**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 2](#_Toc194947099)

[1. ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ 5](#_Toc194947100)

[1.1. Определение актуальной технологии 5](#_Toc194947101)

[1.2. Понятие свёрточных нейронных сетей 7](#_Toc194947102)

[1.3. Трансферное обучение 11](#_Toc194947103)

[1.3.1. Архитектура VGG 12](#_Toc194947104)

[1.3.2. Архитектура ResNet 13](#_Toc194947105)

[1.3.3 Архитектура MobileNet 14](#_Toc194947106)

[1.3.4. Архитектура EfficientNet 17](#_Toc194947107)

[2. ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ 20](#_Toc194947108)

[2.1. Набор данных 20](#_Toc194947109)

[2.2. Реализация и обучение нейронных сетей 23](#_Toc194947110)

[3. АНАЛИЗ РЕЗУЛЬТАТОВ 26](#_Toc194947111)

[3.1. Используемые метрики 26](#_Toc194947112)

[3.2. Анализ результатов 27](#_Toc194947113)

[4. ПРИМЕНЕНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ 36](#_Toc194947114)

[4.1. Потенциальные применения системы 36](#_Toc194947115)

[4.2. Демонстрационное приложение 37](#_Toc194947116)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc194947117)

[ПРИЛОЖЕНИЕ А 41](#_Toc194947118)

[ПРИЛОЖЕНИЕ Б 42](#_Toc194947119)

ВВЕДЕНИЕ

Современная авиация играет ключевую роль в глобальной транспортной системе, обеспечивая быстрое перемещение людей и грузов на большие расстояния. С развитием технологий разнообразие воздушных судов непрерывно растёт: от пассажирских лайнеров и грузовых самолётов до специализированных моделей для военных и исследовательских задач. Точная идентификация типа воздушного судна по внешним признакам критически важна для решения таких задач, как управление воздушным движением, мониторинг безопасности, логистическое планирование и анализ спутниковых данных.

В последние годы значительно возрос интерес к использованию методов компьютерного зрения для автоматического анализа аэрокосмических изображений. Такие технологии применяются в гражданской и военной сферах, включая мониторинг аэропортов, контроль воздушного движения, автоматическое обнаружение объектов на спутниковых снимках и идентификацию моделей воздушных судов. Современные алгоритмы машинного обучения, такие как свёрточные нейронные сети, демонстрируют высокую эффективность [1] в задачах классификации изображений, но требуют дальнейшего совершенствования при применении к сложным сценам с высоким уровнем шума и разнообразием классов объектов.

Актуальность разработки надёжных методов машинного обучения для классификации воздушных судов обусловлена растущим объёмом спутниковых данных и необходимостью их оперативного анализа. Основная сложность заключается в вариативности ракурсов съёмки, влиянии погодных условий, наличии помех и высокой степени схожести между различными моделями воздушных судов. Совершенствование методов автоматической классификации позволит повысить точность идентификации и сократить нагрузку на специалистов, работающих с аэрокосмическими данными.

Целью данной работы является разработка системы классификации типов воздушных судов по спутниковым изображениям с использованием методов машинного обучения.

Для реализации поставленной цели были решены следующие задачи:

1. Определить наиболее подходящий метод машинного обучения для разработки системы;
2. Определить наиболее подходящую архитектуру свёрточной нейронной сети;
3. Определить метрики для объективного сравнения моделей;
4. Обучить модели на наборе данных спутниковых изображений воздушных судов;
5. Определить наиболее оптимальную модель для заданной цели;
6. Реализовать пользовательский интерфейс.

Объектом исследования является процесс автоматической классификации воздушных судов по изображениям; предметом исследования является применение свёрточных нейронных сетей для решения задачи классификации типов воздушных судов.

Методологической основой работы является использование методов глубокого обучения, в частности, свёрточных нейронных сетей и обучения с переносом для обработки и классификации изображений. В качестве инструментов использовались язык программирования Python, в частности библиотеки TensorFlow и Keras для реализации моделей машинного обучения, а также набор данных спутниковых снимков воздушных судов. Для разработки пользовательского интерфейса применялась библиотека PythonQt, которая позволила создать удобное и эффективное графическое представление результатов классификации.

Практическая значимость работы заключается в разработке системы, способной автоматически идентифицировать типы воздушных судов, что может быть применимо в задачах мониторинга, обороны и авиационной аналитики.

Структура работы включает введение, теоретический обзор современных методов классификации изображений с применением глубокого обучения, описание используемых моделей и методологии экспериментов, результаты тестирования разработанной системы, анализ их эффективности и заключение.

1. ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ

1.1. Определение актуальной технологии

Задача распознавания типов воздушных судов на спутниковых снимках относится к более широкой области классификации изображений — одного из фундаментальных направлений в компьютерном зрении. Классификация изображений предполагает автоматическое отнесение входного изображения к одному из заранее определённых классов на основе визуального анализа его содержимого.

Существует ряд подходов к решению задач классификации изображений, основанных на различных методах обработки и анализа данных. К числу широко применяемых относятся алгоритмы на основе свёрточных нейронных сетей (англ. CNN), а также метод опорных векторов (англ. SVM), метод k-ближайших соседей (англ. K-NN) и метод случайного леса (англ. Random Forest) [2].

Методы, такие как SVM и K-NN, выполняют классификацию на основе извлеченных вручную признаков, полученных в результате предварительной обработки. Эти подходы требуют тщательной настройки параметров и чувствительны к качеству выборки, но при этом остаются интерпретируемыми и пригодными для задач с ограниченными объёмами данных. Метод опорных векторов демонстрирует высокую точность при бинарной классификации и устойчивость к зашумлённым данным, однако плохо масштабируется на задачи с множеством классов. Алгоритм k-ближайших соседей отличается простотой реализации, но требует больших затрат на хранение и обработку, особенно при работе с большими наборами данных. Метод случайного леса показывает высокую точность на структурированных признаках, но требует существенных вычислительных ресурсов при росте сложности задачи [2].

Для выявления наиболее часто применяемых методов и алгоритмов в задачах классификации изображений воздушных судов был проведён библиометрический анализ научных публикаций. С помощью поисковой системы по научным публикациям IEEE Xplore была сформирована выборка из 500 рецензируемых статей по ключевому запросу "aircraft image classification", опубликованных с 2010 по 2025 год. Полученные данные были обработаны с применением программного обеспечения VOSviewer, позволяющего строить визуальные карты научного ландшафта на основе соавторства, частотности ключевых слов и их семантической близости [3].

С помощью VOSviewer было выявлено 29 уникальных ключевых слов, которые встречаются в выборке не менее 25 раз, на основе которых была построена семантическая карта совместной встречаемости ключевых слов, представленная на рисунке 1.

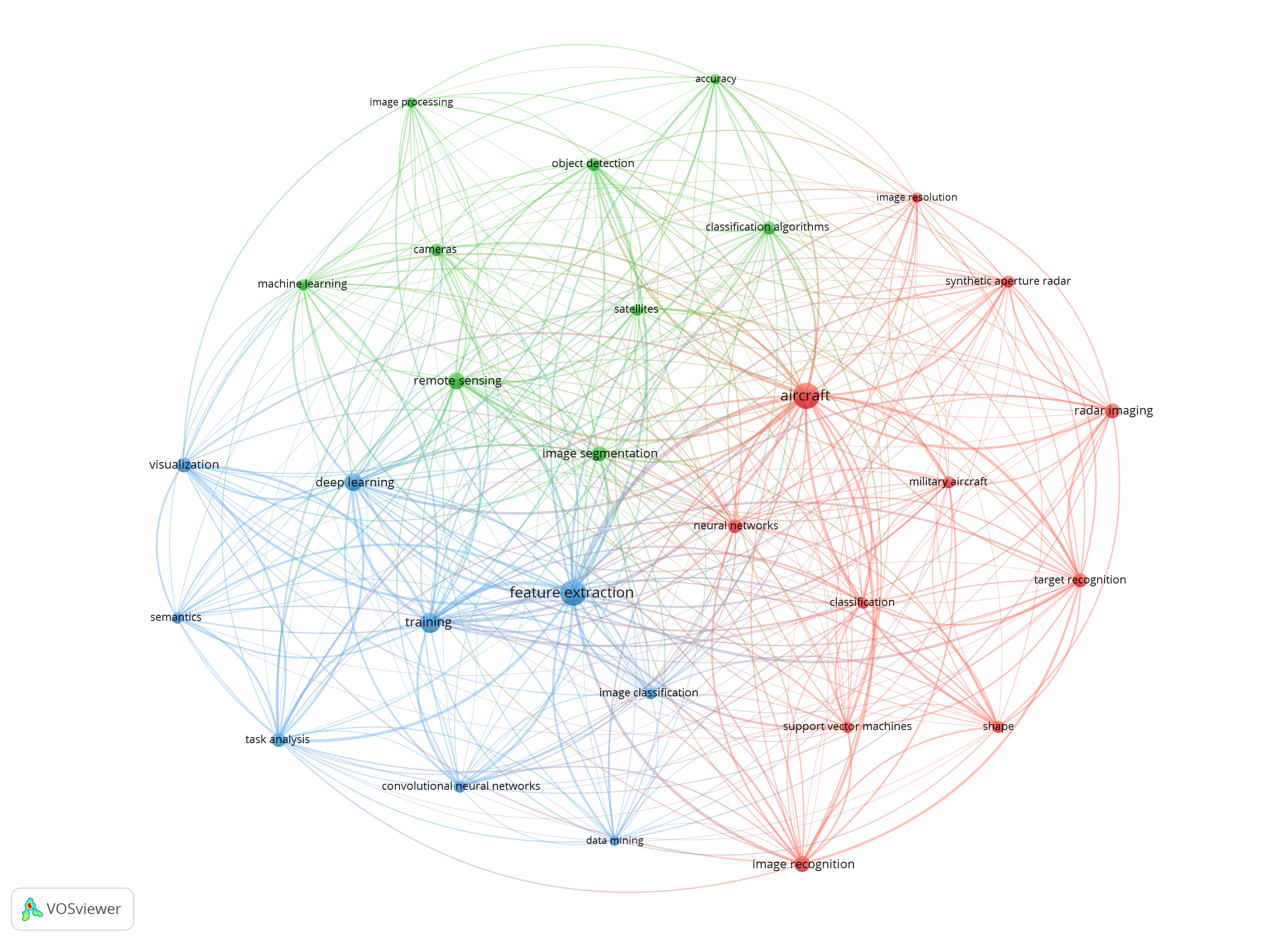


Рисунок 1 – Семантическая карта совместной встречаемости ключевых слов

Анализ карты показывает, что наибольшую плотность связей и центровую позицию в пространстве понятий занимает термин «feature extraction» (извлечение признаков), тесно связанный с понятием «deep learning» (глубокое обучение) и «neural networks» (нейронные сети), что указывает на центральную роль алгоритмов автоматического извлечения релевантных признаков изображения. При этом классические методы опорных векторов, k-ближайших соседей и случайного леса на карте отсутствуют. Однако, на карте присутствует понятие «convolutional neural network» (свёрточные нейронные сети), которое встречалось в выборке 37 раз.

Полученные в результате библиометрического анализа результаты указывают на доминирующее положение методов, основанных на извлечении признаков изображений, таких как нейронные сети и связанные с ними методы глубокого обучения.

На основе проведенного анализа, для разработки алгоритма автоматической классификации воздушных судов по спутниковым изображениям была выбрана технология свёрточных нейронных сетей, которая демонстрирует высокую эффективность в задачах классификации изображений из множества различных областей, включая классификацию спутниковых изображений, изображений автомобилей и другой техники, а также применяемая для распознавания воздушных судов [4–6].

1.2. Понятие свёрточных нейронных сетей

Свёрточные нейронные сети (далее - СНС) являются классом алгоритмов машинного обучения, в основе которых лежит способность автоматически извлекать признаки из входных данных, что особенно эффективно при обработке изображений и видео. Это достигается благодаря применению свёрточных слоев, которые фильтруют информацию и сохраняют только наиболее значимые признаки [7].

Несмотря на непрерывную эволюцию СНС, набирающую все большие темпы за последнее десятилетие, основные части алгоритма остаются неизменными. Первая модель свёрточной нейронной сети – LeNet-5, разработанная в 1998 году для задачи распознавания рукописных цифр и букв с помощью классификации [8], использует в своей основе те же (но модифицированные) основные компоненты, как и выпущенная в 2021 году модель СНС EfficientNetV2-L с более чем 120 миллионами параметров [9].

Архитектуру простейшей СНС можно разбить на 2 основные части, связанные друг с другом и идущие последовательно [8]:

1) Модуль извлечения признаков, состоящий из набора последовательных слоёв свёртки, слоёв активации и слоёв подвыборки.

2) Обучаемый классификационный модуль, представляющий из себя полносвязную нейронную сеть.

Свёрточные слои — основной компонент СНС, обрабатывающий входные данные с помощью фильтров (ядер свёртки). Ядра свертки - матрицы весов небольшого размера, как правило, 3x3. Свёртка позволяет сети выявлять локальные признаки, такие как края, текстуры и другие элементы изображения. Фильтр перемещается по входному признаку, например, градиенту яркости пикселей изображения, и вычисляет свёртку, то есть скалярное произведение значений входных пикселей и весов ядра. В результате получается карта признаков, содержащая информацию о выявленных структурах. Во время обучения нейронной сети градиентный спуск изменяет веса фильтров, чтобы сеть лучше выделяла значимые признаки [7]. На рисунке 2 представлено визуальное представление работы ядра слоя свёртки [10].

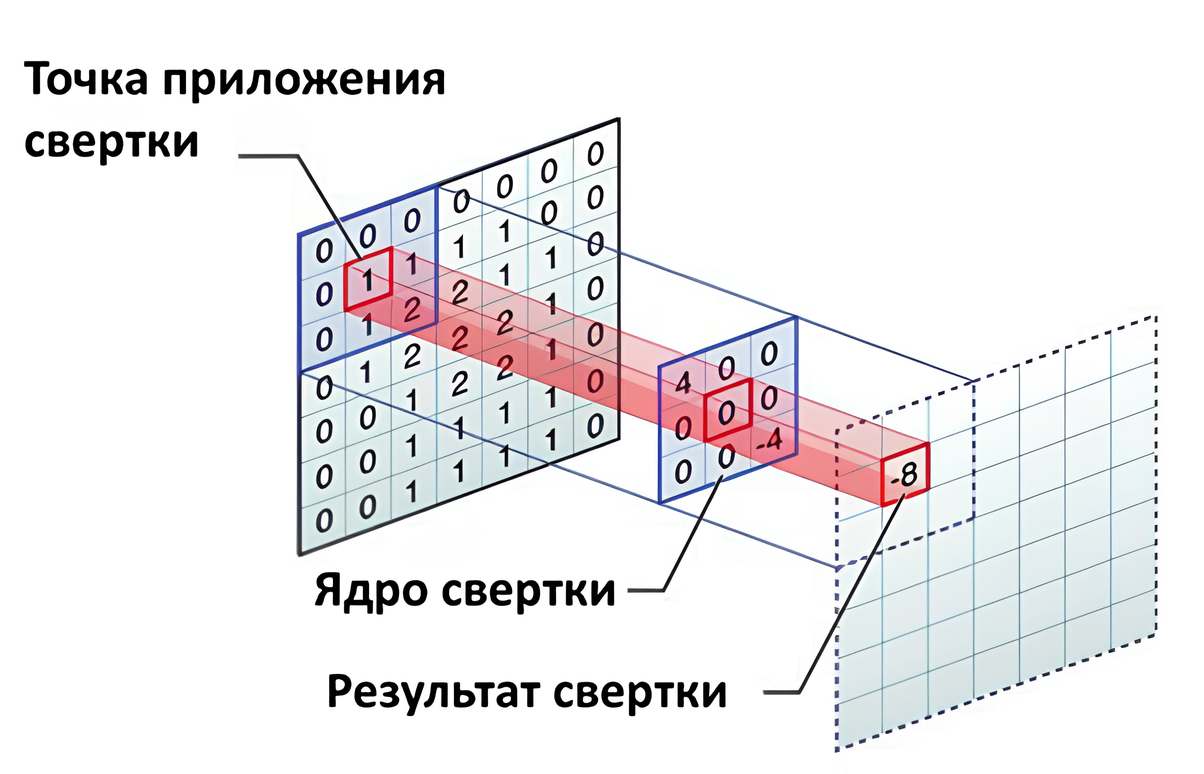


Рисунок 2 - Визуальное представление работы ядра свертки

После применения свёрточных слоёв в большинстве случаев применяется слой активации для введения нелинейности в модель. Это позволяет сети моделировать сложные зависимости между признаками и улучшает её способность к обобщению. Одной из наиболее распространённых функций активации является ReLU (англ. Rectified Linear Unit), которая заменяет все отрицательные значения на ноль, сохраняя положительные значения без изменений [11]. Математическое представление функции активации ReLu:

Другие популярные функции активации включают сигмоидную функцию и гиперболический тангенс (tanh), каждая из которых имеет свои особенности и области применения [11].

Слои подвыборки, в свою очередь, позволяют снизить пространственное разрешение данных, уменьшая размерность и вычислительные затраты, а также способствуют инвариантности к небольшим изменениям в изображении. Существуют несколько разновидностей слоёв подвыборки, однако основные из них: максимальная подвыборка (англ. max pooling) и средняя подвыборка (англ. average pooling), где используется максимальное и среднее значение признаков соответственно [12]. На рисунке 3 представлено визуальное представление работы слоев максимальной и средней подвыборки [13].

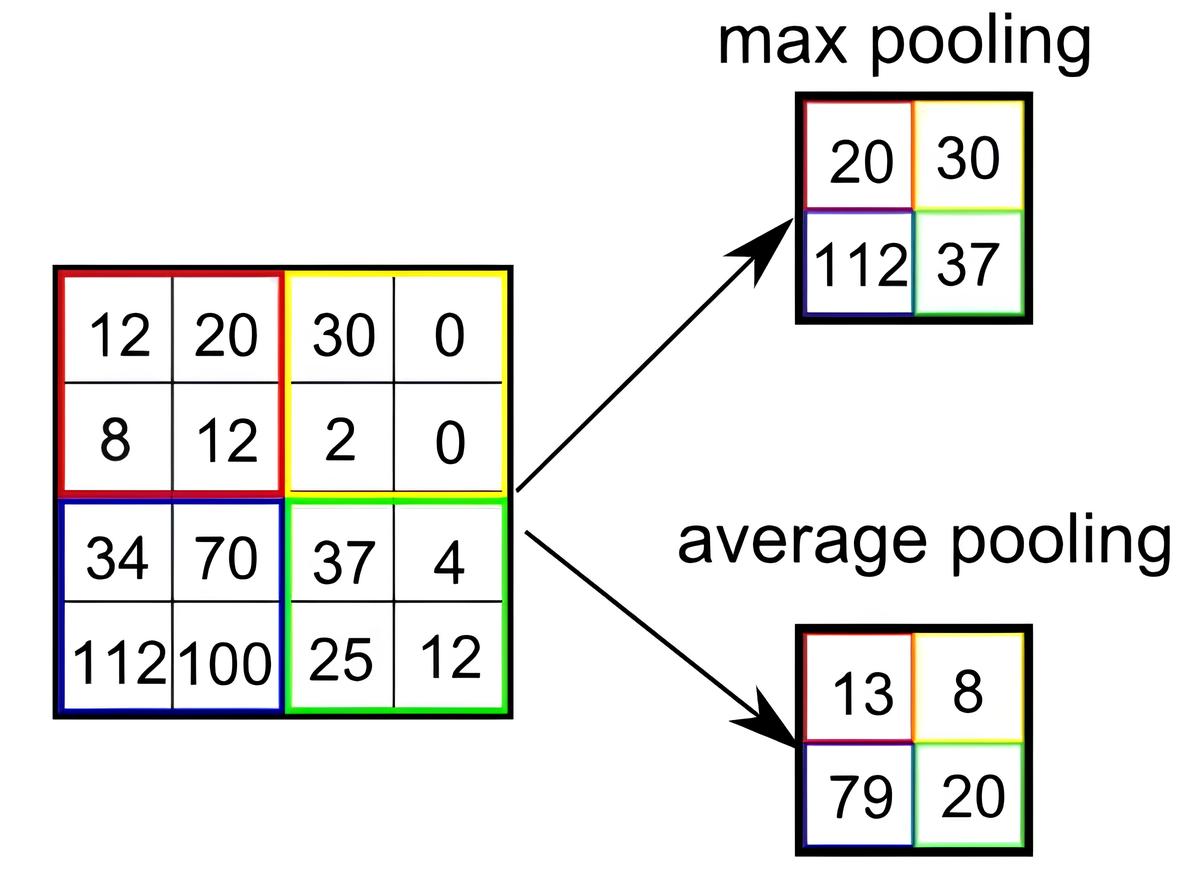


Рисунок 3 - Визуальное представление работы слоев максимальной и средней подвыборки

После выделения основных признаков модуль классификации выполняет интеграцию этих признаков для принятия окончательного решения о принадлежности объекта к определённому классу. Полносвязная нейронная сеть преобразует входной вектор выделенных признаков в выходной вектор вероятностей принадлежности классифицируемого объекта к определённому классу [1].

В задачах классификации на последнем слое часто используется функция активации softmax, которая нормализует выходные значения в диапазон от 0 до 1, представляя собой вероятности принадлежности объекта к определенному классу. Математическое представление функции softmax:

,

где *z* – выходной вектор полносвязного слоя;

*zi –* выходное значение нейрона *i* полносвязного слоя;

*N* – общее количество классов [14].

На рисунке 4 представлена стандартная архитектура простейшей СНС, где «скрытые слои» - модуль выделения признаков, а «классификация» - полносвязная нейронная сеть – классификатор [15].

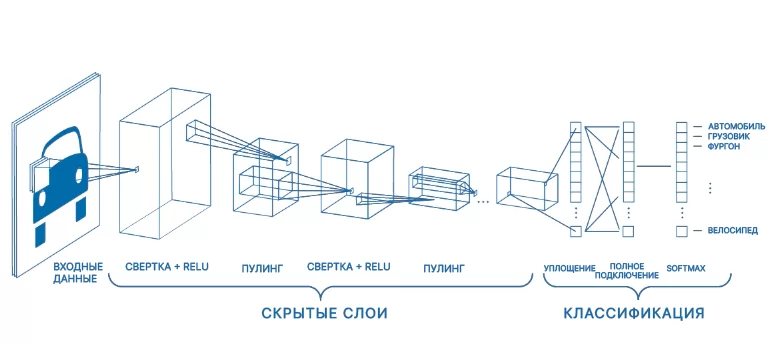


Рисунок 4 - Стандартная архитектура простейшей СНС

1.3. Трансферное обучение

Современные СНС обладают высокой эффективностью в задачах классификации изображений, однако их обучение «с нуля», а именно обучение как модуля извлечения признаков, так и классификационного модуля, требует значительных вычислительных ресурсов и больших объемов данных. Например, обучение модели AlexNet, которая произвела революцию в области машинного обучения в 2012 году, потребовало использования нескольких графических процессоров и заняло значительное время [16].

Для преодоления этих ограничений применяется метод трансферного обучения. Этот подход позволяет использовать веса, полученные моделью при решении одной задачи, для ускорения и улучшения обучения в другой, схожей задаче [17]. В частности, модели, обученные на обширном наборе данных ImageNet Large Scale Visual Recognition Challenge (далее - ILSVRC), содержащем 1,461,406 размеченных изображений [18], широко используются в качестве основы для решения специализированных задач.

В данной работе применяется метод трансферного обучения, при котором параметры базовой модели, предварительно обученной на наборе данных ILSVRC, фиксируются и не обновляются в ходе дальнейшего обучения. Адаптация модели осуществляется исключительно на уровне классификационного слоя, использующего заранее извлечённые признаки.

В следующих разделах рассматриваются архитектуры базовых моделей, к которым была применена методика трансферного обучения для последующего выбора наиболее оптимальной модели для решения задачи классификации воздушных средств на спутниковых изображениях.

1.3.1. Архитектура VGG

Архитектура VGG (Visual Geometry Group), представленная в 2014 году характеризуется последовательным увеличением глубины сети за счёт использования небольших свёрточных фильтров размером 3×3, что позволило значительно улучшить качество классификации изображений по сравнению с более ранними моделями, такими как AlexNet [19].

Модель была представлена в пяти конфигурациях, различающихся глубиной и структурой свёрточных слоёв. Среди них наибольшее распространение получили VGG-16 и VGG-19, содержащие 16 и 19 весовых слоёв соответственно. Эти версии активно используются в задачах компьютерного зрения, например для распознавания промышленных дефектов или для классификации видов растений, благодаря способности извлекать сложные иерархические признаки из изображений [20, 21]. На рисунке 5 представлено визуальное представление архитектуры модели VGG-16.

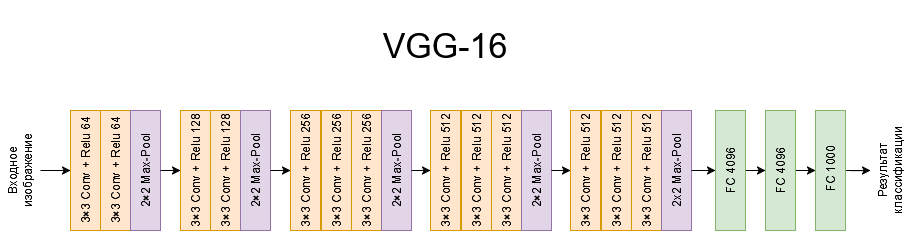


Рисунок 5- Визуальное представление архитектуры модели VGG-16

1.3.2. Архитектура ResNet

Архитектура Residual Network (ResNet) была разработана в 2015 году группой исследователей под руководством Кайминга Хэ с целью решения проблемы затухания градиентов, возникающей при обучении глубоких нейронных сетей. Затухание градиентов приводит к тому, что градиенты ошибки становятся чрезвычайно малыми при обратном распространении ошибки, что затрудняет обновление весов в слоях сети и замедляет процесс обучения [22].

В основе архитектуры ResNet лежит концепция остаточного обучения (residual learning), согласно которой вместо непосредственного приближения некоторой целевой функции , каждый блок слоёв сети обучается аппроксимировать остаточную функцию:

,

где - это желаемая функция, которую сеть должна аппроксимировать, то есть преобразование входного сигнала в более информативное представление, а - это разница между и входными данными [22].

Для реализации остаточного обучения в архитектуре ResNet применяются так называемые пропускные соединения (англ. shortcut connections), которые позволяют входным данным слоя напрямую передаваться в следующие слои, минуя один или несколько промежуточных слоёв. В простейшем случае пропускные соединения реализуются посредством прямого копирования данных без изменений, не увеличивая вычислительную сложность сети. Данная методика позволяет эффективно обучать очень глубокие нейронные сети, избегая проблем с затуханием градиентов. Авторы ResNet показали, что предложенная архитектура позволяет эффективно обучать сверхглубокие сети, например, сеть с 1202 слоями [22].

В оригинальной статье также представлены несколько версий моделей на основе ResNet, различающихся по глубине сети: ResNet-18, ResNet-34, ResNet-50, ResNet-101 и ResNet-152. Число после "ResNet-" указывает на количество слоёв в модели [22].

На рисунке 6 изображено визуальное представление архитектуры модели ResNet-34 [22].

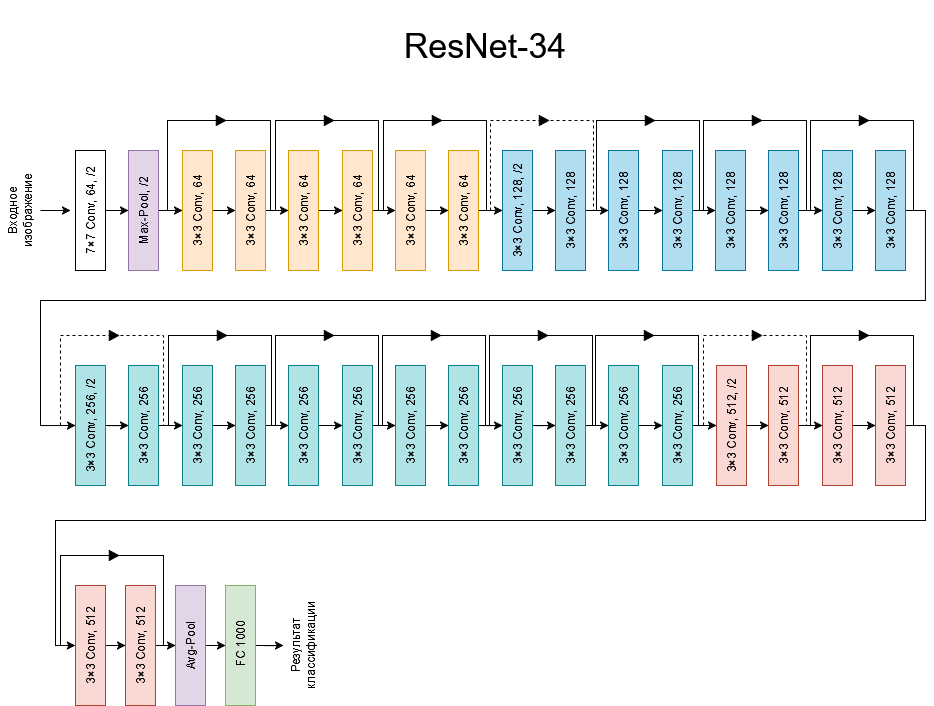


Рисунок 6 – Визуальное отображение архитектуры модели ResNet-34

1.3.3 Архитектура MobileNet

Архитектура MobileNet была представлена в 2017 году группой исследователей из компании Google. MobileNet проектировалась для работы в мобильных устройствах и встроенных системах и, как следствие, большой акцент при разработке уделялся производительности и эффективности. Ключевой особенностью MobileNet является использование DS Conv (depthwise separable convolutions) блоков свёртки, которые позволяют значительно уменьшить количество параметров и вычислительные затраты без существенной потери точности [23].

В стандартных слоях свёртки, применяемых, например, в VGG-19 каждый фильтр применяется ко всем каналам входного изображения, что приводит к значительным вычислительным затратам. В блоках DS Conv используется особый подход, заключающийся в разделении операции свёртки на два этапа [23]:

1. Cвёртка по глубине (англ. depthwise convolution) - каждый фильтр применяется отдельно к каждому каналу входного изображения.

2. Точечная свёртка (англ. pointwise convolution) - 1×1 свёртка, объединяющая признаки, выделенные свёрткой по глубине, и смешивающая информацию между каналами.

Использование DS Conv блоков свёртки сокращает количество операций умножения и сложения примерно в 8–9 раз по сравнению с традиционной свёрточной архитектурой, при этом обеспечивая лишь небольшое снижение точности [23].

На рисунке 7 представлено визуальное представление архитектуры модели на основе MobileNet, а также архитектура блока свёртки DS Conv [23].

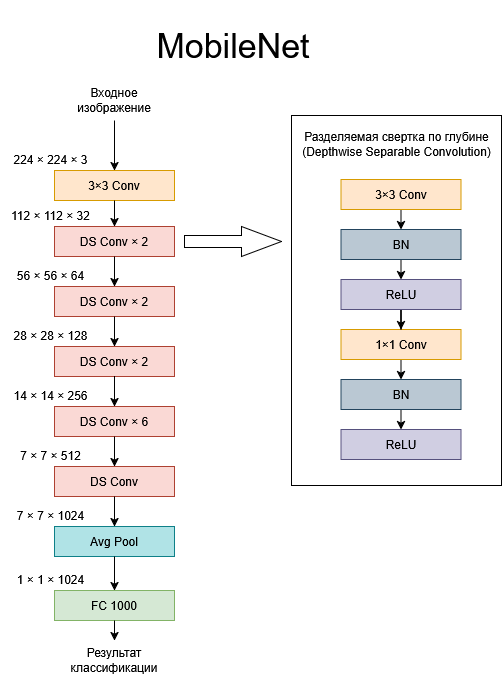


Рисунок 7 - Визуальное представление модели на основе архитектуры MobileNet и блока свёртки DS Conv

В дальнейшем были представлены вторая версия архитектуры в 2018 году (MobileNetV2) и третья версия в 2019 (MobileNetV3). Особенностью MobileNetV2 является использование обратных остаточных блоков свёртки (inverted residuals), которые являются модификацией ResNet блоков, и линейных узких слоев (англ. linear bottlenecks). MobileNetV3, представляя собой прямую эволюцию MobileNetV2, является одной из первых архитектур СНС, при создании которой были использованы методы автоматического поиска архитектуры, такие как NAS (Neural Architecture Search) и NetAdapt. Также в MobileNetV3 была использована функция активации h-swish, пришедшая на замену ReLU, и модифицированные свёрточные блоки, включающие в себя слой squeeze-and-excitation (SE) модули, которые адаптивно изменяют весовые коэффициенты каналов, усиливая наиболее значимые признаки [24, 25].

1.3.4. Архитектура EfficientNet

Архитектура EfficientNet, разработанная в 2019 году, отличается использованием метода комплексного масштабирования, позволяющего одновременно увеличивать глубину (количество слоев), ширину (число каналов в каждом слое) и разрешение (размер входных данных) сети с помощью единого коэффициента. В отличие от традиционных методов, где масштабируется только один параметр, такой подход позволяет добиться лучшего баланса между точностью и вычислительной эффективностью модели [26].

Помимо использования метода комплексного масштабирования, данную архитектуру выделяют блоки свёртки – MBConv. Данные блоки являются модификацией обратных остаточных блоков свёртки (используемых в архитектуре MobileNetV2) с применением механизма squeeze-and-excitation (SE). В модели EfficientNet блоки MBConv представлены в двух вариациях [26]:

1) MBConv1 – базовый вариант блока без расширяющего слоя (expansion layer).

2) MBConv6 - расширенный вариант блока, в котором используется коэффициент расширения 6, т.е. количество каналов во входном тензоре увеличивается в 6 раз. Это достигается за счёт применения 1×1 свёртки, которая расширяет пространство признаков, позволяя модели извлекать более сложные и выразительные признаки.

На рисунке 8 представлено визуальное представление модели на основе архитектуры EfficientNet, а также архитектуры блоков свёртки MBConv1 и MBConv6 [26].

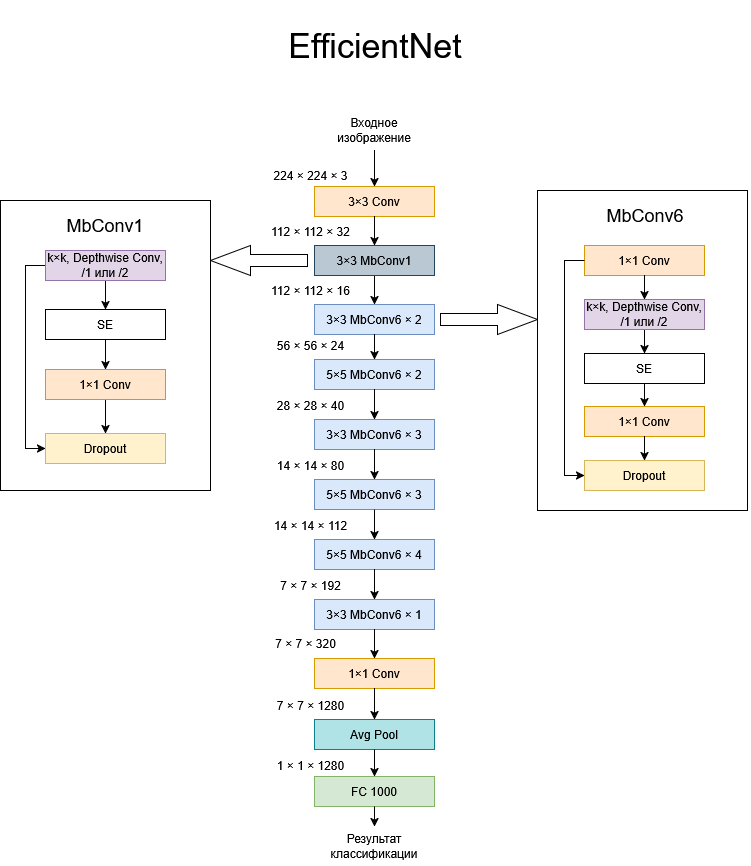
**

Рисунок 8 – Визуальное представление модели на основе архитектуры EfficientNet и архитектуры блоков MbConv1 и MbConv6

В 2021 году была разработана усовершенствованная версия данной архитектуры - EfficientNetV2, особенностью которой было использование нового поколения свёрточных блоков - Fused-MBConv, а также автоматической генерации архитектуры с помощью автоматического алгоритма поиска архитектуры NAS [9].

2. ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

2.1. Набор данных

Существует множество датасетов для разработки систем классификации изображений, как общего назначения, так и специализированных. Среди наиболее известных и широко используемых наборов данных общего назначения можно выделить ImageNet и CIFAR-10, содержащие миллионы размеченных изображений, охватывающих широкий спектр объектов повседневной жизни. Эти датасеты стали основой для разработки и тестирования базовых архитектур глубоких нейронных сетей, таких как VGG, ResNet и InceptionNet [28, 29].

Среди специализированных авиационных датасетов особое место занимает FGVC-Aircraft, включающий иерархически организированные метки для изображений: производитель, семейство, модель. Он подходит для задач тонкой (англ. fine-grained) классификации, однако изображения в этом датасете получены с наземных ракурсов, что существенно ограничивает их использование в задачах автоматического анализа аэрокосмических изображений [30].

Другим значимым набором данных является RarePlanes, включающий как реальные, так и синтетические изображения самолетов, полученные с орбитальных спутников. Несмотря на то, что он ориентирован преимущественно на задачи обнаружения и сегментации, изображения из данного набора данных могут содержать несколько воздушных судов, что затрудняет прямое использование в задачах классификации отдельных объектов [31].

Также заслуживает внимания Aircraft Context Dataset, содержащий изображения воздушных судов как в полёте, так и на земле с аннотациями, подходящими для многозадачного обучения. Однако он, подобно FGVC, не ориентирован на обработку спутниковых данных, так как его изображения получены преимущественно с наземных камер [32].

Таким образом, несмотря на наличие ряда наборов данных, пригодных для классификации воздушных судов, только набор данных Multi-Type Aircraft Remote Sensing Images (далее – MTARSI) был изначально спроектирован как датасет для решения задачи автоматической классификации авиамоделей на основе спутниковых изображений. Данный набор данных включает 9385 спутниковых изображений самолетов 20 различных типов, что делает его достаточно репрезентативным для задач автоматического распознавания воздушных судов. Изображения в MTARSI получены с различных спутниковых снимков, имеют сложные фоновые условия, варьируют по разрешению, освещению и времени съемки, что способствует созданию более устойчивых и обобщаемых моделей машинного обучения. Каждое изображение содержит ровно один самолет, а аннотации были выполнены семью экспертами в области интерпретации аэрокосмических данных. Пространственное разрешение изображений находится в диапазоне от 0,3 до 1,0 м, что обеспечивает высокую детализацию объектов [33].

Структура классов в наборе данных представлена в таблице 1 [33].

Таблица 1 – Структура классов в наборе данных MTARSI

|  |  |  |  |
| --- | --- | --- | --- |
| Тип воздушного судна | Количество изображений | Тип воздушного судна | Количество изображений |
| B-1 | 513 | C-130 | 763 |
| B-2 | 619 | C-135 | 526 |
| B-29 | 321 | C-17 | 480 |
| B-52 | 548 | C-5 | 499 |
| Boeing | 605 | E-3 | 452 |
| F-16 | 372 | F-22 | 846 |
| KC-10 | 554 | C-21 | 491 |
| U-2 | 362 | T-6 | 248 |
| P-63 | 305 | T-43 | 306 |

Как можно заметить из таблицы 1, в наборе данных присутствует классовый дисбаланс. Количество изображений в отдельных классах (например, F-22) в несколько раз превышает количество изображений в других. Это может привести к смещению модели в сторону переобучения на наиболее представленные категории.

На рисунке 9 представлены примеры изображений из набора данных MTARSI с подписями соответствующих им классам.  Рисунок 9 – Примеры изображений из набора данных MTARSI

Используя подпрограмму, код которой представлен в приложении А, был получен средний размер изображений в наборе данных - 218**×**206 пикселей (ширина × высота). Для обучения нейронных сетей и дальнейшего использования в приложении использовался приближенный к данному значению размер изображений - 200**×**200 пикселей.

Для обучения нейронных сетей набор данных был разбит в пропорции 60% от общего числа изображений для обучения, 15% для валидации и 25% для тестирования.

2.2. Реализация и обучение нейронных сетей

Для создания, компиляции, обучения и последующей оценки свёрточных нейронных сетей использовались библиотеки для языка программирования Python – TensorFlow и Keras.

Для предварительной обработки и загрузки изображений использовался класс ImageDataGenerator из модуля tensorflow.keras.preprocessing.image, позволяющий удобно осуществлять генерацию батчей данных из директории. Данные были предварительно структурированы в папки train, validation и test, каждая из которых содержала подкаталоги с изображениями соответствующих классов. Использование flow\_from\_directory() обеспечивало автоматическую разметку и построение батчей с одновременной трансформацией изображений до размера 200×200 пикселей и переводом меток классов в категориальный формат (англ. one-hot encoding). При этом нормализация пиксельных значений не применялась, так как модели обучались на предобученных весах ImageNet, где масштабирование данных производится внутри архитектур [34].

TensorFlow и Keras являются ключевыми инструментами в области глубокого обучения, широко используемыми для разработки и обучения нейронных сетей. TensorFlow, представленный в 2016 году, представляет собой систему машинного обучения, способную работать в различных масштабируемых средах, обеспечивая поддержку как обучения, так и вывода для глубоких нейронных сетей. Keras, в свою очередь, является высокоуровневым API, интегрированным в TensorFlow, который упрощает процесс создания и обучения моделей глубокого обучения [35, 36].

Для загрузки и компиляции моделей была разработана функция load\_and\_compile\_model, позволяющая загружать предобученные модели, такие как VGG16, ResNet50, EfficientNetV2S и другие, с исключением верхних слоев и заданной формы входных данных (200**×**200**×**3 – ширина **×** высота **×** количество каналов). После загрузки базовой модели ее веса фиксировались, предотвращая их обновление во время обучения. Затем к базовой модели добавлялись слои глобального усреднения для уменьшения размерности и полносвязный слой с функцией активации softmax для классификации на 20 классов. Модели компилировались с использованием оптимизатора Adam и функции потерь категориальной кросс-энтропии (англ. categoricalCrossentropy), которая соответствует задаче многоклассовой классификации.

Для обучения моделей была разработана функция train\_model. Процесс обучения состоял из 50 эпох, что обеспечило достаточное число итераций для сходимости модели. В качестве меры контроля за переобучением применялась ранняя остановка обучения в виде сохранения наилучшей версии модели (метод ModelCheckpoint), которая определялась минимальной функцией потерь на валидационном наборе данных. Также для повышения стабильности обучения использовался метод уменьшения скорости обучения (ReduceLROnPlateau), который снижал значение коэффициента скорости обучения (англ. learning rate) в 2 раза при отсутствии улучшений в течение 3 эпох [37, 38].

Для последующего анализа и оценки эффективности модели в ходе обучения сохранялись метрики точности первого, второго и третьего порядков (англ. top-1 accuracy, top-2 accuracy, top-3 accuracy).

После завершения обучения производилась финальная оценка модели на тестовом наборе данных с помощью функции evaluate\_model. Помимо метрик точности и функции потери (англ. loss) на тестовом наборе данных вычислялось среднее время инференса (англ. inference). Полученные результаты фиксировались в файл evaluation\_results.csv, что позволило произвести детальный сравнительный анализ производительности различных архитектур нейронных сетей.

Полный код программы для создания, компиляции, обучения и оценки моделей представлен в приложении Б.

3. АНАЛИЗ РЕЗУЛЬТАТОВ

3.1. Используемые метрики

Для анализа эффективности полученных моделей использовались следующие метрики: точность первого порядка, точность второго и третьего порядка, значение функции потерь на тестовой выборке, а также среднее время инференса одного изображения. Эти показатели позволили не только определить наилучшую модель среди исследуемых, но и провести комплексный анализ её производительности.

Точность первого порядка отражает долю изображений, для которых модель правильно определила класс с наибольшей вероятностью. Однако, учитывая высокую визуальную схожесть между отдельными типами воздушных судов, было целесообразно расширить анализ за счёт метрик точности второго и третьего порядков, учитывающих правильность предсказания среди двух и трёх наиболее вероятных предсказаний соответственно. Такой подход позволяет более гибко оценивать способность модели к обобщению в условиях близости классов, связанной с визуальной схожестью общих признаков воздушных судов.

Значение функции потерь, а именно функции категориальной перекрестной энтропии (англ. categorical crossentropy loss), использованной в качестве основной меры ошибки, характеризует расхождение между предсказанным распределением вероятностей и истинной меткой класса. Данная функция является стандартом для задач многоклассовой классификации и отражает уверенность модели в предсказаниях: чем ниже значение функции потерь, тем выше точность модели и её способность дифференцировать классы [39].

Дополнительно для оценки практической применимости моделей была использована метрика среднего времени инференса одного изображения, измеряемая в секундах. Под инференсом понимается процесс обработки входного изображения через обученную модель с целью получения предсказания — в данном случае, отнесения изображения к одному из классов воздушных судов.

3.2. Анализ результатов

На основе метрик, зафиксированных в ходе обучения моделей, были построены графики зависимости точности предсказаний на валидационном наборе данных от количества эпох (рисунок 10), а также графики изменения значения функции потерь (loss) на тех же данных в процессе обучения моделей (рисунок 11).

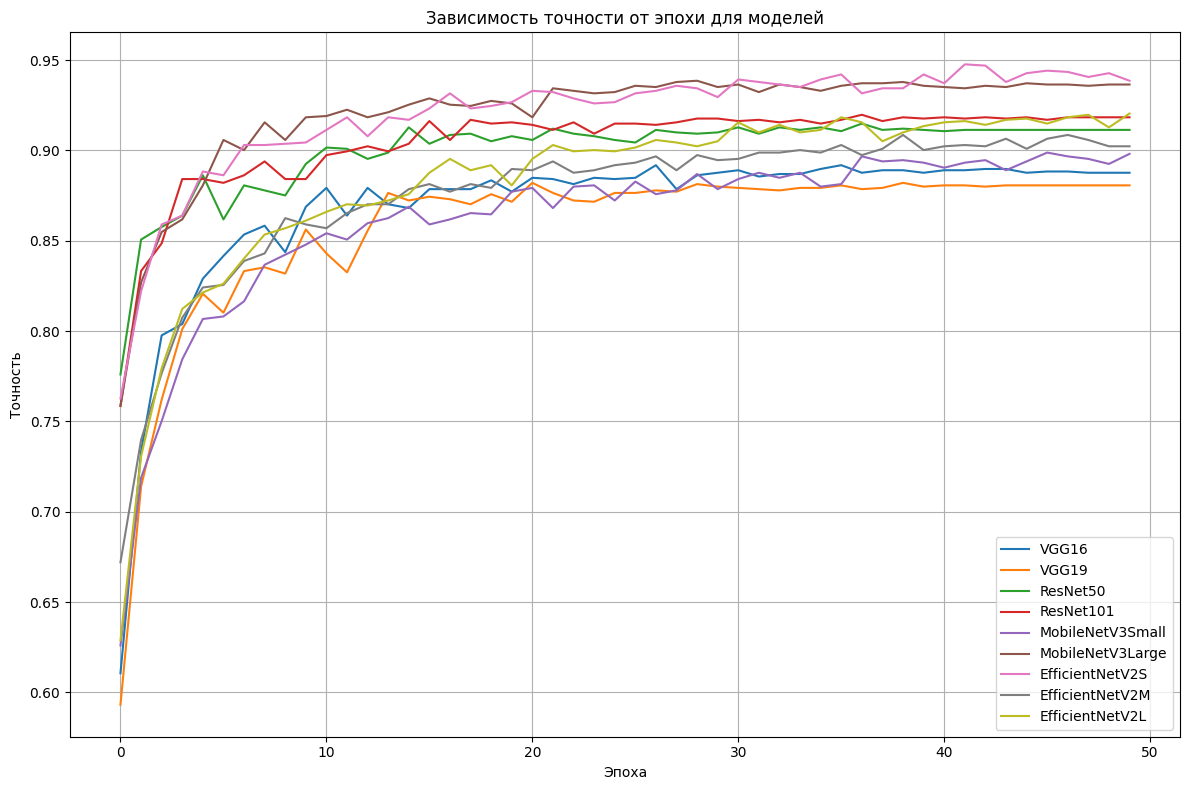


Рисунок 10 – Графики зависимости точности от эпохи

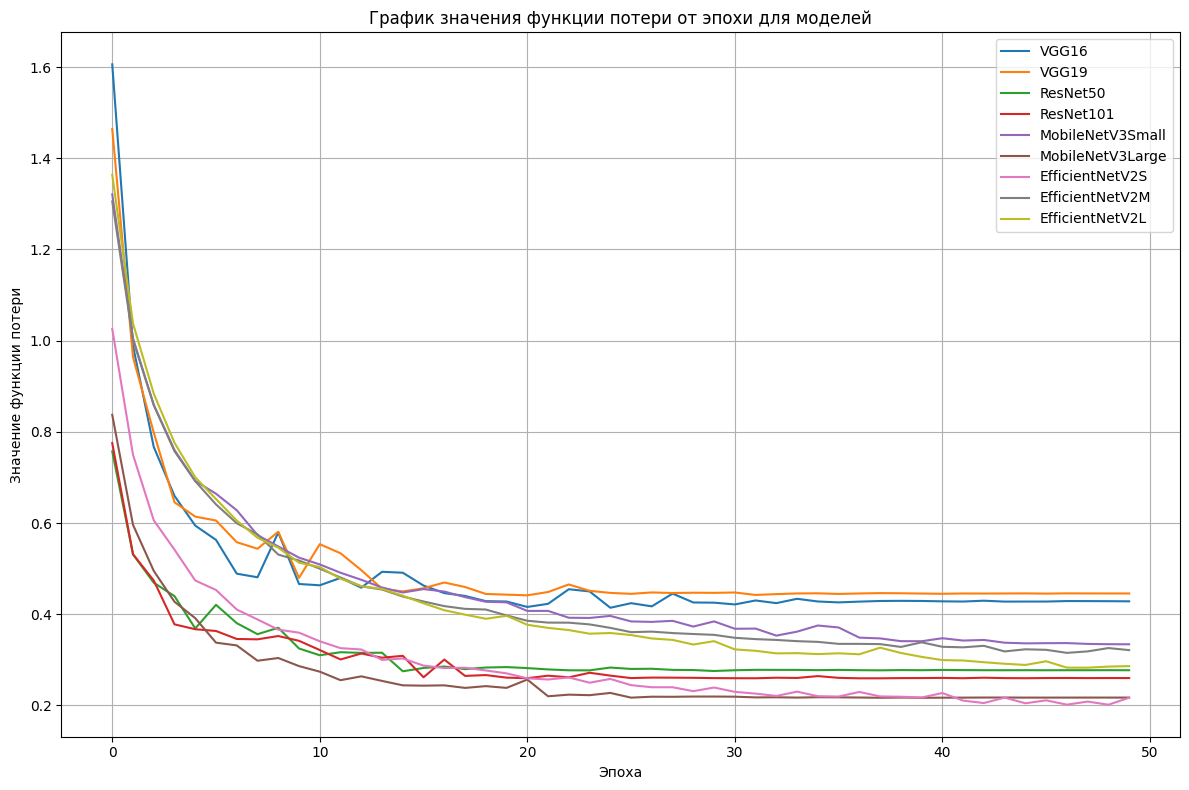


Рисунок 11 – Графики зависимости значения функции потерь от эпохи

Из графиков на рисунках 12 и 13 можно сделать ряд наблюдений относительно динамики обучения различных моделей. На графике зависимости точности от количества эпох видно, что большинство моделей демонстрируют быстрый рост точности на начальных этапах обучения, после чего темп улучшений замедляется. Архитектуры EfficientNetV2S и EfficientNetV2M, показывают более стабильное повышение точности и достигают лучших результатов по сравнению с другими моделями.

Анализ графиков зависимости функции потерь подтверждает данную тенденцию. Для всех моделей наблюдается значительное снижение значения loss на первых эпохах, после чего значения функции потерь выходят на плато. При этом можно отметить, что модели семейства EfficientNet демонстрируют более плавное уменьшение loss и достигают меньших финальных значений функции потерь, что может свидетельствовать о лучшем качестве обучения. В то же время модели, такие как VGG16 и VGG19, демонстрируют более выраженные колебания, что может указывать на менее стабильный процесс обучения.

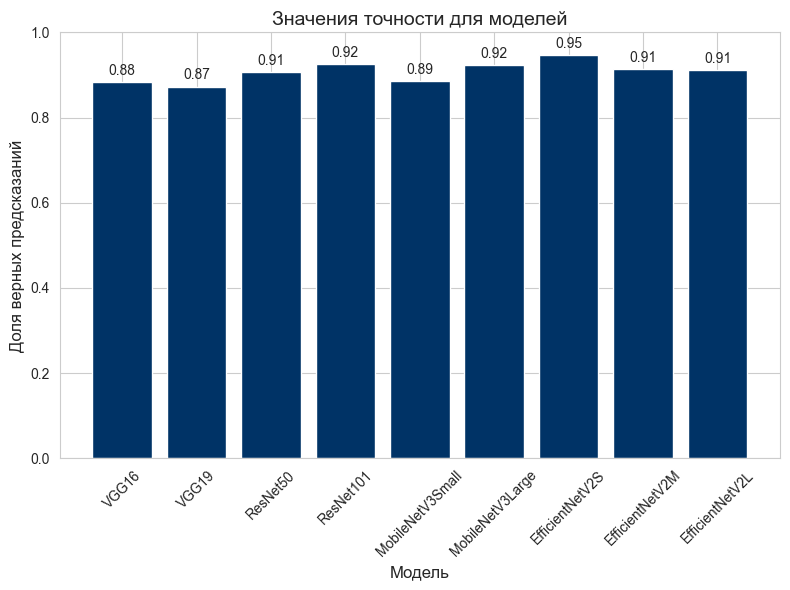
На основе данных, полученных из оценки моделей на тестовом наборе данных, были построены графики, отображающие ключевые метрики, включая точность предсказаний моделей (рисунок 12), значение функции потерь (рисунок 13), а также среднее время обработки одного изображения (рисунок 14). 

Рисунок 12 – График значений точности для каждой модели

Из анализа графика точности следует, что наивысшие значения метрики точности достигаются моделями ResNet101, MobileNetV3Large и EfficientNetV2S. При этом наилучший результат продемонстрировала модель EfficientNetV2S, обеспечив корректную классификацию 95% изображений тестовой выборки. Модели ResNet101 и MobileNetV3Large показали немного более низкие значения — около 92%, однако их точность также характеризуется как высокая. Остальные модели продемонстрировали менее высокие, но все же приемлемые результаты. Так, точность моделей ResNet50, EfficientNetV2M и EfficientNetV2L составила 91%, что также свидетельствует об их пригодности для решения задачи. Модели MobileNetV3Small, VGG16 и VGG19 уступают лидерам по точности, достигая значений 89%, 88% и 87% соответственно, что делает их менее предпочтительными.

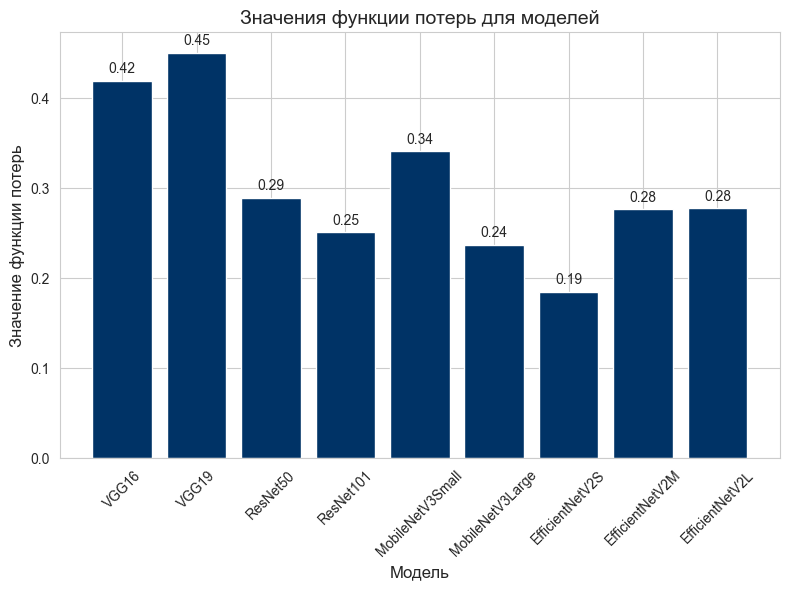


Рисунок 13 – График значений функции потерь для каждой модели

График значений функции потерь подтверждает выводы, сделанные на основе анализа точности. Наименьшее значение функции потерь продемонстрировала модель EfficientNetV2S — 0.19, что свидетельствует о высокой уверенности модели в собственных предсказаниях. Несколько более высокие, но также низкие значения наблюдаются у моделей MobileNetV3Large (0.24) и ResNet101 (0.25). Модели EfficientNetV2M, EfficientNetV2L и ResNet50 занимают промежуточные позиции, демонстрируя значения функции потерь в пределах 0.28–0.29. В то же время MobileNetV3Small, несмотря на приемлемую точность, характеризуется более высоким значением функции потерь (0.34), что может указывать на меньшую уверенность в классификации. Наиболее неудачные результаты по этому показателю показали модели VGG16 и VGG19 с функцией потерь 0.42 и 0.45 соответственно, что коррелирует с их низкой точностью и указывает на слабую способность к генерализации.

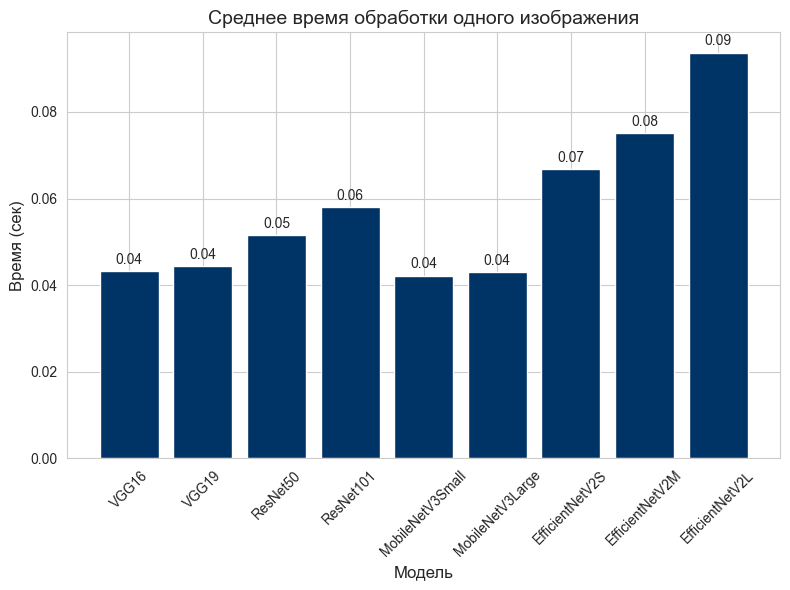


Рисунок 14 – График значений среднего времени обработки одного изображения для каждой модели

На графике, отражающем среднее время обработки одного изображения, видно, что наибольшие затраты по времени демонстрирует модель EfficientNetV2L — 0.09 секунды, что объясняется её высокой сложностью и большим числом параметров. За ней следует EfficientNetV2M (0.08 с) и EfficientNetV2S (0.07 с), которые также требуют относительно больше времени на обработку изображения по сравнению с более простыми архитектурами. Модели ResNet101 и ResNet50 показывают умеренное время обработки — 0.06 и 0.05 секунды соответственно. Наименьшее значение среднего времени инференса зафиксировано у моделей VGG16, VGG19, MobileNetV3Small и MobileNetV3Large — все они обрабатывают изображение в среднем за 0.04 секунды.

Учитывая результаты сравнительного анализа, дальнейшее рассмотрение сводится к подробной оценке модели EfficientNetV2S, как показавшей наилучшие значения точности, значения функции потерь и времени инференса.

Для модели EfficientNetV2S была построена столбчатая диаграмма топ-1, топ-2 и топ-3 точности, отражающая доли верных предсказаний, полученных на тестовой выборке при учёте одного, двух и трёх лучших вариантов. Полученная диаграмма представлена на рисунке 15.

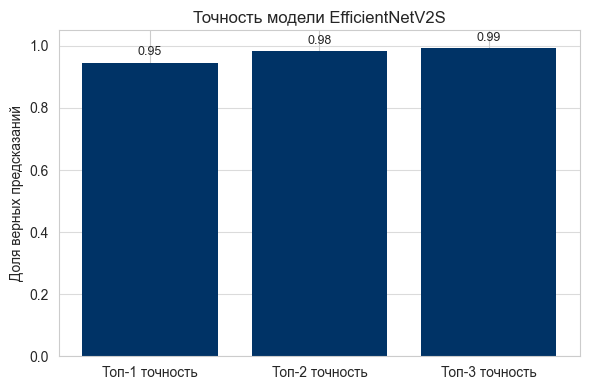


Рисунок 15 - Топ-1, топ-2 и топ-3 точность для модели EfficientNetV2S

На представленном графике видно, что при учёте только самого вероятного класса (топ-1) модель достигает точности около 95%. При расширении диапазона до двух наиболее вероятных вариантов (топ-2) точность возрастает примерно до 98%, а при учёте трёх вариантов (топ-3) приближается к 99%. Такой рост показателей свидетельствует о высоком уровне обобщающей способности модели EfficientNetV2S и указывает на то, что среди первых трёх предложенных вариантов она практически всегда включает правильный класс.

Была сформирована матрица ошибок для модели EfficientNetV2S по предсказаниям, полученным на основе тестового набора данных, показывающая распределение корректных и ошибочных классификаций по каждому из 20 классов воздушных судов. Полученная матрица ошибок представлена на рисунке 16.

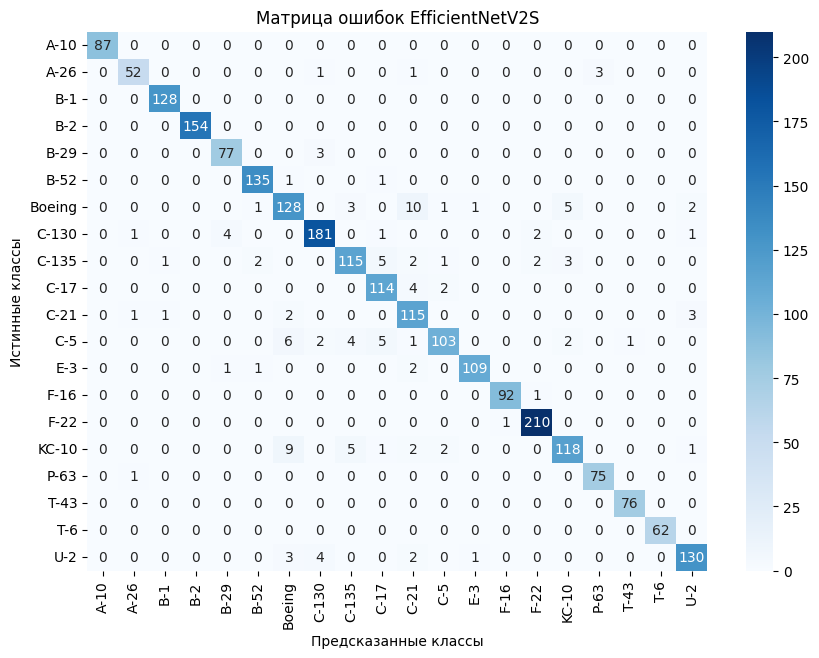


Рисунок 16 – Матрица ошибок для модели EfficientNetV2S

На диагонали матрицы наблюдаются высокие значения, что свидетельствует о большом числе правильных предсказаний: например, F-22, B-2 и C-130 были правильно классифицированы в 210, 154 и 181 случаях соответственно. Ошибки классификации преимущественно сосредоточены между визуально близкими типами, такими как C-21 и Boeing (10 ошибок), и KC-10 и Boeing (9 ошибок), что указывает на пересечения признаков между этими классами и их внешнее сходство. Отсутствие значимых ошибок между существенно различающимися классами подтверждает высокую обобщающую способность модели. Таким образом, матрица ошибок демонстрирует устойчивую и надёжную работу модели EfficientNetV2S на уровне индивидуальных классов.

Также для модели EfficientNetV2S был построен график средних значений вероятности, которую модель присваивает каждому классу воздушного судна при верном и неверном предсказании. График был построен на основе предсказаний модели на тестовом наборе данных. Полученный график представлен на рисунке 17.



Рисунок 17 – График средних значений вероятности принадлежности классу

На полученном графике для всех классов наблюдается чёткое разделение: при правильной классификации средняя предсказанная вероятность оказывается значительно выше (около 0.9 и более), в то время как при ошибках она существенно ниже (около 0.2-0.3). Полученные результаты свидетельствует о высокой уверенности модели в верных предсказаниях и указывает на способность модели дифференцировать классы с минимальным уровнем «ложной» уверенности. Общая тенденция высокой средней вероятности для верных классификаций и низкой — для неверных говорит о хорошей калибровке сети и указывает на её надёжную работу в задаче классификации.

На основании проведённого сравнительного анализа моделей свёрточных нейронных сетей и более детального изучения результатов, полученных с использованием архитектуры EfficientNetV2S, можно заключить, что данная модель демонстрирует высокие показатели эффективности в рамках датасета MTARSI. Однако необходимо подчеркнуть, что такие результаты получены исключительно на этом наборе данных, и для подтверждения устойчивости и обобщающей способности модели требуется дополнительное тестирование на других выборках изображений.

4. ПРИМЕНЕНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ

4.1. Потенциальные применения системы

Разработанная система классификации воздушных судов может функционировать как автономное приложение, так и как компонент более крупной информационно-аналитической платформы. Как самостоятельный продукт система может быть использована специалистами в области геоинформационных систем, аналитиками разведывательных служб, а также операторами центров мониторинга в гражданской авиации. Например, сотрудники ситуационных центров аэропортов или региональных диспетчерских пунктов могут применять приложение для оперативного анализа спутниковых или аэрофотоснимков, получаемых из внешних источников, с целью определения типа и принадлежности обнаруженного воздушного судна.

Система классификации может быть также интегрирована в автоматизированные комплексы обработки спутниковых данных для мониторинга воздушной активности вблизи охраняемых или стратегически важных объектов. Возможные военные применения включают наблюдение за передвижением авиационных единиц, а также поддержку разведывательных и аналитических задач. В ряде публикаций, подчёркивается, что точная классификация типов воздушных судов может служить ценным источником разведывательной информации и повысить эффективность работы аналитиков, обрабатывающих аэрокосмические данные [40, 41].

Разработанная система также может найти применение в рамках гуманитарных миссий и ликвидации последствий катастроф, включая поиск потерпевших крушение воздушных судов. Возможность точной идентификации типа самолёта на спутниковом снимке повышает оперативность и релевантность отклика со стороны аварийно-спасательных служб, особенно в условиях ограниченного доступа к зоне происшествия. Аналогично, в рамках задач экологического мониторинга, идентификация воздушных судов может использоваться для анализа загрязнений, вызванных авиацией, и отслеживания активности вблизи охраняемых природных территорий.

4.2. Демонстрационное приложение

Для наглядной демонстрации работы разработанной системы была разработана демонстрационная версия приложения, позволяющая пользователю взаимодействовать с моделью классификации через удобный графический интерфейс.

Приложение было создано с использованием языка программирования Python и библиотеки PyQt5, основанной на кроссплатформенной библиотеке Qt. Данная библиотека дает возможность создавать графические интерфейсы используя виджеты (англ. widgets). Виджеты в PyQt5 представляют собой базовые строительные блоки интерфейса пользователя. Они могут быть кнопками, текстовыми полями, метками, окнами и другими элементами управления, которые пользователь видит на экране и с которыми взаимодействует.

СПИСОК ЛИТЕРАТУРЫ

1. Сикорский О.С. Обзор свёрточных нейронных сетей для задачи классификации изображений // Новые информационные технологии в автоматизированных системах. 2017. № 20. C. 37–42.

2. Sanghvi K. [и др.]. A Survey on Image Classification Techniques // SSRN Electronic Journal. 2021.

3. VOSviewer - Visualizing scientific landscapes // VOSviewer [Электронный ресурс]. URL: https://www.vosviewer.com// (дата обращения: 25.03.2025).

4. Jain A. [и др.]. Exploring New Convolutional Neural Network Architectures for Satellite Image Categorization: A Comparative Study 2024.C. 73–77.

5. Saputra W. S. J. [и др.]. Car Classification Based on Image Using Transfer Learning Convolutional Neural Network 2022.C. 324–327.

6. Li Y. [и др.]. Aircraft Detection in Remote Sensing Images Based on Deep Convolutional Neural Network 2017.C. 942–945.

7. Маршалко Д.В., Кубанских О.В. Архитектура свёрточных нейронных сетей // Ученые записки Брянского государственного университета. 2019. № 4 (16). C. 10–13.

8. Lecun Y. [и др.]. Gradient-based learning applied to document recognition // Proceedings of the IEEE. 1998. № 11 (86). C. 2278–2324.

9. Tan M., Le Q. V. EfficientNetV2: Smaller Models and Faster Training // 2021.

10. Самое главное о нейронных сетях. Лекция в Яндексе // Хабр [Электронный ресурс]. URL: https://habr.com/ru/companies/yandex/articles/307260/ (дата обращения: 25.03.2025).

11. Parhi R., Nowak R. D. The Role of Neural Network Activation Functions // IEEE Signal Processing Letters. 2020. (27). C. 1779–1783.

12. Zafar A. [и др.]. A Comparison of Pooling Methods for Convolutional Neural Networks // Applied Sciences. 2022. № 17 (12). C. 8643.

13. Введение в архитектуры нейронных сетей / Хабр [Электронный ресурс]. URL: https://habr.com/ru/companies/oleg-bunin/articles/340184/?hl=ru\_RU&fl=ru (дата обращения: 25.03.2025).

14. Нгуен К. Т, Боровик И. Г Сравнение функций активации softmax и сигмоида в контексте многоклассовой классификации распознавания объектов // Международный научно-исследовательский журнал. 2024. № 6 (144).

15. Bot N. CNN Introduction // Lukewarm Takes [Электронный ресурс]. URL: https://lukewarmtakes.net/2021/09/17/cnn-introduction/ (дата обращения: 25.03.2025).

16. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks // Communications of the ACM. 2017. № 6 (60). C. 84–90.

17. Федотов И.с Трансферное обучение в компьютерном зрении: применение и перспективы // Вестник науки. 2023. № 9 (66) (1). C. 135–137.

18. Russakovsky O. [и др.]. ImageNet Large Scale Visual Recognition Challenge // 2015.

19. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // 2015.

20. Siddharth T., Kirar B. S., Agrawal D. K. Plant Species Classification Using Transfer Learning by Pretrained Classifier VGG-19 // 2022.

21. Apostolopoulos I. D., Tzani M. Industrial object, machine part and defect recognition towards fully automated industrial monitoring employing deep learning. The case of multilevel VGG19 // Journal of Ambient Intelligence and Humanized Computing. 2023. № 8 (14). C. 10263–10276.

22. He K. [и др.]. Deep Residual Learning for Image Recognition // 2015.

23. Howard A. G. [и др.]. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // 2017.

24. Sandler M. [и др.]. MobileNetV2: Inverted Residuals and Linear Bottlenecks // 2019.

25. Howard A. [и др.]. Searching for MobileNetV3 // 2019.

26. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // 2020.

27. Gong Y. [и др.]. A survey on dataset quality in machine learning // Information and Software Technology. 2023. (162). C. 107268.

28. Deng J. [и др.]. ImageNet: A large-scale hierarchical image database 2009.C. 248–255.

29. CIFAR-10 and CIFAR-100 datasets [Электронный ресурс]. URL: https://www.cs.toronto.edu/~kriz/cifar.html (дата обращения: 25.03.2025).

30. Maji S. [и др.]. Fine-Grained Visual Classification of Aircraft // 2013.

31. Shermeyer J. [и др.]. RarePlanes: Synthetic Data Takes Flight // 2020.

32. Steininger D. [и др.]. The Aircraft Context Dataset: Understanding and Optimizing Data Variability in Aerial Domains 2021.C. 3816–3825.

33. Wu Z.-Z. [и др.]. A benchmark data set for aircraft type recognition from remote sensing images // Applied Soft Computing. 2020. (89). C. 106132.

34. tf.keras.preprocessing.image.ImageDataGenerator | TensorFlow v2.16.1 // TensorFlow [Электронный ресурс]. URL: https://www.tensorflow.org/api\_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (дата обращения: 25.03.2025).

35. Abadi M. [и др.]. TensorFlow: A system for large-scale machine learning // 2016.

36. Павлов Е. М., Рыжов А. В. Применение нейронных сетей для распознавания образов // Бюллетень науки и практики. 2023. № 12 (9). C. 52–58.

37. Team K. Keras documentation: ModelCheckpoint [Электронный ресурс]. URL: https://keras.io/api/callbacks/model\_checkpoint/ (дата обращения: 25.03.2025).

38. Team K. Keras documentation: ReduceLROnPlateau [Электронный ресурс]. URL: https://keras.io/api/callbacks/reduce\_lr\_on\_plateau/ (дата обращения: 25.03.2025).

39. Реализация функции потерь в Python // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/736536/ (дата обращения: 07.04.2025).

40. Youssef Y. B. [и др.]. Aircraft Type Classification in Remote Sensing Images using Deep Learning 2020.C. 1–6.

41. Azam F. [и др.]. Aircraft Classification Based on PCA and Feature Fusion Techniques in Convolutional Neural Network // IEEE Access. 2021. (9). C. 161683–161694.

ПРИЛОЖЕНИЕ А

Код программы для получения среднего размера изображений в наборе данных

def calculate\_average\_image\_size(dataset\_path):

    total\_width, total\_height, image\_count = 0, 0, 0

    for class\_folder in os.listdir(dataset\_path):

        class\_path = os.path.join(dataset\_path, class\_folder)

        if os.path.isdir(class\_path):

            for image\_name in os.listdir(class\_path):

                image\_path = os.path.join(class\_path, image\_name)

                try:

                    with Image.open(image\_path) as img:

                        width, height = img.size

                        total\_width += width

                        total\_height += height

                        image\_count += 1

                except Exception as e:

                    print(f"Ошибка при обработке файла {image\_path}: {e}")

    if image\_count == 0:

        return (0, 0)  # Нет изображений

    average\_width = total\_width / image\_count

    average\_height = total\_height / image\_count

    return average\_width, average\_height

# Путь к папке

dataset\_path = "MTARSI\_Dataset"

average\_size = calculate\_average\_image\_size(dataset\_path)

print(f"Средний размер изображений: {average\_size[0]:.2f}x{average\_size[1]:.2f}")

ПРИЛОЖЕНИЕ Б

Код программы для создания, компиляции, обучения и оценки моделей

TARGET\_SIZE = (200, 200)

# Генератор без нормализации

datagen\_default = ImageDataGenerator()

# Генераторы для стандартной загрузки данных

train\_generator = datagen\_default.flow\_from\_directory(

    directory=r"processed\_dataset/train",

    target\_size=TARGET\_SIZE,

    color\_mode="rgb",

    batch\_size=16,

    class\_mode="categorical",

    seed=42

)

validation\_generator = datagen\_default.flow\_from\_directory(

    directory=r"processed\_dataset/validation",

    target\_size=TARGET\_SIZE,

    color\_mode="rgb",

    batch\_size=16,

    class\_mode="categorical",

    seed=42

)

test\_generator = datagen\_default.flow\_from\_directory(

    directory=r"processed\_dataset/test",

    target\_size=TARGET\_SIZE,

    color\_mode="rgb",

    batch\_size=1,

    class\_mode="categorical",

    shuffle=False,

    seed=42

)

# Функция загрузки и компиляции модели

def load\_and\_compile\_model(model\_name, input\_shape=(200, 200, 3), num\_classes=20):

    if model\_name == 'VGG16':

        base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'VGG19':

        base\_model = VGG19(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'ResNet50':

        base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'ResNet101':

        base\_model = ResNet101(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'MobileNetV3Small':

        base\_model = MobileNetV3Small(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'MobileNetV3Large':

        base\_model = MobileNetV3Large(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'EfficientNetV2S':

        base\_model = EfficientNetV2S(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'EfficientNetV2M':

        base\_model = EfficientNetV2M(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    elif model\_name == 'EfficientNetV2L':

        base\_model = EfficientNetV2L(weights='imagenet', include\_top=False, input\_shape=input\_shape)

    base\_model.trainable = False

    model = tf.keras.Sequential([

        base\_model,

        tf.keras.layers.GlobalAveragePooling2D(),

        tf.keras.layers.Dense(num\_classes, activation='softmax')

    ])

    # Компиляция модели с метриками Top-1, Top-2 и Top-3 Accuracy

    model.compile(optimizer=Adam(),

                  loss=CategoricalCrossentropy(),

                  metrics=[

                      tf.keras.metrics.TopKCategoricalAccuracy(k=1, name='top\_1\_accuracy'),

                      tf.keras.metrics.TopKCategoricalAccuracy(k=2, name='top\_2\_accuracy'),

                      tf.keras.metrics.TopKCategoricalAccuracy(k=3, name='top\_3\_accuracy')

                  ])

    return model

# Функция обучения модели

def train\_model(model, train\_generator, validation\_generator, model\_name):

    csv\_logger = CSVLogger(f'Training\_logs\{model\_name}\_imagenet\_training\_log.csv', append=True)

    checkpoint = ModelCheckpoint(

        filepath=f'{model\_name}\_best\_model.h5',

        monitor='val\_loss',

        save\_best\_only=True,

        mode='min',

        verbose=1

    )

    reduce\_lr = ReduceLROnPlateau(

        monitor='val\_loss',

        factor=0.5,

        patience=3,

        verbose=1,

        min\_delta=1e-4,

        cooldown=2,

        min\_lr=1e-7

    )

    history = model.fit(

        train\_generator,

        epochs=50,

        validation\_data=validation\_generator,

        callbacks=[csv\_logger, checkpoint, reduce\_lr]

    )

    return history

# Функция для вычисления метрик

def evaluate\_model(model, test\_generator, model\_name):

    # Рассчитываем среднее время инференса

    inference\_times = []

    for \_ in range(len(test\_generator)):

        start\_time = time.time()

        X\_test\_batch, \_ = test\_generator.next()  # Получаем следующий батч

        model.predict(X\_test\_batch, verbose=0)

        inference\_times.append(time.time() - start\_time)

    avg\_inference\_time = np.mean(inference\_times)

    # Оценка модели и извлечение метрик

    results = model.evaluate(test\_generator, verbose=0)

    metric\_names = model.metrics\_names

    loss = results[0]

    top\_1 = results[metric\_names.index('top\_1\_accuracy')]

    top\_2 = results[metric\_names.index('top\_2\_accuracy')]

    top\_3 = results[metric\_names.index('top\_3\_accuracy')]

    return {

        'Model': model\_name,

        'Loss': loss,

        'Top-1 Accuracy': top\_1,

        'Top-2 Accuracy': top\_2,

        'Top-3 Accuracy': top\_3,

        'Avg Inference Time (s)': avg\_inference\_time

    }

# Основной процесс

def main():

    # Список используемых моделей

    models = [

        'VGG16', 'VGG19', 'ResNet50', 'ResNet101',

        'MobileNetV3Small', 'MobileNetV3Large',

        'EfficientNetV2S', 'EfficientNetV2M', 'EfficientNetV2L'

    ]

    csv\_filename = 'evaluation\_results.csv'

    for model\_name in models:

        print(f"Training {model\_name}")

        model = load\_and\_compile\_model(model\_name)

        train\_model(model, train\_generator, validation\_generator, model\_name)

        # Загрузка лучшей версии модели

        best\_model = tf.keras.models.load\_model(f'{model\_name}\_best\_model.h5')

        # Оценка модели

        print(f"Evaluating {model\_name}")

        metrics = evaluate\_model(best\_model, test\_generator, model\_name)

        # Вывод результатов в консоль

        for key, value in metrics.items():

            print(f"{key}: {value}")

        # Сохранение результатов после каждой модели (результаты добавляются, если файл существует)

        metrics\_df = pd.DataFrame([metrics])

        if not os.path.isfile(csv\_filename):

            metrics\_df.to\_csv(csv\_filename, index=False, mode='w')

        else:

            metrics\_df.to\_csv(csv\_filename, index=False, header=False, mode='a')

        print(f"Results for {model\_name} saved to {csv\_filename}\n")

# Запуск основного процесса

main()