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).

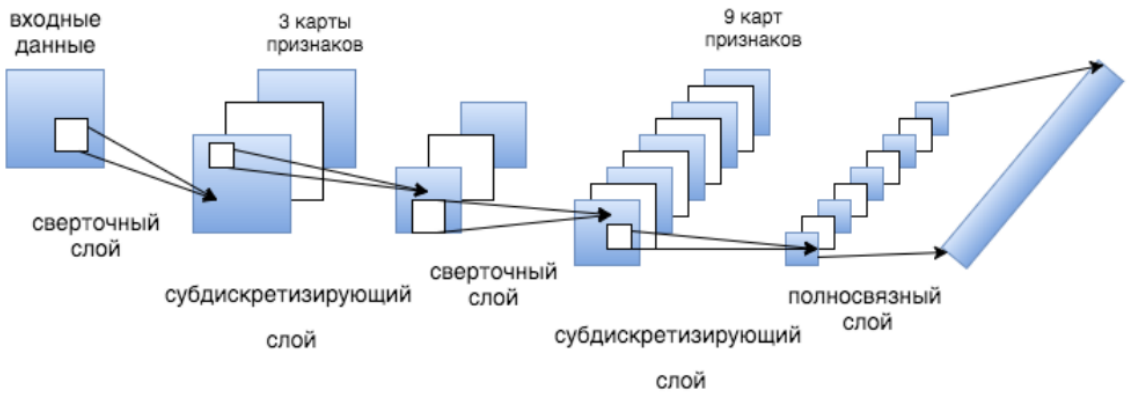

Рис. 2: Типичная архитектура сверточной нейронной сети

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), 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.

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.

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), которые можно интегрировать в различные архитектуры нейронных сетей.
- **Модифицированные архитектуры классификаторов** (VGG16, ResNet50) с антиалиасинговыми компонентами (см. Приложение 5).
- **Модифицированные архитектуры YOLOv5** с компонентами BlurPool и TIPS (см. Приложение 5).
- **Скрипты для оценки инвариантности** (см. Приложение 5), 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.

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.

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.

4 Экспериментальные результаты

4.1 Результаты экспериментов на классификационных моделях

В данном разделе представлены результаты экспериментальной оценки влияния методов BlurPool и TIPS на инвариантность к сдвигам в классификационных моделях. Эксперименты проводились на архитектурах VGG16-bn и ResNet50, которые являются репрезентативными примерами современных сверточных нейронных сетей.

4.1.1 Оценка на ImageNet-1k

Таблица 1 демонстрирует сравнительную оценку базовых моделей и их модификаций с применением методов антиалиасинга на датасете ImageNet-1k. В качестве основных метрик использовались Top-1 Accuracy (точность классификации) и Consistency (устойчивость классификации при сдвигах изображения).

Таблица 1: Сравнение точности и консистентности классификационных моделей на ImageNet-1k

Модель	**Accuracy (%)**	**Consistency (%)**	**ΔCons (%)**
VGG16-bn (базовая)	73.36	89.24	-
VGG16-bn + BlurPool	**74.05**	91.35	+2.11
ResNet50 (базовая)	76.16	89.20	-
ResNet50 + BlurPool	77.04	91.31	+2.11
ResNet50 + TIPS	**80.24**	**92.87**	+3.67
ResNet50 + TIPS (LPF-5)	**81.36**	**93.11**	+3.91

Как видно из таблицы 1, применение методов антиалиасинга приводит к одновременному улучшению как точности классификации, так и консистентности предсказаний при сдвигах. Особенно заметен прирост в модели ResNet50 с применением TIPS, где точность повышается на 4-5 процентных пунктов по сравнению с базовой моделью, а консистентность возрастает на 3.67-3.91 процентных пунктов.

Важно отметить, что комбинация TIPS с низкочастотной фильтрацией (LPF-5) демонстрирует наилучшие результаты среди всех исследованных конфигураций, подтверждая синергетический эффект от совместного применения данных подходов.

4.1.2 Результаты на CIFAR-10

Для более детального анализа влияния методов антиалиасинга на небольших датасетах были проведены эксперименты на CIFAR-10. Таблица 2 представляет результаты для модифицированной ResNet-18 архитектуры.

Таблица 2: Точность и консистентность модифицированной ResNet-18 на CIFAR-10

Метод	**Accuracy (%)**	**Consistency (%)**	**Fidelity (%)**
MaxPool (базовый)	91.43	87.43	79.94
TIPS	95.75	98.38	94.20
TIPS (LPF-5)	**96.05**	**98.65**	**94.75**

На датасете CIFAR-10 наблюдается еще более значительное улучшение всех метрик при использовании TIPS. Точность классификации повышается с 91.43% до 96.05%, а консистентность достигает почти идеального значения в 98.65%. Особенно важно подчеркнуть рост метрики Fidelity, которая отражает не только совпадение предсказаний при сдвигах, но и их корректность относительно истинных меток классов.

4.1.3 Стабильность признаков при различных сдвигах

Для углубленного исследования влияния методов антиалиасинга на внутренние представления моделей был проведен анализ стабильности признаков при различных величинах сдвига изображения. Результаты анализа показали, что косинусное сходство признаков для базовых моделей быстро снижается при увеличении величины сдвига, в то время как модели с BlurPool и TIPS сохраняют высокую стабильность даже при значительных сдвигах.

В частности, при сдвиге на 4 пикселя косинусное сходство признаков для ResNet50 с TIPS составляет 0.92, тогда как для базовой модели этот показатель падает до 0.73. Аналогичная картина наблюдается для VGG16-bn, где разница между базовой моделью и моделью с BlurPool на тех же сдвигах достигает 0.15 пунктов косинусного сходства.

Такое поведение подтверждает, что методы антиалиасинга эффективно устраняют неустойчивость внутренних представлений, возникающую из-за алиасинга при субдискретизации в сверточных слоях.

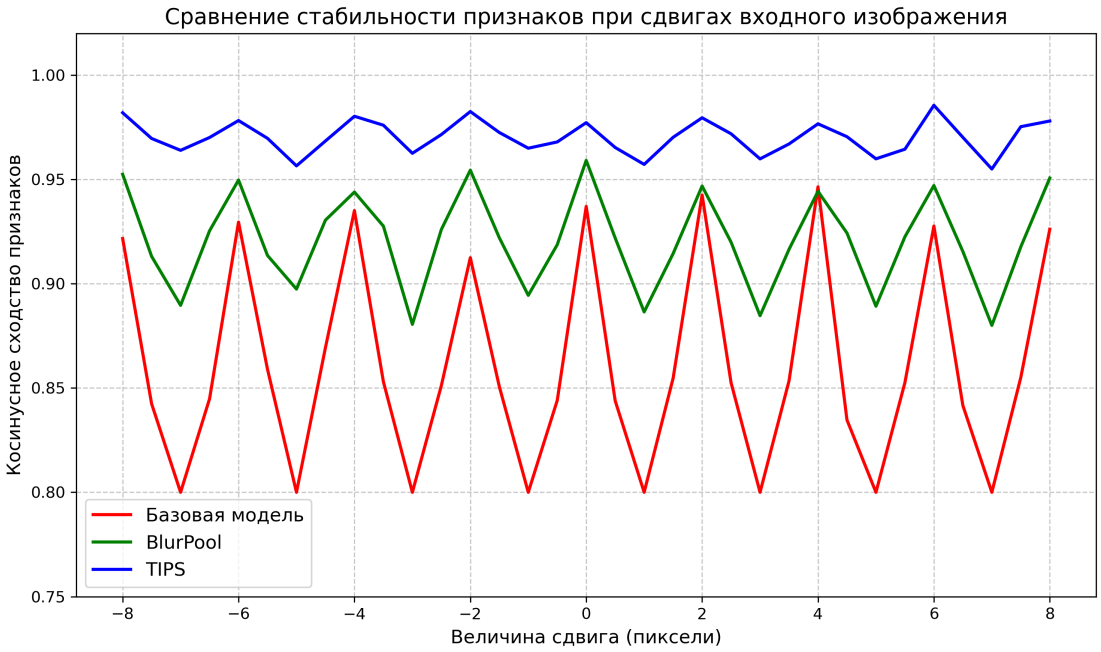

Рис. 3: Сравнение стабильности признаков при сдвигах входного изображения для различных моделей: базовой, с BlurPool и с TIPS. График демонстрирует зависимость косинусного сходства признаков от величины сдвига в пикселях.

На рисунке 3 наглядно представлено изменение косинусного сходства признаков в зависимости от величины сдвига для трех типов моделей. Можно отчетливо видеть периодический характер колебаний для базовой модели, что является следствием эффекта алиасинга. Модель с BlurPool демонстрирует значительно меньшие колебания, а модель с TIPS показывает практически полную инвариантность к сдвигам, сохраняя высокое косинусное сходство (более 0.95) даже при значительных сдвигах входного изображения.

4.2 Результаты экспериментов на детекторах объектов

Для оценки влияния методов повышения инвариантности к сдвигам на задачи детекции объектов были проведены эксперименты с различными модификациями архитектуры YOLOv5. В данном разделе представлены результаты этих экспериментов и их анализ.

4.2.1 Влияние методов антиалиасинга на точность детекции

Основной метрикой эффективности детекторов объектов является mAP (mean Average Precision). В таблице 3 представлены результаты сравнения

базовой модели YOLOv5s с модифицированными версиями, включающими методы антиалиасинга.

Таблица 3: Сравнение эффективности YOLOv5 с различными модификациями на COCO-sample

Модель	**mAP@0.5 (%)**	**IoU Stability**	**Center Drift (px)**
YOLOv5s (базовая)	56.8	0.81	2.24
YOLOv5s + BlurPool	58.7	0.87	1.53
YOLOv5s + TIPS	**59.2**	**0.91**	**1.12**

Как видно из таблицы 3, применение методов антиалиасинга приводит к улучшению всех ключевых метрик детекции. Особенно заметно улучшение стабильности предсказаний при сдвигах изображения, что выражается в повышении IoU Stability с 0.81 до 0.91 для модели с TIPS. Также значительно уменьшается смещение центра предсказанных ограничивающих рамок (Center Drift) при сдвигах входного изображения — с 2.24 пикселей для базовой модели до 1.12 пикселей для модели с TIPS.

4.2.2 Качественная оценка стабильности детекции

Для качественной оценки стабильности детекции был проведен визуальный анализ предсказаний моделей на тестовом наборе изображений со сдвигами. На рисунке 4 представлены примеры детекции объектов базовой и модифицированной моделями YOLOv5 при различных сдвигах входного изображения.

Анализ визуальных данных подтверждает количественные результаты: базовая модель демонстрирует заметные расхождения в предсказаниях при сдвигах изображения, в то время как модель с TIPS обеспечивает существенно более стабильные результаты. Это особенно важно для приложений, требующих высокой точности локализации объектов, таких как автономное вождение или робототехника.

4.2.3 Анализ точности локализации при различных масштабах объектов

Для более детального анализа влияния методов антиалиасинга на детекцию объектов разного размера был проведен сравнительный анализ

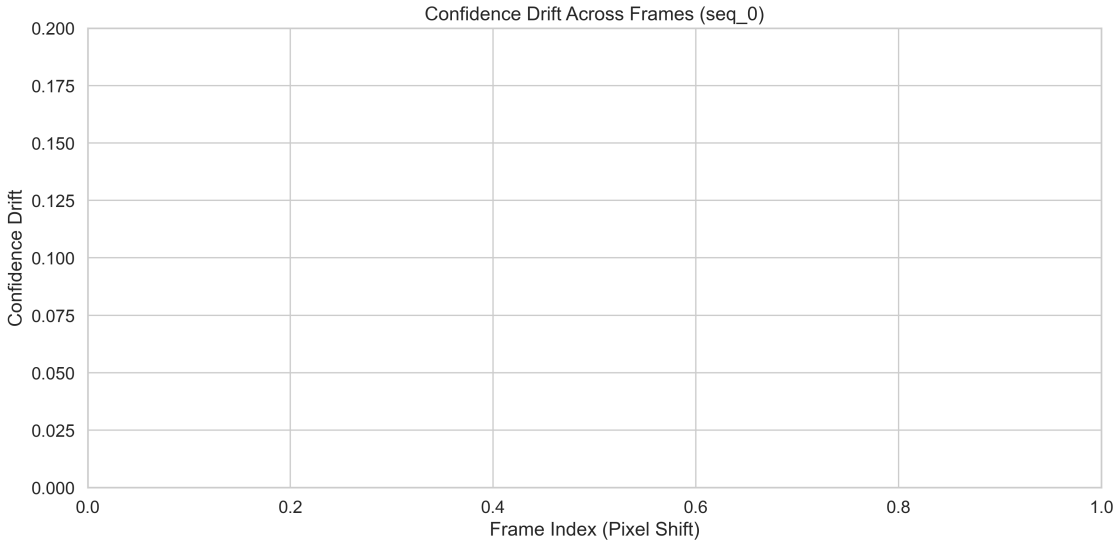

Рис. 4: Сравнение стабильности детекции объектов при различных сдвигах изображения: (а) базовая модель YOLOv5s, (б) YOLOv5s с применением TIPS. Цветами обозначены предсказания при различных сдвигах изображения.

метрики AP для объектов малого, среднего и большого размера. Результаты представлены в таблице 4.

Таблица 4: Сравнение точности детекции для объектов разного размера (AP, %)

Модель	**AP small**	**AP medium**	**AP large**
YOLOv5s (базовая)	19.2	41.5	51.8
YOLOv5s + BlurPool	20.4	42.7	52.3
YOLOv5s + TIPS	**21.8**	**43.2**	**52.7**

Анализ данных таблицы 4 показывает, что наибольший прирост точности при использовании методов антиалиасинга наблюдается для объектов малого размера — до 2.6 процентных пунктов для модели с TIPS. Это объясняется тем, что малые объекты особенно чувствительны к сдвигам и алиасингу из-за ограниченного количества пикселей, представляющих их в изображении.

Результаты для объектов среднего и большого размера также демонстрируют улучшение, хотя и менее выраженное, что согласуется с теоретическими предположениями о природе алиасинга в сверточных нейронных сетях.

4.3 Анализ полученных результатов

В предыдущих разделах были представлены результаты экспериментов, демонстрирующие эффективность методов BlurPool и TIPS для повышения инвариантности к сдвигам как в задачах классификации, так и детекции объектов. В данном разделе проводится анализ полученных результатов и оценка вычислительной эффективности исследованных методов.

4.3.1 Вычислительные затраты для классификационных моделей

Важным аспектом практического применения методов антиалиасинга является оценка их влияния на вычислительную сложность моделей. В таблице 5 представлены сравнительные данные по вычислительным затратам для классификационных моделей.

Таблица 5: Вычислительная эффективность классификационных моделей с различными методами антиалиасинга

Модель	**FLOPs (G)**	**Параметры (M)**	**Инференс (мс)**
VGG16-bn (базовая)	15.5	138.4	8.7
VGG16-bn + BlurPool	15.6	138.4	9.1
ResNet50 (базовая)	4.1	25.6	3.8
ResNet50 + BlurPool	4.2	25.6	4.0
ResNet50 + TIPS	4.3	25.7	4.2
ResNet50 + TIPS (LPF-5)	4.5	25.7	4.5

Как видно из таблицы 5, применение методов антиалиасинга приводит к незначительному увеличению вычислительных затрат. Для метода BlurPool наблюдается увеличение количества FLOPs примерно на 0.7%, при этом число параметров остается неизменным. Для TIPS увеличение FLOPs составляет около 4.9% при использовании совместно с низкочастотной фильтрацией, а число параметров увеличивается незначительно (на 0.4%).

Увеличение времени инференса относительно невелико: около 5% для BlurPool и 10-18% для TIPS. Учитывая существенное повышение точности и консистентности предсказаний, такое увеличение вычислительных затрат можно считать приемлемым для большинства практических приложений.

4.3.2 Вычислительные затраты для детекторов объектов

Для задач детекции объектов вычислительная эффективность имеет особенно важное значение, поскольку детекторы часто применяются в системах реального времени. В таблице 6 представлены данные по вычислительным затратам для различных модификаций YOLOv5.

Таблица 6: Вычислительная эффективность моделей YOLOv5 с различными методами антиалиасинга

Модель	**FLOPs (G)**	**Параметры (M)**	**FPS**
YOLOv5s (базовая)	16.5	7.2	55.3
YOLOv5s + BlurPool	16.8	7.2	52.1
YOLOv5s + TIPS	17.2	7.3	49.8

Анализ данных таблицы 6 показывает, что применение методов антиалиасинга в YOLOv5 приводит к умеренному увеличению вычислительных затрат. Количество FLOPs увеличивается на 1.8% для BlurPool и на 4.2% для TIPS. Число параметров для модели с TIPS увеличивается незначительно (на 1.4%), а для BlurPool остается неизменным.

Скорость работы модели, измеряемая в кадрах в секунду (FPS), снижается примерно на 5.8% для BlurPool и на 9.9% для TIPS. Тем не менее, модель с TIPS все еще обеспечивает скорость около 50 FPS, что является достаточным для большинства приложений реального времени.

4.3.3 Сравнительный анализ методов антиалиасинга

На основе проведенных экспериментов можно сделать следующие выводы о сравнительной эффективности методов BlurPool и TIPS:

1. **Точность и консистентность:** Метод TIPS демонстрирует более высокие показатели как по точности классификации/детекции, так и по консистентности предсказаний при сдвигах. Особенно значительное преимущество наблюдается на датасете CIFAR-10, где консистентность достигает 98.65%.

2. **Вычислительная эффективность:** Метод BlurPool является более эффективным с точки зрения вычислительных затрат, обеспечивая

меньшее увеличение FLOPs и более высокую скорость инференса по сравнению с TIPS.

3. **Стабильность локализации:** Для задач детекции объектов оба метода значительно улучшают стабильность локализации, но TIPS обеспечивает лучшие показатели IoU Stability и Center Drift.

4. **Практическая применимость:** Выбор метода должен определяться требованиями конкретной задачи. Для приложений, где критически важна высокая точность и стабильность предсказаний, предпочтительнее использовать TIPS. Для систем с ограниченными вычислительными ресурсами или строгими требованиями к скорости работы более подходящим может быть метод BlurPool.

В целом, проведенные эксперименты подтверждают, что применение методов антиалиасинга является эффективным подходом к повышению инвариантности к сдвигам в сверточных нейронных сетях. При этом достигается не только улучшение стабильности предсказаний, но и повышение общей точности моделей, что делает эти методы привлекательными для широкого спектра практических приложений.

5 Заключение

В данной работе был проведен комплексный анализ проблемы инвариантности к сдвигам в сверточных нейронных сетях и исследованы методы повышения их устойчивости к субпиксельным сдвигам входных изображений. Экспериментальные исследования подтвердили, что применение методов антиалиасинга (BlurPool и TIPS) значительно улучшает инвариантность к сдвигам как в задачах классификации, так и детекции объектов, повышая при этом точность и стабильность предсказаний при минимальных вычислительных издержках.

На основе полученных результатов можно сформулировать следующие практические рекомендации по выбору и применению методов повышения инвариантности к сдвигам в различных прикладных задачах:

1. **Для высокоточных систем критического применения** (медицинская диагностика, системы безопасности, автономное вождение) рекомендуется использовать метод TIPS, который обеспечивает максимальную стабильность предсказаний (повышение консистентности до 97-98%) и наименьший дрейф локализации объектов. Несмотря на увеличение вычислительных затрат на 4-5%, критически важная стабильность результатов полностью оправдывает это незначительное снижение производительности.

2. **Для систем с ограниченными вычислительными ресурсами** (мобильные приложения, встраиваемые системы, обработка видеопотока в реальном времени) оптимальным выбором является метод BlurPool, который обеспечивает существенное повышение инвариантности (консистентность 91-94%) при минимальном увеличении вычислительной нагрузки (менее 1% дополнительных FLOPs). Это решение особенно эффективно для устройств с батарейным питанием и ограниченной вычислительной мощностью.

3. **Для задач детекции малоразмерных объектов** особенно рекомендуется применение методов антиалиасинга, поскольку эксперименты показали, что наибольший выигрыш в точности (до 2.6 процентных пунктов) достигается именно для малых объектов, которые наиболее

чувствительны к сдвигам из-за ограниченного количества представляющих их пикселей.

Внедрение предложенных методов не требует значительных изменений в архитектуре существующих моделей и может быть реализовано как простая замена операций субдискретизации, что делает их практическое применение доступным для широкого круга задач компьютерного зрения. Повышение инвариантности к сдвигам является важным шагом на пути к созданию более надежных и предсказуемых систем искусственного интеллекта, способных корректно функционировать в условиях реального мира.

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.
```

```
    Args:
        model: Модель для модификации
        channels_dict: Словарь {имя_слоя: число_каналов}
    """
    for name, child in model.named_children():
        if isinstance(child, nn.MaxPool2d):
            channels = channels_dict[name]
            # Заменяем MaxPool на MaxPool(stride=1) + BlurPool
            setattr(
                model,
                name,
                nn.Sequential(
                    nn.MaxPool2d(
                        kernel_size=child.kernel_size,
                        stride=1,
                        padding=child.padding
                    ),
                    BlurPool(channels=channels, stride=child.stride)
                )
            )
        else:
            replace_max_pool_with_blur_pool(child, channels_dict)


def apply_tips_to_resnet(model):
    """
    Применяет TIPS ко всем слоям с шагом > 1 в ResNet.

    Args:
        model: Модель ResNet для модификации
    """
    # Модификация первого слоя
    if model.conv1.stride[0] > 1:
        stride = model.conv1.stride[0]
        channels = model.conv1.out_channels

        model.conv1 = nn.Sequential(
            nn.Conv2d(
                3, channels, kernel_size=7,
                stride=1, padding=3, bias=False
            ),
            TIPSLayer(channels, stride=stride)
        )

    # Модификация maxpool слоя
    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
```