. "Neural Architecture Search with Reinforcement Learning". В: *arXiv preprint arXiv:1611.01578* (2017). URL: https://arxiv.org/abs/1611.01578.

[7] Mingxing Tan и Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". В: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, с. 6105—6114. URL: https://proceedings.mlr.press/v97/tan19a/tan19a.pdf.

[8] Alexey Dosovitskiy и др. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". В: *arXiv preprint arXiv:2010.11929* (2020). URL: url=%7Bhttps://arxiv.org/abs/2010.11929%7D.

[9] Joseph Redmon и др. "You Only Look Once: Unified, Real-Time Object Detection". В: *arXiv preprint arXiv:1506.02640* (2015). URL: https://arxiv.org/abs/1506.02640.

[10] Rahima Khanam и Muhammad Hussain. "YOLOv11: An Overview of the Key Architectural Enhancements". В: *arXiv preprint arXiv:2410.17725* (2024). URL: https://arxiv.org/pdf/2410.17725.

[11] Jason Yosinski и др. “How transferable are features in deep neural networks?” В: *arXiv preprint arXiv:1411.1792* (2014). URL: https://arxiv.org/pdf/1411.1792.

[12] Qiuqiang Kong и др. “PANNs: Large-scale Pretrained Audio Neural Networks for Audio Pattern Recognition”. В: *arXiv preprint arXiv:1912.10211* (2019). URL: https://arxiv.org/abs/1912.10211.

[13] Yuan Gong, Yu-An Chung и James Glass. “AST: Audio Spectrogram Transformer”. В: *arXiv preprint arXiv:2104.01017* (2021). URL: https://arxiv.org/abs/2104.01017.

[14] Karol J Piczak. *ESC: Dataset for Environmental Sound Classification*. Dataset. 2015. URL: https://paperswithcode.com/dataset/esc-50.

[15] MIT. *ast-finetuned-audioset-10-10-0.4593*. Hugging Face Model Hub. 2022. URL: https://huggingface.co/MIT/ast-finetuned-audioset-10-10-0.4593.