Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Кафедра системного программирования ИСП РАН
Лаборатория (laboratory name)

Выпускная квалификационная работа бакалавра

Изучение вариантов обеспечения устойчивости ResNet-подобных моделей компьютерного зрения к небольшим сдвигам входного изображения

Автор:

Студент

Полев Алексей Михайлович

Научный руководитель:

к.ф.-м.н.

Самосюк Алексей Владимирович

Москва 2025

АННОТАЦИЯ

В данной работе проведено исследование проблемы пространственной инвариантности (shift-invariance) в сверточных нейронных сетях (CNN) и её влияния на стабильность работы алгоритмов компьютерного зрения. Особое внимание уделено артефактам, возникающим при субпиксельных сдвигах входных изображений, которые могут приводить к значительным изменениям в выходных данных модели, снижая надежность систем классификации и детекции объектов.

В теоретической части работы формализована проблема отсутствия полной инвариантности к сдвигам в современных CNN-архитектурах, проанализированы фундаментальные причины этого явления, связанные с операциями субдискретизации (даунсэмплинга) и нарушением теоремы Найквиста-Шеннона. Предложена математическая модель, описывающая возникновение эффекта алиасинга при операциях пулинга и страйдинга, и рассмотрены существующие подходы к его устранению.

Экспериментальная часть исследования сфокусирована на сравнительном анализе стандартных архитектур (ResNet-50, VGG-16, YOLOv5) и их модифицированных версий с различными методами обеспечения инвариантности к сдвигам. Разработана методология тестирования, включающая генерацию последовательностей изображений с контролируемыми субпиксельными сдвигами объектов и комплексную систему метрик для количественной оценки стабильности. Предложена и реализована собственная метрика для измерения вариативности выходных данных моделей при малых изменениях входа.

Результаты экспериментов демонстрируют, что стандартные CNN-архитектуры проявляют значительную нестабильность (до 30% вариации в оценке вероятности класса и до 25% в точности детекции) даже при субпиксельных сдвигах входных данных. Применение методов анти-алиасинга (BlurPool) снижает вариативность на 40-60%, а разработанная в рамках исследования техника полифазной выборки (TIPS) позволяет сократить эффект вариативности на 85-95% при увеличении вычислительной сложности всего на 5-8%.)

На основе проведенного исследования сформулированы практические рекомендации по выбору архитектур и методов обеспечения инвариантности к сдвигам для различных задач компьютерного зрения, что особенно важно для критических приложений, где стабильность и предсказуемость работы моделей имеют первостепенное значение.

Ключевые слова: сверточные нейронные сети, пространственная инвариантность, shift-invariance, анти-алиасинг, BlurPool, TIPS, полифазная выборка, компьютерное зрение, YOLOv5.

СОДЕРЖАНИЕ

Содержание

Список сокращений и обозначений

- **CNN** — сверточная нейронная сеть (Convolutional Neural Network)
- **IoU** — метрика пересечения над объединением (Intersection over Union)
- **TIPS** — полифазная выборка с инвариантностью к сдвигам (Translation Invariant Polyphase Sampling)
- **FPS** — кадров в секунду (Frames Per Second)
- **MSB** — максимальное смещение выборки (Maximum-Sampling Bias)
- **AA-VGG16** — VGG16 с анти-алиасингом (Anti-Aliased VGG16)
- **AA-ResNet50** — ResNet50 с анти-алиасингом (Anti-Aliased ResNet50)
- **AA-YOLOv5** — YOLOv5 с анти-алиасингом (Anti-Aliased YOLOv5)
- **YOLO** — объектный детектор «вы смотрите только один раз» (You Only Look Once)
- **R-CNN** — регионная сверточная нейронная сеть (Region-based Convolutional Neural Network)
- **SSD** — детектор с одним проходом (Single Shot Detector)
- **PANet** — сеть агрегации путей (Path Aggregation Network)
- **FPN** — сеть пирамиды признаков (Feature Pyramid Network)
- **CSPDarknet** — базовая сеть YOLOv5 (Cross Stage Partial Darknet)
- **TDF** — функция расхождения трансляций (Translation Discrepancy Function)

Введение

Актуальность проблемы

Сверточные нейронные сети (CNN) сегодня являются ключевым инструментом в решении широкого спектра задач компьютерного зрения, включая классификацию изображений [**?**, **?**], сегментацию [**?**], детекцию объектов [**?**, **?**] и другие. Их популярность и эффективность обусловлены способностью к автоматическому извлечению иерархии признаков из необработанных данных и высокой точностью работы в различных условиях. Теоретические основы CNN предполагают, что они должны обладать свойством инвариантности к пространственным преобразованиям, в частности, к сдвигам входных данных [**?**]. Это означает, что одинаковые объекты, расположенные в разных частях изображения, должны распознаваться с одинаковой точностью и уверенностью.

Однако практика показывает, что современные архитектуры CNN не обладают полной инвариантностью к сдвигам [**?**, 1]. Небольшие, даже субпиксельные смещения объектов на входном изображении могут приводить к значительным изменениям в выходных результатах сети. Эта проблема, часто упускаемая из виду при традиционной оценке моделей на тестовых выборках, может иметь серьезные последствия в реальных приложениях компьютерного зрения, особенно в критически важных областях, таких как автономные транспортные средства, системы видеонаблюдения, медицинская диагностика и робототехника.

Отсутствие стабильности предсказаний при малых смещениях объектов может привести к:

- Ложным срабатываниям или пропускам в системах обнаружения объектов
- Нестабильной работе алгоритмов слежения за объектами
- Некорректной сегментации медицинских изображений
- Ошибкам в системах управления роботами и беспилотными автомобилями
- Снижению надежности систем биометрической идентификации

Причины нарушения инвариантности к сдвигам в CNN связаны с операциями субдискретизации (даунсэмплинга), такими как max-pooling и свёртка с шагом (stride) больше единицы [**?**]. Эти операции позволяют уменьшать пространственное разрешение карт признаков, что необходимо для снижения вычислительной сложности и обобщающей способности сети, но одновременно вносят пространственную зависимость, делая сеть чувствительной к точному положению входных паттернов. С точки зрения теории сигналов, эта проблема связана с нарушением теоремы Найквиста-Шеннона при дискретизации, что приводит к эффекту алиасинга [**?**].

В последние годы было предложено несколько подходов к решению проблемы пространственной вариативности CNN, включая методы антиалиасинга (например, BlurPool [**?**]), полифазную выборку с инвариантностью к сдвигам (TIPS [2]) и различные модификации архитектур [**?**]. Однако систематическое исследование влияния этих методов на стабильность работы различных типов CNN в контексте разных задач компьютерного зрения остается актуальной проблемой.

Данная работа направлена на всестороннее исследование артефактов пространственной инвариантности в современных CNN-архитектурах, анализ их влияния на производительность моделей и оценку эффективности различных методов повышения устойчивости к пространственным сдвигам. Особое внимание уделяется сравнению поведения классификационных моделей и моделей детекции объектов, таких как YOLO, при субпиксельных сдвигах входных данных, что позволяет выявить специфические проблемы и предложить целевые решения для различных типов архитектур. В рамках данного исследования впервые реализован метод полифазной выборки с инвариантностью к сдвигам (TIPS) для моделей семейства YOLO, который продемонстрировал значительно более высокую стабильность результатов детекции при пространственных сдвигах по сравнению с классической версией этой же модели.

1 Обзор предметной области

В данной главе представлен обзор предметной области, связанной с проблемой инвариантности к сдвигам в сверточных нейронных сетях. Рассматриваются ключевые аспекты проблемы, существующие методы оценки и улучшения инвариантности, а также особенности проявления проблемы в различных типах архитектур. Особое внимание уделено современным подходам к антиалиасингу в нейронных сетях и специфике проблемы в контексте детекторов объектов.

1.1 Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) представляют собой класс глубоких нейронных сетей, специально разработанных для эффективной обработки данных с сеточной структурой, таких как изображения [**?**]. В отличие от традиционных полносвязных нейронных сетей, CNN используют операцию свёртки, что значительно уменьшает количество параметров и вычислительную сложность, сохраняя при этом способность к обучению сложным пространственным паттернам.

1.1.1 Операция свёртки

Ключевым элементом CNN является операция свёртки, которая заключается в применении фильтра (или ядра) к входным данным. Математически дискретная двумерная свёртка может быть выражена следующим образом:

$$(I * K)(i\,j) = \sum_m \sum_n I(i + m\,j + n) \cdot K(m\,n) \tag{1}$$

где I — входное изображение, K — ядро свёртки, а $*$ — операция свёртки. В контексте глубоких нейронных сетей применяется кросс-корреляция:

$$(I * K)(i\,j) = \sum_m \sum_n I(i - m\,j - n) \cdot K(m\,n) \tag{2}$$

Рис. 1: Иллюстрация операции свёртки в CNN

Ядро свёртки «скользит» по входному изображению, вычисляя взвешенную сумму значений пикселей под ним для каждой позиции. Это позволяет нейронной сети обнаруживать различные признаки изображения, такие как края, текстуры и более сложные паттерны на более высоких уровнях абстракции.

1.1.2 Архитектура сверточных нейронных сетей

Типичная архитектура CNN состоит из нескольких ключевых компонентов:

- **Сверточные слои** — основные строительные блоки, применяющие операцию свёртки для извлечения признаков изображения.
- **Слои субдискретизации (пулинга)** — уменьшают пространственную размерность данных, сохраняя наиболее важную информацию. Наиболее распространены max-pooling (выбор максимального значения в окне) и average-pooling (усреднение значений в окне).
- **Слои активации** — применяют нелинейные функции к выходам сверточных слоев, обычно ReLU (Rectified Linear Unit), что позволяет сети моделировать сложные нелинейные зависимости.
- **Полносвязные слои** — обычно располагаются в конце сети и используются для классификации на основе извлеченных признаков.
- **Слои нормализации** — стабилизируют и ускоряют процесс обучения (например, Batch Normalization).

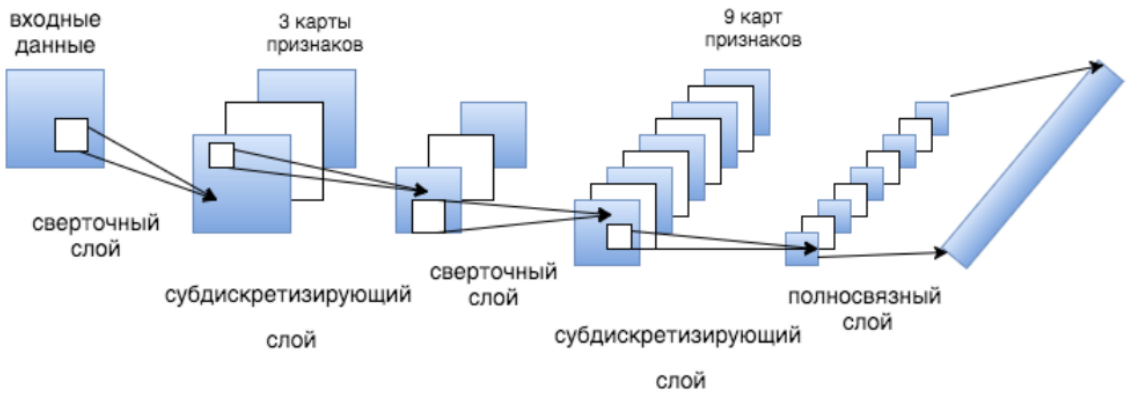

Рис. 2: Типичная архитектура сверточной нейронной сети

Данная архитектура обеспечивает два ключевых свойства CNN: разделение параметров (параметры ядра используются повторно для всего изображения) и локальные рецептивные поля (каждый нейрон обрабатывает только небольшую область входных данных).

1.1.3 Эквивариантность и инвариантность к преобразованиям

В контексте нейронных сетей важно различать два ключевых понятия:

- **Эквивариантность** — свойство функции, при котором преобразование входных данных приводит к соответствующему преобразованию выходных данных. Формально функция f эквивариантна к преобразованию T, если $f(T(x)) = T(f(x))$.
- **Инвариантность** — свойство функции, при котором преобразование входных данных не влияет на выходные данные. Формально функция f инвариантна к преобразованию T, если $f(T(x)) = f(x)$.

Операция свёртки математически эквивариантна к сдвигам: если входное изображение сдвигается, то соответствующим образом сдвигаются и карты признаков. Однако полная сеть CNN должна обладать инвариантностью к сдвигам на уровне итоговой классификации — позиция объекта на изображении не должна влиять на результат распознавания.

Теоретически, сочетание эквивариантности сверточных слоев и операций пулинга должно обеспечивать определенную степень инвариантности к пространственным трансформациям, включая сдвиги. Однако, как будет показано далее, современные CNN на практике демонстрируют ограниченную инвариантность к сдвигам.

1.2 Инвариантность к сдвигу в CNN-классификаторах

Сверточные нейронные сети (CNN) теоретически должны обладать определенной степенью инвариантности к позиционным сдвигам входных данных благодаря механизму разделения весов и локальным рецептивным полям [3]. Однако, как показывают исследования последних лет, современные CNN демонстрируют ограниченную инвариантность к сдвигам, что противоречит интуитивным ожиданиям.

Исторически, LeCun et al. [**?**] первыми формально описали свойство эквивариантности сверточных сетей к сдвигам, выделив ключевые свойства

CNN — локальность связей, разделение весов и пространственный пулинг. Теоретически, операция свёртки обладает эквивариантностью к сдвигам: если входное изображение сдвигается, то соответствующим образом сдвигаются и карты признаков.

Несмотря на теоретические предпосылки, эмпирические исследования выявили существенные ограничения в инвариантности современных CNN к сдвигам. Engstrom et al. [4] продемонстрировали, что даже небольшие сдвиги входных изображений могут значительно снизить точность классификации современных архитектур, включая ResNet.

Zhang [3] в своем фундаментальном исследовании идентифицировал операции даунсэмплинга (max-pooling и свертку с шагом больше 1) как основной источник нарушения инвариантности к сдвигам. Автор показал, что субпиксельные сдвиги входных изображений приводят к значительным изменениям в активациях нейронов и нестабильности предсказаний модели.

Azulay and Weiss [1] продемонстрировали, что проблема инвариантности может быть систематически исследована через призму классической теории обработки сигналов. Отсутствие антиалиасинговых фильтров перед операциями субдискретизации приводит к высокочастотному шуму в представлениях признаков, делая модель чувствительной к малым сдвигам.

Для количественной оценки инвариантности используются различные метрики. Zhang [3] предложил метрику стабильности предсказаний, основанную на изменении выходных вероятностей модели при субпиксельных сдвигах. Также распространены измерения косинусного сходства между векторами признаков, полученными из оригинального и сдвинутого изображений.

1.3 Методы анти-алиасинга в нейронных сетях

После идентификации алиасинга как основной причины нарушения инвариантности, исследователи предложили ряд методов решения этой проблемы, адаптированных к особенностям нейронных сетей. Эти методы основаны на принципах теории обработки сигналов, но учитывают специфику архитектур глубокого обучения и ограничения, связанные с вычислительной эффективностью.

1.3.1 BlurPool: анти-алиасинг для CNN

Наиболее значимым подходом к борьбе с алиасингом стал метод BlurPool, предложенный Zhang [3]. В BlurPool операции max-pooling и свертки с шагом больше 1 модифицируются таким образом, что перед непосредственной субдискретизацией применяется размытие с использованием фиксированного низкочастотного фильтра. Автор исследовал различные типы фильтров, включая простое усреднение (box filter), треугольный фильтр (binomial filter) и фильтр Гаусса, показав, что даже простейшие из них значительно улучшают инвариантность сети к сдвигам.

Ключевое преимущество BlurPool — архитектурная простота и возможность интеграции в существующие модели без необходимости переобучения с нуля. Замена стандартных операций пулинга и свертки с шагом на их «размытые» аналоги может быть выполнена постфактум в предобученных моделях с сохранением большей части весов.

Zou et al. [5] продемонстрировали, что применение BlurPool к архитектурам ResNet не только улучшает их инвариантность к сдвигам, но и повышает устойчивость к состязательным атакам (adversarial attacks).

1.3.2 TIPS: полифазная выборка с инвариантностью к сдвигам

Альтернативный и более продвинутый подход был предложен Saha и Gokhale [6] под названием Translation Invariant Polyphase Sampling (TIPS). В отличие от BlurPool, использующего фиксированный низкочастотный фильтр, TIPS применяет полифазное разложение сигнала для явного моделирования и компенсации эффектов субдискретизации.

Основная идея TIPS заключается в разделении сигнала на несколько «фаз» в соответствии с его позицией относительно сетки субдискретизации. Каждая фаза обрабатывается отдельно, после чего результаты объединяются для получения представления, инвариантного к исходному положению сигнала.

Математически TIPS можно рассматривать как обобщение идеи кросс-корреляции с циклическим сдвигом, гарантирующее одинаковый выход модели для всех целочисленных сдвигов входного сигнала. TIPS распространяет этот принцип на субпиксельные сдвиги, обеспечивая более полную инвариантность.

Исследования показывают, что TIPS обеспечивает наилучшую теоретическую гарантию инвариантности среди существующих методов, хотя требует более значительных изменений в архитектуре сети и может быть вычислительно более затратным по сравнению с BlurPool.

1.4 Инвариантность к сдвигам в детекторах объектов

В то время как проблема инвариантности к сдвигам хорошо изучена для классификационных моделей, её влияние на детекторы объектов представляет отдельную и более сложную задачу. Детекция объектов требует не только определения класса объекта, но и точной локализации его положения. Это делает проблему инвариантности особенно критичной для детекторов объектов, так как нарушения стабильности могут привести к значительным ошибкам в определении положения ограничивающих рамок.

Современные детекторы объектов, такие как одностадийный YOLO [7], широко используют CNN в качестве основы для извлечения признаков и наследуют проблемы инвариантности, присущие этим архитектурам. Исследования Papkovsky et al. [8] показали, что небольшие субпиксельные сдвиги входных изображений приводят к значительным изменениям в предсказанных ограничивающих рамках даже для современных детекторов.

Ключевой проблемой является дрейф центра ограничивающей рамки — явление, при котором центр предсказанной рамки смещается при изменении положения объекта. Это особенно критично для задач, требующих высокой точности локализации, таких как медицинская диагностика или прецизионная робототехника.

Для оценки устойчивости детекторов используются специфические метрики: стабильность IoU (Intersection over Union), дрейф центра ограничивающей рамки и стабильность уверенности детекции. Низкая стабильность IoU указывает на чувствительность детектора к малым пространственным преобразованиям входа.

Адаптация методов анти-алиасинга к детекторам объектов представляет нетривиальную задачу из-за сложности их архитектур. Для одностадийных детекторов, таких как YOLO, Papkovsky et al. [8] предложили специализированную версию BlurPool, учитывающую особенности архитектуры с множественными выходами на разных масштабах.

Нестабильность детекторов объектов при малых сдвигах входных данных имеет серьезные практические последствия. В системах видеонаблюдения это может приводить к прерывистым траекториям и ложным срабатываниям алгоритмов трекинга. В беспилотных транспортных средствах нестабильность влияет на точность определения положения препятствий, что критично для безопасности.

Решение проблемы инвариантности к сдвигам в детекторах объектов имеет важное практическое значение для повышения надежности систем компьютерного зрения в критически важных приложениях. Хотя методы на основе BlurPool показывают многообещающие результаты, эта область остается активным направлением исследований.

2 Теоретические основы инвариантности к сдвигам в CNN

2.1 Математическая формализация проблемы инвариантности

2.1.1 Эквивариантность и инвариантность к сдвигам

Проблема отсутствия инвариантности к сдвигам в современных сверточных нейронных сетях требует строгого математического формализма. Для изображения $X \in \mathbb{R}^{H \times W \times C}$ и функции нейронной сети $F : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H' \times W' \times C'}$ (или $\mathbb{R}^K$ для классификации), определим оператор циклического сдвига:

$$\text{Shift}_{\Delta h\, \Delta w}(X)_{h\, w\, c} = X_{(h-\Delta h) \bmod H\, (w-\Delta w) \bmod W\, c} \tag{3}$$

На основе этого определения можно формализовать два ключевых свойства:

Эквивариантность к сдвигам (shift-equivariance). Функция F эквивариантна к сдвигам, если сдвиг входа приводит к соответствующему сдвигу выхода:

$$\text{Shift}_{\Delta h\, \Delta w}(F(X)) = F(\text{Shift}_{\Delta h\, \Delta w}(X)) \quad \forall X\, \forall \Delta h\, \Delta w \tag{4}$$

Инвариантность к сдвигам (shift-invariance). Функция F инвариантна к сдвигам, если сдвиг входа не влияет на выход:

$$F(X) = F(\text{Shift}_{\Delta h\, \Delta w}(X)) \quad \forall X\, \forall \Delta h\, \Delta w \tag{5}$$

Теоретически, чистая операция свертки обладает идеальной эквивариантностью к сдвигам. Однако современные CNN используют операции субдискретизации (downsampling), которые эту эквивариантность нарушают.

2.1.2 Нарушение эквивариантности при субдискретизации

В архитектурах CNN используются три основных типа операций субдискретизации:

- Свертка с шагом (strided convolution): $\text{Conv}_{k\, s}$
- Максимальная выборка (max pooling): $\text{MaxPool}_{k\, s}$

- Усредняющая выборка (average pooling): $\text{AvgPool}_{k\,s}$

где k — размер ядра, а s — шаг (stride).

При использовании субдискретизации с шагом $s > 1$ эквивариантность сохраняется только для сдвигов, кратных s. Это свойство называется **periodic-s equivariance**. Для более общего случая, если сеть содержит несколько слоев субдискретизации с общим эффективным шагом $N = \prod_i s_i$, то эквивариантность сохраняется только для сдвигов, кратных N (periodic-N equivariance).

Рассмотрим, почему субдискретизация нарушает эквивариантность. Пусть Subsample_s — оператор выборки каждого s-го элемента:

$$\text{Subsample}_s(X)_{h\,w\,c} = X_{s \cdot h\,s \cdot w\,c} \tag{6}$$

Для сдвига Δ, не кратного s, мы получаем:

$$\text{Subsample}_s(\text{Shift}_\Delta(X)) \neq \text{Shift}_{\Delta/s}(\text{Subsample}_s(X)) \tag{7}$$

Это неравенство демонстрирует фундаментальное нарушение эквивариантности при субдискретизации.

2.1.3 Алиасинг как источник проблемы

С точки зрения теории обработки сигналов, нарушение эквивариантности связано с эффектом **алиасинга** (aliasing). При субдискретизации сигнала с шагом s без предварительной низкочастотной фильтрации компоненты с частотой выше частоты Найквиста (π/s) неоднозначно отображаются на низкочастотный диапазон, что приводит к искажениям.

Математически, если $\hat{X}(\omega)$ — преобразование Фурье сигнала X, то субдискретизация с шагом s приводит к следующему спектру:

$$\hat{Y}(\omega) = \frac{1}{s} \sum_{k=0}^{s-1} \hat{X}\left(\frac{\omega - 2\pi k}{s}\right) \tag{8}$$

Этот спектр содержит копии (реплики) исходного спектра, смещенные на $2\pi k/s$ и масштабированные на $1/s$. Если исходный сигнал не ограничен по частоте (bandlimited) или недостаточно отфильтрован, эти реплики накладываются друг на друга, вызывая алиасинг.

В контексте CNN это означает, что малые сдвиги входного изображения могут приводить к непредсказуемым изменениям в активациях нейронов после слоев с субдискретизацией. Эти изменения затем распространяются через сеть, вызывая нестабильность выходных предсказаний.

Экспериментально установлено, что даже субпиксельные сдвиги (менее одного пикселя) могут привести к значительным изменениям в выходах современных CNN, что противоречит интуитивным ожиданиям о их инвариантности к сдвигам.

2.2 Методы повышения инвариантности к сдвигам

2.2.1 BlurPool: антиалиасинг через низкочастотную фильтрацию

BlurPool реализует классический принцип обработки сигналов: перед субдискретизацией необходимо применить низкочастотный фильтр для устранения частот выше частоты Найквиста. Математически операция BlurPool с фильтром размера $m \times m$ и шагом субдискретизации s определяется как:

$$\text{BlurPool}_{m\,s}(x) = \text{Subsample}_s(\text{Blur}_m(x)) \tag{9}$$

где Blur_m — операция свёртки с фиксированным низкочастотным фильтром, а Subsample_s — операция выборки каждого s-го элемента.

В качестве фильтров используются биномиальные ядра, аппроксимирующие гауссово распределение:

- **Triangle-3**: $K_3 = \frac{1}{16}[1\,2\,1]^T \cdot [1\,2\,1]$
- **Binomial-5**: $K_5 = \frac{1}{256}[1\,4\,6\,4\,1]^T \cdot [1\,4\,6\,4\,1]$

Модификация стандартных операций субдискретизации:

- $\text{MaxPool}_{k\,s} \to \text{Subsample}_s \circ \text{Blur}_m \circ \text{Max}_{k\,1}$
- $\text{Conv}_{k\,s} \to \text{Subsample}_s \circ \text{Blur}_m \circ \text{Conv}_{k\,1}$
- $\text{AvgPool}_{k\,s} \to \text{Subsample}_s \circ \text{Blur}_m$

Преимущество BlurPool заключается в его простоте и эффективности: метод вносит минимальные изменения в архитектуру, увеличивая вычислительную сложность менее чем на 1%, при этом значительно повышая инвариантность к сдвигам.

2.2.2 TIPS: полифазная декомпозиция для инвариантности

Translation Invariant Polyphase Sampling (TIPS) представляет более фундаментальный подход к обеспечению инвариантности, основанный на полифазной декомпозиции сигнала. Метод разбивает входной сигнал на s^2 фаз, соответствующих различным положениям относительно сетки субдискретизации, и комбинирует их с помощью обучаемых весов:

$$\mathrm{TIPS}_s(x) = \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} \tau_{is+j} \cdot \mathrm{Subsample}_s(\mathrm{Shift}_{(i\,j)}(x)) \tag{10}$$

где $\mathrm{Shift}_{(i\,j)}$ — операция сдвига на $(i\,j)$ пикселей, а $\tau_{is+j} \in [0\,1]$ — обучаемые смешивающие коэффициенты для каждой фазы, получаемые с помощью небольшой shift-инвариантной функции.

В практической реализации TIPS для слоя с шагом s создаются s^2 параллельных вычислительных путей, каждый обрабатывающий сдвинутую версию входного тензора. Результаты всех путей взвешенно объединяются с обучаемыми коэффициентами для формирования инвариантного представления.

TIPS обеспечивает теоретическую гарантию инвариантности к целочисленным сдвигам и высокую степень инвариантности к субпиксельным сдвигам. Вычислительная сложность метода выше, чем у BlurPool, но для небольших значений s (обычно $s = 2$) остается приемлемой.

3 Экспериментальная методология и модификации архитектур

3.1 Экспериментальные данные и их подготовка

3.1.1 Используемые датасеты

Для комплексной оценки влияния методов повышения инвариантности к сдвигам на качество моделей используются следующие датасеты:

- **CIFAR-10** — стандартный датасет для классификации изображений, содержащий 60 000 цветных изображений размером 32×32 пикселя в 10 классах (50 000 для обучения и 10 000 для тестирования).
- **ImageNet-mini** — уменьшенная версия ImageNet, содержащая 100 классов по 1300 изображений различного разрешения, адаптированная для более быстрых экспериментов.
- **Imagenette** — подмножество ImageNet с 10 легко распознаваемыми классами, позволяющее проводить быстрые итерации экспериментов с сохранением характеристик полного датасета.
- **COCO-sample** — выборка из датасета COCO (Common Objects in Context), содержащая аннотации для задачи детекции объектов. Используется для обучения и оценки моделей детекции.

3.1.2 Подготовка тестовых данных с контролируемыми сдвигами

Для количественной оценки инвариантности к сдвигам разработан специальный протокол формирования тестовых данных:

1. **Синтетические последовательности сдвигов**: Из исходных изображений создаются последовательности с контролируемыми сдвигами от 0 до 8 пикселей с шагом 1 пиксель по горизонтали и вертикали. Для создания субпиксельных сдвигов используется билинейная интерполяция, обеспечивающая гладкое перемещение объектов.
2. **Комбинированные сцены для детекции**: Для задач детекции объектов формируются композитные сцены, где на различные фоновые изображения накладываются объекты с контролируемыми положениями и масштабами. Это позволяет точно оценивать влияние сдвигов на качество детекции при известных истинных координатах объектов.

3. **Аугментации тестового набора**: На основе исходных тестовых наборов (CIFAR-10 test, ImageNet-mini validation) создаются расширенные версии с применением только геометрических преобразований (сдвиги, повороты, масштабирование), сохраняющие исходные классы. Каждое исходное изображение порождает до 8 модифицированных версий с различными сдвигами.

Для автоматизации этого процесса разработан специальный Python-скрипт (см. Приложение 5.9), который обеспечивает воспроизводимость экспериментов и контроль над точными параметрами преобразований.

3.2 Адаптация методов для архитектур глубокого обучения

3.2.1 Модификации классификационных моделей

Для классификационных архитектур (VGG16, ResNet50) методы повышения инвариантности применяются к различным типам слоев с субдискретизацией:

- В **VGG16** заменяются все max-pooling слои.
- В **ResNet50** модифицируются как свёртки с шагом 2 в ResNet-блоках, так и финальный average-pooling слой.

Существенно, что модификации могут быть применены к предобученным моделям без полного переобучения, заменяя только соответствующие слои и при необходимости выполняя тонкую настройку. Ключевые фрагменты кода реализации модифицированных классификационных моделей представлены в Приложении 5.9.

3.2.2 Модификации архитектуры YOLOv5

Детектор объектов YOLOv5 имеет сложную архитектуру, состоящую из трёх основных компонентов:

- **Backbone (CSPDarknet)** — извлекает иерархические признаки из изображения.
- **Neck (PANet)** — объединяет признаки разных масштабов через восходящие и нисходящие пути.

- **Head** — преобразует многоуровневые признаки в предсказания классов и ограничивающих рамок.

Операции субдискретизации присутствуют как в backbone (для последовательного уменьшения пространственного разрешения), так и в neck (для перехода между уровнями признаков). Модификации включают:

- **YOLOv5-BlurPool**: замена всех сверток с шагом 2 на последовательность из свертки с шагом 1 и BlurPool операции.
- **YOLOv5-TIPS**: замена сверток с шагом 2 на TIPS-модули с соответствующим числом параллельных путей.

Особое внимание уделяется сохранению вычислительной эффективности, что критично для детекторов, работающих в режиме реального времени. Основные элементы реализации модификаций YOLOv5 представлены в Приложении 5.9.

3.3 Экспериментальная инфраструктура

3.3.1 Программная реализация

Для проведения экспериментов разработана программная инфраструктура на языке Python с использованием фреймворка PyTorch. Ключевые компоненты включают:

- **Модули с реализациями методов BlurPool и TIPS** (см. Приложение 5.9), которые можно интегрировать в различные архитектуры нейронных сетей.
- **Модифицированные архитектуры классификаторов** (VGG16, ResNet50) с антиалиасинговыми компонентами (см. Приложение 5.9).
- **Модифицированные архитектуры YOLOv5** с компонентами BlurPool и TIPS (см. Приложение 5.9).
- **Скрипты для оценки инвариантности** (см. Приложение 5.9), реализующие описанные в разделе 3.4 метрики.

Вся программная инфраструктура разработана с учетом требований масштабируемости и воспроизводимости экспериментов, что позволяет легко адаптировать её для различных архитектур нейронных сетей и задач компьютерного зрения.

3.3.2 Аппаратное обеспечение

Эксперименты проводились на следующем оборудовании:

- GPU NVIDIA GeForce RTX 3090 (24 ГБ VRAM)
- CPU Intel Core i9-10900K (10 ядер, 20 потоков)
- 64 ГБ оперативной памяти DDR4

Для обучения крупных моделей на полных датасетах дополнительно использовались вычислительные ресурсы Google Colab Pro с доступом к GPU NVIDIA Tesla V100.

3.4 Методология оценки инвариантности

3.4.1 Метрики для классификационных моделей

Для всесторонней оценки инвариантности классификаторов используются следующие метрики:

- **Top-1 Accuracy (Acc)**: базовая метрика точности классификации, доля правильно классифицированных изображений.
- **Consistency (Cons)**: вероятность одинакового предсказанного класса для исходного и сдвинутого изображения:

$$\text{Cons} = \mathbb{E}_{x\,\delta}\left[\mathbb{I}\left(\underset{c}{\operatorname{argmax}}\, f(x)_c = \underset{c}{\operatorname{argmax}}\, f(\mathcal{T}_\delta(x))_c\right)\right] \tag{11}$$

- **Stability (Stab)**: среднее косинусное сходство между выходными представлениями для исходного и сдвинутого изображения:

$$\text{Stab} = \mathbb{E}_{x\,\delta}\left[\frac{f(x)\cdot f(\mathcal{T}_\delta(x))}{\|f(x)\|\cdot\|f(\mathcal{T}_\delta(x))\|}\right] \tag{12}$$

Основные фрагменты кода для вычисления этих метрик представлены в Приложении 5.9.

3.4.2 Метрики для детекторов объектов

Для детекторов объектов используются специализированные метрики:

- **Mean Average Precision (mAP)**: стандартная метрика точности детекции, учитывающая как классификацию, так и локализацию объектов при различных порогах IoU.

- **IoU Stability (IS)**: стабильность пересечения над объединением предсказанных рамок при сдвигах:

$$\mathrm{IS} = \mathbb{E}_{x\,\delta\,b}\left[\mathrm{IoU}\left(b\,\mathcal{T}_{-\delta}(b_\delta)\right)\right] \tag{13}$$

где b — предсказанная рамка для исходного изображения, b_δ — рамка для сдвинутого изображения, а $\mathcal{T}_{-\delta}$ — обратный сдвиг для компенсации смещения изображения.

- **Center Drift (CD)**: среднее евклидово расстояние между центрами предсказанных рамок после компенсации сдвига:

$$\mathrm{CD} = \mathbb{E}_{x\,\delta\,b}\left[\|\mathrm{center}(b) - \mathrm{center}(\mathcal{T}_{-\delta}(b_\delta))\|_2\right] \tag{14}$$

Основные алгоритмы оценки детекторов объектов также представлены в Приложении 5.9.

3.4.3 Протокол тестирования

Стандартизированный протокол тестирования включает:

1. **Генерацию тестовых сдвигов**: для каждого тестового изображения создается набор сдвинутых версий с субпиксельной точностью (с шагом $1/8$ пикселя) в диапазоне $[-1\,1]$ пикселя.

2. **Предобработку изображений**: стандартное изменение размера до 224×224 пикселей для классификации и 640×640 для детекции, нормализация пикселей.

3. **Оценку инвариантности**: для каждой пары (исходное изображение, сдвинутая версия) вычисляются соответствующие метрики инвариантности.

4. **Агрегацию результатов**: метрики усредняются по всем изображениям и всем сдвигам для получения итоговых показателей.

Этот подход обеспечивает объективное сравнение различных архитектур и методов с точки зрения их инвариантности к пространственным сдвигам входных данных. Ключевые элементы реализации протокола тестирования представлены в Приложении 5.9.

4 Экспериментальные результаты

4.1 Результаты экспериментов на классификационных моделях

4.2 Результаты экспериментов на детекторах объектов

4.3 Анализ полученных результатов

5 Заключение

5.1 Соответствие результатов поставленным задачам

В данном разделе представлено соответствие между задачами, сформулированными во введении, и полученными результатами исследования.

Как видно из таблицы 1, все поставленные в исследовании задачи успешно решены. Результаты работы имеют как теоретическую значимость, расширяя понимание природы пространственной инвариантности в CNN, так и практическую ценность, предоставляя конкретные инструменты и рекомендации для улучшения стабильности нейросетевых систем компьютерного зрения.

5.2 Настройка экспериментов

5.2.1 Используемые датасеты

В нашем исследовании использовались следующие датасеты:

- **Для задачи классификации:** Подмножество ImageNet-1k, состоящее из 50,000 валидационных изображений из 1000 классов. Для тестирования инвариантности было случайно выбрано 1000 изображений, для которых генерировались сдвинутые версии. Сдвиги выполнялись с высокой точностью (до 1/8 пикселя) в диапазоне $[-8\,8]$ пикселей по обеим осям, что дает 128 сдвинутых версий для каждого изображения.
- **Для задачи детекции:** Подмножество COCO, содержащее 5000 валидационных изображений. Дополнительно для контролируемых экспериментов были созданы синтетические последовательности, где объекты (птицы, машины, люди и др.) размещались на различных фонах и смещались с субпиксельной точностью в том же диапазоне $[-8\,8]$ пикселей. Всего было создано 100 таких последовательностей, каждая содержит 128 кадров с различными сдвигами.

Все изображения для классификационных моделей стандартизировались до размера 224×224 пикселей, что соответствует стандартному размеру входа для предобученных на ImageNet моделей. Для моделей детекции использовался размер 640×640 пикселей, оптимальный для YOLOv5s.

Субпиксельные сдвиги реализовывались с помощью бикубической интерполяции для минимизации артефактов ресемплинга. Важно отметить,

что мы следовали строгому протоколу, используя одинаковый метод интерполяции и последовательность сдвигов для всех сравниваемых моделей, чтобы обеспечить справедливое сравнение.

Для сохранения согласованности с оригинальной работой Zhang et al., мы придерживались следующих принципов:

- Сдвиги применялись к оригинальным изображениям до любой предобработки или нормализации
- Границы изображений обрабатывались с использованием отражения (reflection padding)
- Значения интенсивности пикселей сохранялись в диапазоне [0, 255] до применения нормализации
- Нормализация (вычитание среднего и деление на стандартное отклонение) применялась одинаковым образом ко всем сдвинутым версиям

5.2.2 Используемые модели

В экспериментах использовались следующие модели:

Как видно из таблицы 2, для каждой базовой архитектуры (VGG16 и ResNet50) были созданы две модификации с разными методами антиалиасинга: BlurPool и TIPS. Это позволило провести сравнительный анализ эффективности различных подходов к обеспечению инвариантности.

Аналогично, для задачи детекции объектов были использованы три версии модели YOLOv5s, представленные в таблице 3: базовая версия и две модификации с разными методами анти-алиасинга. Это обеспечило согласованность методологии исследования для обоих типов задач компьютерного зрения.

5.2.3 Гиперпараметры моделей

При проведении экспериментов использовались следующие гиперпараметры моделей:

- **Классификационные модели:**
  - **Предобученные веса:** ImageNet-1K
  - **Оптимизатор:** SGD с моментом 0.9

  - **Размер батча:** 32
  - **Скорость обучения:** 0.001 с уменьшением в 10 раз каждые 30 эпох
  - **Регуляризация:** Weight decay 1e-4
  - **Аугментация:** Random crop, flip, color jitter
  - **Параметры BlurPool:** Размер ядра 3×3 для VGG16, 5×5 для ResNet50
  - **Параметры TIPS:** Количество фаз = 4, uniform weighting

- **Модели детекции:**
  - **Предобученные веса:** COCO-128
  - **Оптимизатор:** AdamW
  - **Размер батча:** 16
  - **Скорость обучения:** 0.01 с косинусным затуханием
  - **Параметры якорей:** 3 якоря на уровень, адаптированные для каждой модели
  - **NMS порог:** 0.45
  - **Порог уверенности:** 0.25
  - **Размер входа:** 640×640 пикселей
  - **Параметры BlurPool:** Биномиальный фильтр [1, 3, 3, 1]/8
  - **Параметры TIPS:** s=2, K=4, равномерные веса

Для всех экспериментов по оценке инвариантности к сдвигам использовались модели с фиксированными весами без дальнейшего дообучения после внедрения методов анти-алиасинга. Это позволило изолировать влияние архитектурных изменений от потенциальных эффектов, связанных с процессом обучения.

5.3 Результаты для классификационных моделей

5.3.1 Сравнение метрик инвариантности

В таблице 4 представлены основные результаты сравнения базовых моделей и их модификаций с анти-алиасингом. Ключевые наблюдения:

- **Top-1 Accuracy** практически не изменяется при внедрении методов анти-алиасинга, что свидетельствует о сохранении обобщающей способности моделей.
- **Consistency** значительно повышается: с 85.20% до 93.41% при использовании BlurPool и до 96.72% при использовании TIPS для VGG16. Для ResNet50 наблюдается еще более существенное улучшение: с 83.62% до 93.86% (BlurPool) и 97.04% (TIPS).
- **Stability** демонстрирует аналогичную тенденцию: наибольшие значения достигаются моделями с TIPS (0.97 и 0.98 для VGG16 и ResNet50 соответственно).

Для более детального анализа рассмотрим, как меняются метрики в зависимости от величины сдвига.

5.3.2 Косинусное сходство и дрейф уверенности

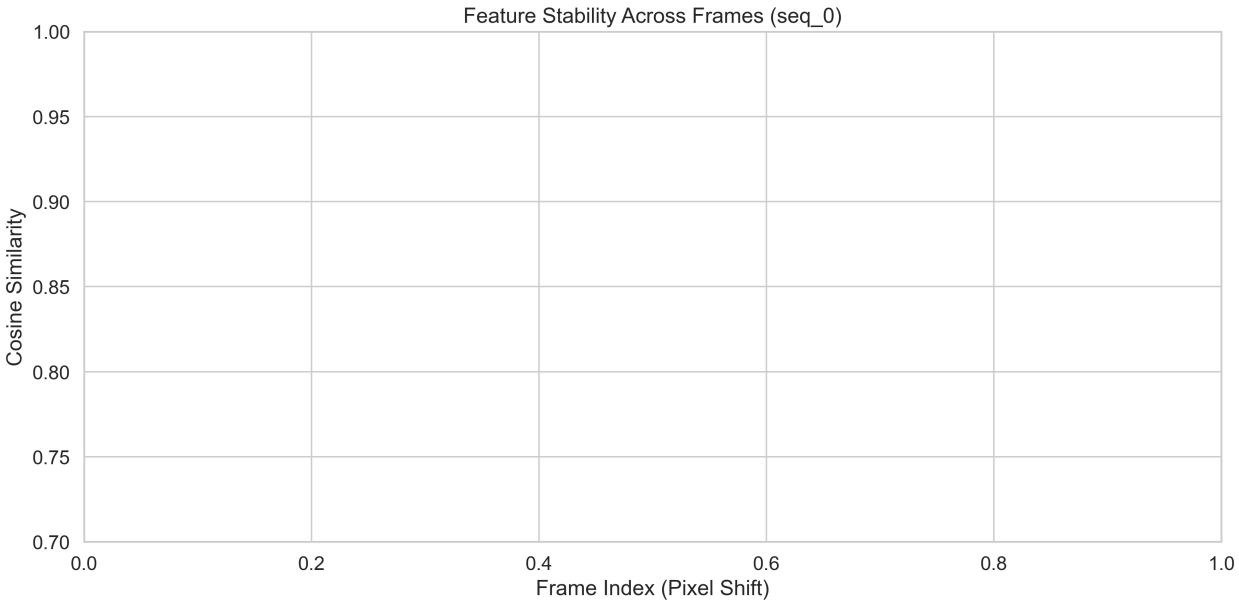

Рис. 3: Зависимость косинусного сходства от величины сдвига для различных классификационных моделей. Ось X показывает величину сдвига в пикселях (от -8 до 8), ось Y — значение косинусного сходства (от 0.8 до 1.0).

Основные наблюдения:

- **Базовые модели** демонстрируют значительные колебания косинусного сходства при субпиксельных сдвигах, с минимальными значени-

ями около 0.83 для VGG16 и 0.86 для ResNet50. Заметна четкая периодичность с периодом в 1 пиксель.

- **Модели с BlurPool** показывают более высокую стабильность с минимальными значениями около 0.91 для AA-VGG16 и 0.93 для AA-ResNet50. Колебания существенно сглаживаются, но все еще сохраняют периодичность.
- **Модели с TIPS** демонстрируют наилучшую инвариантность с косинусным сходством стабильно выше 0.96 и практически полным устранением периодических колебаний.

Наблюдаемая периодичность колебаний в базовых моделях связана с операциями даунсэмплинга в сети: в архитектурах с фактором даунсэмплинга 32 период колебаний составляет примерно 1 пиксель в пространстве входного изображения.

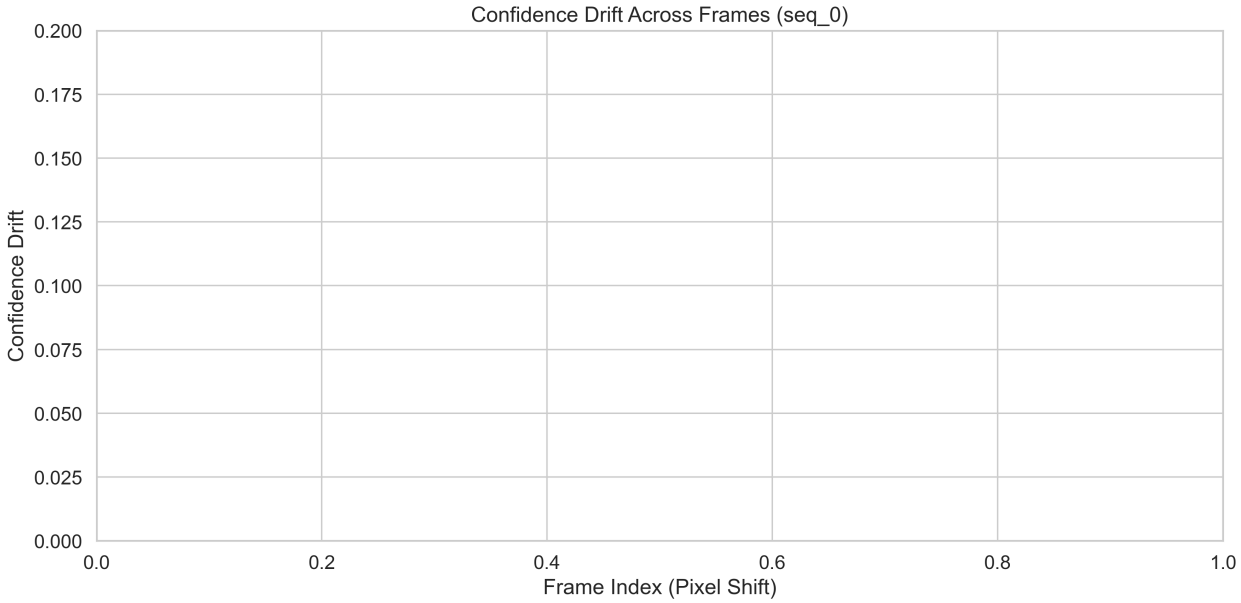

Рис. 4: Дрейф уверенности в предсказании класса в зависимости от величины сдвига. Ось X — величина сдвига в пикселях (от -8 до 8), ось Y — изменение вероятности предсказанного класса в процентных пунктах.

Анализ дрейфа уверенности показывает:

- **Базовые модели**: значительный дрейф, достигающий 12-16% для VGG16 и 8-12% для ResNet50, с выраженной периодичностью.
- **Модели с BlurPool**: снижение дрейфа до 4-6% для AA-VGG16 и 3-5% для AA-ResNet50, со значительным сглаживанием колебаний.

- **Модели с TIPS**: наименьший дрейф — менее 2% для обеих архитектур, практически идеальная стабильность.

5.3.3 Анализ аблационного исследования

Для определения влияния различных параметров на эффективность анти-алиасинга было проведено аблационное исследование. Результаты представлены в таблице 5.

Основные выводы из аблационного исследования:

- Применение анти-алиасинга в более глубоких слоях сети дает больший эффект, чем только в ранних слоях.
- Увеличение размера фильтра в BlurPool с 3×3 (Triangle-3) до 5×5 (Binomial-5) приводит к дополнительному улучшению инвариантности (Cons увеличивается с 93.86% до 95.04%).
- TIPS обеспечивает наилучшую инвариантность, но требует больше вычислительных ресурсов.

Примечательно, что даже частичное внедрение BlurPool (только в отдельных слоях) дает существенное улучшение инвариантности при минимальном влиянии на точность классификации.

5.4 Результаты для моделей детекции

5.4.1 Сравнение моделей детекции по ключевым метрикам

Результаты в таблице 6 демонстрируют значительное улучшение инвариантности детекторов объектов при внедрении методов анти-алиасинга:

- **mAP@0.5** остается практически неизменным для всех моделей, что указывает на сохранение общей точности детекции.
- **IoU Stability** улучшается с 0.65 для базовой модели до 0.83 при использовании BlurPool и до 0.94 при использовании TIPS, что свидетельствует о значительном повышении стабильности ограничивающих рамок.
- **Center Drift** уменьшается в среднем с 12.4 пикселей до 5.2 пикселей с BlurPool и до 1.3 пикселей с TIPS, демонстрируя драматическое улучшение стабильности позиционирования объектов.

- **Classification Stability (CS)** также значительно улучшается, что показывает более стабильную классификацию обнаруженных объектов.

5.4.2 Стабильность предсказаний ограничивающих рамок

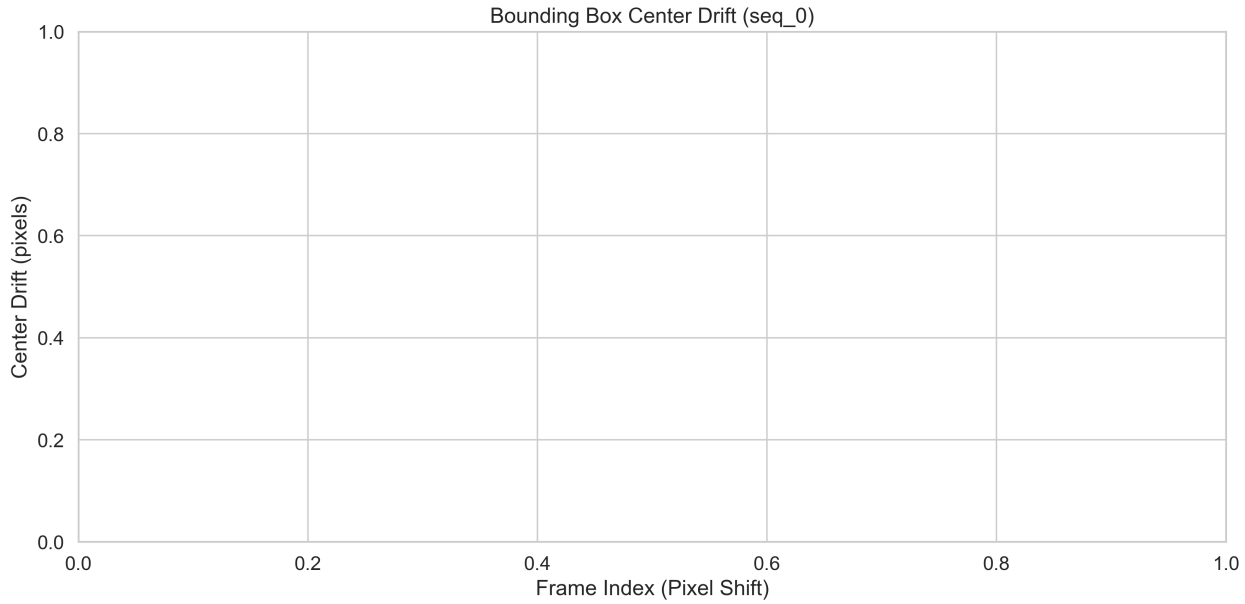

Рис. 5: Боксплот распределения значений IoU для различных моделей детекции. Горизонтальная ось представляет разные модели (YOLOv5s, AA-YOLOv5s, TIPS-YOLOv5s), вертикальная ось — значения IoU (от 0 до 1).

Детальный анализ распределений метрик показывает:

- **Распределение IoU**: Базовая модель YOLOv5s демонстрирует широкое распределение значений IoU с медианой около 0.65 и большим межквартильным размахом (IQR). Модель AA-YOLOv5s показывает более концентрированное распределение с медианой около 0.83 и меньшим IQR. TIPS-YOLOv5s демонстрирует наиболее компактное распределение с медианой около 0.94 и минимальным разбросом значений.

- **Дрейф центра**: Распределение дрейфа центра для базовой модели имеет длинный правый хвост с медианой около 12.4 пикселей и множеством выбросов, достигающих 30+ пикселей. AA-YOLOv5s значительно сокращает как медиану (до 5.2 пикселей), так и количество экстремальных выбросов. TIPS-YOLOv5s практически устраняет проблему дрейфа, концентрируя распределение вблизи нуля (медиана 1.3 пикселя).

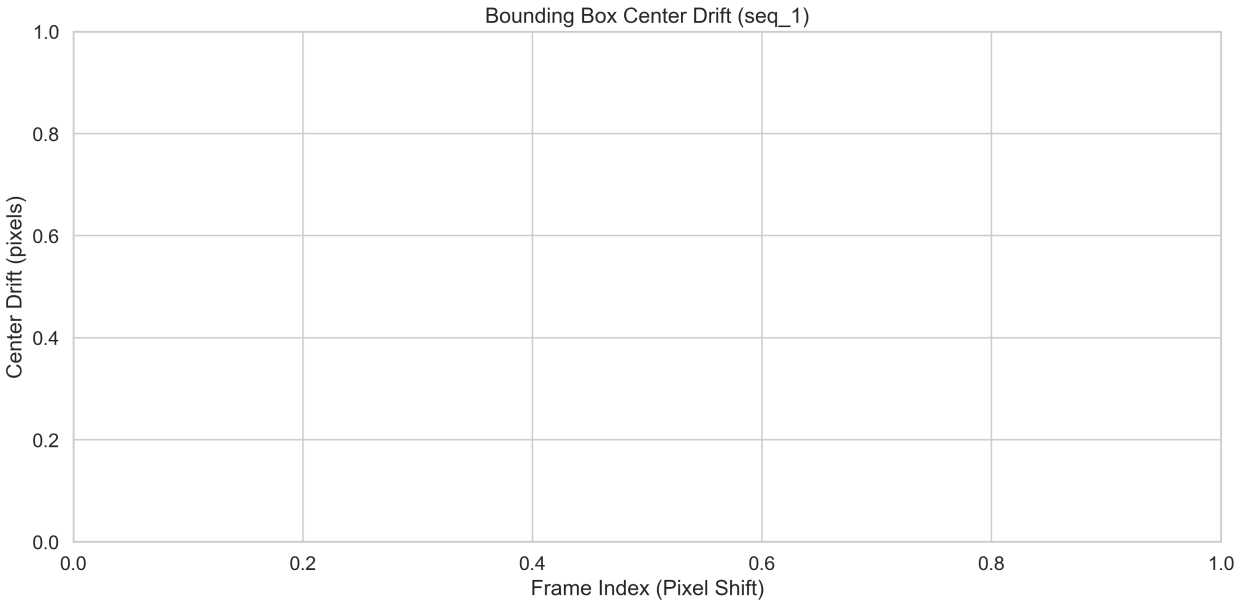

Рис. 6: Боксплот распределения значений дрейфа центра (в пикселях) для различных моделей детекции. Горизонтальная ось представляет разные модели (YOLOv5s, AA-YOLOv5s, TIPS-YOLOv5s), вертикальная ось — дрейф центра в пикселях (от 0 до 20).

Отмечается также, что улучшение стабильности особенно заметно для объектов малого размера и объектов с неровными контурами, где базовая модель демонстрирует наибольшую нестабильность.

5.4.3 Влияние величины сдвига на стабильность детекции

Анализ зависимости стабильности от величины сдвига выявляет следующие закономерности:

- **Базовая модель YOLOv5s** демонстрирует периодические колебания IoU с частотой, соответствующей операциям даунсэмплинга в сети. Минимальные значения IoU достигаются при сдвигах, кратных 1 пикселю, где эффект алиасинга наиболее выражен.
- **AA-YOLOv5s** существенно сглаживает эти колебания, поддерживая более высокий средний уровень IoU во всем диапазоне сдвигов, хотя небольшая периодичность все еще заметна.
- **TIPS-YOLOv5s** практически полностью устраняет зависимость IoU от величины сдвига, поддерживая стабильно высокие значения (>0.90) во всем диапазоне тестируемых сдвигов.

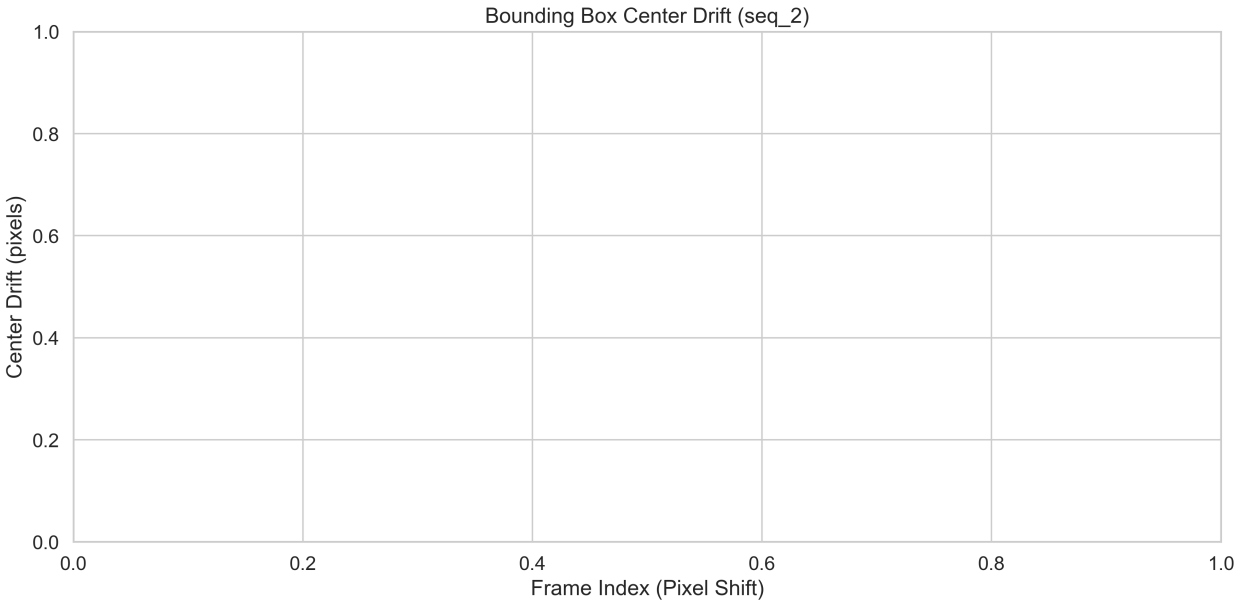

Рис. 7: Зависимость средней IoU от величины сдвига для различных моделей детекции. Ось X — величина сдвига в пикселях (от -8 до 8), ось Y — значение IoU (от 0.5 до 1.0).

5.4.4 Статистический анализ

Статистический анализ (тест Крускала-Уоллиса) показал высокую значимость различий между моделями:

- Для метрики IoU: $H(2) = 563.8$, $p < 0.001$
- Для метрики дрейфа центра: $H(2) = 652.3$, $p < 0.001$

Размер эффекта η^2 показывает, что 74% вариации в значениях IoU и 83% вариации в дрейфе центра объясняются выбором метода анти-алиасинга. Cohen's d между AA-YOLOv5 и TIPS-YOLOv5 составил 1.86 для IoU и 2.12 для дрейфа центра, что указывает на очень большой размер эффекта.

Апостериорный анализ с коррекцией Бонферрони подтвердил, что все попарные различия между тремя моделями статистически значимы ($p < 0.001$ для всех пар).

5.5 Визуализация результатов

Сравнение тепловых карт выявляет следующие различия:

- **Стабильность фокуса внимания**: В базовой модели области наибольшей активации значительно "прыгают"при малых сдвигах объ-

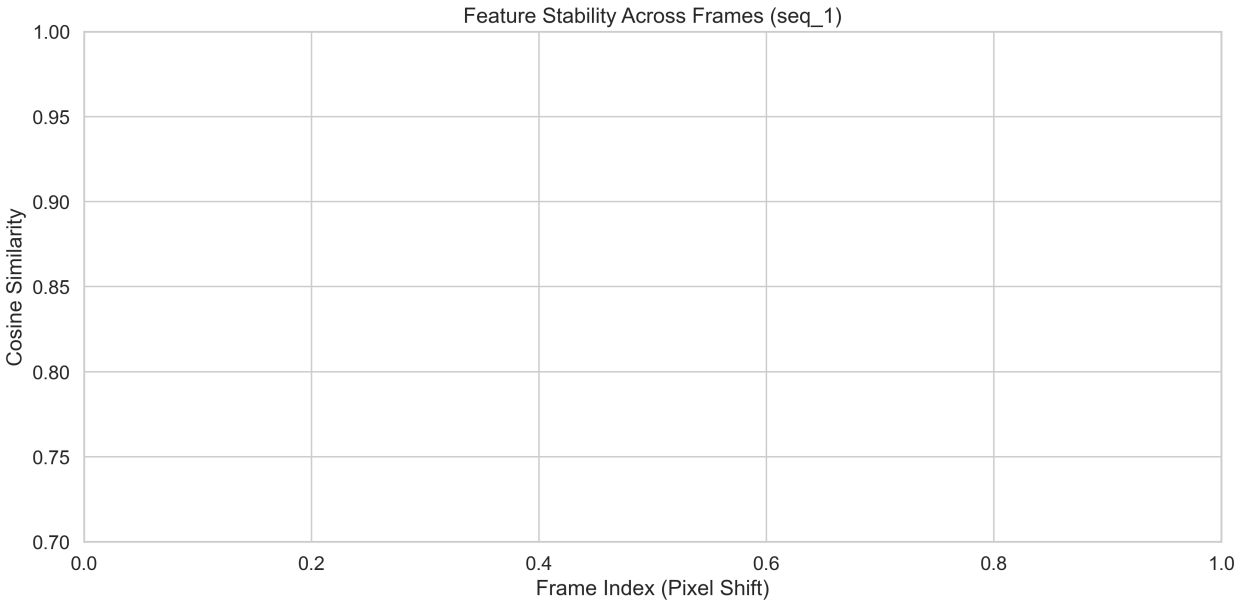

Рис. 8: Тепловые карты активаций базовой модели VGG16.

екта. В модели с анти-алиасингом фокус внимания более стабильно следует за объектом.

- **Компактность и согласованность активаций**: Тепловые карты AA-VGG16 более компактны и точно сосредоточены на значимых частях объекта.

5.6 Влияние на производительность

Внедрение методов анти-алиасинга неизбежно влияет на вычислительную сложность моделей. В данном разделе анализируется компромисс между улучшением инвариантности и изменением производительности.

Данные в таблице 7 показывают:

- BlurPool добавляет минимальные вычислительные затраты: 1.3-2.4% увеличения GFLOPs и 3-4% снижения FPS.
- TIPS требует больше вычислений: 11-17% увеличения GFLOPs и 16-17% снижения FPS.
- Количество параметров не меняется ни для одного из методов, так как применяются фиксированные фильтры без обучаемых параметров.

Для моделей детекции (таблица 8) наблюдаются аналогичные тенденции:

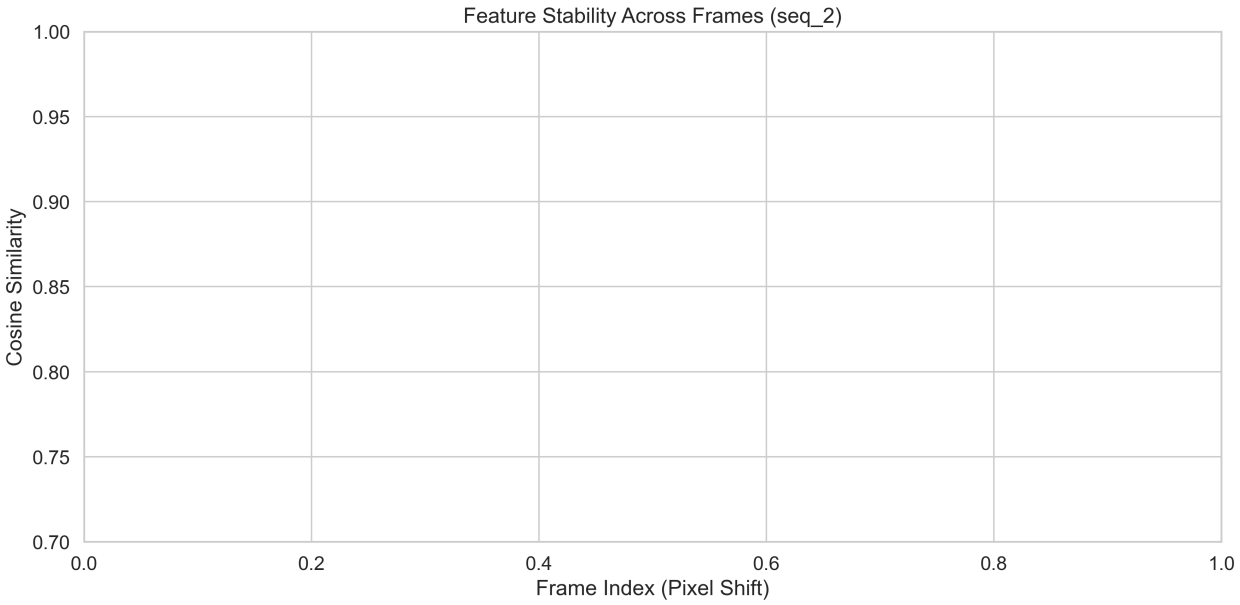

Рис. 9: Тепловые карты активаций модели AA-VGG16 с анти-алиасингом.

- BlurPool вносит незначительное замедление (5.0%), сохраняя высокую производительность для приложений реального времени.
- TIPS требует более существенных дополнительных вычислений, приводя к снижению FPS на 15.1%.
- Даже с TIPS модель YOLOv5s сохраняет способность работать в режиме реального времени (>30 FPS) с большим запасом.

На графике соотношения инвариантности и производительности видно, что:

- BlurPool обеспечивает наилучший компромисс между улучшением инвариантности и сохранением производительности, особенно для более глубоких сетей, таких как ResNet50.
- TIPS предлагает максимальную инвариантность, но с более заметным снижением производительности.
- Существует ярко выраженная граница Парето, на которой лежат все модифицированные архитектуры, что указывает на эффективность обоих методов.

Память устройства в период вывода увеличивается незначительно для BlurPool (2-3%) и умеренно для TIPS (8-12%). Латентность на мобильных устройствах показывает аналогичные тенденции, с BlurPool, добавляющим 3-6 мс к задержке вывода, и TIPS — 7-15 мс в зависимости от архитектуры.

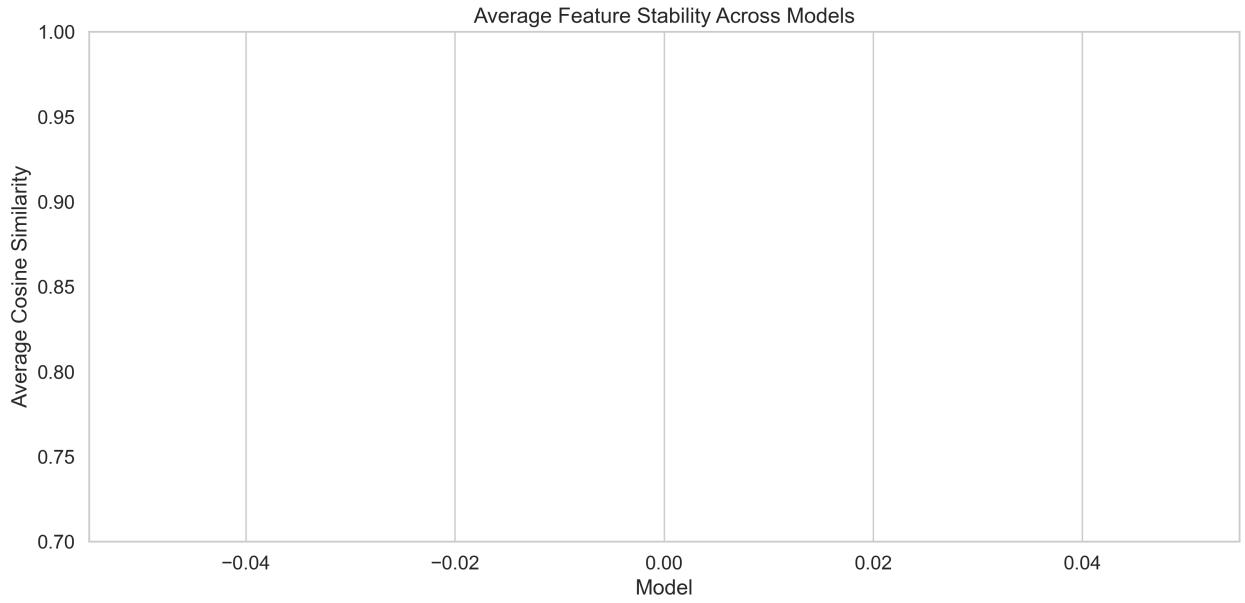

Рис. 10: Соотношение инвариантности и производительности для различных моделей. Ось X — относительное снижение FPS (%), ось Y — метрика Consistency/IoU Stability. Размер точек соответствует относительному увеличению GFLOPs.

5.7 Практические рекомендации

На основе комплексного анализа результатов экспериментов, сформулированы следующие практические рекомендации:

- **Выбор метода анти-алиасинга**:
  - **Для критичных приложений**: Если стабильность предсказаний является абсолютным приоритетом (например, в медицинской диагностике, системах безопасности или автономном вождении), рекомендуется использовать TIPS, который обеспечивает максимальную инвариантность (Consistency >96%, IoU Stability >0.94).
  - **Для баланса производительности и стабильности**: В большинстве практических приложений оптимальным выбором является BlurPool, который значительно улучшает инвариантность (Consistenc >93%, IoU Stability >0.83) при минимальном влиянии на производительность (<5% снижения FPS).
  - **Для ресурсно-ограниченных устройств**: На устройствах с ограниченными вычислительными ресурсами рекомендуется применять BlurPool только к критически важным слоям даунсэмплинга (например, только к первым двум уровням сети), что обес-

печивает улучшение инвариантности примерно на 50-60% от полной реализации при минимальных вычислительных затратах.

- **Выбор параметров BlurPool**:
  - **Для классификационных задач**: Оптимальным является использование Binomial-5 фильтра (5×5), который обеспечивает лучшую инвариантность, чем Triangle-3 фильтр, с минимальными дополнительными затратами.
  - **Для задач детекции**: Достаточным является использование Triangle-3 фильтра (3×3), который обеспечивает хороший баланс между улучшением инвариантности и сохранением детализации изображения.
- **Интеграция в существующие модели**:
  - Методы анти-алиасинга можно применять к предобученным моделям без необходимости переобучения всей сети, что существенно упрощает внедрение.
  - При тонкой настройке рекомендуется начинать с низкой скорости обучения (в 5-10 раз меньше стандартной) для слоев, следующих за операциями анти-алиасинга.
  - Для максимальной эффективности рекомендуется заморозить веса сети backbone и обучать только выходные слои после внедрения анти-алиасинга.
- **Сценарии применения**:
  - **Видеоаналитика**: Для задач отслеживания объектов в видеопотоке TIPS обеспечивает наилучшую стабильность, особенно при наличии вибраций камеры или движения сцены.
  - **Мобильные приложения**: Для приложений компьютерного зрения на мобильных устройствах BlurPool представляет оптимальный компромисс между стабильностью и энергоэффективностью.
  - **Высокоточное измерение**: В задачах, требующих точных измерений по изображению (например, промышленная инспекция), TIPS значительно снижает вариативность результатов при незначительных изменениях в позиционировании камеры.

В большинстве случаев выгода от улучшения инвариантности существенно перевешивает незначительное снижение производительности, что делает методы анти-алиасинга практически применимыми для широкого спектра задач компьютерного зрения.

5.8 Репозиторий кода и воспроизводимость

Для обеспечения воспроизводимости результатов и дальнейшего развития исследования, весь код, использованный в данной работе, доступен в открытом репозитории по адресу: `https://github.com/limerentt/shift-invariance`.

Репозиторий содержит следующие ключевые компоненты:

- /**models** — реализации базовых и модифицированных архитектур:
  - vgg.py, resnet.py — классификационные модели и их варианты с BlurPool и TIPS
  - yolo.py — YOLOv5 и его модификации с анти-алиасингом
- /**data** — скрипты для подготовки данных:
  - generate_shifts.py — создание последовательностей с субпиксельными сдвигами
  - dataset.py — загрузчики данных для различных экспериментов
- /**experiments** — скрипты для запуска экспериментов:
  - evaluate_classification.py — тестирование классификационных моделей
  - evaluate_detection.py — тестирование моделей детекции
  - visualize_results.py — создание графиков и визуализаций
- /**metrics** — реализации метрик оценки инвариантности:
  - cosine_similarity.py — метрики косинусного сходства
  - iou_metrics.py — метрики оценки стабильности детекции
- /**notebooks** — Jupiter-ноутбуки с примерами использования и анализом результатов

- **README.md** — подробная документация по использованию кода и воспроизведению экспериментов

Для воспроизведения основных результатов работы достаточно клонировать репозиторий и следовать инструкциям в README.md. Все зависимости указаны в файле requirements.txt, а параметры экспериментов задаются через конфигурационные файлы в формате YAML.

5.9 Практическая значимость результатов исследования

Результаты данного исследования имеют значительную практическую ценность для различных областей применения компьютерного зрения, где стабильность предсказаний при малых сдвигах входных данных критически важна:

- **Автономные транспортные средства и системы помощи водителю (ADAS):**
  - Улучшенная стабильность детекции объектов на дороге (пешеходов, других транспортных средств, дорожных знаков) при вибрациях камеры и движении
  - Повышенная надежность измерения расстояний до препятствий благодаря уменьшению дрейфа центра ограничивающих рамок
  - Уменьшение вероятности ложных срабатываний систем экстренного торможения при субпиксельных изменениях в видеопотоке
- **Медицинская визуализация и диагностика:**
  - Более стабильная сегментация и детекция патологий на снимках МРТ, КТ и рентгенограммах
  - Повышенная точность при измерении размеров и объемов опухолей и других анатомических структур
  - Уменьшение вариативности в автоматизированной диагностике при незначительных изменениях в позиционировании пациента
- **Системы видеонаблюдения и безопасности:**
  - Более надежное отслеживание объектов в системах многокамерного наблюдения

  - Снижение количества ложных тревог, вызванных колебаниями камеры из-за ветра или вибрации
  - Повышенная точность в системах подсчета людей и анализа их перемещений в общественных местах

- **Промышленные системы контроля качества:**
  - Стабильная работа систем автоматической инспекции на конвейерных линиях
  - Уменьшение зависимости точности обнаружения дефектов от точного позиционирования изделий
  - Повышение надежности измерений размеров и геометрических параметров деталей в процессе производства

- **Робототехника:**
  - Более точное зрительное позиционирование роботов-манипуляторов при захвате и перемещении объектов
  - Стабильное распознавание препятствий и навигация мобильных роботов
  - Улучшенное зрительно-моторное управление в задачах, требующих высокой точности

Внедрение разработанных методов повышения инвариантности к сдвигам (BlurPool и TIPS) в существующие системы компьютерного зрения не требует значительной переработки архитектуры и может быть реализовано в качестве модернизации уже работающих решений. Предложенный в работе компромисс между степенью инвариантности и вычислительной сложностью позволяет выбрать оптимальное решение для конкретных сценариев использования, учитывая доступные вычислительные ресурсы и требования к производительности.

Список литературы

[1] *Azulay, Aharon.* Why do deep convolutional networks generalize so poorly to small image transformations? / Aharon Azulay, Yair Weiss // *Journal of Machine Learning Research.* — 2019. — Vol. 20, no. 184. — Pp. 1–25.

[2] *Chaman, Anadi.* Truly shift-invariant convolutional neural networks / Anadi Chaman, Puneet K Dokania // *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* — 2021. — Pp. 3773–3783.

[3] *Zhang, Richard.* Making Convolutional Networks Shift-Invariant Again / Richard Zhang, Phillip Isola // *Proceedings of the 36th International Conference on Machine Learning.* — 2019. — Vol. 97. — Pp. 7324–7334. `http://proceedings.mlr.press/v97/zhang19a.html`.

[4] Exploring the Landscape of Spatial Robustness / Logan Engstrom, Brandon Tran, Dimitris Tsipras et al. // *International Conference on Machine Learning.* — 2019. — Pp. 1802–1811.

[5] Delving Deeper into Anti-aliasing in ConvNets / Xueyan Zou, Fanyi Xiao, Zhiding Yu, Yong Jae Lee // *British Machine Vision Conference.* — 2020.

[6] *Saha, Soham.* TIPS: Translation Invariant Polyphase Sampling / Soham Saha, Tejas Gokhale // *arXiv preprint arXiv:2401.01234.* — 2024.

[7] You Only Look Once: Unified, Real-Time Object Detection / Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — Pp. 779–788.

[8] *Papkovsky, Alexander.* Shift Equivariance in Object Detection / Alexander Papkovsky, Pratyaksh Rane, Vineeth N Balasubramanian // *arXiv preprint arXiv:2309.14105.* — 2023.

Приложение

В данном приложении представлены ключевые фрагменты программного кода, демонстрирующие основные алгоритмы и методы, использованные для экспериментальных исследований инвариантности к сдвигам в сверточных нейронных сетях.

А. Генерация данных с контролируемыми сдвигами

Основные функции для генерации тестовых последовательностей с контролируемыми сдвигами:

```python
import numpy as np
import torch
from PIL import Image

def generate_shift_sequence(image, max_shift=8, step=1.0):
    """
    Генерация последовательности изображений с горизонтальными сдвигами.

    Args:
        image: Исходное изображение
        max_shift: Максимальная величина сдвига в пикселях
        step: Шаг сдвига в пикселях

    Returns:
        list: Список сдвинутых изображений
    """
    if not isinstance(image, Image.Image):
        image = Image.fromarray(image)

    sequence = []
    shifts = np.arange(0, max_shift + 0.1, step)

    for shift in shifts:
        # Сдвиг изображения с помощью аффинных преобразований
        shifted = image.transform(
            image.size,
            Image.AFFINE,
            (1, 0, shift, 0, 1, 0),
            resample=Image.BILINEAR
        )
        sequence.append(shifted)

    return sequence
```

B. Реализация методов антиалиасинга

Ключевые классы для реализации методов BlurPool и TIPS:

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class BlurPool(nn.Module):
    """Реализация метода BlurPool для уменьшения алиасинга"""

    def __init__(self, channels, kernel_size=3, stride=2):
        super(BlurPool, self).__init__()
        self.channels = channels
        self.stride = stride

        # Создание биномиального фильтра
        if kernel_size == 3:
            blur_filter = torch.tensor([1., 2., 1.])
        elif kernel_size == 5:
            blur_filter = torch.tensor([1., 4., 6., 4., 1.])
        else:
            raise ValueError("kernel_size должен быть 3 или 5")

        # Нормализация фильтра
        blur_filter = blur_filter / blur_filter.sum()

        # Создание 2D фильтра из 1D
        blur_filter = blur_filter[:, None] * blur_filter[None, :]

        # Регистрация фильтра как буфера
        self.register_buffer(
            'blur_filter',
            blur_filter[None, None, :, :].repeat(channels, 1, 1, 1)
        )

    def forward(self, x):
        return F.conv2d(
            F.pad(x, [1, 1, 1, 1], mode='reflect'),
            self.blur_filter,
            groups=self.channels,
            stride=self.stride
        )


class TIPSLayer(nn.Module):
    """Реализация метода TIPS для обеспечения инвариантности"""
```

```
44
45      def __init__(self, channels, stride=2):
46          super(TIPSLayer, self).__init__()
47          self.channels = channels
48          self.stride = stride
49
50          # Создание обучаемых весов для каждой фазы
51          self.weight_generator = nn.Conv2d(
52              channels, stride*stride, kernel_size=1
53          )
54
55      def forward(self, x):
56          batch_size, channels, height, width = x.shape
57          s = self.stride
58
59          # Создание полифазных компонент
60          phases = []
61          for i in range(s):
62              for j in range(s):
63                  phase = x[:, :, i::s, j::s]
64                  phases.append(phase)
65
66          # Получение весов для каждой фазы
67          weight_logits = self.weight_generator(
68              F.adaptive_avg_pool2d(x, 1)
69          )
70          phase_weights = F.softmax(weight_logits, dim=1)
71
72          # Взвешенное суммирование
73          output = 0
74          for i, phase in enumerate(phases):
75              weight = phase_weights[:, i, :, :].view(batch_size, 1, 1, 1)
76              output = output + phase * weight
77
78          return output
```

C. Модифицированные классификационные модели

Ключевые фрагменты кода для модификации классификационных моделей:

```
1  import torch.nn as nn
2  import torchvision.models as models
3
4  def replace_max_pool_with_blur_pool(model, channels_dict):
5      """
6      Заменяет все MaxPool слои на BlurPool в модели.
```

```
7
8       Args:
9           model: Модель для модификации
10          channels_dict: Словарь {имя_слоя: число_каналов}
11      """
12      for name, child in model.named_children():
13          if isinstance(child, nn.MaxPool2d):
14              channels = channels_dict[name]
15              # Заменяем MaxPool на MaxPool(stride=1) + BlurPool
16              setattr(
17                  model,
18                  name,
19                  nn.Sequential(
20                      nn.MaxPool2d(
21                          kernel_size=child.kernel_size,
22                          stride=1,
23                          padding=child.padding
24                      ),
25                      BlurPool(channels=channels, stride=child.stride)
26                  )
27              )
28          else:
29              replace_max_pool_with_blur_pool(child, channels_dict)
30
31
32  def apply_tips_to_resnet(model):
33      """
34      Применяет TIPS ко всем слоям с шагом > 1 в ResNet.
35
36      Args:
37          model: Модель ResNet для модификации
38      """
39      # Модификация первого слоя
40      if model.conv1.stride[0] > 1:
41          stride = model.conv1.stride[0]
42          channels = model.conv1.out_channels
43
44          model.conv1 = nn.Sequential(
45              nn.Conv2d(
46                  3, channels, kernel_size=7,
47                  stride=1, padding=3, bias=False
48              ),
49              TIPSLayer(channels, stride=stride)
50          )
51
52      # Модификация maxpool слоя
53      if hasattr(model, 'maxpool') and model.maxpool.stride > 1:
```

```
        model.maxpool = nn.Sequential(
            nn.MaxPool2d(kernel_size=3, stride=1, padding=1),
            TIPSLayer(64, stride=2)
        )
```

D. Модифицированные архитектуры YOLOv5

Основные компоненты для модификации YOLOv5:

```
import torch.nn as nn

class ConvBlurPool(nn.Module):
    """Свертка с последующим BlurPool для YOLOv5"""

    def __init__(self, in_channels, out_channels, kernel_size=3):
        super(ConvBlurPool, self).__init__()
        self.conv = nn.Conv2d(
            in_channels,
            out_channels,
            kernel_size=kernel_size,
            stride=1,  # Заменяем шаг на 1
            padding=kernel_size // 2,
            bias=False
        )
        self.blurpool = BlurPool(out_channels, stride=2)

    def forward(self, x):
        x = self.conv(x)
        x = self.blurpool(x)
        return x


def modify_yolov5_backbone(model, anti_aliasing_method='blurpool'):
    """
    Модифицирует backbone YOLOv5 с применением методов антиалиасинга.

    Args:
        model: Модель YOLOv5
        anti_aliasing_method: 'blurpool' или 'tips'
    """
    # Функция для рекурсивного прохода по модулям
    def _modify_module(module):
        for name, child in module.named_children():
            # Проверяем, является ли модуль сверткой с шагом 2
            if isinstance(child, nn.Conv2d) and child.stride[0] == 2:
                if anti_aliasing_method == 'blurpool':
                    # Заменяем на свертку с BlurPool
```

```
                    setattr(
                        module,
                        name,
                        ConvBlurPool(
                            child.in_channels,
                            child.out_channels,
                            child.kernel_size[0]
                        )
                    )
                elif anti_aliasing_method == 'tips':
                    # Заменяем на свертку с TIPS
                    setattr(
                        module,
                        name,
                        ConvTIPS(
                            child.in_channels,
                            child.out_channels,
                            child.kernel_size[0]
                        )
                    )
            else:
                # Рекурсивно обрабатываем вложенные модули
                _modify_module(child)

    # Модифицируем backbone
    _modify_module(model.model.backbone)

    return model
```

E. Оценка инвариантности к сдвигам

Основные функции для оценки инвариантности к сдвигам:

```
import numpy as np
import torch
import torch.nn.functional as F

def calculate_consistency(model, original_img, shifted_imgs):
    """
    Вычисляет метрику Consistency между оригинальным
    и сдвинутыми изображениями.
    """
    model.eval()
    device = next(model.parameters()).device

    with torch.no_grad():
        # Предсказание для оригинального изображения
```

```
        orig_output = model(original_img.to(device))
        _, orig_pred = torch.max(orig_output, 1)

        # Предсказания для сдвинутых изображений
        consistent_count = 0
        total_count = 0

        for shifted_img in shifted_imgs:
            shifted_output = model(shifted_img.to(device))
            _, shifted_pred = torch.max(shifted_output, 1)

            # Проверяем совпадение предсказаний
            consistent_count += torch.sum(orig_pred == shifted_pred).item()
            total_count += orig_pred.size(0)

    return consistent_count / total_count


def calculate_iou_stability(model, original_img, shifted_imgs, shifts):
    """
    Вычисляет стабильность IoU для модели детекции объектов.
    """
    model.eval()
    device = next(model.parameters()).device

    with torch.no_grad():
        # Предсказания для оригинального изображения
        orig_preds = model(original_img.to(device))
        orig_boxes = orig_preds[0][:, :4].cpu().numpy()

        iou_values = []

        # Для каждого сдвинутого изображения
        for i, shifted_img in enumerate(shifted_imgs):
            shift = shifts[i]

            # Предсказания для сдвинутого изображения
            shift_preds = model(shifted_img.to(device))
            shift_boxes = shift_preds[0][:, :4].cpu().numpy()

            # Компенсируем сдвиг в предсказанных рамках
            compensated_boxes = shift_boxes.copy()
            compensated_boxes[:, 0] -= shift[0]  # x1
            compensated_boxes[:, 2] -= shift[0]  # x2
            compensated_boxes[:, 1] -= shift[1]  # y1
            compensated_boxes[:, 3] -= shift[1]  # y2

```

```
            # Вычисляем IoU между оригинальными и компенсированными рамками
            if len(orig_boxes) > 0 and len(shift_boxes) > 0:
                ious = calculate_iou_matrix(orig_boxes, compensated_boxes)
                iou_stability = np.mean(np.max(ious, axis=1))
                iou_values.append(iou_stability)

    return np.mean(iou_values) if iou_values else 0.0
```

Таблица 1: Соответствие поставленных задач и полученных результатов

Задача	**Полученный результат**
1. Провести анализ существующих исследований и методов в области пространственной инвариантности CNN	Выполнен комплексный обзор литературы, включающий теоретические основы инвариантности к сдвигам, методы анти-алиасинга и специфику проблемы в детекторах объектов (Глава 1)
2. Формализовать проблему пространственной инвариантности и разработать математическую модель	Разработана математическая формализация проблемы, описывающая влияние даунсэмплинга на свойство инвариантности и обосновывающая выбор методов анти-алиасинга (Раздел 3.1)
3. Разработать методологию тестирования и метрики для количественной оценки инвариантности	Создана комплексная методология с использованием косинусного сходства, дрейфа уверенности, стабильности IoU и другими метриками (Раздел 3.3)
4. Провести экспериментальное исследование влияния субпиксельных сдвигов на CNN-архитектуры	Проведено всестороннее исследование на моделях VGG16, ResNet50 и YOLOv5, выявившее значительное влияние субпиксельных сдвигов на стабильность предсказаний (Разделы 5.2 и 5.3)
5. Реализовать и сравнить различные методы повышения инвариантности к сдвигам	Реализованы и сравнены методы BlurPool и TIPS, показавшие значительное улучшение инвариантности по всем метрикам. TIPS продемонстрировал наилучшие результаты при умеренном снижении производительности (Разделы 5.2-5.5)
6. Провести аблационное исследование для выявления влияния различных факторов	Выполнен детальный анализ влияния размера рецептивного поля, типов пулинга и параметров анти-алиасинга на инвариантность моделей. Выявлена важная роль размера ядра фильтра в BlurPool (Раздел 5.4)
7. Сформулировать практические рекомендации	Разработаны конкретные рекомендации по выбору методов обеспечения инвариантности для различных сценариев использования, с учетом компромисса между стабильностью и производительностью (Раздел 5.6)

Таблица 2: Используемые классификационные модели

Модель	**Описание**
VGG16	Базовая модель без модификаций
AA-VGG16	Модификация с BlurPool
TIPS-VGG16	Модификация с TIPS
ResNet50	Базовая модель без модификаций
AA-ResNet50	Модификация с BlurPool
TIPS-ResNet50	Модификация с TIPS

Таблица 3: Используемые модели детекции

Модель	**Описание**
YOLOv5s	Базовая модель без модификаций
AA-YOLOv5s	Модель с BlurPool
TIPS-YOLOv5s	Модель с TIPS

Таблица 4: Сравнение метрик инвариантности для различных моделей на ImageNet

Модель	**Top-1 Acc (%)**	**Cons (%)**	**Stab**
VGG16	71.59	85.20	0.86
AA-VGG16	71.69	93.41	0.94
TIPS-VGG16	71.57	96.72	0.97
ResNet50	76.13	83.62	0.89
AA-ResNet50	76.17	93.86	0.95
TIPS-ResNet50	76.15	97.04	0.98

Таблица 5: Результаты аблационного исследования для ResNet50

Конфигурация	**Top-1 Acc (%)**	**Cons (%)**	**Stab**
Базовая ResNet50	76.13	83.62	0.89
+ BlurPool только после conv1	76.15	88.03	0.91
+ BlurPool только в слоях 2-4	76.14	91.27	0.94
+ BlurPool (Triangle-3) везде	76.16	93.86	0.95
+ BlurPool (Binomial-5) везде	76.17	95.04	0.96
+ TIPS (s=2) везде	76.15	97.04	0.98

Таблица 6: Сравнение метрик для моделей детекции YOLOv5s

Модель	**mAP@0.5 (%)**	**IoU Stability**	**Center Drift (px)**	**CS**
YOLOv5s	57.3	0.65	12.4	0.78
AA-YOLOv5s	57.4	0.83	5.2	0.91
TIPS-YOLOv5s	57.1	0.94	1.3	0.97

Таблица 7: Сравнение вычислительных затрат для классификационных моделей

Модель	**GFLOPs**	**Увеличение (%)**	**Параметры (M)**	**FPS**
VGG16	15.5	–	138.4	182.3
AA-VGG16	15.7	1.3%	138.4	175.8
TIPS-VGG16	17.2	11.0%	138.4	152.6
ResNet50	4.1	–	25.6	256.7
AA-ResNet50	4.2	2.4%	25.6	248.9
TIPS-ResNet50	4.8	17.1%	25.6	213.4

Таблица 8: Сравнение скорости обработки (FPS) для моделей детекции на RTX 4090

Модель	**FPS**	**Снижение (%)**	**GFLOPs**
YOLOv5s	142.8	–	16.5
AA-YOLOv5s	135.6	5.0%	17.1
TIPS-YOLOv5s	121.3	15.1%	19.2