

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	2
2	Разработка пайплайна обучения нейросетей для задачи классификации изображений	4
2.1	Теоретические основы и архитектуры нейросетей для классификации изображений	4
2.1.1	Обоснование выбора архитектур для работы	6
2.2	Аугментации данных и реализация процесса обучения моделей . .	7
2.2.1	Аугментации данных для EfficientNet	8
2.2.2	Пайплайн обучения (EfficientNet)	9
2.2.3	Аугментации данных и процесс обучения YOLOv11	10
2.3	Тестирование и сравнение выбранных архитектур на различных данных	10
2.3.1	Описание тестовых сценариев и метрик	11
2.3.2	Эксперименты с EfficientNet B0	13
2.3.3	Эксперименты с EfficientNet B7	15
2.3.4	Эксперименты с YOLOv11	17
2.3.5	Сводное сравнение и выводы по тестированию	20
3	Разработка пайплайна обучения нейросетей для задачи классификации звуков	21

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 не только понимания 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 ограниченности ресурсов и сложности сбора крупных размеченных датасетов.

Существенной частью системы является интеграция с облачными вычислительными мощностями Yandex Cloud, что позволяет пользователям запускать обучение моделей удаленно, 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 сравнение различных архитектур моделей и их обучение, с учетом того, что ресурсы ограничены, а также создание скриптов с динамическим изменением гиперпараметров обучения и генеральной совокупности данных для подачи на бекенд-часть сервиса.

Задачи проекта:

1. Провести аналитический обзор литературы и выбрать лучшие архитектуры моделей под разные домены машинного обучения, также рассмотреть способы аугментации данных.
2. Выбрать способ обучения моделей, а также реализовать полное/fine-tuning обучение разных архитектур.

3. Сравнить полученные результаты полученные на разных данных/методах обучения и сделать оптимальный выбор модели под задачи 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 полносвязанных 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 полносвязанных 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 основанных 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 эмбединги 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 задачах 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-clс	1.6	70.0	89.4	0.5
YOLOv11s-clс	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 изображений.

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 EfficientNet-B7 была взята лишь для исследования - "Насколько можно увеличить метрики на малом объеме 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 объектами.

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 EfficientNet

Для моделей семейства EfficientNet 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 тренировочного 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 (EfficientNet)

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 предтренине 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 объектов. 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-clс, s-clс) 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.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- **Precision (Точность предсказания):** Доля объектов, действительно относящихся к классу, среди всех объектов, которые модель отнесла к этому классу.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

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

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

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

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

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:

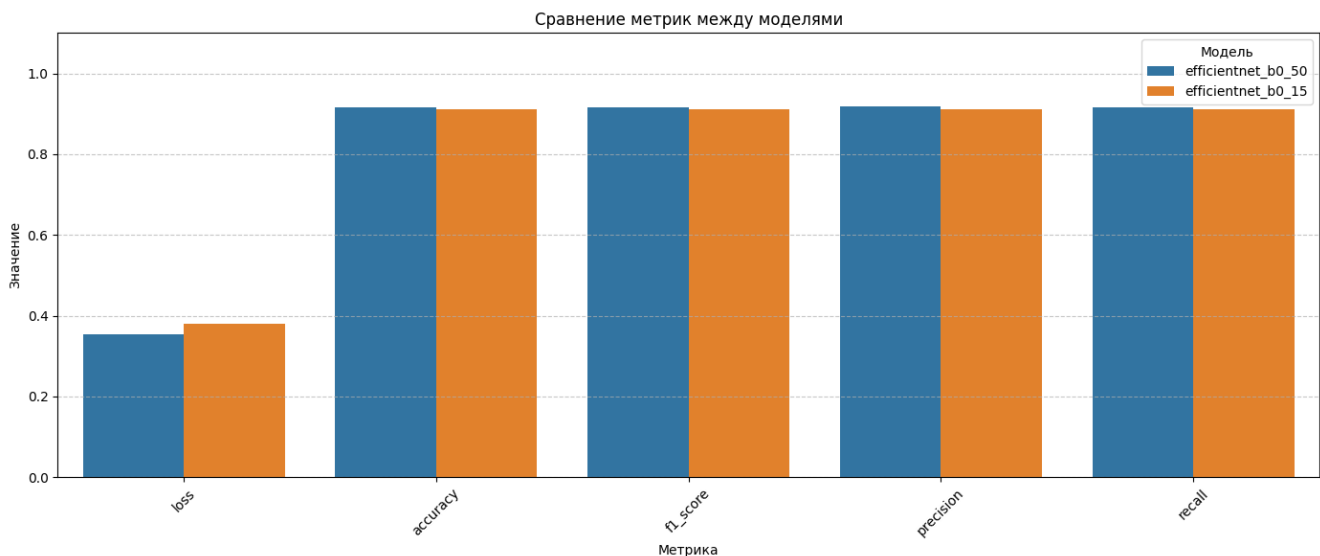
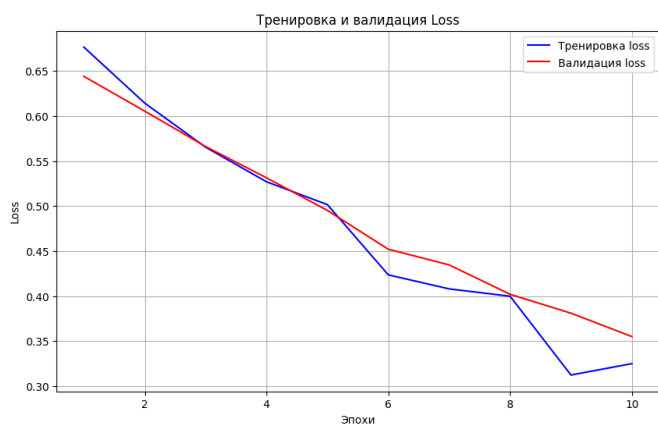
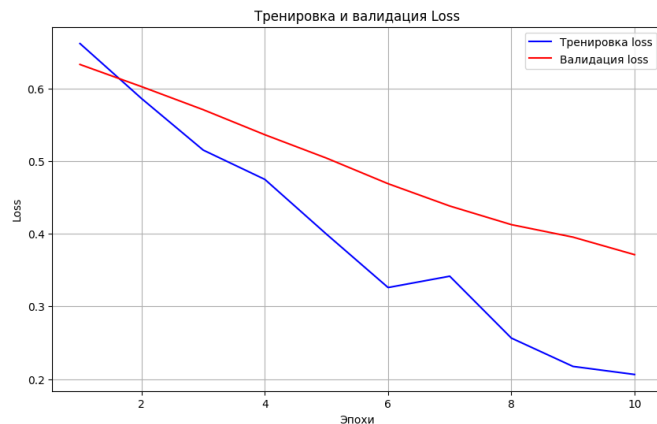


Рис. 1: Метрики efficientnet_b0_50 vs efficientnet_b0_15

Заметим, что метрики идентичны. Сравним графики лоссов:



(a) 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:

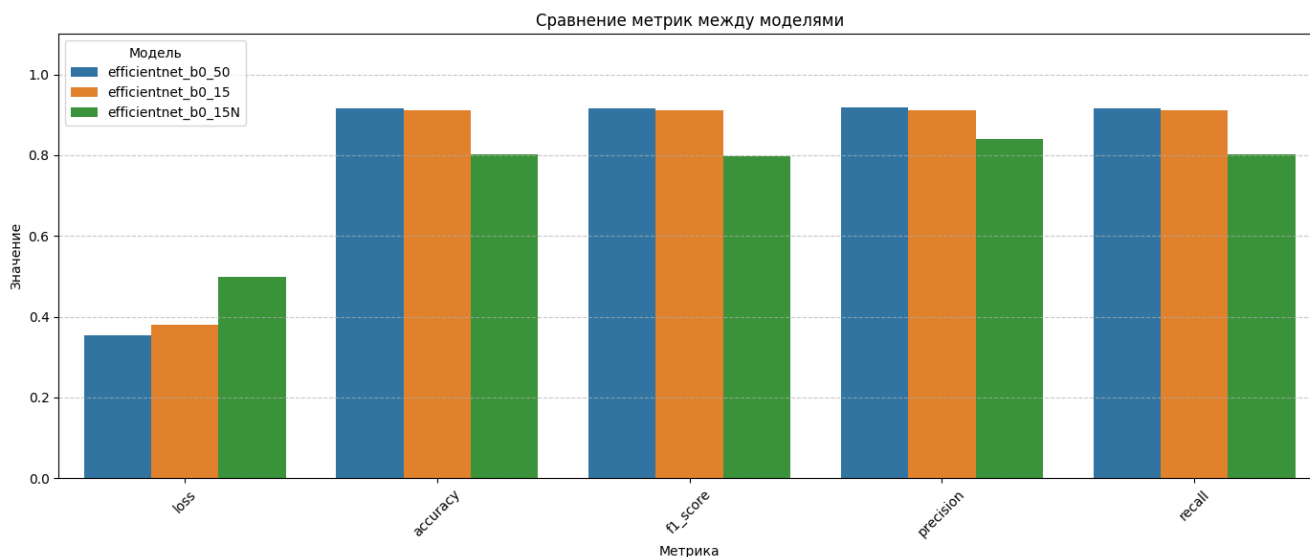


Рис. 3: Метрики efficientnet_b0_50 vs efficientnet_b0_15 vs efficientnet_b0_15N

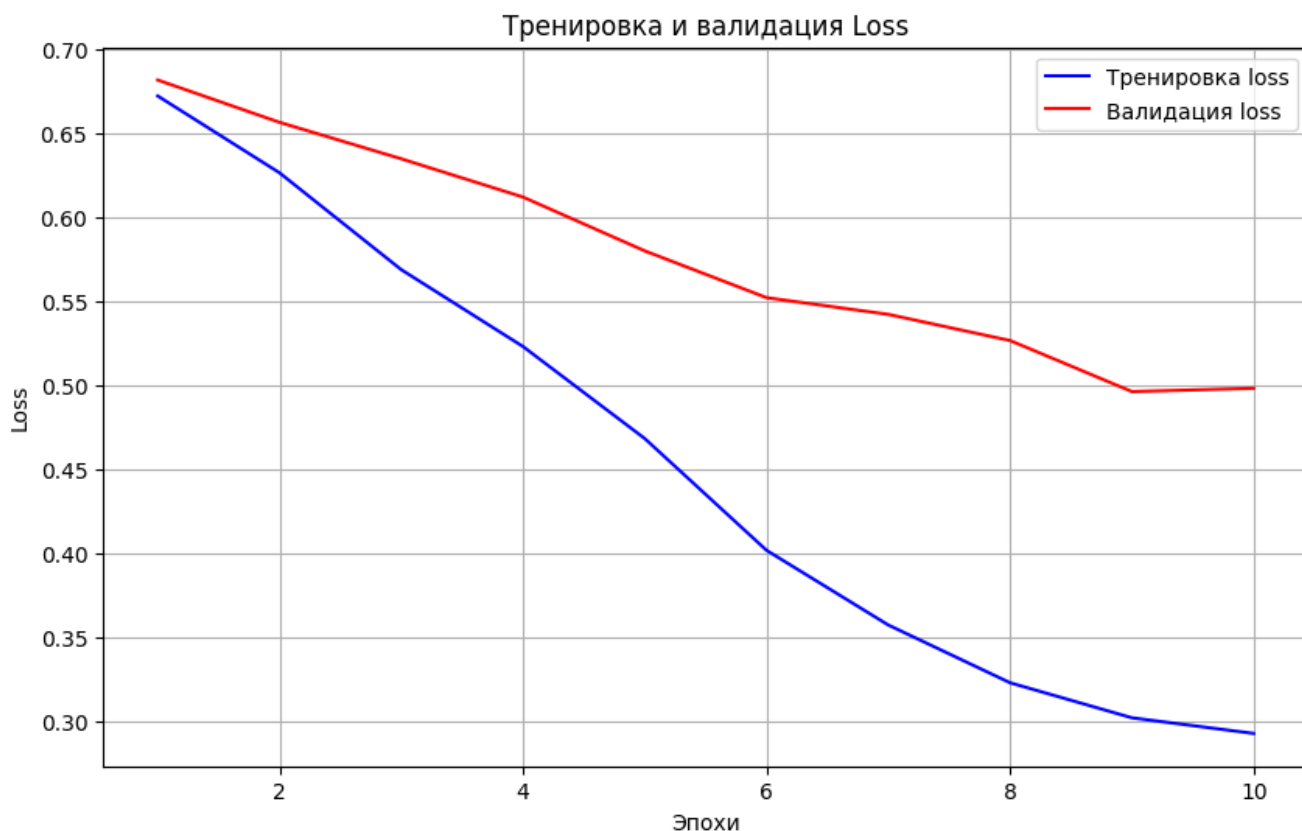


Рис. 4: Loss efficientnet_b0_15N

В среднем качество инференса упало на 10%, однако показатель Ассигасу ≈ 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 видеопамати \geq в 10 раз больше, чем у младшей модели, посмотрим, смогли ли мы улучшить наши метрики, потратив столько ресурсов:

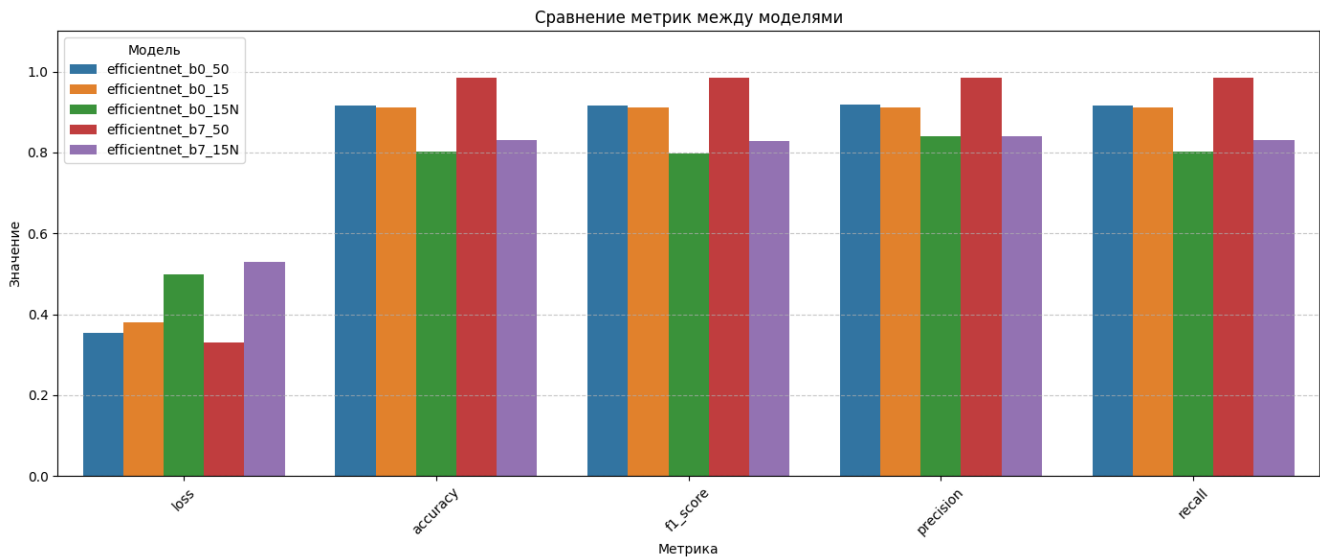


Рис. 5: Метрики моделей семейства EfficientNet

Примечание: В ходе тестирования было выявлено, что 10 эпох для обучения модели B7 недостаточно для нахождения экстремума функции потерь, Val Loss ≈ 0.5305 , после дополнительных 10 эпох, Val Loss ≈ 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:

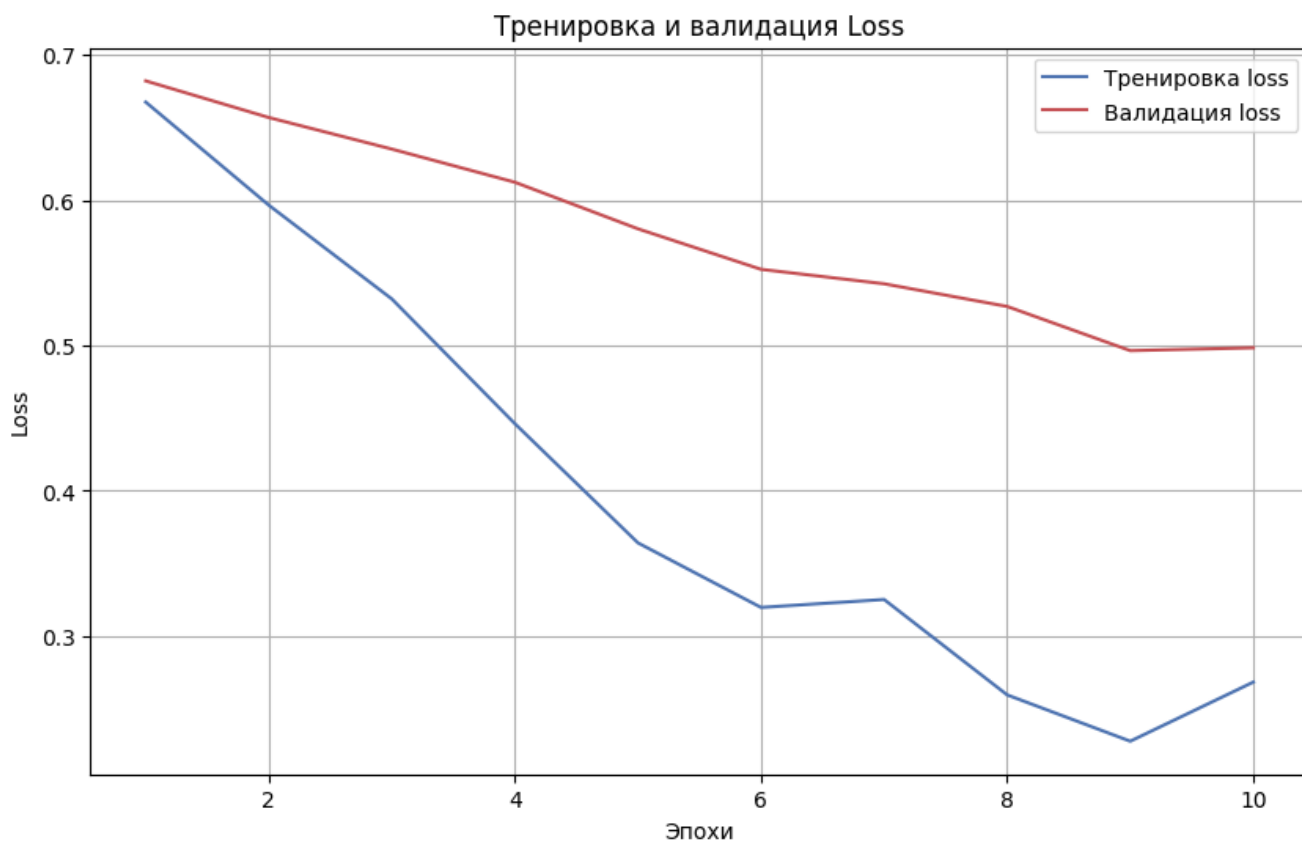
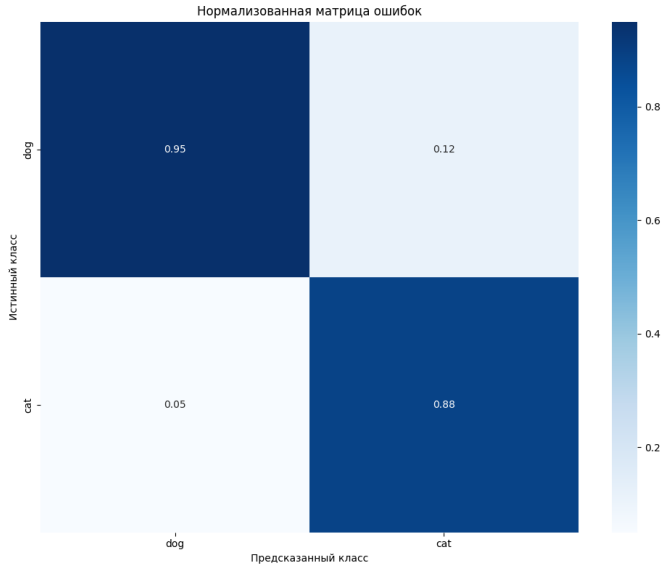


Рис. 6: Loss efficientnet_b7_50

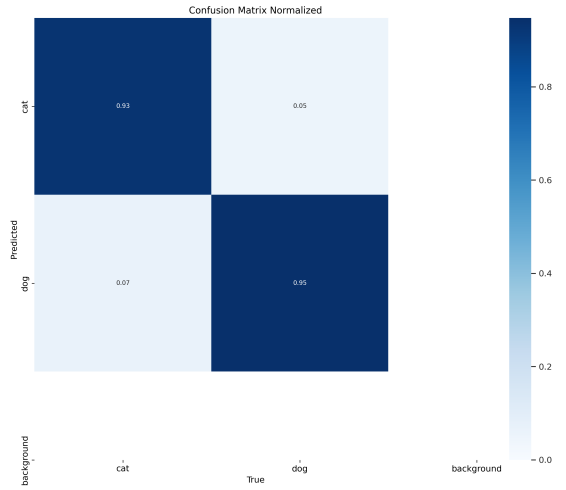
Это подтверждает нашу гипотезу о том, что EfficientNet B7 практически не обучается на знакомых классах, так как падение Val Loss совсем незначительное, при том что метрики около 1.

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:

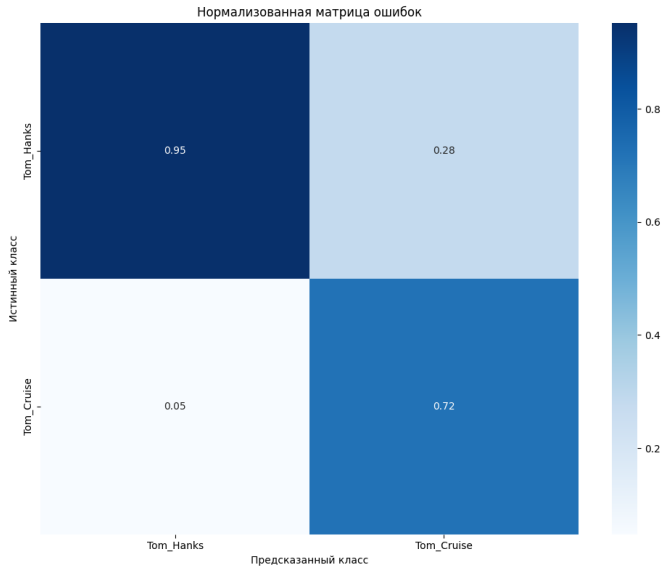


(a) EfficientNet B0

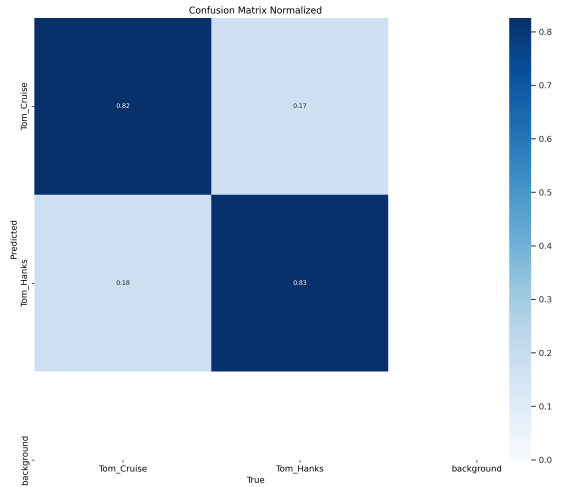


(б) yolo11n-clс

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



(a) EfficientNet B0



(б) yolo11n-clс

Рис. 8: Сравнение Confusion Matrix для EfficientNet B0 и yolo11n-clс (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-clс* 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 Transfer Learning (*Подробности см. в ноутбук*):

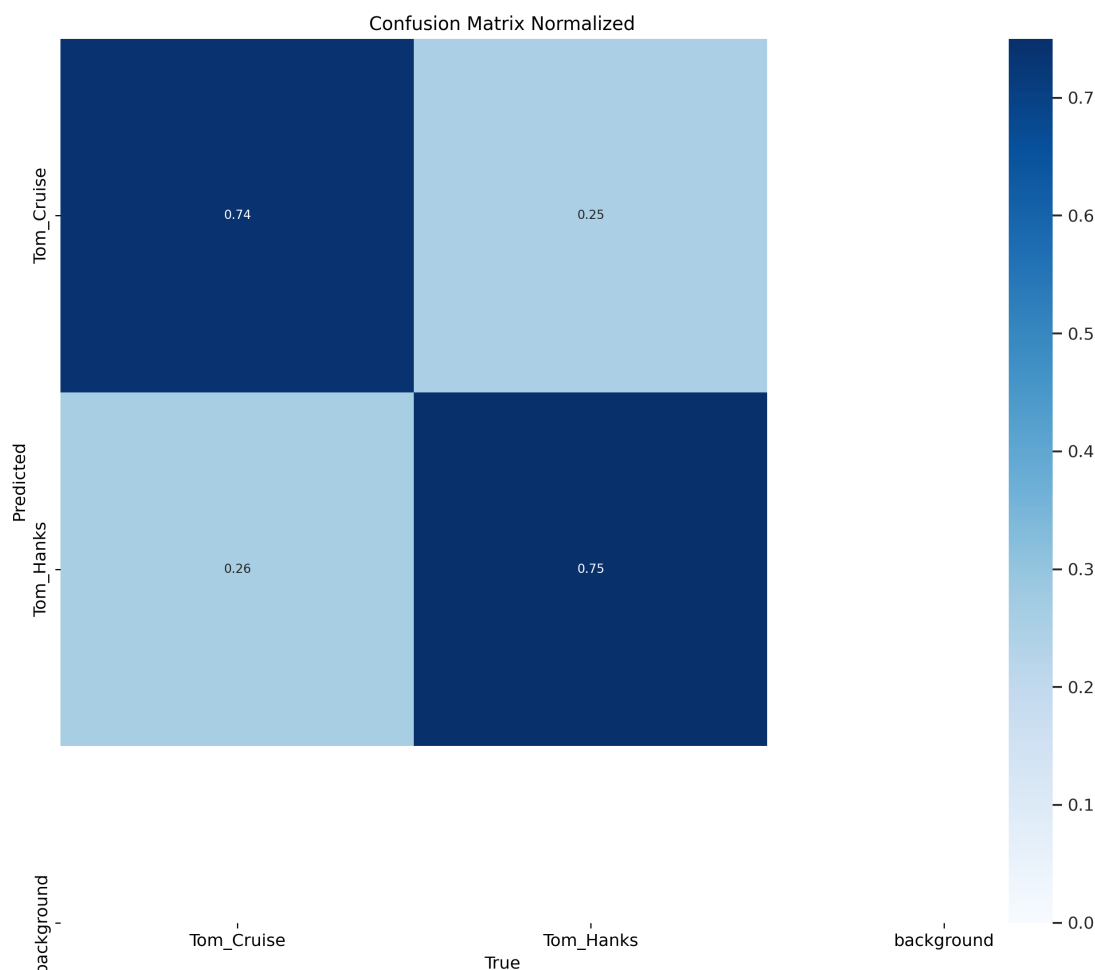


Рис. 9: Confusion Matrix для yolo11n-cls (Transfer Learning) на датасете Celeb-15

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:

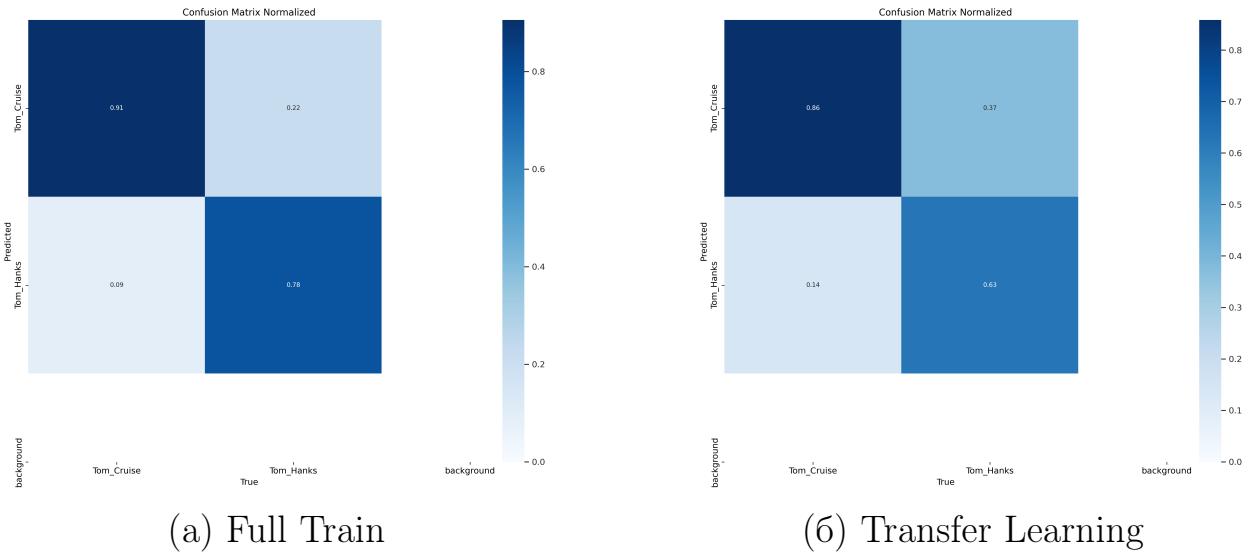


Рис. 10: Сравнение Confusion Matrix для yolo11s-cls с разными методиками обучения на датасете Celeb-15.

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

Список использованной литературы

- [1] Alex Krizhevsky, Ilya Sutskever и Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. B: *Advances in neural information processing systems*. T. 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”. B: *arXiv preprint arXiv:1409.1556* (2014). URL: <https://arxiv.org/abs/1409.1556>.
- [3] Christian Szegedy и др. “Going deeper with convolutions”. B: *arXiv preprint arXiv:1409.4842* (2014). URL: <https://arxiv.org/abs/1409.4842>.
- [4] Kaiming He и др. “Deep residual learning for image recognition”. B: *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”. B: *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”. B: *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”. B: *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”. B: *arXiv preprint arXiv:2010.11929* (2020). URL: [url=%7Bhttps://arxiv.org/abs/2010.11929%7D](https://arxiv.org/abs/2010.11929).
- [9] Joseph Redmon и др. “You Only Look Once: Unified, Real-Time Object Detection”. B: *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”. B: *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?”
B: *arXiv preprint arXiv:1411.1792* (2014). URL: <https://arxiv.org/pdf/1411.1792>.