Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Национальный исследовательский университет «Высшая школа экономики»

Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Сазанович Никита Валерьевич

SIM2REAL ДЛЯ АВТОНОМНОГО ВОЖДЕНИЯ С ИСПОЛЬЗОВАНИЕМ СКРЫТОГО ПРОСТРАНСТВА

Выпускная квалификационная работа

по направлению подготовки

01.04.02 Прикладная математика и информатика

образовательная программа

«Программирование и анализ данных»

Научный руководитель:
кандидат физико-математических наук, доцент департамента информатики,
Подкопаев Антон Викторович

Консультант:
заведующий центром анализа данных и машинного обучения,
Шпильман Алексей Александрович

Рецензент:
исследователь лаборатории алгоритмов мобильных роботов JetBrains,
Филатов Антон Юрьевич

Санкт-Петербург 2021

ОГЛАВЛЕНИЕ

ГЛАВА 1. ВВЕДЕНИЕ

В последнее время компании и исследователи демонстрируют все большие успехи в алгоритмах автономного вождения автомобилей. Реальностью становятся автопилоты автомобилей на магистралях и выполнение процедуры паркования. Значительный прогресс достигнут и в автопилотах общего назначения для езды в городе. Такие алгоритмы разрабатываются индустриальными компаниями, и в них, как правило, вкладывают множество ресурсов, лет и усилий людей.

Одним из направлений для создания таких алгоритмов являются методы машинного обучения. Классической работой в этой области была статья [1]. Авторы продемонстрировали алгоритм обучения, который оперировал на исходных изображениях и преобразовывал их в команды в пределах одной модели машинного обучения. Эта работа имела сильные эмпирические результаты. А именно, предложенный алгоритм управлял автомобилем на протяжении 10 миль автомагистрали без единого вмешательства человеческого водителя.

Создание таких алгоритмов также содержит следующие трудности. Во-первых, сбор данных в реальном мире для процедуры обучения, как правило, трудоемок и затратен. Во-вторых, выученная политика вождения строго следует использованной при сборе политике, и адаптация алгоритма к новому стилю вождения потребует сбора данных заново. В-третьих, для обеспечения безопасного поведения алгоритма в экстремальных условиях возникают проблемы этического характера при сборе данных и тестировании. К примеру, для корректного поведения в случае пешехода, перебегающего дорогу перед автомобилем, необходимо собрать и включить такие ситуации в тренировочный набор, что может быть проблематично. Наконец, трудностью таких алгоритмов является высокая размерность входных данных и факторов, которые задают их распределение. Чтобы учесть эти факторы, процедура сбора данных должна быть тщательно спланирована и учитывать возможные вариации окружения.

Использование симулятора для сбора данных и тестирования алгоритмов способно разрешить эти трудности. В то же время недавние исследования [2, 3] представляют алгоритмы автономного вождения, обученные на командах управления из симулятора, которые имеют сравнимые результаты во время эксплуатации в реальном мире.

В этой работе я разрабатываю такую модель машинного обучения для переноса алгоритма из симулятора в реальный мир (Sim2Real), провожу ее обуче-

ние и тестирование на платформе Duckietown и исследую улучшения модели, которые основаны на темпоральном порядке данных.

Практической частью моей работы, необходимой для дальнейшего исследования, был сбор и обработка симуляционных и реальных наборов данных для автономного вождения, реализация простейшего Sim2Real метода, основанного на базовой сверточной модели, и метода Sim2Real обучения с трансляцией изображений, не использующего меток из реального мира.

Исследовательской частью моей работы была разработка улучшений Sim2Real методов, которые вводят дополнительные ограничения на представления в скрытом пространстве, основанные на темпоральном порядке данных, и сравнение таких методов с другими подходами. Второй составляющей исследования было рассмотрение качества переноса таких методов для разных симуляционных датасетов.

Таким образом, моими задачами были: 1) сбор нескольких симуляционных датасетов и набора данных из реального мира; 2) реализация методов Sim2Real для переноса алгоритма автономного вождения из симулятора в реальный мир; 3) разработка дополнительных темпоральных ограничений для улучшения трансфера модели; 4) проведение экспериментов и сравнение результатов моделей и окружений для осуществления переноса; 5) анализ полученных моделей.

Структура этой работы следующая. В главе 2 я описываю предметную область автономного вождения. В главе 3 я рассматриваю релевантные работы для реализации таких алгоритмов и проведения трансфера. В главе 4 я описываю процесс сбора и обработки данных, реализацию методов Sim2Real и способ введения дополнительных темпоральных ограничений. В главе 5 я представляю результаты и анализ экспериментов. Наконец, в главе 6 я подвожу итоги работы и описываю дальнейшие направления ее развития.

ГЛАВА 2. ПРЕДМЕТНАЯ ОБЛАСТЬ

2.1. Автономное вождение

Задача автономного вождения заключается в управлении транспортным средством без участия человека. Транспортным средством может выступать как автомобиль, так и мобильный робот. Эта область является важным направлением разработки и исследований в последние годы. Было сформировано и активно

развивается множество компаний, целью которых является разработка алгоритма автономного вождения для того или иного транспортного средства.

Все подходы можно разбить на два больших класса: последовательный конвейер и обработка от начала до конца. Подход последовательного конвейера заключается в том, что весь алгоритм состоит из десятков модулей, каждый из которых отвечает за строго поставленную ему задачу, а результаты работы одного блока становятся входом следующего. В частности, первый блок получает в качестве входа данные датчиков, такие как камера и лидар, а результат последнего блока задает команды управления. В качестве примера, чаще всего приводится разбиение всего алгоритма на четыре части: восприятие, локализация, планирование и контроль. Исторически такие алгоритмы появились первыми и являлись стандартным методом построения индустриального автопилота. Второй подход обработки от начала до конца заключается в разработке блока или модели, которая совмещает все функции и задачи и напрямую преобразует сырые показания датчиков в команды управления. Этот подход обрел популярность с ростом использования машинного обучения и глубокого обучения в частности.

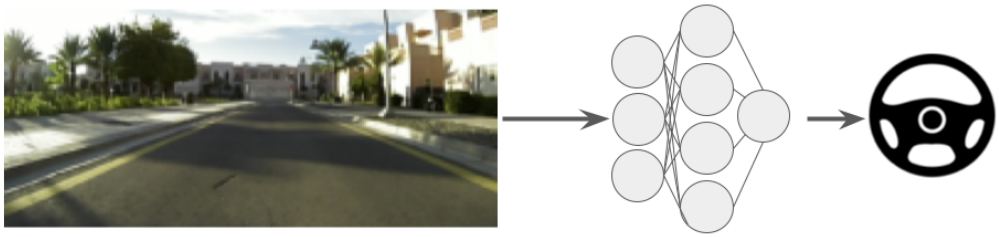

Рисунок 1 – Схема подхода машинного обучения от начала до конца для алгоритма автономного вождения.

Методы машинного обучения становятся повсеместными в алгоритмах автономного вождения. Эти методы способны принести пользу всему набору компонент алгоритма автономного вождения. С помощью машинного обучения можно улучшить алгоритм локализации, провести более точную сегментацию изображения, уточнить положения других объектов относительно транспортного средства или дополнить алгоритм планирования движения. Другим, как уже упоминалось ранее, более базовым подходом использования машинного обуче-

ния является создание модели для управления автомобилем от начала до конца. В таком подходе модель обучается выделять признаки из данных датчиков и преобразовывать их в команды управления. Принцип работы такого подхода схематично представлен на рисунке 1. В данной работе пойдет речь именно о таком построении алгоритма автономного вождения.

Такой подход имеет следующие преимущества и недостатки. Одним из наиболее важных преимуществ является простота такого подхода. Ранее алгоритм управления был составлен из десятка модулей, каждый из которых был сложно устроен и должен был согласовываться с другими. Для построения алгоритма в таком подходе нужен лишь набор данных пар изображение-команда, архитектура и время для обучения модели. Гипотеза заключается в том, что такая модель, имея доступ к исходным данным сенсоров, будет способна объединить восприятие, локализацию, планирование и контроль в операциях внутри модели и, что более важно, использовать их полнее и эффективнее. Ведь частая критика подхода последовательного конвейера состоит в том, что ошибка в одном из блоков сократит или исказит входные данные, и последующие блоки не смогут ее исправить и, как следствие, корректно выполнить контроль автомобиля. Совмещая все эти функции в рамках одной модели, результаты их работы могут взаимодействовать произвольным образом, создавать избыточность для предотвращения ошибок, и вводить новые или устранять привычные нам стадии обработки. Главные недостатки такого подхода заключаются в следующем. В то время как такая модель может работать корректно и предоставлять разумные команды, при непонятном поведении предсказаний нет возможности провести разбор, почему это поведение возникает. Такая модель представляет собой набор математических операций, и вещественные числа на промежуточных слоях модели не несут особого значения, а их рассмотрение не приводит к ответу. В последнее время были разработаны способы объяснения таких моделей машинного обучения, но они все еще не позволяют достичь ясности объяснения, которая возможна при подходе последовательного конвейера. Вторым важным недостатком является необходимость огромного объема данных для обучения таких моделей. Такие данные собираются месяцами и годами специальными операторами. Этот процесс трудоемкий и дорогостоящий. Более того, важен процесс сборки таких данных. К примеру, если автомобиль рассчитан на работу в снежную или нелетную погоду, то такие же данные должны быть широко

представлены в тренировочном наборе. Важно и соотношение таких данных со стандартными условиями, ведь если таких данных небольшое количество, то модель может потратить весь бюджет обучения на приспособление к стандартным условиями. Тем не менее такие алгоритмы обработки от начала до конца все чаще разрабатываются и используются индустриальными компаниями.

Рассмотрев преимущества и недостатки, перейдем к формальной постановке задачи автономного вождения методом обработки от начала до конца. В этой задаче мы имеем набор данных показаний сенсоров и команд управления $D = \{x_i, y_i\}_{i=1}^{|D|}$, где $x_i \in X$ – это входные данные с сенсоров, $y_i \in Y$ – соответствующие им команды управления. Сенсорами могут выступать камеры, лидар и другие устройства. Команды управления обычно представлены векторами небольшой размерности. Например, это может быть вектор размерности два, где первое число означает поворот рулевого колеса, а второе число показывает ускорение автомобиля. В задаче требуется предоставить политику вождения, или модель, $f : X \to Y$ для регрессии команд по данным сенсоров. Чаще всего такая модель является параметризованной. А именно, модель f имеет набор веществennозначных параметров $\theta \in R^N$ и, используя их, преобразует входные данные к результату $f_\theta : X \to Y$. В контексте этой работы, в качестве показаний сенсоров я рассматриваю изображение с фронтальной камеры, а в качестве команд управления я принимаю вектор размерности два, который представляет напряжения приводов на левом и правом колесах. Изображения до обработки будут иметь размер 640x480 и иметь три канала цвета. Такой выбор обоснован платформой сбора данных, которая будет описана в секции 4.1.

Оценивание таких политик вождения проводят метриками из двух категорий: онлайн и оффлайн. Онлайн метрики показывают насколько хорошо автомобиль управляется алгоритмом в режиме реального времени. Для этого автомобиль помещается на дорожную полосу, и контроль передается алгоритму. После того как проходит заданное время или автомобиль достигает назначения, высчитывается насколько хорошо эта задача была выполнена. К примеру, индустриальным стандартом считается частота вмешательств человека в процесс управления в среднем на 1000 километров. Такие метрики хорошо отражают финальную цель автопилота, но очень трудоемки и времязатратны. Оффлайн метрики характеризуют способность модели повторять команды экспертов по собранным данным. Такие метрики могут вычисляться автоматически и

используются для постоянной итерации над алгоритмом. В своей работе я буду оценивать модели используя оффлайн метрики. Для этого я буду использовать общепринятые среднее абсолютное отклонение (САО) и среднее квадратичное отклонение (СКО). Для вычисления этих метрик для модели f_θ необходим тестовый набор данных $D_{test} = \{x_i,y_i\}_{i=1}^{|D_{test}|}$. Тогда среднее абсолютное отклонение предсказаний модели вычисляется как:

$$\text{САО}(f_\theta, D_{test}) = \frac{1}{|D_{test}|}\sum_{i=1}^{D_{test}}\frac{1}{|Y|}||f_\theta(x_i) - y_i||_1 \tag{1}$$

а среднее квадратичное отклонение как:

$$\text{СКО}(f_\theta, D_{test}) = \frac{1}{|D_{test}|}\sum_{i=1}^{D_{test}}\frac{1}{|Y|}||f_\theta(x_i) - y_i||_2^2 \tag{2}$$

2.2. Sim2Real

Стандартным правилом в машинном обучении является использование тренировочного датасета, который имеет то же самое распределение или характеристики, что и тестовой набор данных, или область, в которой будет проводиться эксплуатация модели. А именно, если нам нужен автопилот для симулятора, то для обучения требуется лишь собрать данные в нем. А, чтобы обучить алгоритм автономного вождения для реального мира, нужен набор данных из реального мира, сбор которого является более затратной процедурой. Целью области Sim2Real является создание методов, которые будут обучаться на симуляционных данных, но при этом показывать хорошие результаты при эксплуатации обученной модели в реальном мире. Эта область отвечает на следующий вопрос: как, используя симуляционные данные, обучить модель для реального мира? Разработка таких методов имеет огромное практическое значение. При наличии такого метода пропадает необходимость в ресурсоемком сборе данных в реальном мире. Но такой трансфер осуществить на практике довольно сложно. Поэтому методы и исследования демонстрируют результаты при наличии небольшого набора данных в реальном мире или использовании входных данных из реального мира, но не конечных результатов. Стоит отметить, что сбор неразмеченных данных гораздо проще. Например, в открытом доступе находится огромное количество изображений, аудио файлов или текстов из реального мира. Затратным, как правило, является разметка таких данных.

Способом оценки методов Sim2Real является вычисление доменного несоответствия. Оценка основана на том факте, что обученные внутри домена модели, а именно для симулятора на симуляционных данных и для реального мира на реальных, достигают лучшего результата по сравнению с трансфером из других доменов. Они задают в некотором смысле верхнюю границу эффектности внутри домена. Тогда доменное несоответствие выражается тем, насколько эффективность модели ниже после переноса в целевой домен по сравнению с моделью обученной внутри домена. В области Sim2Real оценивается несоответствие эффективности перенесенной модели из симулятора с моделью, обученной на данных реального мира. Формальный способ определения такой оценки для автономного вождения будет представлен в секции 2.3.

Подходы Sim2Real можно объединить в следующие категории:

— метод рандомизации характеристик симуляционных доменов;
— перенос высокоуровневых политик;
— установление соответствия между входными данными симулятора и реального мира;
— и другие.

Я кратко опишу основные идеи подходов в каждой группе.

Одним из основополагающих методов Sim2Real стала рандомизация доменов [4]. Авторы этой статьи предложили обучать модель не в фиксированной версии симулятора, а в множестве версий симулятора, которые отличаются задающими их параметрами. Гипотеза авторов в том, что при обучении модели в десятке различных симуляторов реальный мир будет лишь очередной версией симулятора, и поэтому модель будет лучше готова к работе в ней. В статье авторы рассматривали задачу захвата объекта роботизированной рукой, и рандомизацию симулятора они проводили такими способами как изменение начального положения объекта, смена цвета стола, использование различного набора и количества посторонних объектов.

Мотивация для обучения высокоуровневых политик следующая. Весь процесс получения выходных данных может быть разбит на две части: выделение низкоуровневых признаков из входных данных и их дальнейшее преобразование в результат. Например, для задачи на изображениях низкоуровневыми признаками могут служить результаты сегментации, которые в дальнейшем используются для обучения высокоуровневой политики. Тогда при трансфере

такой модели, вторую часть можно полностью сохранить, а часть выделения низкоуровневых признаков обучить используя сторонние данные или небольшое количество размеченных реальных данных. К примеру, если переиспользовать результаты сегментации на датасете автономного вождения CityScapes [5] модели DenseASPP [6] для реального мира, то необходимости в сборе реальных данных для переноса высокоуровневой политики нет. Один из методов этой группы для задачи автономного вождения детально описан в секции 3.2.

Методы установления соответствия между входными данными нацелены на разрешение ковариативного сдвига, или сдвига входных данных. Работы в этой группе полагают, что входные данные в симуляторе и реальном мире имеют одинаковую структуру, но сдвинуты относительно друг друга. Восстановив это преобразование, модель в симуляторе можно перенести в реальный мир путем применения к данным из реального мира такого преобразования. Примером такого подхода служит следующая статья [7]. В ней авторы решают задачу захвата объекта роботизированной рукой путем обучения политики на командах из симулятора и одновременной трансляции изображения из симулятора в реальный мир для установления соответствия, используя генеративно-состязательные сети [8].

2.3. Sim2Real для автономного вождения

Sim2Real методы были разработаны и исследованы и для задачи автономного вождения. Целью методов в этом контексте является обучение политики управления в симуляторе и ее дальнейший перенос в реальный мир.

Более формально, в Sim2Real для автономного вождения имеются два набора данных: $D^{sim} = \{x_i,y_i\}_{i=1}^{|D^{sim}|}$ в симуляторе и $D^{real} = \{x_i,y_i\}_{i=1}^{|D^{real}|}$ в реальном мире. Как правило, эти датасеты разбиты на тренировочную, валидационную и тестовую части: $D^{sim}_{train}, D^{sim}_{val}, D^{sim}_{test}$ и $D^{real}_{train}, D^{real}_{val}, D^{real}_{test}$. При использовании оффлайн метрик частым выбором оценки произвольной модели f является вычисление САО и/или СКО из секции 2.1 на D^{real}_{test}. Задачей в Sim2Real является получение как можно лучшего результата метрик на тестовом наборе реальных данных, используя только данные из симулятора D^{sim}. Также в общей постановке задачи допускается использование неразмеченных примеров из реального мира $X^{real}_{train} = \{x_i\}_{i=1}^{|D^{real}_{train}|}$, которые состоят из множества изображений в D^{real}_{train}. Как видно, в такой постановке целью методов Sim2Real является либо обучение обобщающейся модели в симуляторе, либо использование реальных

изображений и пар изображение-команда в симуляторе, чтобы в результате получить модель для реального мира.

Методом оценки эффективности метода Sim2Real служит доменное несоответствие для модели. Эта оценка вводится относительно одной из метрик в секции 2.1. Определим доменное несоответствие для СКО. Чтобы ввести его, рассмотрим обученную на данных из реального мира модель f_{real}. Это модель с произвольной архитектурой, которая при обучении имеет доступ к полному набору данных D^{real}_{train} и D^{real}_{val} из реального мира, что запрещено в постановке Sim2Real, и поэтому она используется только для оценки. Так как эта модель обучается на реальных данных, то, как правило, значение полученной метрики СКО на D^{real}_{test} является нижней границей для всех методов Sim2Real, которые не имеют доступ к командам из реального мира. Тогда для любой модели Sim2Real $f_{sim2real}$ мы можем определить доменное несоответствие относительно СКО как:

$$GAP_{\text{СКО}}(f_{sim2real}, f_{real}, D^{real}_{test}) = \text{СКО}(f_{sim2real}, D^{real}_{test}) - \text{СКО}(f_{real}, D^{real}_{test}) \quad (3)$$

Обычно доменное несоответствие считается для фиксированной модели в реальном мире и тестового датасета. Поэтому чаще всего эта оценка будет зависеть только от рассматриваемой модели Sim2Real и записываться как $GAP_{\text{СКО}}(f_{sim2real})$. Доменное несоответствие относительно САО вводится аналогично.

По виду этой оценки, можно заметить, что значение близкое к 0 получится при вычислении этой метрики от модели, обученной в реальном мире. Если же мы вычислим доменное несоответствие для модели f_{sim}, обученной только в симуляторе, то $GAP_{\text{СКО}}(f_{sim})$ будет, как правило, верхней границей для любого метода Sim2Real. Поэтому, иногда рассматривают эффективность метода относительно того, насколько получается улучшить, то есть уменьшить, доменное несоответствие по сравнению с моделью, обученной только в симуляторе. Я буду использовать такой способ оценки для своего метода в главе 5.

В общих чертах Sim2Real методы для автономного вождения следуют классификации подходов в секции 2.2. В следующей главе я детально опишу два таких метода из двух разных классов: работу [2] для обучения политики управления по высокоуровневым признакам и работу [3] для установления со-

ответствия между распределениями картинок из симулятора и реальности для переноса контроллера, обученного только в симуляторе.

ГЛАВА 3. ОБЗОР ЛИТЕРАТУРЫ

3.1. Подход обучения алгоритма от начала до конца

В обзоре литературы, я сначала опишу классический подход обучения алгоритма от начала до конца. В статье [1] авторы представили один из первых методов машинного обучения единой модели для автономного вождения. В статье обучается традиционная сверточная нейронная сеть для преобразования исходных данных с единственной камеры в передней части автомобиля в команды управления. Авторы продемонстрировали хорошие результаты вождения алгоритмом в реальном мире, что было неожиданно и вызвало интерес к подходам обучения от начала до конца для алгоритмов автономного вождения.

Новизна этого подхода состояла в том, что обучение происходило, используя глубокое обучение на исходных данных без явной декомпозиции задачи на части, такие как детекция границ дороги или определение позиции автомобиля в полосе. Нейронная сеть автоматически строила внутренние представления, которые ей были необходимы, для корректных предсказаний заданных команд управления автомобиля. Авторы предположили, что такой подход лучше явной декомпозиции, так как модель имеет дополнительную степень свободы в том, какие признаки или представления вычислять для исходного изображения. Ведь в некоторой степени привычная декомпозиция задачи алгоритма вождения разбита на стадии, которые понятны человеку, но они могут быть не оптимальны. Другим преимуществом является возможность сокращения задержки, с которой вычисляются команды, по сравнению с методом последовательного конвейера, так как все стадии могут вычисляться одновременно в одной модели.

Технические детали этой реализации следующие. Авторами было собрано 72 часа данных вождения людьми. Данные были собраны на разных дорогах и в различных условиях освещения и погоды. Исследователи собрали большинство данных в центре города, но также включили данные с автомагистралей, туннелей и грунтовых дорог. Сбор выполнялся на тестовых автомобилях, которые записывали изображения с камер и повороты рулевого колеса, выполняемые человеческим водителем. Используя эти изображения и команды, была обучена сверточная сеть для предсказания управления. Модель минимизирова-

ла среднеквадратическую ошибку команд. Такая модель содержала около 250 тысяч вещественнозначных параметров и оперировала при частоте 30 кадров в секунду. Процедура обучения метода схематично изображена на рисунке 2.

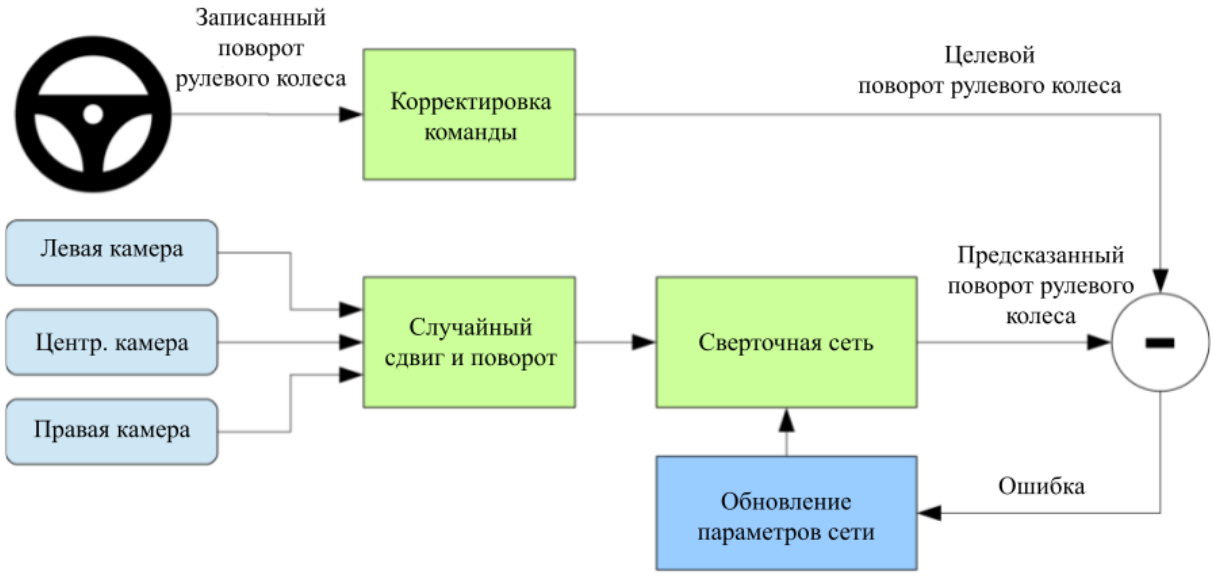

Рисунок 2 – Подход обучения алгоритма от начала до конца.

В качестве подтверждения метода, авторы показали 98% автономность их алгоритма вождения в городе и управление без вмешательств человеческого водителя на протяжении 10 миль автомагистрали.

3.2. Использование модульности и абстракций для переноса политики

Другой интересной статьей в области методов машинного обучения для автономного вождения стала работа [2]. Целью этой работы была разработка Sim2Real алгоритма, который бы показывал хорошую переносимость из симулятора в реальный мир благодаря модульности и абстракциям в архитектуре алгоритма.

Авторы заметили, что методы машинного обучения от начала до конца для автономного вождения часто требуют большого объема данных и сложны в обобщении для всех возможных дорожных условий. Ведь для того, чтобы алгоритм имел разумные предсказания для каких-то условий, кадры в этих же условиях, как правило, должны быть в тренировочном наборе. Симуляция таких условий представляет дешевую, безопасную и разнообразную среду для обучения. Но стандартные методы машинного обучения испытывают сложности в переносе обученных политик вождения из симулятора в реальность.

Для решения этой проблемы авторы предлагают подход, который находится на границе методов последовательного конвейера и обработки от начала до конца. Ключевой идеей стало использованием промежуточного представления изображения результатом его сегментации. Тогда управление и его обучение в симуляторе состоит из двух стадий: сегментация изображения в симуляторе и предсказания точек движения по этой сегментации. Для преобразования точек движения в команды управления авторы используют пропорционально-дифференцирующий регулятор [9]. Особенность же работы заключается в том, что модуль, который предсказывает точки движения по сегментации, может быть перенесен без изменений из симулятора в реальный мир. А именно, обучение модуля сегментации в используемом симуляторе не является трудозатратным, так как сегментация может быть легко получена используя внутреннее состояние симулятора во время сбора данных. А модуль сегментации для реального мира может быть взят из моделей в открытом доступе или обучен на стандартных наборах данных. В приведенной работе авторы самостоятельно обучают модули сегментации для архитектуры ERFNet [10] на публичном наборе данных CityScapes [5]. Таким образом, обучив лишь модуль предсказаний точек движения на командах из симулятора, можно осуществить эффективный перенос подхода из симулятора в реальный мир.

Авторы собрали данные для обучения политики управления в симуляторе с варьирующимися условиями и окружениями. Как результат было протестировано автономное управление мобильного робота алгоритмом в различных дорожных условиях. Авторами было выбрано три маршрута в городских условиях длиной 1.0, 0.7, 1.1 километров и оценено количество строгих и слабых нарушений, совершаемых во время их выполнения. Предложенный алгоритм, который был обучен полностью в симуляторе и не имел доступ к данным из реального мира, совершал не более одного строгого и не более пяти слабых нарушений на любом из маршрутов.

3.3. Метод трансляции изображений для обучения без команд реального мира

В работе [3] авторы представили альтернативный Sim2Real подход, использующий трансляцию изображений из симулятора в реальность и обратно, для установления соответствия между доменами и лучшего переноса в реальный мир. Мотивацией авторов послужило то же существующее несоответствие

между эффектностью моделей, которые обучены полностью в симуляционных условиях, в выполнении управления в симуляторе и реальном мире.

Для решения этого авторы предлагают использовать изображения из реального мира в качестве дополнительной информации для алгоритма. А именно, авторы основывают свой подход на возможности модели свободно преобразовывать условия в симуляторе к соответствующим условиям в реальном мире и обратно. Имея такое качественное и реалистичное преобразование, можно осуществить трансфер следующим образом. Можно обучить такое преобразование на изображениях из симулятора и реального мира и политику управления на парах изображение-команда в симуляторе. Тогда во время эксплуатации в реальном мире изображения, получаемые в реальных условиях, можно транслировать в соответствующие условия в симуляторе и использовать команду, которую модель предскажет для транслированных условий. Авторы улучшают описанную процедуру и используют промежуточное скрытое пространство, через которое происходят трансляции и предсказание команд. Тогда эксплуатация упрощается следующим образом. Чтобы получить команды управления в реальном мире, нужно перевести реальное изображение в это скрытое пространство и использовать уже обученный контроллер для предсказания команд по представлению в скрытом пространстве.

Для сбора симуляционных данных авторы использовали собственный симулятор с закрытым кодом. Авторы генерировали окружения и собирали оттуда изображения и команды агента-эксперта, которым выступал настроенный пропорциональный контроллер. Всего авторы собрали 77755 симуляционных примеров. Сбор данных в реальном мире для использования изображений в тренировочном процессе и оффлайн тестирования авторы проводили самостоятельно на дорогах без посторонних автомобилей. Человеческий водитель управлял автомобилем, который логировал изображения и команды. Всего авторы получили 77057 примеров из реального мира.

Предложенная модель была обучена на тренировочной части симуляционного и реального наборов данных. В качестве функции потерь для предсказания команд использовалось среднее абсолютное отклонение. Авторы получили хорошие результаты оффлайн метрик в реальности. Они также провели онлайн тестирование алгоритма, в котором ему было предоставлено управление на про-

тяжении 3 километров. Алгоритм авторов не потребовал вмешательств со стороны водителя во время всего маршрута, тем самым показав лучший результат.

3.4. Выводы

В литературе представлены алгоритмы, которые позволяют выучить алгоритм автономного вождения от начала до конца. Из-за сложностей сбора данных и адаптации к новым условиям, в последнее время в литературе активно появляются методы Sim2Real для этой задачи. Такие методы, как правило, оперируют над высокоуровневым представлением изображений или устанавливают соответствие между симуляционным и реальным доменом во время обучения. Однако в литературе не исследованы Sim2Real методы использующие ограничения, основанные на темпоральном порядке, который неотделим от данных непрерывного вождения.

ГЛАВА 4. МЕТОД

4.1. Сбор симуляционных и реальных данных

Для проведения исследования переноса моделей автономного вождения из симулятора в реальный мир мне необходимо было собрать и подготовить наборы данных. Сбор данных был устроен на платформе Duckietown [11]. Так как разработка алгоритмов на полноразмерных автономных автомобилях обходится дорого, многие исследователи прибегают к разработке и тестированию своих алгоритмов на миниатюрных версиях улиц и городов. Платформа Duckietown и была разработана с целью предоставления более широкого доступа к области автономного вождения. Я использовал эту платформу как для сбора реальных, так и для сбора симуляционных данных.

В данной работе я рассматриваю два симуляционных набора данных и один набор данных из реального мира. Симуляционные наборы отличаются картой, в рамках который был сгенерирован набор. Карты отличаются сложностью окружения и конфигурацией дороги, что оказывает влияние на требуемый алгоритм автономного вождения. Я собирал данные в каждом из этих доменов с расчетом на то, чтобы каждый набор имел приблизительно одинаковое количество примеров.

Мои данные состоят из изображений с единственной камеры, расположенной в передней части мобильного робота. Для каждого изображения

предоставлены команды человеческого водителя или настроенного алгоритма управления. Эти команды закодированы парой веществennозначных чисел $< \text{Vleft} >, < \text{Vright} >$, каждое из которых находится в диапазоне $[-1, 1]$, и обозначает напряжение привода левого и правого колеса соответственно. Для симуляционных наборов данных эти команды представляли собой результат работы алгоритма автономного вождения, использующего внутреннее представление симулятора и специально настроенные для каждой карты параметры алгоритма. В реальном мире эти команды соответствовали человеческому эксперту.

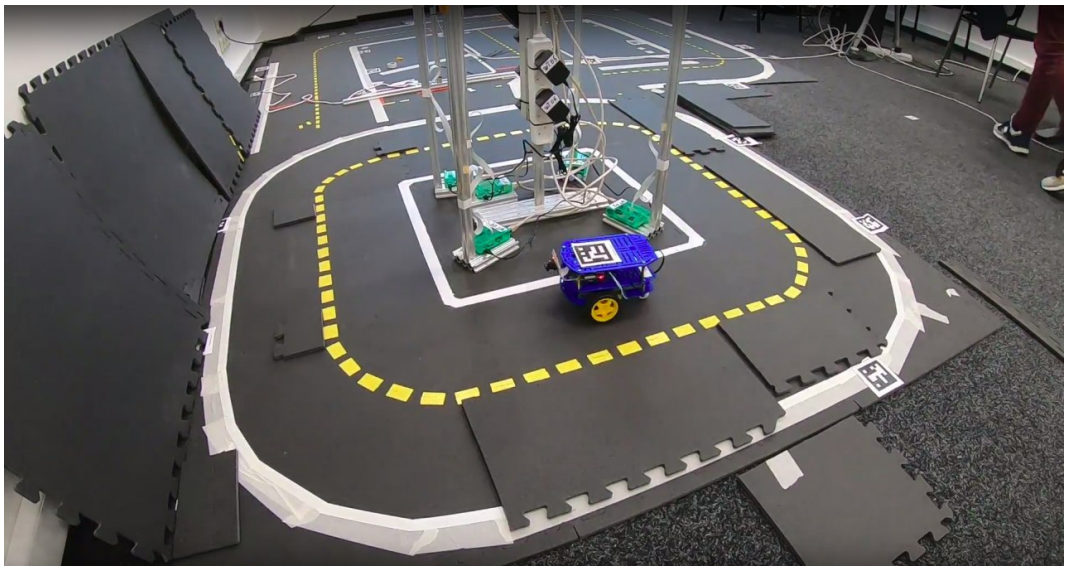

Рисунок 3 – Платформа Duckietown в реальном мире.

Для сбора данных в реальном мире используется робот на сконструированных дорогах. На рисунке 3 можно видеть пример такого робота и одно из возможных окружений реального мира Duckietown. Платформа предоставляет программное обеспечение для настройки и управления такого робота. В качестве дополнительной функциональности это программное обеспечение имеет возможность логировать информацию, которую получает и исполняет робот. В частности, таким образом можно устроить запись изображений с камеры в передней части робота и команд, которые исполняются на его колесах. Каждый такой запуск робота с логированием позволяет получить один эпизод данных из реального мира. Для сбора данных в большом объеме я использовал открытую базу данных таких логов управления роботов от участников сообщества Duckietown. Это база публично доступна по адресу: `http://logs.duckietown.org`. В ней содержатся сотни записей управления роботом. Чтобы сформировать набор данных из реального мира, я скачал и извлек данные из 61 эпизода, ко-

торые суммарно состоят из 66515 пар изображение-команда. Далее этот набор данных я буду обозначать как **REAL-DT**.

Для сбора симуляционных данных использовался открытый код симулятора описанной выше платформы Duckietown [12]. Целью этого симулятора было создание окружения, которое бы максимально близко описывало реальную среду Duckietown. Поэтому визуально симуляционный мир напоминает треки, которые используются в реальном мире. Пример изображения с камеры свободного обзора в симуляторе представлен на рисунке 4. Как можно видеть в сравнении с реальным миром на рисунке 3, пропорции дорог, ширина и цвет линий в симуляторе соответствуют реальным. Код симулятора позволяет его легко дополнять и изменять. В частности, код позволяет заменять карту на которой тестируется агент, и предоставляет интерфейс для выполнения в нем любого алгоритма агента.

Рисунок 4 – Симулятор платформы Duckietown.

Для сбора первого набора данных в симуляторе, была использована карта простого дорожного цикла с несколькими объектами на заднем плане. В качестве эксперта выступал пропорционально-дифференцирующий регулятор [9], оперирующий на внутреннем состоянии симулятора, которое включало позицию робота в полосе и координаты дороги и разметки. Внутренние коэффициенты этого регулятора были подобраны для плавного и непрерывного вождения в рамках заданной карты. Совместив симулятор и реализовав регулятор был получен код, который позволяет собирать заданное число кадров и соответствующих команд. Это было устроено следующим образом. Каждый эпизод сбора

данных начинается в случайной координате полосы дороги, и робот располагается с небольшим отклонением от прямого направления движения. Далее код регулятора взаимодействует со средой симулятора Duckietown на протяжении 512 шагов. В это же время происходит логирование изображений из симулятора и команд регулятора. Используя этот код, я провел 130 эпизодов сбора данных. В итоге было получено 66560 пар примеров в этом симуляционном окружении. Количество эпизодов было выбрано так, чтобы общий размер симуляционного датасета был приблизительно равен размеру реального. Этот набор данных далее я буду обозначать как **SIM-LP**.

Для сравнения особенностей переноса из разных симуляционных сред, я собрал в симуляторе второй набор данных. Как упоминалось ранее, его основное отличие состоит в карте, которая использовалась для генерации мира. Картой для этой среды был город с четырехсторонними перекрестками и множеством объектов, которые располагались вне дорожных полос. Примерами таких объектов являются светофоры, автобусы, здания и деревья. Другой отличительной особенностью являлось наличие дорожной разметки красного цвета. Эта разметка обозначала место, где автомобилю следует остановиться на перекрестке, если ему надо уступить дорогу другим участникам дорожного движения. Так как в первом симуляционном окружении картой был дорожный цикл, то в красной разметке там не было необходимости. Также стоит отметить, что красная разметка была представлена в эпизодах реального мира. Хотя в контексте этой работы я рассматриваю управление без посторонних участников, визуальные сходства этой среды могут способствовать переносу обученной политики в реальный мир. Структура сбора данных в этой среде следовала описанной процедуре для первого симуляционного набора. Пропорционально-дифференцирующий регулятор был заново настроен для плавного вождения в контексте этой карты. Используя код, связывающий симулятор и регулятор, сбор происходил в течение 512 шагов после случайного расположения робота в полосе. Как и в первом наборе, было собрано 130 эпизодов, что суммарно составило 66560 кадров. К данному набору я буду далее отсылаться как **SIM-IS**.

Платформа Duckietown стандартизирована таким образом, что реальные изображения и картинки из симулятора имеют размер 640x480, а управление, как уже упоминалось ранее, представлено вектором размерности 2, который обозначает напряжения приводов левого и правого колес. Таким образом, я обо-

значаю изображения как x_i, а команды – y_i. Тогда набор данных представляет собой множество пар таких примеров. Так как данные были собраны эпизодами, мне дополнительно известен их временной, или темпоральный, порядок. Поэтому в этой нотации я обозначаю эпизод данных как e_j, и он представляет собой множество пар $e_j = \bigsqcup_{i=1}^{n_{e_j}} \{x_i, y_i\}$. А набор данных D представляет собой множество таких эпизодов: $D = \bigsqcup_{j=1}^{n_D} e_j$. В моей работе я рассматриваю три таких набора данных: $D^{REAL-DT}$, D^{SIM-LP} и D^{SIM-IS}, в которых количество эпизодов есть $n_{D^{REAL-DT}} = 61$, $n_{D^{SIM-LP}} = 130$ и $n_{D^{SIM-IS}} = 130$ соответственно.

Для данных я использую следующую процедуру предобработки. Я обрезаю верхнюю треть изображения, так как она в основном содержит фон и не влияет на решения по управлению. Этот шаг изображение размера 640x480 приводит к размеру 640x320. Многие статьи по методам машинного обучения для автономного вождения [1, 2] показали, что нет необходимости в подаче на вход высокоразмерных изображений. В частности, глубокие модели машинного обучения способны обнаружить в изображениях небольшого разрешения закономерности достаточные для разумного автономного управления. Поэтому я сжимаю обрезанное изображение размера 640x320 в десять раз до размера 64x32. Примеры изображений после этих шагов обработки из всех рассматриваемых наборов данных представлены на рисунке 5. Как видно, по этим картинкам человеческий водитель может принимать правильные решения в управлении, что является практическим правилом при подготовке данных для моделей машинного обучения.

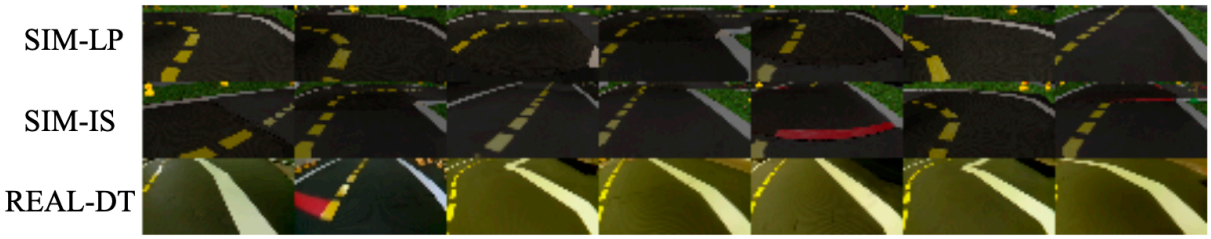

Рисунок 5 – Примеры предобработанных изображений каждого из трех наборов данных.

Дополнительным шагом предобработки изображений был перевод дискретных значений каналов цветов к вещественнозначному диапазону $[-1; 1]$. Так как значения цветов принимают значения от 0 до 255, функция перевода t значения цвета c выглядела как:

$$t(c) = (c/255) * 2 - 1 \quad (4)$$

Другим важным шагом в предобработке данных для дальнейшего обучения является разбиение набора данных на тренировочную и тестовую части. Хорошим правилом также является выделение валидационной части для получения несмещенной оценки тестовых метрик при настройке модели. Я следую общепринятой пропорции 80:10:10 для тренировочной, валидационной и тестовой частей. Так как мои данные сгруппированы в эпизоды, я проводил разбиение эпизодов, а не примеров по отдельности. Ведь смысл разбиения состоит в том, чтобы проверить обобщающую способность модели на новых данных. Проведя разбиение не учитывая границ эпизодов, могло оказаться, что кадры из одного эпизода будут как в тренировочной, так и в тестовой частях, что противоречит требованию новизны тестовой части.

Для поиска лучшего разбиения эпизодов наборов данных, я следовал следующей процедуре.

Сначала я группировал случайным образом эпизоды для достижения требуемой пропорции содержащихся в них примеров. Так как все симуляционные эпизоды состояли из 512 примеров, то разбив все эпизоды в пропорции 80:10:10, я разбивал в таком же соотношении и примеры. А именно, учитывая что каждый набор данных из симулятора состоял из 130 эпизодов, для этих датасетов это означало, что 104, или 80%, случайных эпизодов составляли тренировочную часть, 13, или 10%, других случайных эпизодов – валидационную, и оставшиеся 13, или 10%, эпизодов – тестовую часть. Для реального набора данных разбиение было устроено более сложным способом, так как эпизоды имели разное количество кадров. Я упорядочивал эпизоды случайным образом. Потом я расширял префикс эпизодов, которые составят тестовую часть, пока общее число примеров в них было меньше 10%. После определения границы тестовой части, я повторял эту процедуру далее для валидационной части. А именно, я набирал эпизоды в валидационную часть, пока общее число примеров было меньше 10%. Определив границу валидационной части, все оставшиеся эпизоды отправлялись в тренировочную часть. Таким образом, приблизительные части примеров, которые попадали в каждую выборку, соответствовали пропорции 80:10:10.

После случайной группировки, которая устроена описанным выше способом и отличается для симуляционного и реального набора, я оценивал, наскольк-

ко близкими получались распределения команд между частями набора данных. В идеальном случае, эти распределения должны оказаться одинаковыми, чтобы во время обучения команды управления были репрезентативными при валидации и тестировании. Для оценки разности распределений я использовал дискретные диапазоны всего пространства команд, что есть квадрат с координатами противоположных углов $(-1; -1)$ и $(1; 1)$, и вычислял полную вариацию дискретных вероятностей оказаться в каждом из диапазонов.

Я строил случайное разбиение и оценку средней попарной разности тренировочной, валидационной и тестовой частей 100 раз, и выбирал в качестве итогового разбиение то, которое имело минимальную среднюю разность. В итоге симуляционные наборы данных содержали 104, 13, 13 эпизодов и 53248, 6656, 6656 примеров, а реальный набор данных содержал 47, 7, 7 эпизодов и 51131, 8461, 6923 примеров в тренировочной, валидационной и тестовой частях соответственно.

Собрав и выполнив предобработку и разбиение наборов данных, я перешел к реализации моделей машинного обучения и процесса их тренировки.

4.2. Реализация базовой сверточной модели

В литературе наиболее простым и популярным методом обучения модели автономного вождения является создание сверточной архитектуры и обучение на целевом наборе данных. Во время обучения сверточная модель находит и выделяет закономерности в изображениях для предсказания команд. Такие модели показывают хорошую обобщаемость в домене, на котором они обучались. Например, обучив модель на наборе данных изображение-команда из симулятора, получается модель, способная предсказывать команды в похожих дорожных ситуациях симулятора. Аналогично, можно обучить модель для реального мира имея реальный набор изображений и команд. Важным требованием для таких моделей является наличие большого числа примеров, и, что более принципиально, команд для изображений. Так как в моей работе я собрал три обширных датасета, каждый из которых содержит изображения и соответствующие команды, я реализовал и обучил такой подход.

В качестве модели я выбрал архитектуру, которая подробно описана в моей статье [13]. Для полноты я приведу описание этого метода в контексте этой работы.

Архитектура сети в большой степени следует принципам, заложенным в статье [1]. В частности, я составляю ее из пяти сверточных слоев для выделения признаков из цветных изображений размера 64x32, и двух полносвязных слоев для преобразования этих признаков в команды управления, а именно векторов размера 2. Каждый сверточный слой использует небольшое количество фильтров для избежания переобучения. Пять сверточных слоев используют 2, 12, 24, 36 и 48 фильтров соответственно. Между каждой парой слоев расположена стандартная операция MaxPool с ядром 2, которая сжимает высоту и ширину изображения в два раза, а в качестве признаков выбирает максимальное значение в области размером равным ядру, то есть 2x2.

Важной отличительной чертой рассматриваемой архитектуры было использование независимых компонент из статьи [14]. Эти слои дополняют сверточные путем добавления перед ними общепринятых операций BatchNorm [15] и Dropout [16]. Цель такого расширения сверточного слоя в создании независимости последовательных слоев. Это критично для моделей машинного обучения, так как при получении входа, который отличается от тренировочного распределения, каждый из слоев будет принимать независимые решения, что приводит к лучшей обобщающей способности. В то же время, если слои сопряжены, то ошибка в первом слое будет усилена в несколько раз последующими слоями. Это особенно важно в контексте автономного вождения, так как предсказания в нестандартных ситуациях критичны для безопасного управления автомобилем. Архитектура использует Dropout с вероятностью 0.01 и с вероятностью 0.05 для дополнения первого и четырех последующих сверточных слоев соответственно. Стандартный Dropout был заменен на Spatial Dropout [17] из-за его эффективности для сверточных слоев. Два полносвязных слоя используют стандартный Dropout с вероятностью 0.05. Также, последний сверточный слой не использует функции активации, что важно для сохранения линейности предсказаний от предыдущего слоя.

После стадии подбора гиперпараметров, финальная версия такой архитектуры обучается с функцией потерь, представленной среднеквадратической ошибкой (СКО). Оптимизация проводится методом Adam [18] с коэффициентом обучения $1e{-}2$. Модель использует стадию предварительного разогрева коэффициента обучения в течение первых 10% шагов обучения, и после этого вы-

полняет его затухание обратно пропорциональное квадратному корню текущего шага.

После обучения эта модель может быть использована для предсказания команд по цветным изображениям с размером, согласованным с процедурой предобработки из секции 4.1. А именно, архитектура преобразует изображение размера 64x32 с тремя каналами цвета в вектор размера 2, представляющий команды управления. Более детальная постановка и результаты экспериментов для таких базовых сверточных моделей будут представлены в секции 5.1.

4.3. Реализация метода обучения с трансляцией изображений

Основной целью моей работы была разработка модели, которая способна обучаться при отсутствии команд в реальном мире. Такая модель должна обучаться на примерах изображение-команда из симулятора и использовать набор изображений из реального мира для установления соответствия между доменами.

В качестве базовой версии такой модели, которую я дальше буду называть Sim2Real, я реализовал следующий подход. Целью этого подхода является совмещение двух моделей: трансляторов из симулятора в реальный мир и наоборот, и модуля автономного вождения. Их объединение устроено путем использования общего скрытого пространства. Общее скрытое пространство критично для этой модели, так как представления изображений в нем универсальны и могут быть использованы как для генерации изображений в реальном мире и симуляторе, так и для предсказания команд, соответствующих этому представлению. Высокоуровневая схема архитектуры представлена на рисунке 6.

Всего в архитектуре 10 модулей, которые разделены на несколько компонент: генераторы (Enc^{sim}, Enc^{real}, Enc^{shared}, Dec^{sim}, Dec^{real}, Dec^{shared}), дискриминаторы (Dis^{sim}, Dis^{real}), сжимающий модуль ($Downstreamer$) и контроллер ($Controller$). Генераторы и дискриминаторы образуют описанные выше трансляторы, контроллер формирует модуль автономного вождения, а сжимающий модуль преобразует представления между высокоразмерным и низкоразмерным скрытыми пространствами. Далее мы рассмотрим подробнее каждую из этих компонент. В этой схеме можно явно выделить четыре пространства: пространство изображений симулятора Sim, в котором находятся изображения из симуляционного набора данных D^{sim}, аналогичное пространство для реальных изображений $Real$, которое содержит все изображения из D^{real}, высокоразмер-

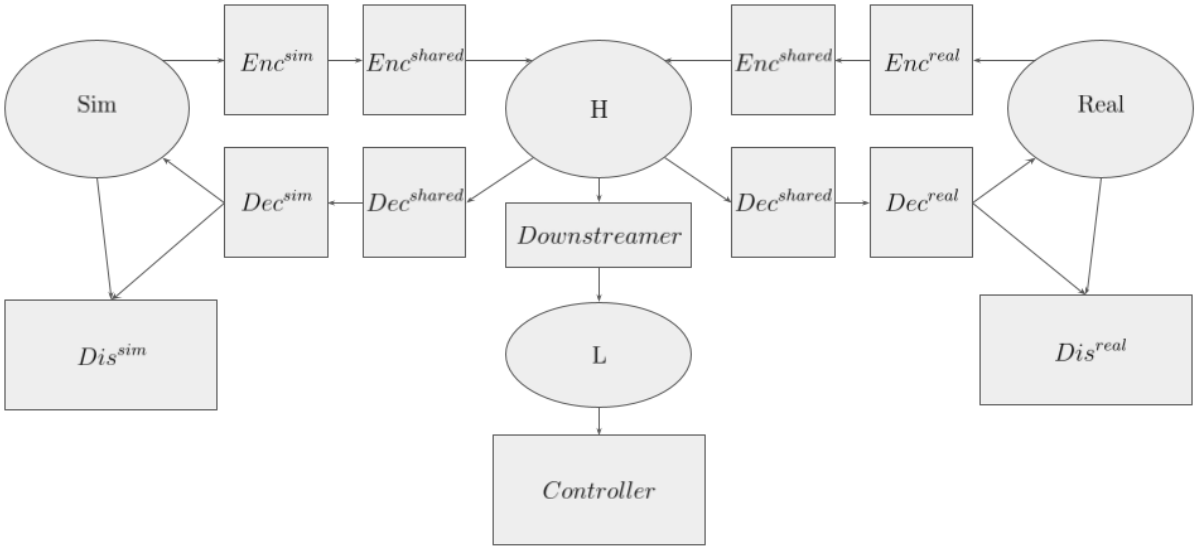

Рисунок 6 – Архитектура метода Sim2Real.

ное пространство скрытых представлений H и низкоразмерное пространство скрытых представлений L.

Компоненты генераторов и дискриминаторов вместе формируют трансляторы изображений из симулятора в реальный мир и обратно. В основе архитектуры трансляторов лежат генеративно-состязательные сети [8]. А именно, я следую варианту, предложенному в статье [19].

Генераторы устроены следующим образом. Эти модели могут использоваться для четырех задач: преобразование изображения из симулятора, или пространства Sim, в пространство высокоразмерных скрытых представлений H, обратное по направлению преобразование точек из H в Sim, а также преобразование изображения реального мира, или пространства $Real$, в пространство H и обратное ему преобразование из H в $Real$. Эти четыре направления можно использовать для достижения различных преобразований. Например, имея изображение в симуляторе, можно выполнить преобразования $Sim \rightarrow H$ и $H \rightarrow Real$, чтобы получить соответствующее ему относительно генераторов изображение в реальном мире. Возможно проверить насколько связны преобразования между скрытым пространством и одним из наблюдаемых пространств, выполнив $Sim \rightarrow H$ и $H \rightarrow Sim$. Наконец, можно проверить насколько информация сохраняется при смене доменов путем выполнения четырех преобразований: $Sim \rightarrow H$, $H \rightarrow Real$, $Real \rightarrow H$, $H \rightarrow Sim$. Эти преобразования

могут быть не настолько важны после обучения модели, но ожидаемые результаты для таких преобразований будут использованы для обучения модулей этой компоненты.

Эти четыре функции реализованы, используя 6 модулей: Enc^{sim}, Enc^{real}, Enc^{shared}, Dec^{sim}, Dec^{real}, Dec^{shared}. Преобразование $Sim \rightarrow H$ выполняется применением к изображению сперва операций, соответствующих модулю Enc^{sim}, и потом операций модуля Enc^{shared}. Преобразование $Real \rightarrow H$ выполнено модулями Enc^{real} и Enc^{shared}. Как видно общий модуль Enc^{shared}, используется как часть обоих кодировщиков. Такое объединение высокоуровневых преобразований для изображений из симулятора и реального мира следует статье [19]. Два обратных преобразования реализованы похожим способом, но используют другой набор модулей. Преобразование $H \rightarrow Sim$ выполнено модулями Dec^{shared} и Dec^{sim}, а преобразование $H \rightarrow Real$ – Dec^{shared} и Dec^{real}. Здесь тоже используется похожее сопряжение модулей для симулятора и реального мира, но оно случается в начальных слоях операций, так как оба этих преобразования исходят из общего пространства скрытых представлений H.

Теперь я опишу структуру каждого из 6 модулей генераторов. Так как Enc^{sim} и Enc^{real} выполняют одинаковую задачу на для разных доменов, то они имеют одинаковую архитектуру. На вход каждый из этих модулей принимает изображение 64x32x3 и преобразует его в промежуточное представление размера 16x8x64. Более детально, такое преобразование устроено применением трех сверточных слоев и трех остаточных слоев [20]. Первый сверточный слой преобразует изображение к 16 каналам, второй сжимает изображение в 2 раза и преобразует к 32 каналам, и последний опять сжимает и выдает 64 канала. В результате из вектора 64x32x3 получается вектор 16x8x64. После применяются три остаточных слоя, которые не меняют размерность. После каждого сверточного слоя применяется LeakyReLU с коэффициентом 0.01. Остаточный слой использует свертки 3x3, InstanceNormalization [21] и ReLU в качестве функции активации. Модуль Enc^{shared} является общим для двух преобразований и представлен одним остаточным слоем с аналогичными параметрами. Для улучшения процесса тренировки к этому модулю был также добавлен слой гауссовского шума, который искажает представление прибавлением сэмпла из стандартного нормального распределения $\mathcal{N}_{16\text{x}8\text{x}64}(0; 1)$ во время тренировки, и оставляет представления неизменными во время эксплуатации. В результате модули,

представляющие преобразования $Sim \to H$ и $Real \to H$, переводят изображения 64x32x3 в вектор размера 16x8x64.

Обратные по направлению преобразования устроены похожим образом, за исключением обратного порядка слоев и использования транспонированных сверток. Оба преобразования сначала применяют общий модуль Dec^{shared}. Этот модуль оперирует на представлениях размер 16x8x64, и аналогично Enc^{shared} состоит из одного остаточного слоя, который не меняет размерность. Dec^{sim} и Dec^{real} имеют одинаковую архитектуру, которая состоит из трех остаточных слоев и трех транспонированных сверток. Транспонированные свертки представляют собой обратные к стандартным сверткам операции, и результат их применения разжимает изображение, как правило, за счет уменьшения числа каналов. Более детально, первая транспонированная свертка преобразует представление 16x8x64 к размеру 32x16x32, вторая – к 64x32x16, и последняя – к 64x32x3. Между этими слоями применяется та же функция активации LeakyReLU. После финального слоя, применяется гиперболический тангенс, который приводит каждое значение к диапазону $[-1; 1]$. Таким образом преобразования $H \to Sim$ и $H \to Real$ переводят точку из высокоразмерного скрытого пространства H к точке в исходном пространстве изображений.

Ввиду отсутствия парных изображений типа симулятор-реальный мир и следуя рамкам подхода генеративно-состязательных сетей, модули Dis^{sim} и Dis^{real} представляют дискриминаторы. Целью обучения дискриминаторов является их ответ на вопрос, было ли входное изображение оригинальным, или оно было получено в ходе преобразований модулей генераторов. Мотивация такого подхода в том, что при невозможности отличить изначальные и сгенерированные изображения, мы можем утверждать, что наша процедура генерации выдает реалистичные изображения. Так как Dis^{sim} и Dis^{real} выполняют одинаковую задачу, но для разных доменов (Dis^{sim} для симулятора, а Dis^{real} для реального мира), то они имеют одинаковую архитектуру. Каждый дискриминатор был представлен четырьмя сверточными слоями, сжимающими изображение в 2 раза и увеличивающими число каналов, и финальным сверточным слоем с сигмоидой в качестве функции активации. Более детально, дискриминатор получает изображение, оригинальное или преобразованное, размера 64x32x3, и первая свертка преобразует его к размеру 32x16x16, вторая – к 16x8x32, третья – к 8x4x64, а четвертая – к 4x2x128. После каждого сверточного слоя исполь-

зуется LeakyReLU с коэффициентом 0.01. Последний сверточный слой имеет ядро размером 1x1 и поэтому преобразует лишь число каналов к размеру 4x2x1. Применив сигмоиду, получается вектор вероятностей того, оригинальное ли это изображение. Замечу, что число не одно, а их целый вектор, в котором каждый элемент обозначает вероятность, посчитанную по разному участку изображения.

Сжимающий модуль $Downstreamer$ переводит точки в высокоразмерном скрытом пространстве H к точкам в низкоразмерном пространстве L. А именно, этот модуль из вектора размера 16x8x64 получает вектора размера 128. Это преобразование выполнено с целью более устойчивого обучения контроллера и применения дополнительных ограничений, которые рассчитаны на оперирование в небольших пространствах. Такое преобразование выполнено сперва распрямлением вектора 16x8x64 в вектор размера 8192, применением BatchNorm слоя [22], LeakyReLU и полносвязным слоем, который приводит вектор к размеру 128. Эти низкоразмерные вектора будут использованы для дополнительных ограничений в секции 4.4 и для модуля контроллера.

Наконец, $Controller$ представляет модуль, который возвращает вектор размера 2, обозначающий команды управления, что и является исходной задачей алгоритма автономного вождения. Этот модуль представлен двумя последовательными блоками, которые получают вектора размера 32 и 2 соответственно. Каждый блок устроен схожим с блоком сжимающего модуля образом. А именно, применяется BatchNorm слой, LeakyReLU, Dropout и полносвязный слой. Dropout используется с вероятностью обнуления значения равной 0.1. После этих двух блоков применяется гиперболический тангенс, который приводит значения вектора размера 2 к диапазону $[-1; 1]$. Это в точности совпадает с определением команд управления в данной задаче.

Далее я опишу процесс обучения и устройство функции потерь. Для определения функции потерь я буду обозначать изображения симуляционного и реального набора данных как x_i^s и x_i^r соответственно. Аналогичным добавлением индекса я буду различить команды из симулятора и реального мира: y_i^s и y_i^r. В более общем виде, для изображений индексы будут обозначать пройденные в процессе преобразований пространства. К примеру, взяв изображение x_i^r из реального мира и применив $Real \to H$ и $H \to Real$, соответствующее изображение будет обозначено как x_i^{rhr}.

Полная функция потерь генераторов имеет следующий вид:

$$L_{generator} = \alpha_1 L_{ll\_dir} + \alpha_2 L_{ll\_cyc} + \alpha_3 L_{kl\_dir} + \alpha_4 L_{kl\_cyc} + \alpha_5 L_{recon} + \\ + \alpha_6 L_{gen\_gan} + \alpha_7 L_{control} \tag{5}$$

Функция потерь дискриминаторов:

$$L_{discriminator} = \alpha_6 L_{dis\_gan} \tag{6}$$

Функция потерь контроллера:

$$L_{controller} = \alpha_7 L_{control} \tag{7}$$

Рассмотрим отдельные составляющие функций потерь для соответствующих компонент. L_{ll_dir} представляет собой компоненту функции потерь, которая отвечает за качество восстановления изображения из скрытого пространства в прямом цикле. А именно, рассмотрим подробнее пример для изображения из симулятора x_i^s. Применив преобразование $Sim \rightarrow H$ и $H \rightarrow Sim$, получаем изображение x_i^{shs}, и в идеальном случае эти преобразования должны происходить без потерь. Это можно оценить с помощью $L_{ll\_dir\_s}$, что равно среднему L1 расстоянию между значениями изображений:

$$L_{ll\_dir\_s} = \frac{1}{|Sim|} ||x_i^s - x_i^{shs}||_1 \tag{8}$$

В вероятностной интерпретации это выражение представляет функцию правдоподобия преобразованного изображения относительно оригинального для распределения Лапласа. Аналогично определена часть $L_{ll\_dir\_r}$, отвечающая за качество восстановления для реального мира. Тогда L_{ll_dir} равна сумме $L_{ll\_dir\_s}$ и $L_{ll\_dir\_r}$. L_{ll_cyc} определена аналогичным образом только для полных циклов преобразований. А именно, полным циклом преобразований для симулятора определим последовательное применение четырех преобразований: $Sim \rightarrow H$, $H \rightarrow Real$, $Real \rightarrow H$, $H \rightarrow Sim$. Тогда вычислив среднее L1 расстояние между x_i^s и x_i^{shrhs} мы оценим качество полного цикла преобразований для симулятора. Значение L_{ll_cyc} равно сумме соответствующих средних L1 расстояний для симулятора и реального мира.

Компоненты L_{kl_dir}, L_{kl_cyc} и L_{recon} оперируют над представлениями в скрытом пространстве. Первые две компоненты регуляризуют представления, а последняя оценивает качество восстановления скрытых представлений во время полного цикла преобразований. Более подробно, я предполагаю, что скрытые представления должны иметь стандартное нормальное распределение. Чтобы заставить модель это учитывать при оптимизации, L_{kl_dir} и L_{kl_cyc} оценивают расстояние Кульбака-Лейблера (KL) между нормальным распределением, которое задается покомпонентно скрытым представлением изображения, и соответствующим стандартным нормальным распределением. L_{kl_dir} равно сумме расстояний KL для скрытых представлений прямого цикла, а именно x_i^{sh} и x_i^{rh}. L_{kl_cyc} задается суммой расстояний KL для скрытых представлений полного цикла, то есть x_i^{shrh} и x_i^{rhsh}. L_{recon} оценивает насколько согласованы скрытые представления в разных стадиях преобразований полного цикла. А именно, эта компонента функции потерь есть сумма средних L1 расстояний между x_i^{sh} и x_i^{shrh} для симулятора и x_i^{rh} и x_i^{rhsh} для реального мира.

Компоненты L_{gen_gan} и L_{dis_gan} представляют состязательные функции потерь для генераторов и дискриминаторов соответственно. В такой постановке генераторы пытаются максимизировать вероятность того, что сгенерированные ими изображения классифицируются дискриминаторами как оригинальные, а дискриминаторы пытаются точно провести классификацию: классифицировать оригинальные изображения как оригинальные, и сгенерированные как сгенерированные. Функцией потерь здесь выступает бинарная кросс-энтропия. Более формально, рассмотрим как эта функция выглядит для генератора в домене симулятора. Задача генератора здесь состоит в том, чтобы реальные изображения x_i^r после трансляции дискриминатор принимал за симуляционные. А именно:

$$L_{gen\_gan\_s} = -\sum_{k=1}^{|Dis|} log(Dis_k^{sim}(x_i^{rhs})) \tag{9}$$

где $|Dis|$ – это размерность выходного вектора дискриминатора, а Dis_k^{sim} – это k-ый выход в распрямленном векторе вероятностей дискриминатора для симулятора. Задачей дискриминатора здесь является правильная классификация изображений из симулятора x_i^s и, в отличие от генератора, в определении фальшивости транслированных изображений x_i^{rhs}. Более формально, в домене симулятора эта функция выглядит как:

$$L_{dis\_gan\_s} = -\sum_{k=1}^{|Dis|}(log(Dis_k^{sim}(x_i^s)) + log(1 - Dis_k^{sim}(x_i^{rhs}))) \quad (10)$$

Аналогично эти функции определены для реального мира. Тогда сумма соответствующих частей для симулятора и реального мира и задают L_{gen_gan} и L_{dis_gan}.

Последняя составляющая компонента $L_{control}$ напрямую оптимизирует точность предсказаний команд на наборе данных. Точность предсказаний оценивается как для оригинальных изображений из симулятора x_i^s, так и для преобразованных в домен реального мира изображений x_i^{shr}. Во время поиска гиперпараметров я оптимизировал функцию потерь контроллера для команд управления. В финальной версии функция потерь представлена среднеквадратическим отклонением (СКО), и выглядит как:

$$L_{control} = \frac{1}{|Y|}(||Controller(x_i^s) - y_i^s||_2^2 + ||Controller(x_i^{shr}) - y_i^s||_2^2) \quad (11)$$

Мотивация такой постановки в том, что оптимизация на транслированных из симулятора в реальный мир изображениях даже для симуляционных команд способствует модели во время эксплуатации в реальном мире.

Для полноты описания я приведу коэффициенты компонент функций потерь, использованные в финальной версии модели: $\alpha_1 = 100, \alpha_2 = 100, \alpha_3 = 0.1, \alpha_4 = 0.1, \alpha_5 = 0.1, \alpha_6 = 10, \alpha_7 = 1$.

Тренировка компонент модели осуществлялась совместно. А именно, на каждом оптимизационном шаге сэмплируются $n_{episodes}$ эпизодов из симуляционного и столько же эпизодов из реального набора данных. Из каждого эпизода выбирается подпоследовательность кадров длины $batch\_size$. Все функции потерь вычисляются используя выбранные симуляционные изображения и команды и изображения из реального мира. Далее, используя уникальные оптимизаторы для генераторов, дискриминаторов и контроллера, проводился один шаг оптимизации. В финальной версии модели, я использовал оптимизатор Adam с коэффициентом обучения $1e{-}4$ для генераторов и дискриминаторов, стохасти-

ческий градиентный спуск с коэффициентом обучения $1e{-}2$ для контроллера, и гиперпараметры $n_{episodes} = 2$ и $batch\_size = 32$.

В следующей секции я опишу какие дополнения функции потерь, основанные на темпоральном порядке, были исследованы мной.

4.4. Использование темпорального порядка

Для потенциального улучшения Sim2Real трансфера, я рассмотрел дополнительные ограничения использующие темпоральный порядок. Это область, которая не была исследована ранее в литературе методов Sim2Real для алгоритмов автономного вождения. Но информация, которая содержится в порядке кадров и изменении соседних изображений, может быть полезна для установления соответствия между симулятором и реальным миром и, следовательно, лучшего переноса.

Как ранее было определено, в задаче имеются два набора данных: в симуляторе D^s и реальном мире D^r. Эти два набора данных состоят из эпизодов: $D^s = \bigsqcup_{j=1}^{n_{D^s}} e_j^s$ и $D^r = \bigsqcup_{j=1}^{n_{D^r}} e_j^r$. Каждый эпизод содержит изображения и команды в темпоральном порядке: $e = \bigsqcup_{i=1}^{n_e} \{x_i, y_i\}$. Основываясь на этом, я ввожу ограничения на скрытые представления соответствующих кадров в эпизодах. Для вычисления этих ограничений я использую представления в низкоразмерном скрытом пространстве L, то есть после применения сжимающего модуля $Downstreamer$.

Я рассмотрел два ограничения, основанные на обучении без учителя для представлений:

— ограничение Triplet Loss на то, чтобы представления любого кадра эпизода были ближе в скрытом пространстве к своим соседям в темпоральном порядке, чем к другим кадрам внутри того же или другого эпизода;

— ограничение Temporal Cycle-Consistency на то, чтобы отношение близости между представлениями кадров двух эпизодов было согласованным при поиске ближайшего соседа в скрытом пространстве.

Далее я опишу каждое из ограничений формально.

Ограничение Triplet Loss (TL) основано на идее, изначально предложенной для методов классификации лиц в компьютерном зрении [23]. В статье авторы предлагали оптимизировать одноименную функция потерь, где лица, принадлежащие одному человеку должны были быть ближе друг к другу, чем к лицам других людей. Я ввожу его с похожей мотивацией для задачи алгоритма

автономного вождения. Я оптимизирую скрытые представления таким образом, чтобы точки соответствующие соседним кадрам в эпизоде находилось в пространстве ближе друг к другу, чем к другим кадрам внутри того же или другого эпизода. Более формально, эта функция высчитывается для каждого кадра в батче. Процесс вычисления аналогичен для симулятора и реального мира, поэтому рассмотрим случай для симулятора. Имея кадр x_i^s, его скрытые представления в пространстве L аналогично нотации в секции 4.3 будут равны x_i^{shl}. Скрытые представления следующего за ним кадра аналогично равны x_{i+1}^{shl}. Я задаю отступ расстояния M_{TL} в скрытом пространстве, который обозначает минимум того, насколько соседние кадры должны быть ближе друг к другу относительно других кадров. Обозначив случайный кадр из случайного эпизода, симуляционного или реального, как x_j^d (где индекс d заменяет индекс s симуляционного или r реального домена) и его скрытые представления этого как x_j^{dhl}, TL ограничение вычисляется как:

$$TL = max(d(x_i^{shl}, x_{i+1}^{shl}) - d(x_i^{shl}, x_j^{dhl}) + M_{TL}, 0) \tag{12}$$

где d – это функция расстояния в пространстве скрытых представлений. В финальной версии алгоритма я использую L2 норму в качестве расстояния. Схематично TL ограничение изображено на рисунке 7.

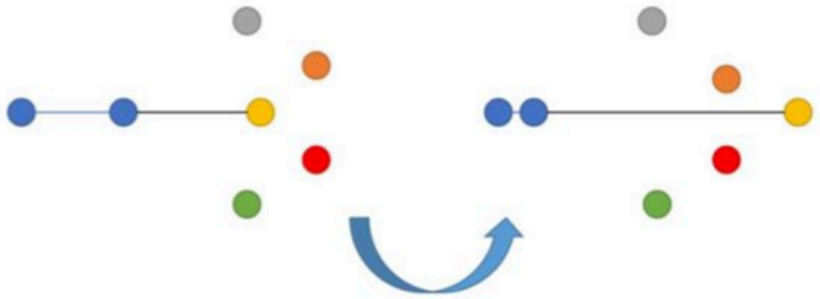

Рисунок 7 – Ограничение Triplet Loss. Слева изображено то, как могли выглядеть скрытые представления соседних кадров до, а справа после использования TL ограничения.

Ограничение Temporal Cycle-Consistency (TCC) в изначальной постановке решает задачу темпорального выравнивания видео [24]. Я предлагаю ввести

ограничение TCC для задачи автономного вождения следующим образом. Для его вычисления необходима пара эпизодов e_a и e_b (из какого домена каждый из эпизодов неважно). При оптимизации выбирается подпоследовательность размера n_{cycles} номеров кадров из каждого эпизода: $\{a_i\}_{i=1}^{n_{cycles}}$ и $\{b_i\}_{i=1}^{n_{cycles}}$. Переобозначим скрытые представления для соответствующих кадров каждого эпизода как $u_i = x_{a_i}^{dhl}$ и $v_j = x_{b_j}^{dhl}$. Найдем непрерывного ближайшего соседа для каждого u_i как:

$$\begin{aligned} \widetilde{v_i} &= \sum_{j}^{n_{cycles}} \alpha_j v_j \\ \alpha_j &= \frac{e^{-||u_i - v_j||_2^2}}{\sum_{k=1}^{n_{cycles}} e^{-||u_i - v_k||_2^2}} \end{aligned} \tag{13}$$

Тогда можно обозначить оценки того, насколько близко непрерывный сосед $\widetilde{v_i}$ к изначальным кадрам эпизода v, как:

$$\widetilde{z}_{ik} = softmax(-||\widetilde{v_i} - u_k||_2^2) \tag{14}$$

Наконец ограничение TCC есть:

$$TCC = -\sum_{k=1}^{n_{cycles}} z_{ik} log(\widetilde{z}_{ik}) \tag{15}$$

где $z_{ik} = \mathbb{1}[k = i]$. То есть, вектор z имеет единицу в индексе, соответствующему номеру изначального кадра u_i, а TCC оптимизирует бинарную кросс-энтропию между точными метками изначального кадра и оценок относительно непрерывной версии ближайшего соседа в другом эпизоде. Схематично TCC ограничение представлено на рисунке 8.

Оба эти ограничения могут быть включены в модель следующим образом. Как было описано в секции 4.3, на каждом оптимизационном шаге сэмплируются несколько реальных и симуляционных эпизодов. Используя подпоследовательности выбранных кадров, могут быть вычислены TCC и TL ограничения. Я добавляю их с коэффициентами α_{TL} и α_{TCC} к функции потерь генераторов, приведенной формулой 5. Эти ограничения могут быть использованы как отдельно, так и вместе. В финальной версии алгоритма, я использую для ограничений

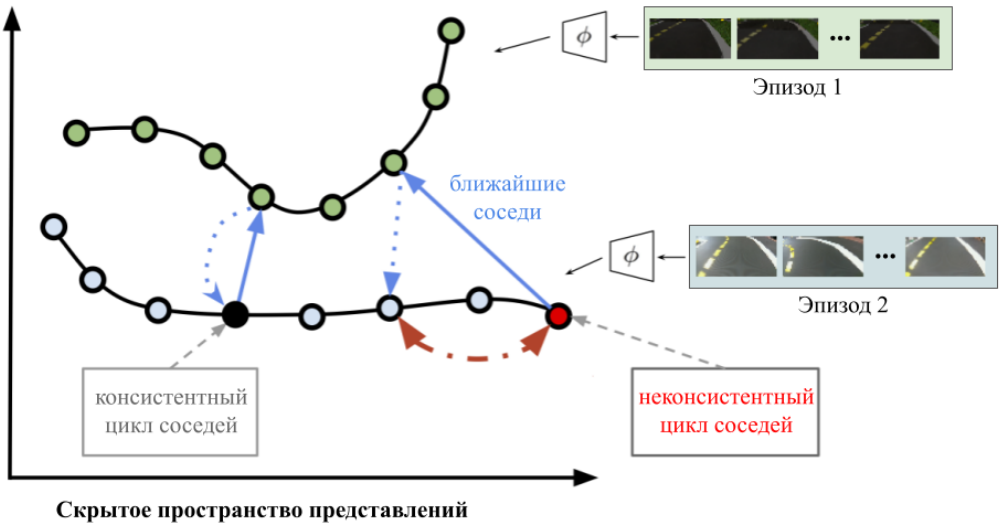

Рисунок 8 – Ограничение Temporal Cycle-Consistency.

следующие значения гиперпараметров: $M_{TL} = 1.0$, $\alpha_{TL} = 0.1$, $n_{cycles} = 20$ и $\alpha_{TCC} = 0.1$.

4.5. Выводы и результаты по главе

В начале этой главы я описал используемую в моей работе среду Duckietown, и процедуру сбора двух симуляционных датасетов и одного набора данных из реального мира. Я описал шаги предобработки собранных изображений. Далее я представил сверточную модель, которая может использоваться для получения качественных предсказаний внутри тренировочного домена. Я обосновал архитектуру и процедуру обучения этой модели. После я определил и описал Sim2Real модель, которую я разработал для осуществления переноса модели из симулятора в реальный мир. Эта модель для обучения использует симуляционные изображения и команды и реальные изображения. Я представил архитектуру, функции и составляющие модули этой модели. Я определил и привел мотивацию для использованных функций потерь. Наконец, я описал дополнительные ограничения, которые я разработал для лучшего переноса модели. Эти ограничения исходят из темпорального порядка эпизодов данных.

ГЛАВА 5. ЭКСПЕРИМЕНТЫ

5.1. Постановка

В своих экспериментах я сравниваю 6 методов обучения.

SIM. Сверточная модель, описанная в секции 4.2, которая обучается только на изображениях и командах одного симуляционного датасета. Эта модель служит примером того, как метод обучения, не учитывающий изображения из реального мира, переносится в реальный мир.

REAL. Сверточная модель с той же архитектурой, описанной в секции 4.2, но которая обучается на изображениях и командах из реального мира. Эта модель служит верхней границей эффективности в реальном мире, так как она использует команды из реального мира. В то время как в постановке задачи Sim2Real, команды из реального мира недоступны.

SIM2REAL. Модель, использующая реальные изображения для лучшего переноса предсказаний команд управления из симулятора в реальный мир. Эта модель была описана в секции 4.3. Данная модель использует изображения и команды из одного симуляционного датасета и только изображения из набора данных реального мира. Наличие этой модели в постановке экспериментов покажет, насколько эффективны Sim2Real методы по сравнению с тривиальным переносом модели SIM и как дополнительные ограничения влияют на финальный трансфер.

SIM2REAL+TL. SIM2REAL модель с дополнительным Triplet Loss ограничением, описанным в секции 4.4. Это ограничение оперирует над понятием соседства в эпизоде.

SIM2REAL+TCC. SIM2REAL модель с дополнительным Temporal Cycle-Consistency ограничением из секции 4.4. Это ограничение основано на согласованности отношения близости представлений между эпизодами.

SIM2REAL+TL+TCC. SIM2REAL модель, которая включает Triplet Loss и Temporal Cycle-Consistency ограничения. Эта модель служит примером совместимости разных ограничений, основанных на темпоральном порядке.

Обучение этих моделей происходило на трех наборах данных: SIM-LP, SIM-IS и REAL-DT. SIM модель была обучена на обоих симуляционных наборах данных, REAL модель – на единственном наборе данных из реального мира, а все варианты SIM2REAL были обучены для трансфера SIM-LP $\rightarrow$ REAL-DT и SIM-IS $\rightarrow$ REAL-DT. Как было описано ранее в секции 4.1, все датасеты были разбиты на тренировочную, валидационную и тестовую части. Обучение моделей всегда происходит на тренировочной части соответствующего набора

данных, а все метрики в этой секции приведены для тестовых частей. Гиперпараметры были подобраны на валидационной части наборов данных.

Все модели использовали особенную структуру оптимизационного шага, описанную в соответствующих секциях. В то же время все модели обучались в течение 100000 шагов.

Описав постановку экспериментов, в следующей секции я приведу результаты и сравнение работы моделей для двух обозначенных трансферов из симулятора в реальный мир.

5.2. Сравнение и анализ результатов

Сравнение моделей я проведу для двух постановок переносов: SIM-LP $\rightarrow$ REAL-DT и SIM-IS $\rightarrow$ REAL-DT. Для получения более достоверных результатов я выполнил обучение каждой модели для 5 случайных инициализаций процесса обучения.

Сначала я представлю результаты переноса моделей из более простого окружения SIM-LP в реальный мир. Окружение SIM-LP соответствует простому дорожному циклу с несколькими посторонними объектами. Результаты средних значений и стандартных отклонений обоих тестовых метрик из секции 2.1 приведены в таблице 1. Результаты абсолютного и относительного доменного несоответствия для средних значений тестовых метрик представлены в таблице 2.

Как и ожидалось, REAL модель имеет лучшие значения метрик CAO и СКО в реальном мире, а SIM модель показывает худшие значения. Все варианты модели SIM2REAL лежат внутри этого диапазона. Лучшим относительно CAO и СКО вариантом модели оказалась SIM2REAL+TCC, которая дополнительно поддерживает отношение непрерывного ближайшего соседа между эпизодами. Отмечу, что все четыре варианта имеют схожие результаты в симуляторе, но благодаря дополнительным ограничениями позволяют точнее перенести модель из симулятора в реальный мир. Доменное несоответствие базового метода SIM2REAL равно 28.67% относительно CAO и 20.00% относительно СКО. Это показывает, что перенос SIM2REAL модели в реальный мир в 4-5 раз эффективнее, чем перенос SIM модели, обученной только в симуляторе. Использование темпоральных ограничений позволяет дальше сократить доменное несоответствие относительно CAO до 24.86% и относительно СКО до 15.61%.

Таблица 1 – Результаты методов для переноса из SIM-LP в REAL-DT. Оценки метрик построены на 5 запусках.

Метод	SIM-LP				REAL-DT			
	САО		СКО		САО		СКО	
	сред.	откл.	сред.	откл.	сред.	откл.	сред.	откл.
SIM	0.0036	0.0001	49e-6	5e-6	0.1182	0.0190	0.0248	0.0062
SIM2REAL	0.0364	0.0021	0.0027	0.0006	0.0657	0.0020	0.0084	0.0008
SIM2REAL+TL	0.0366	0.0044	0.0029	0.0008	0.0643	0.0016	0.0082	0.0003
SIM2REAL+TCC	0.0359	0.0036	0.0024	0.0004	**0.0629**	**0.0018**	**0.0075**	**0.0010**
SIM2REAL+TL+TCC	0.0357	0.0034	0.0025	0.0004	0.0631	0.0059	0.0078	0.0016
REAL	0.1586	0.0348	0.0354	0.0148	0.0446	0.0025	0.0043	0.0004

Таблица 2 – Доменное несоответствие методов в реальном мире для переноса из SIM-LP в REAL-DT.

Метод	САО		СКО	
	Абсолютное	Относительное	Абсолютное	Относительное
SIM	0.0736	100.00%	0.0205	100.00%
SIM2REAL	0.0211	28.67%	0.0041	20.00%
SIM2REAL+TL	0.0197	26.77%	0.0039	19.02%
SIM2REAL+TCC	**0.0183**	**24.86%**	**0.0032**	**15.61%**
SIM2REAL+TL+TCC	0.0185	25.14%	0.0035	17.07%
REAL	0.0	0.00%	0.0	0.00%

Второй перенос моделей был проведен из более сложного окружения симулятора SIM-IS в реальный мир. Эта среда симулировала город с четырехсторонними перекрестками и множеством объектов на заднем плане, таких как деревья, здания, светофоры и автобусы. Как было отмечено ранее, эта среда по содержанию больше похожа на окружение реального мира. Результаты средних значений и стандартных отклонений обоих тестовых метрик для такого переноса приведены в таблице 3. Результаты абсолютного и относительного доменного несоответствия для средних значений тестовых метрик представлены в таблице 4.

Аналогично первому трансферу, методы SIM и REAL задают соответственно нижнюю и верхнюю границу эффективности в реальном мире. Лучшим для этого переноса среди SIM2REAL вариантов относительно САО и СКО явля-

Таблица 3 – Результаты методов для переноса из SIM-IS в REAL-DT. Оценки метрик построены на 5 запусках.

Метод	SIM-IS				REAL-DT			
	САО		СКО		САО		СКО	
	сред.	откл.	сред.	откл.	сред.	откл.	сред.	откл.
SIM	0.0147	0.0014	0.0012	0.0002	0.0935	0.0083	0.0162	0.0021
SIM2REAL	0.0552	0.0059	0.0080	0.0007	0.0624	0.0019	0.0078	0.0004
SIM2REAL+TL	0.0567	0.0043	0.0089	0.0014	0.0590	0.0015	0.0070	0.0005
SIM2REAL+TCC	0.0560	0.0023	0.0080	0.0002	0.0600	0.0023	0.0069	0.0002
SIM2REAL+TL+TCC	0.0568	0.0038	0.0084	0.0009	**0.0587**	**0.0009**	**0.0066**	**0.0001**
REAL	0.1377	0.0432	0.0317	0.0161	0.0446	0.0025	0.0043	0.0004

Таблица 4 – Доменное несоответствие методов в реальном мире для переноса из SIM-IS в REAL-DT.

Метод	САО		СКО	
	Абсолютное	Относительное	Абсолютное	Относительное
SIM	0.0489	100.00%	0.0119	100.00%
SIM2REAL	0.0178	36.40%	0.0035	29.41%
SIM2REAL+TL	0.0144	29.45%	0.0027	22.69%
SIM2REAL+TCC	0.0154	31.49%	0.0026	21.85%
SIM2REAL+TL+TCC	**0.0141**	**28.83%**	**0.0023**	**19.33%**
REAL	0.0	0.00%	0.0	0.00%

ется SIM2REAL+TL+TCC. Этот метод применяет оба темпоральных ограничения. Опять же, результаты метрик в симуляторе находятся в одном диапазоне, в то время как использующие дополнительные ограничения методы показывают лучшую переносимость в реальный мир. Доменное несоответствие метода SIM2REAL относительно САО есть 36.40% и относительно СКО есть 29.41%. При использовании темпоральных ограничений доменное несоответствие сокращается до 28.83% относительно САО и до 19.33% относительно СКО.

Интересно также сравнить перенос моделей в один и тот же реальный набор данных из двух разных симуляционных датасетов. Лучшие средние результаты для переноса из SIM-LP в REAL-DT были САО равное 0.0629 и СКО равное 0.0075. В то же время при трансфере моделей из SIM-IS в REAL-DT лучшими результатами были САО равное 0.0587 и СКО равное 0.0066. Таким образом

трансфер из симуляционной среды SIM-IS по сравнению с SIM-LP позволяет улучшить в абсолютном смысле метрики САО на 6.68% и СКО на 12%. Как было замечено в секции 4.1, окружение SIM-LP более реалистично, и результаты экспериментов подтверждают то, что трансфер с такой среды является более эффективным.

5.3. Исследование полученных моделей

Для дальнейшего понимания того, как темпоральные ограничения влияют на характеристики моделей, я провожу следующий анализ базовой модели SIM2REAL и вариантов модели с дополнительными ограничениями. Для сравнения я рассматриваю визуальное качество трансляции моделей и исследую полученные скрытые представления.

Я сравниваю результаты трансляции четырех SIM2REAL моделей, описанных в секции 5.1: SIM2REAL, SIM2REAL+TL, SIM2REAL+TCC, SIM2REAL+TL+TCC. Результаты трансляции показаны на рисунках 9 и 10. На рисунках каждая строка соответствует отдельному примеру. Первая колонка показывает исходное изображение в симуляторе или реальности, а четыре следующих представляют результат трансляции, используя соответствующие кодировщики и декодировщики моделей.

По итогу сравнения, я выявил, что хоть базовая SIM2REAL модель и генерирует разумные трансляции, они могут быть чаще не согласованы в рамках всего эпизода. К примеру, на рисунке 9 можно видеть как меняется освещение на двух последовательных кадрах из симуляционного окружения SIM-LP и двух других последовательных кадрах из SIM-IS при трансляции в базовой модели, но при добавлении любой темпоральной функции потерь, оно остается более консистентным.

Для дальнейшего исследования полученных моделей я разработал процедуру визуализации представлений модели. После преобразования входных изображений кодировщиками я получаю высокоразмерные вектора в скрытом пространстве. После этого я использую сжимающий модуль, который преобразует вектора к размерности 128. Далее я провожу популярный метод визуализации путем стохастического вложения соседей с t-распределением [25] для получения двумерных векторов. Такая процедура позволяет преобразовать любой набор входных изображений к двумерным векторам, которые можно изобразить на плоскости. Отмечу, что в такой постановке, близкие точки на плос-

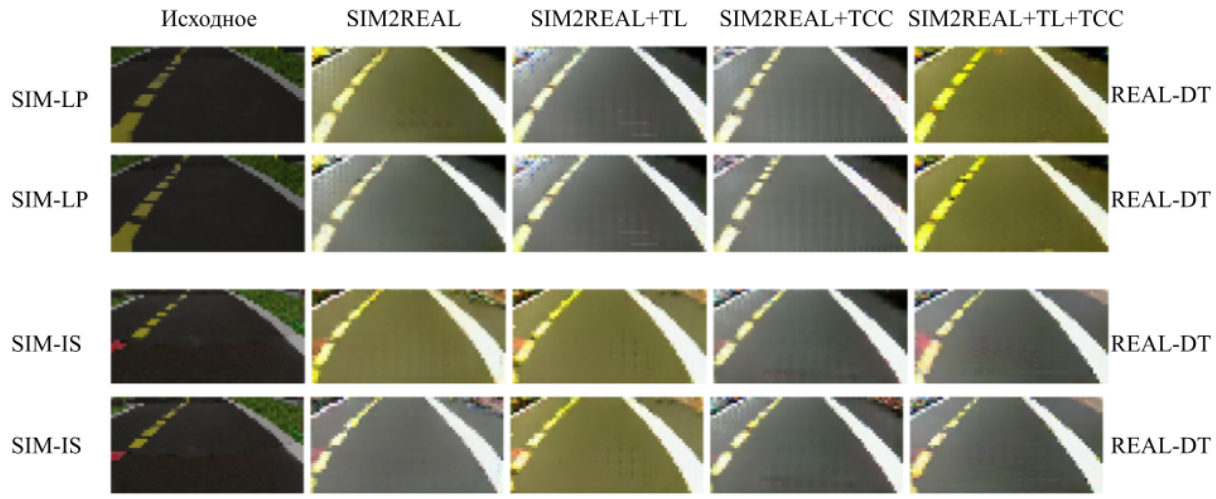

Рисунок 9 – Примеры трансляции изображений из симуляционных окружений в реальность.

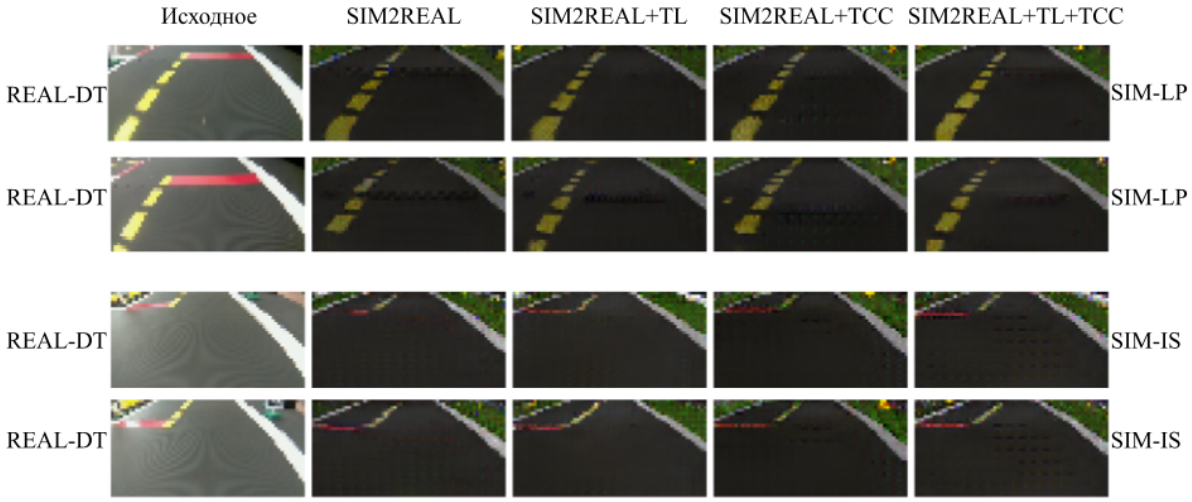

Рисунок 10 – Примеры трансляции изображений из реальности в симуляционные окружения.

кости будут иметь близкие в высокоразменом пространстве представления, что позволяет судить о схожести относительно модели изображений, по которым представления и были получены.

Используя эту процедуру я построил совместную визуализацию представлений для кадров двух наборов данных переноса SIM-LP $\rightarrow$ REAL-DT. На рисунке 11 представлены результаты такой визуализации для базовой SIM2REAL модели и модели с темпоральными ограничениями SIM2REAL+TL+TCC.

На рисунке можно видеть, что использование темпоральной функции потерь делает представления кадров одного эпизода более линейными после стохастического вложения, и таким образом вносит дополнительную структуру, что потенциально способствует трансферу из симулятора в реальность.

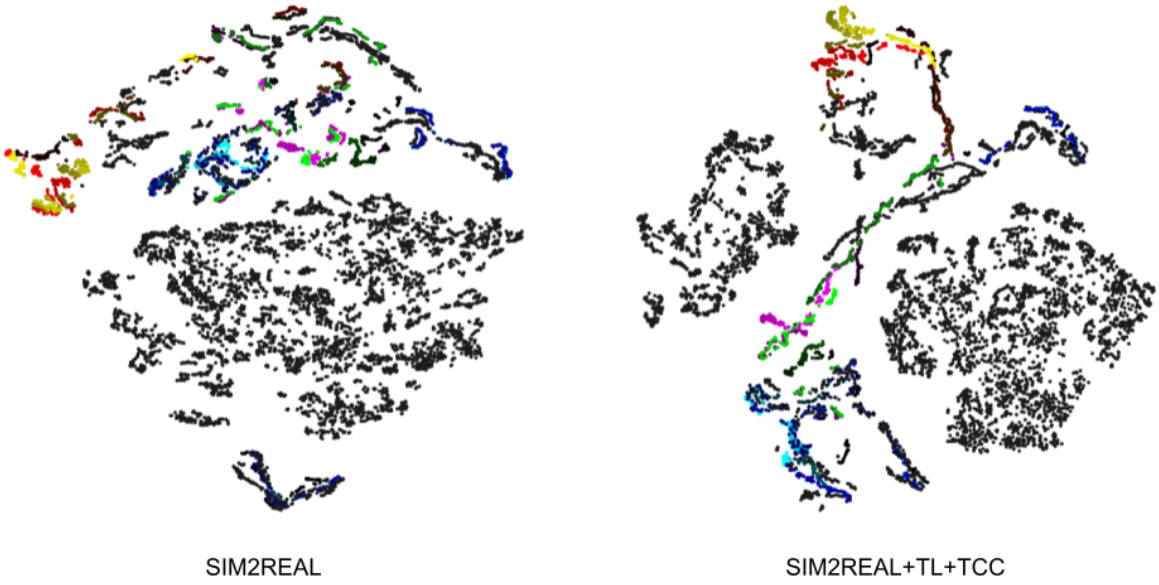

Рисунок 11 – Результаты визуализации скрытых представлений вариантов модели SIM2REAL. Цветами выделены кадры из одного эпизода. Интенсивность цвета показывает насколько близок кадр к началу эпизода.

5.4. Выводы и результаты по главе

В начале этой главы я представил и обосновал условия проведения экспериментов, для достижения целей поставленных в этой работе. Было представлено 6 моделей и описаны особенности каждой из них. Далее были представлены результаты и сравнение этих моделей между собой при переносе из двух симуляционных наборов данных на набор данных из реального мира. Был проведен анализ этих результатов, и в частности, предоставлено объяснение разной степени переносимости между двумя симуляционными доменами. После было проведено исследование полученных моделей путем анализа визуального качества трансляции моделей и визуализации полученных скрытых представлений. В ходе исследования были указаны отличия в визуальном качестве и характеристиках полученных представлений моделей.

ГЛАВА 6. ЗАКЛЮЧЕНИЕ

В процессе этой работы я собрал и обработал симуляционные и реальные наборы данных для автономного вождения и реализовал метод обучения с трансляцией изображений для лучшего переноса из симулятора в реальный мир.

Дополнительно, я предложил использование темпоральных функций потерь. Я сравнил 6 методов обучения и провел анализ полученных результатов.

В итоге я подтвердил, что обучение в симуляторе требует техник Sim2Real для дальнейшего применения в реальном мире. Сравнение показало, что трансляция изображений в качестве Sim2Real метода сокращает в условиях экспериментов доменное несоответствие в среднем относительно САО до 32.54% и относительно СКО до 24.71%. Использование дополнительных ограничений, основанных на темпоральном порядке, позволяет дальше сократить доменное несоответствие в среднем относительно САО до 26.85% и относительно СКО до 17.47%.

У данной работы есть множество продолжений.

Одним из продолжений может быть исследование переноса из симулятора в реальность для других вариаций симулятора и/или явно обозначенных условий реальности. В данной работе я рассматривал две вариации симулятора, соответствующие простому дорожному циклу и городу с перекрестками, и один домен реальности, который совмещал разные условия окружения.

Другим продолжением может стать проведение онлайн тестирования методов, и исследованием эффективности или скорости вывода модели для работы на ограниченных вычислительных ресурсах. В рамках этой работы моей главной целью было доказательство выгоды от использования Sim2Real методов для переноса моделей в реальный мир и исследование влияния темпорального порядка.

По итогам части проводимого в данной работе исследования была опубликована статья [13] на “Workshop on Artificial Intelligence for Autonomous Driving”, проводимого в рамках International Conference on Machine Learning 2020. Код моей работы находится в открытом доступе по адресу: `https://github.com/niksaz/sim2real`.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 End to End Learning for Self-Driving Cars / M. Bojarski [и др.] // ArXiv. — 2016. — Т. abs/1604.07316.

2 Driving Policy Transfer via Modularity and Abstraction / M. Müller [и др.] // CoRL. — 2018.

3 Learning to Drive from Simulation without Real World Labels / A. Bewley [и др.] // 2019 International Conference on Robotics and Automation (ICRA). — 2019. — С. 4818–4824.

4 Domain randomization for transferring deep neural networks from simulation to the real world / J. Tobin [и др.] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 2017. — С. 23–30.

5 The Cityscapes Dataset for Semantic Urban Scene Understanding / M. Cordts [и др.] // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — С. 3213–3223.

6 DenseASPP for Semantic Segmentation in Street Scenes / M. Yang [и др.] // 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2018. — С. 3684–3692.

7 Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping / K. Bousmalis [и др.] // 2018 IEEE International Conference on Robotics and Automation (ICRA). — 2018. — С. 4243–4250.

8 Generative Adversarial Nets / I. Goodfellow [и др.] // Advances in Neural Information Processing Systems. Т. 27 / под ред. Z. Ghahramani [и др.]. — Curran Associates, Inc., 2014.

9 *Nise N. S.* Control Systems Engineering. — John Wiley & Sons, 2007.

10 Efficient ConvNet for real-time semantic segmentation / E. Romera [и др.] // 2017 IEEE Intelligent Vehicles Symposium (IV). — 2017. — С. 1789–1794.

11 Duckietown: An open, inexpensive and flexible platform for autonomy education and research / L. Paull [и др.] // 2017 IEEE International Conference on Robotics and Automation (ICRA). — 2017. — С. 1497–1504.

12 Duckietown Environments for OpenAI Gym / M. Chevalier-Boisvert [и др.]. — 2018. — `https://github.com/duckietown/gym-duckietown`.

13 Imitation Learning Approach for AI Driving Olympics Trained on Real-world and Simulation Data Simultaneously / M. Sazanovich [и др.]. — 2020.

14 Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks / G. Chen [и др.]. — 05.2019.

15 *Ioffe S.*, *Szegedy C.* Batch normalization: Accelerating deep network training by reducing internal covariate shift // arXiv preprint arXiv:1502.03167. — 2015.

16 Dropout: a simple way to prevent neural networks from overfitting / N. Srivastava [и др.] // J. Mach. Learn. Res. — 2014. — Т. 15. — С. 1929–1958.

17 Efficient object localization using Convolutional Networks / J. Tompson [и др.] // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — С. 648–656.

18 *Kingma D. P.*, *Ba J.* Adam: A Method for Stochastic Optimization // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings / под ред. Y. Bengio, Y. LeCun. — 2015.

19 *Liu M.-Y.*, *Breuel T.*, *Kautz J.* Unsupervised Image-to-Image Translation Networks // Advances in Neural Information Processing Systems. Т. 30 / под ред. I. Guyon [и др.]. — Curran Associates, Inc., 2017.

20 Deep Residual Learning for Image Recognition / K. He [и др.] // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — С. 770–778.

21 *Ulyanov D.*, *Vedaldi A.*, *Lempitsky V.* Instance Normalization: The Missing Ingredient for Fast Stylization // ArXiv. — 2016. — Т. abs/1607.08022.

22 *Ioffe S.*, *Szegedy C.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. — Lille, France : JMLR.org, 2015. — С. 448–456. — (ICML'15).

23 *Schroff F.*, *Kalenichenko D.*, *Philbin J.* FaceNet: A unified embedding for face recognition and clustering // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — С. 815–823.

24 Temporal Cycle-Consistency Learning / D. Dwibedi [и др.] // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2019.

25 *Maaten L. V. D.*, *Hinton G. E.* Visualizing Data using t-SNE // Journal of Machine Learning Research. — 2008. — Т. 9. — С. 2579–2605.