МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

БАРАНОВ Дмитрий Владиславович

ДОЛГОВРЕМЕННОЕ ПРЕДСКАЗАНИЕ ПОВЕДЕНИЯ ТРАНСПОРТНЫХ СРЕДСТВ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Магистерская диссертация
специальность 1-31 80 09 Прикладная математика и информатика

Научный руководитель:
Котов Владимир Михайлович
доктор физ.-мат. наук, профессор

Допущена к защите
«_31_»__марта__ 2021 г

Зав. кафедрой дискретной математики и алгоритмики
доктор физико-математических наук,
профессор В.М. Котов

Минск 2021

ОГЛАВЛЕНИЕ

Аннотация

Магистерская диссертация основана на двух подходах к решению задачи предсказания долговременного поведения транспортных средств: использовании революционного метода растеризации окружающей обстановки для извлечения семантики с помощью сверточной нейронной сети и векторного метода кодирования окружающей обстановки с использованием иерархической графовой нейронной сети.

В ходе исследования был разработан принципиально новый способ генерации синтетических данных на основе компьютерной игры с открытым миром GTA V и обучена сеть для их пост-обработки в виде нахождения автомобилей.

Оба вышеназванных подхода были реализованы на основе различных нейронных сверточных и графовых архитектур и фреймворке PyTorch, полученные сети были успешно обучены, а их результаты тщательно сопоставлены.

Annotation

The master thesis is based on two approaches to the problem of long-term vehicle motion prediction: the use of revolutionary technique - rasterization of surrounding environment for better semantic features extraction by neural network and the use of vectorization of environment encoding to be further used as an input to an hierarchical graph neural network.

As a part of study, a brand-new way for the generation of synthetic self-driving dataset was presented, based on the open-world videogame GTA V used as a simulator, along with the training of object-recognition neural network for its post-processing.

Both approaches were fully implemented with different back-end network architectures using open-source framework PyTorch, the corresponding networks were successfully trained and accurately compared to each other.

РЕФЕРАТ

Магистерская диссертация, 54 с., 28 рис., 6 таблиц, 17 формул

НЕЙРОННЫЕ СЕТИ, ГЛУБИННОЕ ОБУЧЕНИЕ, БЕСПИЛОТНЫЕ АВТОМОБИЛИ, ДОЛГОСРОЧНОЕ ПРЕДСКАЗАНИЕ, ПРЕДСКАЗАНИЕ ТРАЕКТОРИЙ, SELF-DRIVING CARS, SDC, GTA V, OBJECT RECOGNITION, ENVIRONMENT RASTERIZATION, TRAJECTORY PREDICTION, ENRIRONMENT VECTORIZATION, МУЛЬТИМОДАЛЬНОЕ ПРЕДСКАЗАНИЕ

Объект исследования – использование нейронных сетей для предсказания движения и поведения транспортных средств.

Цель работы – анализ существующих подходов к долговременному предсказанию поведения транспортных средств; исследование state-of-the-art подходов к построению нейронных сетей и их особенностей в контексте поставленной задачи; изучение принципов реализации нейронных сетей с использованием фреймворка PyTorch; разработка и внедрение оптимизаций в уже существующие подходы к объекту исследования; реализация принципа растеризации и векторизации окружающей среды и их оценка.

Методы исследования – построение нейронных архитектур различной структуры и их оптимизация с учетом особенностей предметной области и данных; разработка методологии и ее применение к использованию GTA V в качестве симулятора для генерации синтетических данных; реализация архитектур, обучение нейронных сетей и сопоставление результатов с уже достигнутыми в данной сфере

Область применения – долгосрочный анализ и предсказание поведения транспортных средств на автомагистралях и перекрестках. Классификация маневров транспортных средств, предсказание траекторий маневров. Системы оценки и автоматического принятия решений для беспилотных автомобилей.

РЭФЕРАТ

Магістэрская дысертацыя, 54 старонкі, 28 ілюстрацый, 6 табліц, 17 формул

НЕЙРОНАВЫЯ СЕТКІ, ГЛЫБІННАЕ НАВУЧАННЕ, БЕСПІЛОТНЫЯ АЎТАМАБІЛІ, ДОЎГАТЭРМІНОВАЕ ПРАДКАЗАННЕ, ПРАДКАЗАННЕ ТРАЕКТОРЫЙ, GOAL-ATTENTION-CAUSE-STIMULUS (GACS), HONDA DRIVING DATASET, SELF-DRIVING CARS, SDC, GTA V, OBJECT RECOGNITION, ENVIRONMENT RASTERIZATION, TRAJECTORY PREDICTION, ENRIRONMENT VECTORIZATION, МУЛЬТЫМАДАЛЬНАЕ ПРАДКАЗАННЕ

Аб'ект даследавання – выкарыстанне нейронавых сетак для прадказання руху и паводзін транспартных сродкаў.

Мэта работы – аналіз існуючых падыходаў да доўгатэрміновага прадказання паводзін транспартных сродкаў; даследаванне state-of-the-art падыходаў да пабудовы нейронавых сетак і іх асаблівасцяў у кантэксце пастаўленай задачы; вывучэнне прынцыпаў рэалізацыі нейронавых сетак з выкарыстаннем фрэймворка PyTorch; распрацоўка і ўкараненне аптымізацый ва ўжо існуючыя падыходы да аб'екта даследавання; рэалізація прынцыпу растэрызацыі і вектарызацыі навакольнага асяроддзя і іх ацэнка.

Метады даследавання – пабудова нейронавых архітэктур рознай структуры і іх аптымізацыя з улікам асаблівасцяў прадметнай вобласці і дадзеных; распрацоўка метадалогіі і яе прымяненне да выкарыстання GTA V у якасці сімулятара для генерацыі сінтэтычных дадзеных; рэалізацыя архітэктур, навучанне нейронавых сетак і супастаўленне вынікаў з ужо дасягнутымі ў дадзенай сферы.

Вобласць прымянення – доўгатэрміновы аналіз і прадказанне паводзін транспартных сродкаў на аўтамагістралях і скрыжаваннях. Класіфікацыя манеўраў транспартных сродкаў, прадказанне траекторый манеўраў. Сістэмы ацэнкі і аўтаматычнага прыняцця рашэнняў для беспілотных аўтамабіляў.

ABSTRACT

Master thesis, 54 p., 28 illustrations, 6 tables, 17 formulas

NEURAL NETWORKS, DEEP LEARNING, SELF-DRIVING CARS, SDC, LONG-TERM PREDICTION, TRAJECTORY PREDICTION, GTA V, OBJECT RECOGNITION, ENVIRONMENT RASTERIZATION, ENRIRONMENT VECTORIZATION, MULTIMODAL PREDICTION

Object of research – usage of neural networks for vehicle long-term motion and behavior prediction.

Objective – analysis of existing approaches to long-term prediction of vehicle behavior; investigation for state-of-the-art approaches to the construction of neural networks and their specifics in the context of the problem; the study of the principles of the implementation of neural networks using the PyTorch framework; development and integration of different optimizations to existing approaches to the object of study; implementation of the principle of rasterization and vectorization of the environment and theirs evaluation.

Research design – construction of neural architectures of various structure and their optimization considering the specifics of the subject field and the data; development of the methodology and its application to the usage of GTA V as a simulator for generating synthetic data; implementation of architectures, training of neural networks and comparison of results with those already achieved in this field.

The scope – long-term analysis and prediction of vehicle behavior on highways and crossroads. Classification of vehicle maneuvers, prediction of maneuver trajectories. Systems of evaluation and automatic decision making for self-driving vehicles.

ВВЕДЕНИЕ

На сегодняшний день люди проводят большую часть своего времени путешествуя; общество в целом выросло зависимым от транспортных средств – в среднем каждый американец проводит более 300 часов за рулем каждый год. Вместе со временем, проведенным за рулем автомобиля, растет и количество несчастных случаев, т.к. текущая транспортная система далека от совершенства. Всемирная организация здравоохранения опубликовала отчет, в котором заявлено, что лишь за 2015 год около 1,25 миллионов человек погибли в результате несчастных случаев на дороге по всему миру.

За последние годы огромную популярность и развитие в автомобильной отрасли получили автономные беспилотные транспортные средства. Данное оживление вызвано недавними прорывами в компьютерном зрении, распознавании изображений и компьютерной перцепции, качество которых стало сопоставимо с человеческим. Однако лишь одного отличного понимания среды еще недостаточно для решения проблемы автономности транспортных средств – беспилотные автомобили должны удовлетворять критериям безопасности в рамках окружения, в котором они действуют.

Одной из самых больших проблем в современных системах помощи водителю (Advanced Driver-Assistance Systems, ADAS) и в решении проблемы полной автономии является человеческий фактор. Беспилотные автомобили и ADAS функционируют в тесной близости с людьми, следовательно, должны уметь предсказывать как поведут себя люди-водители дабы обезопасить окружающую обстановку.

Магистерская диссертация посвящена исследованию, анализу существующих и разработке новых подходов в задаче долгосрочного предсказания поведения транспортных средств, управляемых как человеком, так и автономно.

Людей сложно смоделировать или предсказать в силу их иррациональной натуры. Более того, все люди различны и действует по-разному, из чего следует практически бесконечное количество исходов в каждом дорожном сценарии.

В процессе движения остро необходимо знать, что намереваются делать окружающие транспортные средства, чтобы обеспечить безопасность и оптимальный контроль собственного. Например, на рисунке ниже, если машина №1 собирается выполнить съезд с шоссе, она должна предсказать поведение

остальных участников движения, чтобы совершить безопасный маневр к съезду, если это вообще возможно с точки зрения безопасности.

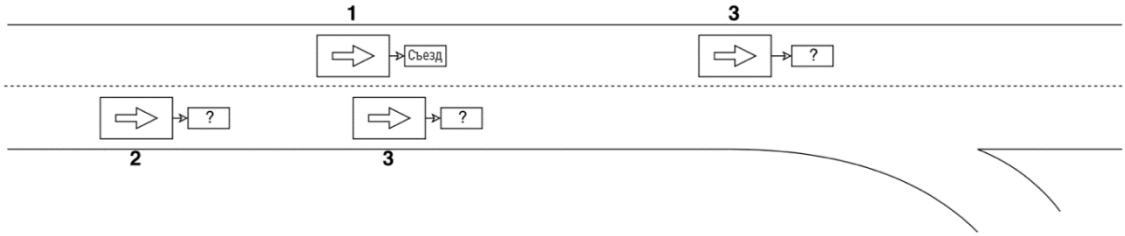


Рисунок 1. Модельная ситуация момента принятия решения машиной №1 для совершения маневра.

Магистерская диссертация является логическим продолжением дипломной работы автора и поделена на две части, в каждой из которых рассматривается независимый подход к решению поставленной задачи.

В дипломной работе акцент смещен в сторону подготовки данных, обогащению данных семантикой и обучению нейронных сетей понимать данную семантическую нагрузку. На основе особо размеченных данных, а именно Honda Research Institute Driving Dataset, строится композитная сверточная нейронная сеть для анализа записей фронтальной камеры и одометрии транспортного средства, обучается и тестируется для предсказания маневров и их причин.

Магистерская диссертация же полностью посвящена революционному взгляду на заданную проблему – в ней решение опирается на принципы растеризации и векторизации истории окружающей обстановки в одном многослойном искусственном кадре и его дальнейшее использование в качестве входа нейронной сети.

В первом разделе второй главы решается проблема отсутствия необходимых данных за счет создания синтетического набора данных, используя компьютерную игру с открытым миром (GTA V) в качестве симулятора и последующим распознаванием и локализацией транспортных средств с помощью нейронной сети Faster R-CNN ResNet-101.

Далее во втором разделе созданный набор данных используется непосредственно в предобработке-растеризации; выводится оптимизируемый функционал с учетом особенностей предметной области; обучается нейронная сеть, способная предсказывать траектории движения транспортных средств на долговременные горизонты.

В третьей главе вводится понятие векторного представления окружающей обстановки, описывается мотивация ее построения и ее достоинства, строится иерархическая графовая нейронная сеть, которая затем обучается для предсказания траекторий.

Разработанные архитектуры реализованы на основе фреймворка PyTorch, обучены и сопоставлены с уже существующими подходами.

В заключении сформулированы направления, идеи и возможности дальнейшей оптимизации и расширения результатов в рамках дальнейших исследований.

ГЛАВА 1

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Временная последовательность

Временная последовательность T определяется как упорядоченная последовательность n действительных переменных:

$$T = \{t_1, t_2, \dots, t_n\}, t_i \in \mathbb{R} \qquad (1.1)$$

Наблюдая некоторый процесс во времени можно собрать упорядоченную коллекцию, равномерно распределенную во времени, действительных наблюдений – временную последовательность. Временная последовательность по определению может быть одномерной или многомерной в зависимости от числа одновременно наблюдаемых величин.

Пусть дана временная последовательность T, тогда подпоследовательностью S последовательности T назовем последовательность длины $m \leq n$, состоящую из непрерывной серии наблюдений из T:

$$S = \{t_k, t_{k+1}, \dots, t_{k+m-1}\} \qquad (1.2)$$

где $1 \leq k \leq n - m + 1$.

1.2 Сверточная нейронная сеть

В 1950-х и 1960-х годах американец Дэвид Хьюбел и швед Торстен Визель начали исследовать зрительную систему кошек и обезьян в медицинской школе Джона Хопкинса. Они опубликовали серию статей, в которых изложена теория о том, что каждый нейрон зрительной коры отвечает за отдельную область поля зрения. Они также постулировали, что визуальная обработка информации происходит каскадом, от нейронов, предназначенных для самых простых форм, до нейронов, которые улавливают более сложные. Открытия в сумме принесли им Нобелевскую премию по физиологии и медицине 1981 года.

В 1980 году японский ученый-компьютерщик Кунихико Фукусима изобрел «неокогнитрон», своего рода нейронную сеть, состоящую из сверточных слоев и слоев с понижением размерности, вдохновившись открытиями Хьюбела и Визеля. Неокогнитрон мог выполнять некоторые основные задачи обработки изображений, такие как распознавание символов.

С 1987 по 1990 год многие исследователи, включая Алекса Вайбеля и Коуичи Ямагути, адаптировали неокогнитрон, внедрив такие инновации как обучение с обратным распространением ошибки, что сильно его облегчило. Затем в 1998 году Ян Лекун разработал LeNet, сверточную нейронную сеть с

пятью сверточными слоями, которая была способна распознавать рукописные цифры почтового индекса с большой точностью. Еще одной важной вехой стала сверточная нейронная сеть AlexNet, созданная украинско-канадским аспирантом Алексеем Крижевским, опубликованная в 2012 году. AlexNet превзошла всех ее соперников в конкурсе ImageNet по распознаванию изображений более чем на 10 процентных пунктов, что ознаменовало начало новой эры в компьютерном зрении.

Примерно до 2015 года задачи с изображениями, такие как распознавание лиц, обычно выполнялись с помощью трудоемких программ с ручным заданием признаком, которые реагировали на черты лица (брови, нос и т.д.). Многие приложения для оптического распознавания текста или распознавания лиц вообще не использовали машинное обучение. Быстрый рост вычислительных мощностей и широкая доступность больших наборов данных, графических процессоров и программного обеспечения для глубокого обучения означали, что примерно в середине 2010-х годов сверточные нейронные сети смогли обеспечить гораздо лучшую точность, чем традиционные методы и стали де-факто стандартом почти для всех задач, связанных с компьютерным зрением в науке и промышленности.

Сверточные нейронные сети (Convolutional Neural Networks, CNNs) отличаются от обычных полносвязных нейронных сетей несколькими ключевыми моментами, которые заключаются в следующем:

1. Нейроны имеют локальную связность, поэтому нет нужны соединять их со всеми выходами предыдущего слоя.
2. Их рецепторные поля (области восприятия) могут перекрываться.
3. В любом из слоев нейроны имеют одинаковые параметры весов применительно ко всему слою.
4. Как правило сверточная нейронная сеть вместо активации сигмоидой использует блок линейной ректификации (Rectified Linear Unit, ReLU)
5. Они чередуют слои свертки со слоями субдискретизации (сэмплинга) или «пулинга»
6. Они могут иметь слои нормализации, чтобы корректировать распределение выходов каждого слоя.

Однако, они все так же используют принцип обучения с учителем и обратное распространение ошибки.

Рассмотрим вышеперечисленные отличия более детально. Во-первых, отметим, что обычная полносвязная нейронная сеть перемножает каждый

входной сигнал согласно предопределенным весам и складывает получившееся результаты. Если же нейроны и веса одинаковые по всему слою (см. пункт 3), то результирующая математическая операция по определению и есть свертка (отсюда и термин сверточные нейронные сети). Если каждый сверточный слой должен иметь такие же пространственные размеры, как и предыдущий в силу свертки, то отсюда следует что рецепторные поля каждого выходного пикселя будут почти полностью перекрываться.

Дабы несколько последовательных слоев не «схлопывались» в один в силу правила композиции линейных преобразований, между ними ставится нелинейность в виде функции $\max(0, x)$ (линейная ректификация, ReLU), где x – выходное значение непосредственно предшествующего слоя свертки. Такая нелинейность менее подвержена эффекту насыщения (как например сигмоида или гиперболический тангенс) и ускоряет обучение, т.к. не активируется на отрицательной полуплоскости и имеет простую производную.

Субдискретизация (сэмплирование) включает в себя взятие всех выходов из некоторой локальной области и получение из них одного выхода: обычно это происходит в форме операции усреднения или взятия максимума (более часто) всех входов. Цель операции – минимально изменить данные, выкинув значительную и избыточную часть информации в конкретном слое, но сохраняя всю полезную информацию. Использование операции взятия максимума может казаться странным, т.к. взятия максимума множества чисел более подвержено влиянию импульсного шума, однако много сверток неявно немножко обладают свойствами сглаживающего фильтра (фильтра высоких частот), поэтому операция максимума не вызывает значимой деградации выхода.

Заметим, что все перечисленные изменения относительно обычных полносвязных сетей все еще описываются тем же самым математическим аппаратом, с небольшими изменениями в части ReLU и сэмплинга, однако последние все еще позволяют градиенту распространяться по системе, а значит допускают использование обратного распространения ошибки для обучения сети.

Следует добавить, что несколько сверточных слоев все же могут располагаться непосредственно друг за другом. Хоть фактически это и эквивалентно одной большой свертке, но такая схема может сильно повлиять на общую вычислительную нагрузку. Например, три свертки 3×3 эквивалентны одной свертке 7×7, т.е. можно обойтись выполнением 29 операций вместо 49.

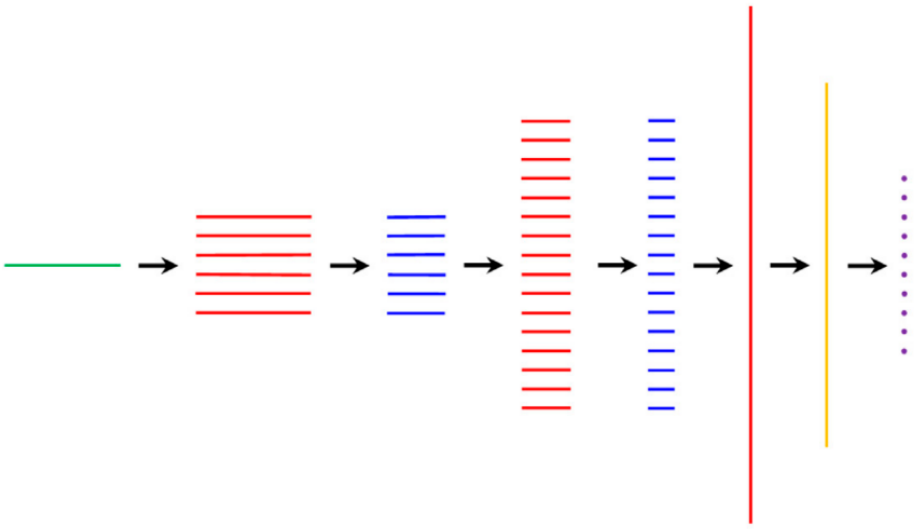
В целом, становится очевидно, что сверточные нейронные сети составляют хорошую альтернативу полносвязным. Одним из их преимуществ является непрерывное движение от локальных операций на изображении к глобальным в

поисках все больших и больших признаков или объектов. Также стоит отметить, что пространственная инвариантность, достигаемая в сверточной нейронной сети, очень важна для ряда задач и совершенно нереализуема в полносвязных сетях.

В процессе сверток растет количеству каналов и уменьшается пространственный размер изображения в силу сэмплинга, поэтому дабы привести размер выхода к желаемому и как-то постобработать результаты свертки в конце сети как правило используется один или несколько полносвязных слоев. Размер последнего слоя определяется на основе желаемого результата и задачи: вероятностей классов в задаче классификации, относительных или абсолютных координат и т.д.

1.2.1 LeNet-5

В 1998 году Ян ЛеКун опубликовал ставший знаковым пример сверточной нейронной сети, использовавшей многое из пунктов, упомянутых выше. Входное изображение параллельно подавалось на вход блоку из 6 нейронов, реализующих свертку, где за каждым нейроном непосредственно шел нейрон субдискретизации. За 6 нейронами субдискретизации далее шел блок из 16 сверточных нейронов, где за каждым снова следовал нейрон субдискретизации.



	Вход	C1	S2	C3	S4	C5	F6	Выход
N	1	6	6	16	16	120	84	10
n×*n*	32×32	28×28	14×14	10×10	5×5	1×1	1	1
r×*r*		5×5	2×2	5×5	2×2	5×5	120	84

Рисунок 1.1 Архитектура LeNet-5

Далее выход пропускался через два полносвязных слоя и поступал на вход классификатору на основе радиально-базисных функций (детали представлены на рисунке 1.1).

Рассмотрим архитектуру сети более подробно:

1. Шесть скрытых слоев в LeNet обозначены как C1, S2, C3, S4, C5, F6, где “C”, “S”, “F” означают соответственно свертку (convolution), субдискретизацию (subsampling), полносвязный слой (fully-connected layer). Таким образом свертка и субдискретизация чередуются вплоть до слоя C5, начиная с которого сеть превращается в полносвязную и более субдискретизация уже не нужна.
2. В табличной части рисунка 1.1 N обозначает число нейронов каждого типа, $n \times n$ - двухмерный пространственный размер слоя (в случае одного числа – одномерный размер), $r \times r$ – размер рецепторных полей каждого нейрона, $s \times s$ – шаг свертки.
3. В теории, входы и выходы сверточных слоев должны соотноситься между собой, т.е. число входов должно быть равно числу выходов, умноженному на r^2. На деле же это справедливо только для C1. В остальных трех случаях нужно учитывать следующее:
    a. Для C3 в теории задумывалось использовать для каждого нейрона всю глубину карты признаков на входе, однако это приводило к чрезмерной вычислительной сложности, поэтому благодаря особой схеме соединения слоев кол-во соединенных нейронов было уменьшено с 96 до 60.
    b. Для C5 весьма неочевидно почему было выбрано использовать именно 120 нейронов, т.к. оригинальная авторская статья не приводит никакую аргументацию. Однако, C5 является полносвязным слоем, поэтому вероятно 120 есть наибольшее число нейронов, которое укладывается в разумную вычислительную сложность.
    c. Для F6 нужно заметить, что перед ним находится полносвязный слой, поэтому число входов больше не зависит от $r \times r$ и равно просто 120. Число 84 появляется из наблюдения, что выход слоя F6 возвращается к формату изображения и представляет из себя изображение 7×12 – некоторую визуальную интерпретацию цифры на входном изображении. Оказывается, что такое представление полезнее чем классическое кодирование 1-из-N для классификации символов, т.к. последний имеет чрезмерную избыточность, когда нужно различать десятки вероятных возможностей, поэтому лучше представить выход как некоторую стилизованную картинку символа.
4. Финальный слой интерпретирует получившуюся «стилизованную» картинку 7×12 как одну из 10 цифр, используя классификатор на основе радиально-базисных функций.
5. Слои субдискретизации S2 и S4 имеют размер входа 2×2 и выполняют операцию усреднения. Они также включают в себя обучаемую линейную

функцию, состоящую из двух параметров: мультипликативного (скейлинг) и аддитивного (смещение).

6. Несмотря на то, что выход 5×5 слоя S4 приводит к тому, что нейроны C5 имеют рецепторное поле 1×1 (было решено использовать двухмерный выход 1×1, так как это будет удобно, в случае если сеть дальше будет расширяться в размере), слой C5 можно справедливо считать полносвязным.

Рассмотрев детали архитектуры, теперь стоит понять почему именно эта архитектура была выбрана для задачи распознавания числовых цифр. В задаче распознавания рукописного почтового индекса цифры представляют из себя неточно расположенные символы, которые также могут иметь неточный масштаб. Кроме того, они подвержены множеству стилевых вариаций и искажений: последние включают в себя локальные аномалии, неправильно расположенные точки, кресты, соединения и шум, который уже присутствовал на материале или случайно попал на него постфактум в виде пятен, потертостей или клякс. Все эти факторы затрудняют прочтение индекса человеком – и, возможно, не менее трудны для машин. Чтобы решить эту проблему, алгоритм считывания должен обеспечивать пространственную, масштабную и искажающую инвариантность.

К счастью, сверточные нейронные сети имеют пространственную инвариантность по построению, благодаря сверткам – одно и то же маленькое ядро реплицируется и применяется по всему слою сети. Аналогично, многие из вариаций, вызванных изменением масштаба, стиля или другими искажениями, могут быть устранены путем за счет отсутствия требования к точному совпадению взаимных расположений небольших признаков. В частности, наборы сверточных слоев, каждый из которых детектирует определенный микро-признак (такие, как например сегменты, пересечения, углы), могут быть скомбинированы таким образом, что они будут нечувствительны небольшим изменениям в расположении микро-признаков. Эффект достигается применением субдискретизации между свертками.

Таким образом видно, что архитектура LeNet воплощает наиболее важные инварианты, используя наборы слоев свертки и субдискретизации. На самом деле многие из упомянутых выше вариативных проблем носят случайных характер, и самый простой способ получить хорошую обобщающую способность здесь – это обучение сети на большом объеме разнородных данных. Однако есть еще один метод, который может помочь в этом, а именно нормализация входных данных путем использования обучаемого мультипликативного параметра и обучаемого аддитивного параметра в слое субдискретизации, которые будут применяться перед подачей выхода на сигмоидную активационную функцию. Дело в том, что если сделать сигмоиду слишком широкой, то она вырождается в линейную

функцию, что приводит к потере необходимой нелинейности, но если сделать ее слишком узкой, то наоборот – сигмоида потеряет свою линейную составляющую и обратное распространение ошибки будет медленным и неэффективным. Очевидно, что нужно соблюдать баланс между этими двумя крайностями, причем в идеале сеть должна быть способна самостоятельно его найти в ходе обучения, используя упомянутые выше параметры для корректировки.

В LeNet слои C1-S4 можно интерпретировать как стадии детекции признаков в пространстве изображения, последующие слои – как стадии детекции более абстрактных признаков, что приводит к заключительному этапу классификации самой цифры. Такая сложная сеть обучается, используя стандартный алгоритм обратного распространения ошибки: несмотря на ее глубину, обучение не является трудноразрешимой задачей, как это было бы при использовании полносвязной нейронной сети такого размера.

Обобщив ключевые аспекты архитектуры, соответствующие современным моделям, имеем следующие особенности:

- Входные изображения фиксированного размера;
- Группировка сверточных и объединяющих слоев в блоки;
- Повторение сверточно-объединяющих блоков в архитектуре;
- Увеличение количества фильтров с глубиной сети;
- Раздельное выделение признаков и классификация;

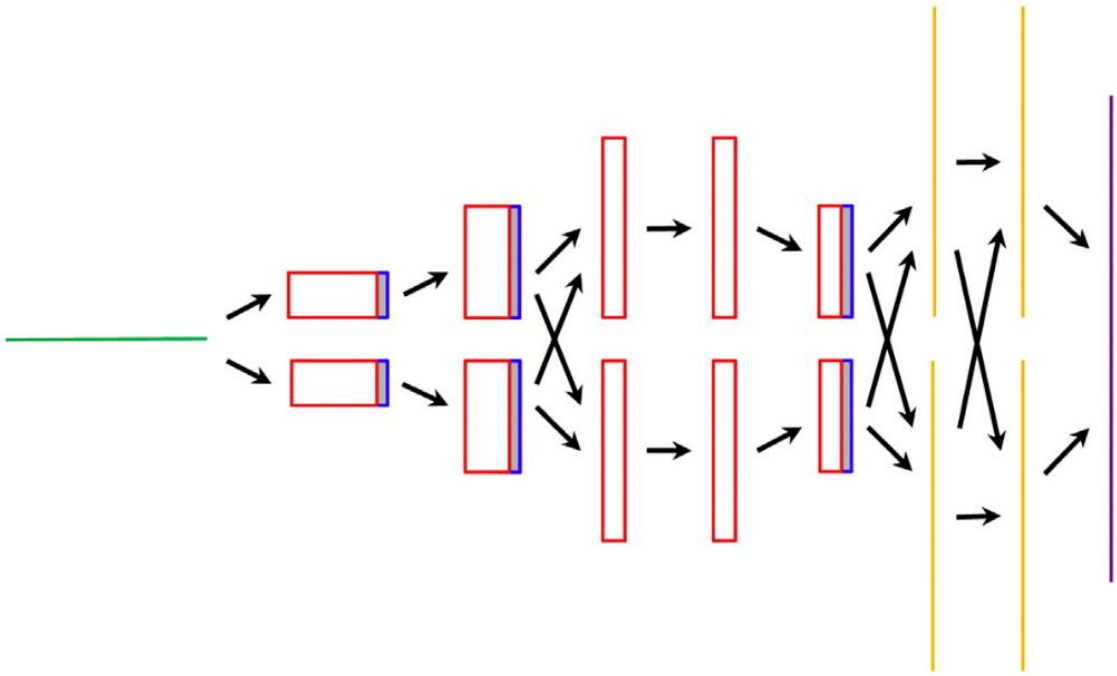
1.2.2 AlexNet

AlexNet была разработана специально для ImageNet Challenge (соревнования по распознаванию объектов на картинках), который состоялся в 2012 году. В самом деле, исключительная ценность этой задачи ведет к возможности мощного развития фундаментальной науки, лежащей в ее основе, а также прорывов в смежных технологиях. Создатели AlexNet задались целью устранить ограничения уже довольно старых сетей, основанных на свертках, которые явно не были на пьедестале классификации.

Чтобы достичь этого, им пришлось радикально улучшить архитектуру сверточной нейронной сети, и это неизбежно привело к созданию очень большого программного комплекса для работы с ней; затем им пришлось значительно ускорить его с помощью графических процессоров - отнюдь не небольшая задача, поскольку это означало повторную оптимизацию программного обеспечения для соответствия аппаратному обеспечению; наконец, им пришлось найти, как снабдить программную систему очень большим обучающим набором - опять же, задача не из легких, поскольку требовалось тщательно обучать беспрецедентно большое количество параметров, и для этого потребовалось несколько нововведений в архитектуре программного комплекса.

Прежде чем продолжить, мы рассмотрим различные детали архитектуры AlexNet. Одной из характерных особенностей архитектуры является горизонтальное разделение изображения по всей сети, чтобы для обсчета обеих получившихся частей использовать два графических процессора. В теории, это должно было наложить серьезные ограничения на архитектуру, но на практике это оказалось относительно легко разрешимой проблемой (на самом деле, это можно было бы рассматривать как одно из самых хитрых нововведений в системе). Решение связано с тем, что довольно разумно позволить каждому графическому процессору отбирать половину изображения и обсчитывать его, а затем в слиянии объединять независимо полученные результаты. Слияние происходит прямо перед последними двумя полносвязными слоями, за которыми следует классификатор в виде многомерной логистической функции.

Одной из особенностей AlexNet, которая отличала ее от LeNet, было использование недавно разработанной нелинейной функции активации ReLU. Авторы обнаружили, что это может ускорить тренировку примерно в 6 раз по



Слой	Вход	C1	C2	C3	C4	C5	F6	F7	Выход
N	3	96	256	384	384	256	4096	4096	1000
$n \times n$	224×224	55×55	27×27	13×13	13×13	13×13	6×6	1	1
$r \times r$		11×11	5×5	3×3	3×3	3×3	1×1	1	1
$s \times s$		4×4	1×1	1×1	1×1	1×1	1×1	1	1
d		3	48	256	192	192	256	4096	4096

Слой	S1	S2	S3
N	96	256	256
$n \times n$	27×27	13×13	6×6
$r \times r$	3×3	3×3	3×3

Рисунок 1.2 Архитектура AlexNet

сравнению с обычной сигмоидной функцией. Поскольку в AlexNet требовалось огромное количество эпох обучения, это было ценным нововведением. Фактически, ReLU не нуждаются в нормализации ввода, чтобы предотвратить их насыщение (очевидно, что линейный отклик никогда не может насыщаться). Тем не менее авторы обнаружили, что по-прежнему полезно включать элемент того, что они называли «нормализацией яркости», аналогичный обучаемым параметрам масштабирования и сдвига сигмоиды в LeNet. Выявлено, что его включение понизило частоту ошибок на 1-2%.

Также стояла задача сильного уменьшения избыточности системы на начальных стадиях. Решение было найдено довольно грубое – использовать шаг свертки 4×4 в слое C1 одновременно с использованием большого рецептивного поля нейрона, 11×11, дабы хоть как-то компенсировать ущерб объему извлекаемой информации от большого шага свертки. Собственно, это место является единственным во всей архитектуре, где используется шаг больший единицы, кроме разве что слоев субдискретизации S1, S2, S3. В этих слоях потеря информации из-за шага компенсируется рецептивным полем размера 3×3 вместо привычного 2×2, которые частично пересекаются, поэтому такая субдискретизация получила название «накладывающаяся субдискретизация».

Незадолго до завершения AlexNet сторонние исследователи представили новую технику под названием «отсев» (dropout). Целью ее было уменьшение числа случаев переобучения сети. Это было достигнуто путем случайного обнуления пропорции (обычно до 50%) весов для каждого цикла обучения; этот довольно неожиданный метод работал весьма хорошо: он не позволял скрытым слоям слишком сильно полагаться на определенные данные, подаваемые в них (другой способ взглянуть на это состоит в том, что имеется случайная выборка из 2^{WH} различных архитектур, поэтому вероятность переобучения любой из них должна быть незначительной: в действительности сеть обучается по множеству независимых путей, поэтому любой индивидуально переобученный путь не должен влиять на общее качество).

Авторы включили эту технику в AlexNet. Для ее применения выходной сигнал каждого нейрона случайным образом устанавливается равным нулю с вероятностью 0,5. Это делается до прямого прогона входных данных, и поэтому затронутые нейроны не участвуют в обратном распространении ошибки. На следующем прямом проходе другой набор выходных сигналов нейронов устанавливается равным нулю с вероятностью 0,5, и снова затронутые нейроны не вносят вклад в обратное распространение; и то же самое повторяется для более поздних проходов. Во время тестирования происходит аналогичная процедура, при которой все выходы нейронов умножаются на 0,5. Фактически, умножение выходов всех нейронов на 0,5 является приближением к взятию статистического геометрического среднего всех распределений вероятностей

выходных локальных нейронов и основывается на том, что среднее геометрическое не слишком далеко от среднего арифметического. "Отсев" был встроен в первые два уровня AlexNet и значительно уменьшил переобучение: основным недостатком его включения было удвоение количества итераций, необходимых для сходимости. Время обучения было удвоено, но эффективность обучения выросла еще больше - все это было намного лучше, чем обычный результат.

AlexNet была обучена с использованием 1,2 миллиона изображений, доступных из задачи ImageNet ILSVRC — это число является подмножеством полных 15 миллионов в базе данных ImageNet. Фактически, ILSVRC-2010 был единственным подмножеством, для которого были доступны тестовые метки, где было около 1000 изображений в каждой из 1000 категорий.

Стоит обратить внимание, что рекордные результирующие метрики были достигнуты не только путем разработки выигрышной архитектуры с нуля и создания правильного набора данных для ее адекватного обучения, но также благодаря сокращению времени обучения до достижимого уровня. В этом отношении решающее значение имела реализация на GPU. Даже с парой графических процессоров время обучения занимало примерно неделю, работая 24 часа в сутки, чтобы справиться с задачей. Без графических процессоров это заняло бы примерно в 50 раз больше времени - скорее всего, около года. Наконец, следует отметить, что графические процессоры обеспечивают очень хорошую и эффективную реализацию из-за своего внутреннего параллелизма и, следовательно, их способности обрабатывать большие наборы данных за меньшее количество циклов. Важно, что каждый слой сверточной нейронной сети полностью однороден и поэтому хорошо адаптирован для параллельной обработки. Также стоит отметить, что графические процессоры в свою очередь хорошо адаптированы для параллельной работы, поскольку они могут напрямую читать и писать в память друг друга, избегая необходимости перемещать данные через память самой машины.

Обобщив ключевые аспекты архитектуры, соответствующие современным моделям, имеем:

- Использование функции активации ReLU после сверточных слоев и многомерной логистической функции для выходного слоя;
- Использование максимального пулинга вместо усредняющего пулинга;
- Использование регуляризации прореживанием между полностью связанными слоями;
- Паттерн, при котором выход сверточного слоя подается непосредственно на другой сверточный слой;

1.2.3 VGG

Развитие глубоких сверточных нейронных сетей для задач компьютерного зрения, казалось, стало в некоторой степени темной магией после AlexNet.

Важной работой, которая стремилась стандартизировать проектирование архитектуры для глубоких сверточных сетей и разрабатывать гораздо более глубокие и более эффективные модели в процессе, была статья 2014 года под названием «Очень глубокие сверточные сети для крупномасштабного распознавания изображений» Карена Симоняна и Эндрю Циссермана. Их архитектура обычно упоминается как VGG в честь названия их лаборатории, Visual Geometry Group в Оксфорде.

Входное изображение пропускается через серию сверточных слоев, использующих фильтры с маленьким размером рецептивного поля: 3×3, что является наименьшим возможным значением, способным уловить понятия верх/низ, лево/право, центр – и функцию ReLu в качестве активации. В некоторых конфигурациях также используются особые линейные трансформации – свертки 1×1. Шаг свертки во всей сети устанавливается равным 1×1. Субдискретизация выполняется в нескольких слоях, выполняющих операцию максимума в окне 2×2 и шагом 2×2, которые, однако, следуют не за всеми сверточными слоями.

Хоть VGG и базировалась на AlexNet, но несколько особенностей выгодно отличало ее от предшественника:

1. Вместо использования больших рецептивных полей нейронов, как это было в AlexNet с размером ядра свертки 11×11 и шагом 4×4, VGG использует очень маленькие рецептивные поля с размером ядра свертки 3×3 и шагом 1×1. Дополнительно это еще и снижает число параметров.
2. VGG включает в себя свертки с размером ядра 1×1, чтобы внести больше нелинейности без изменения рецептивных полей.
3. Свертки с маленьким размером ядра позволяют VGG иметь большее количество слоев, а как известно большее число слоев приводит к лучшему качеству. Другая одновременно созданная с рассматриваемой архитектура, Inception, тоже следует данной идеологии.

Был разработан ряд вариантов архитектуры, хотя две из них упоминаются чаще всего с учетом их производительности и глубины. Они названы по количеству слоев: это VGG-16 и VGG-19 для 16 и 19 обучаемых слоев соответственно. Проектные решения в моделях VGG стали отправной точкой для простого и прямого использования сверточных нейронных сетей.

Обобщив ключевые аспекты архитектуры, соответствующие современным моделям, имеем:

- Использование очень маленьких сверточных фильтров, например, 3×3 и 1×1;
- Использование максимального пулинга с размером 2×2 и шагом тех же размеров;
- Важность объединения сверточных слоев вместе перед использованием пулинга;
- Разработка очень глубоких (16- и 19-слойных) моделей;

1.2.4 Inception

Хотя VGG и обеспечивает феноменальную точность на датасете ImageNet, ее развертывание на графических процессорах даже не самого скромного размера является проблемой из-за огромных вычислительных ресурсов как с точки зрения памяти, так и времени, необходимого для ее применения. Проблема возникает из-за очень широких сверточных слоев. Например, сверточный слой с ядром 3×3, который принимает на вход 512 каналов и выдает 512 каналов, имеет число операций порядка $9 * 512 * 512 \approx 2 * 10^6$.

В процессе операции свертки для некоторого выходного пикселя, каждый выходной канал (один из 512 в примере выше) соединен с каждым входным каналом, поэтому такая архитектура называется сильносвязанной. Inception сеть основана на идее, что большинство активаций в глубокой сети не нужны (нулевые) либо избыточны из-за корреляции между ними. Следовательно, наиболее эффективная архитектура глубокой сети будет иметь разреженную связь между активациями, что означает, что не все 512 выходных каналов будут иметь связь со всеми 512 входными. Существуют методы, который позволяют удалить такие избыточные соединения, что приведет к разреженным весам и связям. Однако вычислительные ядра в пакетах BLAS или CUDA не

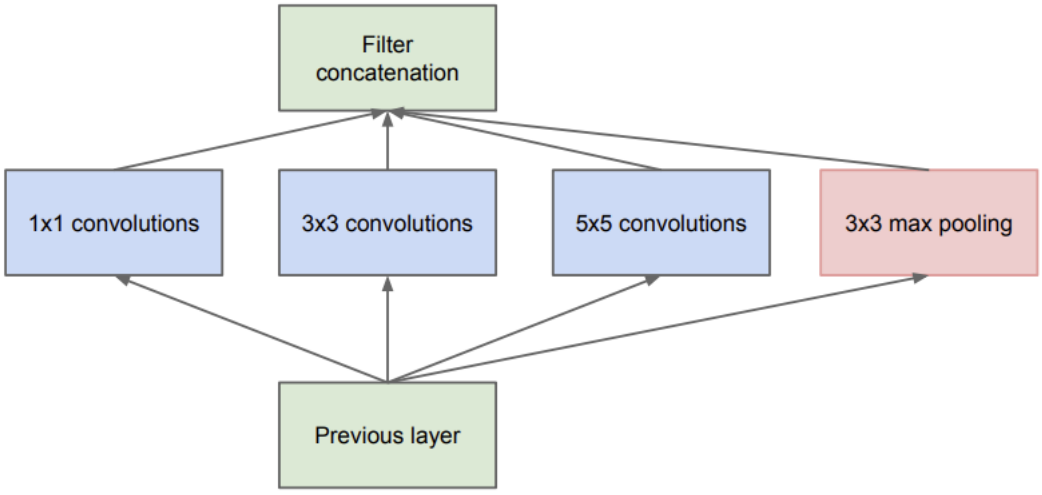


Рисунок 1.3 Базовый Inception модуль

оптимизированы для перемножения разреженных матриц, поэтому считать их будет еще медленнее, чем их плотные версии.

Дабы это побороть в Inception был придуман особый модуль (inception module), который аппроксимировал разреженную сверточную нейронную сеть с помощью конструкции из обыкновенных плотных слоев (рисунок 1.3). Так как только небольшая часть нейронов является полезной, как уже было отмечено выше, число сверточных нейронов с фильтром определенного размера нужно было оставить небольшим. Также, чтобы улавливать детали на различных масштабах, были использованы фильтры размера $5 \times 5, 3 \times 3, 1 \times 1$.

Еще одним важным свойством модуля является наличие свертки 1×1 (так называемого «узкого места»), которая позволяет драматически снизить число вычислительных ресурсов для обсчета изображения. Давайте возьмем в качестве примера первый inception модуль, который имеет 192 канала в качестве входа. Он имеет всего 128 фильтров с размером ядра 3×3 и 32 фильтра с размером ядра 5×5. Число вычислений для фильтров размера 5×5 пропорционально числу параметров $5 * 5 * 32 * 192 \approx 1{,}5 * 10^5$, что может привести к взрыву итогового числа при рассмотрении все более глубоких слоев, когда ширина сети и число фильтров будут расти. Чтобы избежать данной проблемы, inception модуль использует свертки 1×1 перед применением сверток большего размера, чтобы уменьшить число входных каналов перед подачей входа в эти самые свертки. Такие образом в первом inception модуле входные данные сначала передаются в свертки 1×1 всего с 16 фильтрами, а затем уже передаются в свертки размера 5×5 – это сокращает число параметров до $16 * 192 + 5 * 5 * 32 * 16 \approx 1{,}5 * 10^4$.

Интересным изменением, внесенным в Inception, является также замена полносвязных слоев в конце сети на простые глобальные усредняющие субдискретизации, которые усредняют значения каналов по двухмерной карте признаков после последнего сверточного слоя. Такое решение еще сильнее уменьшает число параметров сети (это можно понять на примере сети AlexNet, где полносвязные слои содержат приблизительно 90% от всех параметров сети). Использование глубокой и широкой сети позволяет безболезненно для точности убрать эти финальные полносвязные слои и получить топ-5 точность 93,3% на ImageNet, работая при этом куда быстрее VGG.

Обобщив ключевые аспекты архитектуры, соответствующие современным моделям, имеем:

- Разработка и использование повторяющихся Inception модулей;
- Масштабное использование свертки 1×1 для уменьшения количества каналов;
- Использование обратной связи в нескольких точках сети;
- Разработка очень глубоких (22-слойных) моделей;
- Использование полного усредняющего пулинга;

1.2.5 ResNet

В соответствии с тем, что было продемонстрировано до сих пор, увеличение глубины должно повысить точность сети, если не забывать и бороться с переобучением. Однако проблема с сильно увеличенной глубиной заключается в том, что сигнал, необходимый для изменения весов, который исходит от конца сети путем сравнения истинной метки и предсказания, по мере обратного распространения слабеет и становится очень слабым на ранних слоях из-за увеличения глубины. По сути своей, это означает, что начальные слои практически не обучаются. Данная проблема называется проблемой исчезающего градиента. Вторая проблема, связанная с обучением более глубоких сетей, заключается в решении оптимизационной задачи в все большем пространстве параметров и, поэтому, наивное добавлении слоев, приводит к более высокой ошибке обучения. Данная проблема получила название проблема деградации.

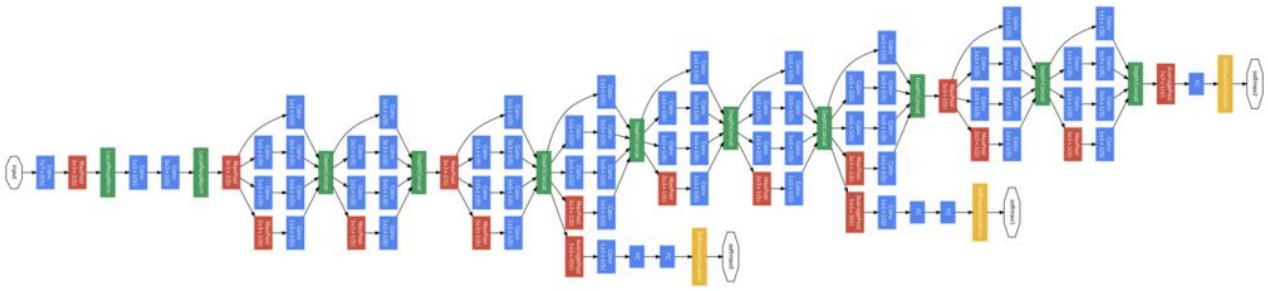


Рисунок 1.4 Архитектура GoogLeNet

Остаточные сети типа ResNet позволяют бороться с этими проблемами благодаря наличию в них специальных остаточных модулей, которые используют соединения быстрого доступа – соединения, где входные данные передаются на следующий уровень как есть или возможно проходя через набор сверток 1×1.

Интуицию почему данный способ работает можно получить в следующем примере. Представим, что имеется сеть A, которая имеет ошибку обучения равную X. Построим сеть B, добавив несколько слоев поверх A и положив в них параметры таким образом, чтобы они никак не видоизменяли выход A (тогда

очевидно, что сеть B должна иметь такую же ошибку обучения как и A, равную X). Обозначим добавленные слои как C. В процессе обучения размер ошибки сети B не может превышать размер ошибки сети A, но т.к. это происходит в действительности, то единственной причиной может служить тот факт, что выучивание тождественного отображения (передачи входа дальше как есть) в добавленных слоях C является нетривиальной задачей для оптимизатора. Дабы облегчить ее и сделать тривиальной, остаточный модуль добавляем прямое соединение между входом и выходом, реализующим тождественное отображение, и слоям C остается только доучить новые признаки на основе имеющихся. Так как C учит только оставшиеся признаки, весь модуль получил название остаточный.

Архитектура в целом аналогична VGG и состоит в основном из фильтров 3x3 с добавлением остаточных модулей и соединений быстрого доступа как описано выше, чтобы сформировать остаточную сеть. Кроме того, как и в Inception, в нем используется глобальная усредняющая субдискретизация, за которой следует слой классификации. Благодаря упомянутым изменениям ResNet была обучена с глубиной сети до 152 слоев. Сеть обеспечивает лучшую точность, чем VGG и Inception, но при этом более эффективен с точки зрения вычислений, чем VGG.

Обобщив ключевые аспекты архитектуры, соответствующие современным моделям, имеем:

- Использование соединений быстрого доступна;
- Создание, использование и неоднократное повторение остаточных блоков;
- Создание очень глубоких (152-слойных) моделей;

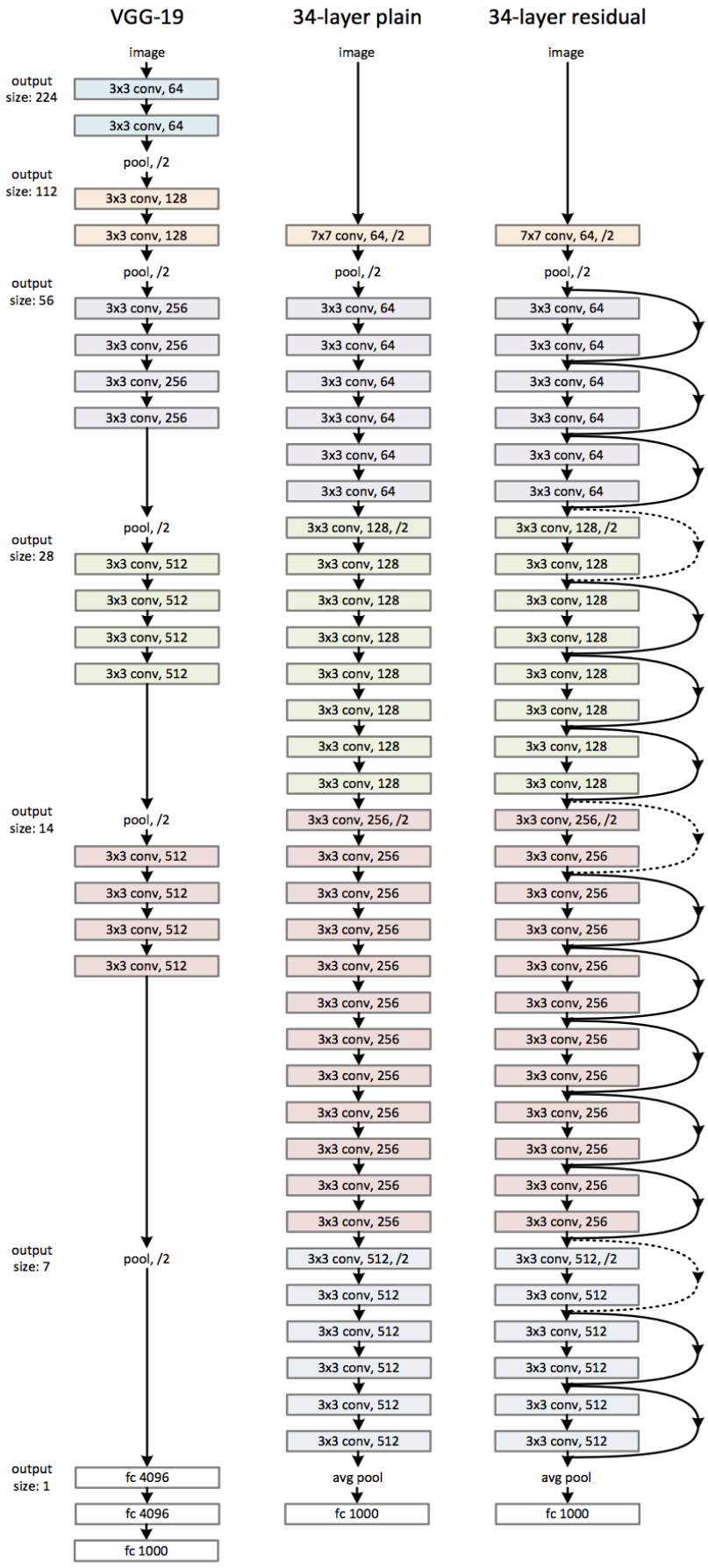


Рисунок 1.5 Архитектура ResNet

ГЛАВА 2

РАСТЕРИЗАЦИОННЫЙ ПОДХОД

2.1 Обзор литературы и существующих подходов

В последнее десятилетие был предложен ряд методов решения проблемы прогнозирования будущего движения участников дорожного движения. Мы сперва рассмотрим подходы, обычно используемые на практике в промышленности. Затем мы обсудим подходы с использованием классического машинного обучения, а также методов глубинного обучения.

Точный прогноз движения других агентов — это важная составляющая автономных систем. В частности, данное прогнозирование тесно связано с планированием собственной траектории SDV, поскольку важно точно оценить будущее состояние окружающей обстановки, чтобы правильно запланировать собственный путь SDV через динамическую среду. Неточное предсказание движения может привести к несчастным случаям, о чем свидетельствует столкновение “Таноса” MIT и “Скайнета” Cornel во время DARPA Urban Challenge 2007 года.

Большинство развернутых в реальном мире автономных систем используют устоявшиеся инженерные подходы для прогнозирования движения. Общий подход состоит в вычислении будущего движения объекта путем распространения его состояния во времени на основе кинематики и допущений базовой физической системы. Оценка состояния обычно включает в себя положение, скорость, ускорение и курс объекта, а для оценки и распространения самого состояния в будущем используются такие методы, как фильтр Калмана. Например, в развернутой системе Honda, фильтр Калмана используется для прогнозирования движения транспортных средств вокруг SDV. Хотя этот подход хорошо работает для краткосрочных прогнозов, его эффективность ухудшается для более длинных временных горизонтов, поскольку модель игнорирует окружающий контекст (например, дороги, других субъектов, правила дорожного движения).

С другой стороны, компонент прогнозирования движения системы Mercedes-Benz использует картографическую информацию в качестве ограничения для вычисления будущего положения автомобиля. Система сначала связывает каждый обнаруженный автомобиль с одной или несколькими полосами движения на карте. Затем генерируются все возможные пути для каждой пары (транспортное средство, связанная с ним полоса) на основе топологии карты, связности полосы и текущего состояния транспортного средства. Такая эвристика дает разумные прогнозы в обычных случаях, однако она плохо масштабируется и не может моделировать необычные или исключительные сценарии.

В качестве альтернативы существующим инженерным подходам, учитывая большие объемы данных для обучения, наш предлагаемый подход автоматически

выучивает тот факт, что транспортные средства обычно подчиняются правилам дорожного движения на дорогах и ограничениям полос движения, но в то же время также способны обрабатывать выбросы, т.е. исключительные ситуации.

2.2 Допущения

Допустим, что мы имеем доступ к потоку информации от датчиков SDV в реальном времени, таким как лидар, радар или камера, установленным на борту транспортного средства.

Далее допустим, что мы уже имеем функционирующую систему, обрабатывающую потоки данных с датчиков, способную в реальном времени обнаруживать и отслеживать перемещение агентов. Например, мы можем воспользоваться любым количеством фильтров Калмана, нашедших весьма широкое применение в промышленности, беря данные с датчиков в качестве входа и выдавая записи каждого индивидуального агента, представляющие оценки его состояния через заданные интервалы времени. Оценка состояния содержит следующую информацию об агенте:

1. bounding box – ограничивающая прямоугольная рамка;
2. координаты местоположения в пространстве;
3. скорость;
4. ускорение;
5. курс;
6. скорость изменения курса;

И наконец мы предполагаем наличия доступа к карте местности, в которой функционирует SDV, обобщающей наличие дорог и пешеходных переходов, направления полос и другую релевантную гео-информацию, и информацию карты.

2.3 Формализация задачи

Обозначим всю статическую информацию, связанную с картой, как $\mathcal{M}$. Набор дискретных моментов времени оценок состояний обозначим $\mathcal{T} = \{t_1, t_2, \dots, t_T\}$, где временной интервал между последовательными оценками постоянен и равен 0,1 секунды (оценки дискретизируются с частотой 10Hz).

Обозначим оценку состояния агента i в момент времени t_j как s_{ij}, где $i = 1, \dots, N_j$, N_j - количество уникальных агентов, отслеживаемых в момент времени t_j. Заметим, что в общем случае это число не является константой, потому что новые агенты могут появляться в радиусе действия датчиков, а старые – исчезать.

Имея данные $\mathcal{M}$ и оценки состояния всех агентов до момента времени t_j включительно (обозначим S_j), наша задача – предсказать последовательность будущих состояний $[s_{i(j+1)}, s_{i(j+2)}, \dots, s_{i(j+H)}]$, где H – означает количество шагов,

на которое мы предсказываем будущие состояния (горизонт предсказания). Без ограничения общности, упростим нашу задачу с предсказания оценок всех агентов до предсказания оценки одного агента i, а именно его будущие местоположения $\left[x_{i(j+1)}, x_{i(j+2)}, \dots, x_{i(j+H)}\right], \left[y_{i(j+1)}, y_{i(j+2)}, \dots, y_{i(j+H)}\right]$

Прошлые и будущие местоположения в пространстве представлены в агенто-центрической системе координат, т.е. в системе координат с центром в положении агента в момент времени t_j, где ось oX направлена вперед по ходу движения, а ось oY направлена от агента в сторону «левой руки водителя».

2.4 Описание процесса растеризации

Для описания растеризации сначала введем понятие векторного слоя, образованного набором полигонов и линий, принадлежащих некоторому общему типу. Например, в случае элементов карты мы имеем векторный слой дорог, пешеходных переходов и так далее. Для растеризации векторного слоя в пространстве RGB каждому векторному слою вручную присваивается цвет из фиксированного набора различных цветов RGB, которые делают разницу между слоями наиболее заметной.

Рисунок 2.1 Шаг №0. Исходное изображение bird-eye-view перекрестка

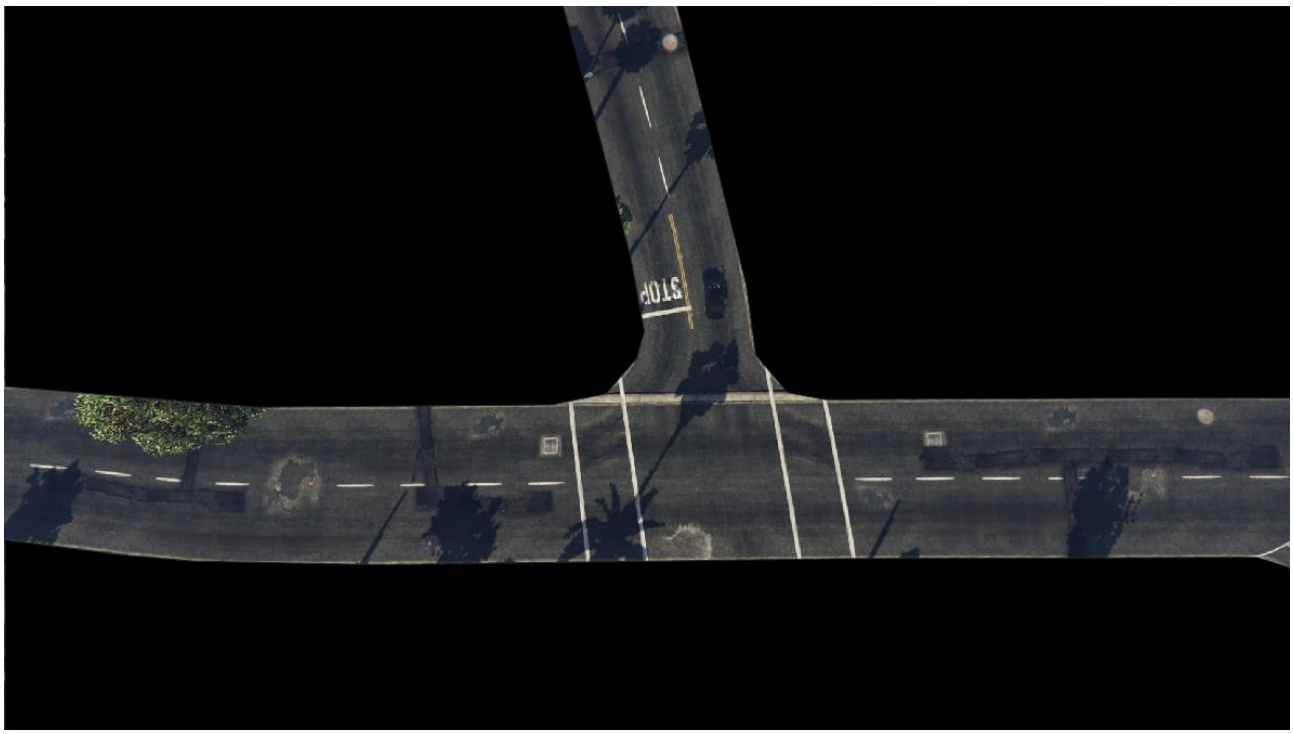


Рисунок 2.2 Шаг №1. Нанесение слоя, отсекающего лишнюю часть изображения

Рисунок 2.3 Шаг №2. Нанесение слоя дорожного полотна

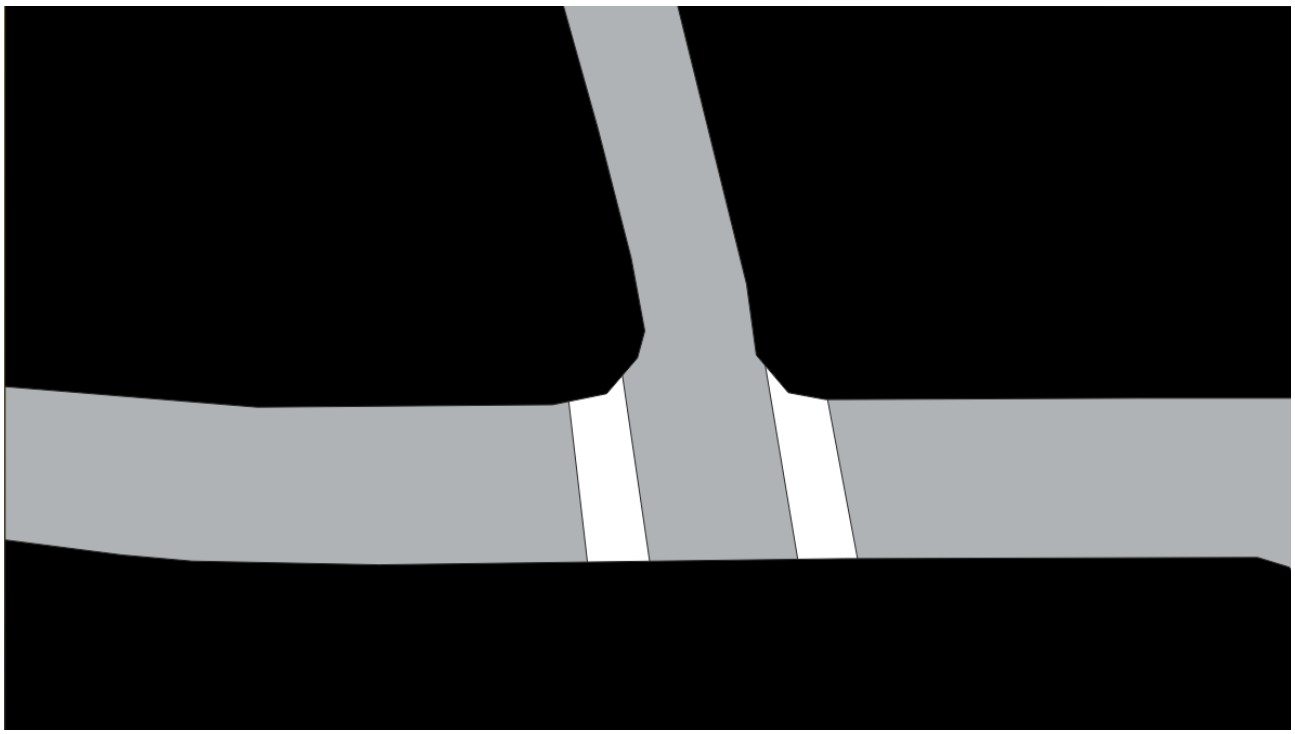


Рисунок 2.4 Шаг №3. Нанесение слоя пешеходных переходов

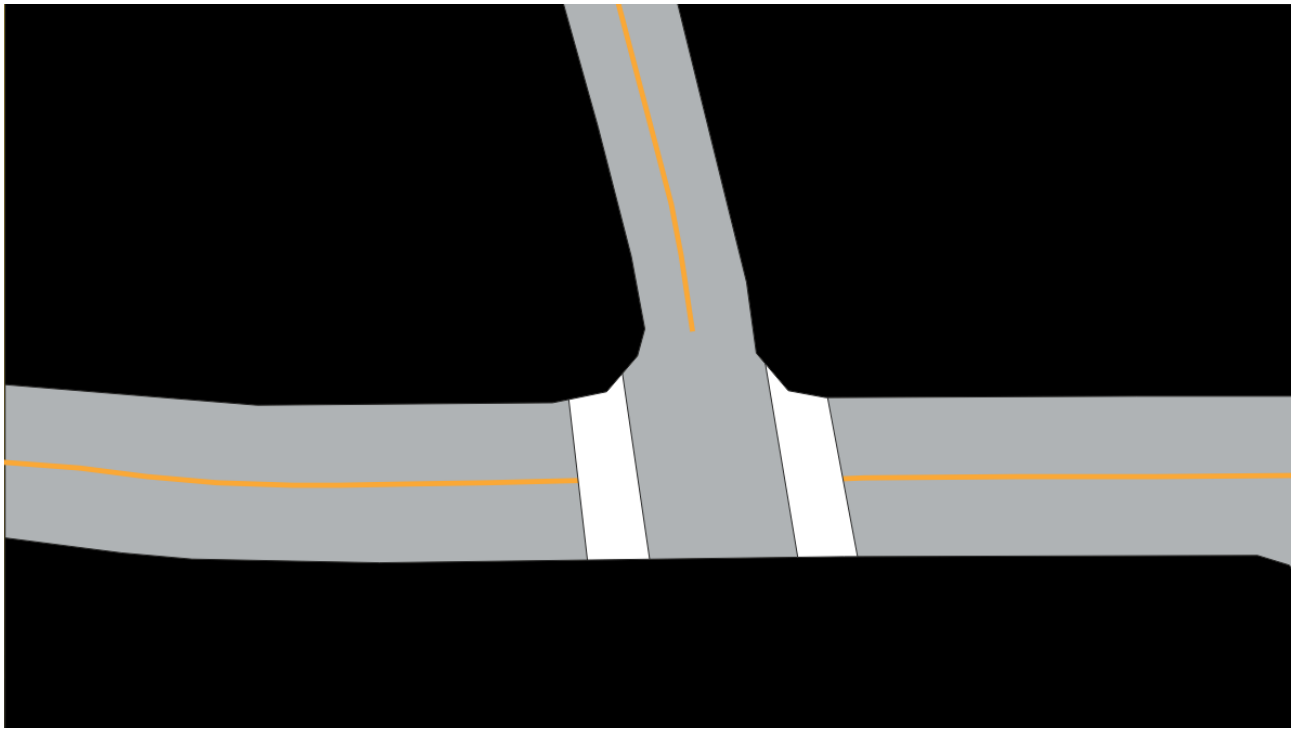


Рисунок 2.5. Шаг №4. Нанесение слоя дорожной разметки

Единственный слой, не имеющий определенного цвета RGB, — это слой, кодирующий направление полосы. Вместо присвоения отдельного цвета RGB перейдем в пространство HSV с максимальным значением насыщенности и значения цвета, а в качестве оттенка используем направление в градусах полосы в данной точке (см. рисунок 2.6, ось Hue).

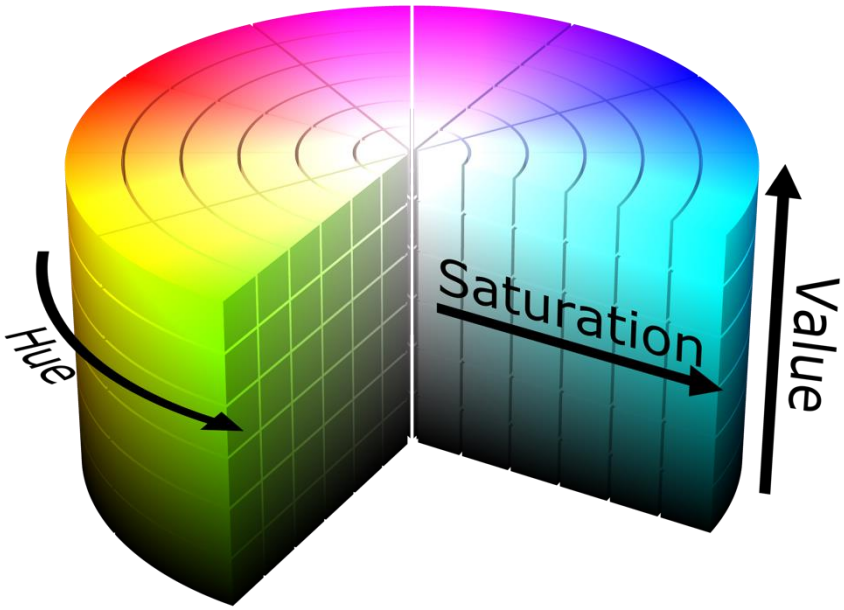


Рисунок 2.6 HSV цилиндр, цвет изменяется согласно углу Hue

Затем мы преобразуем HSV в цветовое пространство RGB, таким образом кодируя направление движения каждой полосы в результирующем растровом изображении. Таким образом, полосы, направленные в противоположных направлениях, будут представлены цветами диаметрально противоположными друг другу на HSV цилиндре.

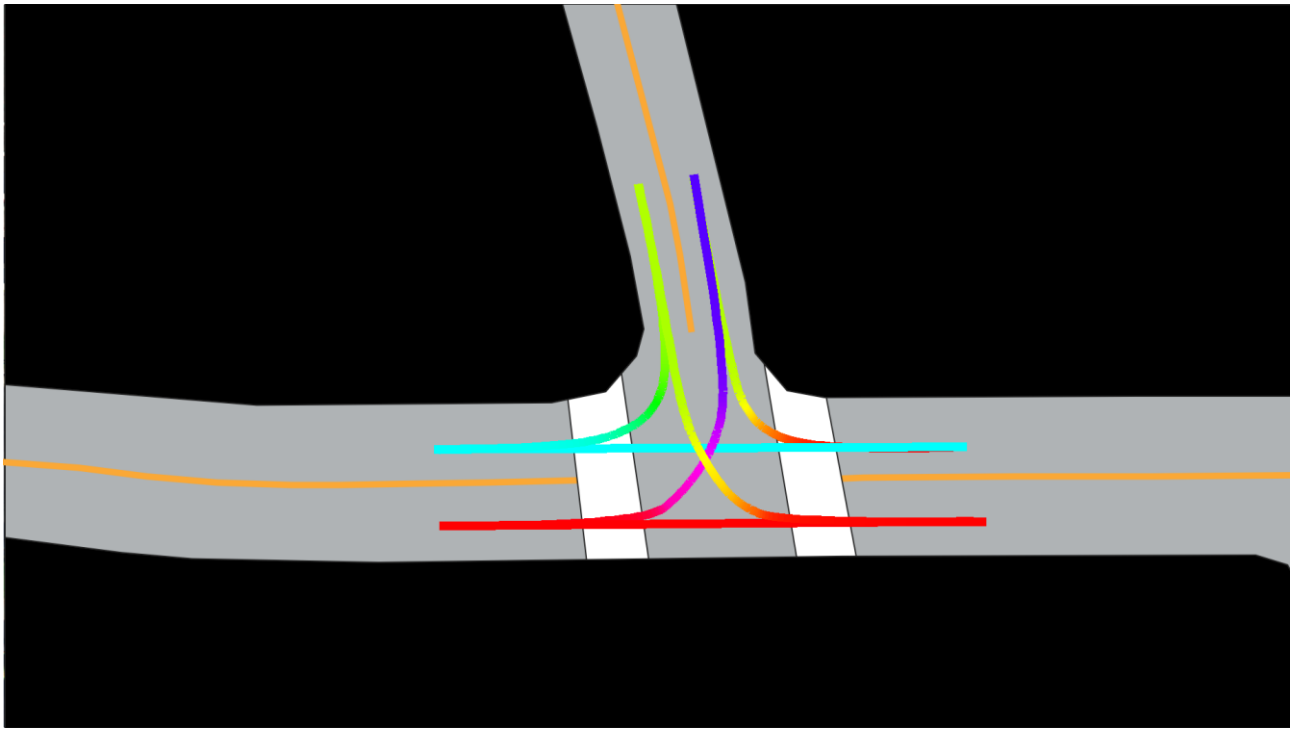


Рисунок 2.7 Шаг №5. Нанесение возможных проездов по перекрестку

Как только цвета определены, векторные слои растеризируются один за другим поверх другого в порядке от слоев, представляющих большие площади,

такие как полигоны дорог, к слоям, представляющим более тонкие структуры, такие как полосы или bounding box агентов. Важным параметром является разрешение одного пикселя, которое мы устанавливаем равным 0,1 м, учитывая компромисс между размером изображения и способностью представлять мелкие детали.

Как обсуждалось ранее, мы заинтересованы в представлении контекста для каждого агента в отдельности. Для представления контекста вокруг i-го агента, отслеживаемого в момент времени t_j, мы создаем растеризованное изображение I_{ij}, размером $n \times n$ таким образом, чтобы агент располагался в пикселе (w, h) в пределах I_{ij}, где w представляет ширину, h - высоту, отмеренные от нижнего левого угла изображения. Изображение поворачивается так, чтобы направление агента указывало верх. Направления полосы вычисляются относительно курса агента, а затем кодируются в пространстве HSV. Мы устанавливаем $n = 400$, интересующий нас агент расположен в $w = 150$ и $h = 50$, так что 25 м перед агентом и 5 м за растеризованы. Наконец, мы окрашиваем интересного нам в данный момент агента отлично от других транспортных средств (как показано на рисунке 2.8, он окрашен в красный цвет, а другие агенты - в желтый).

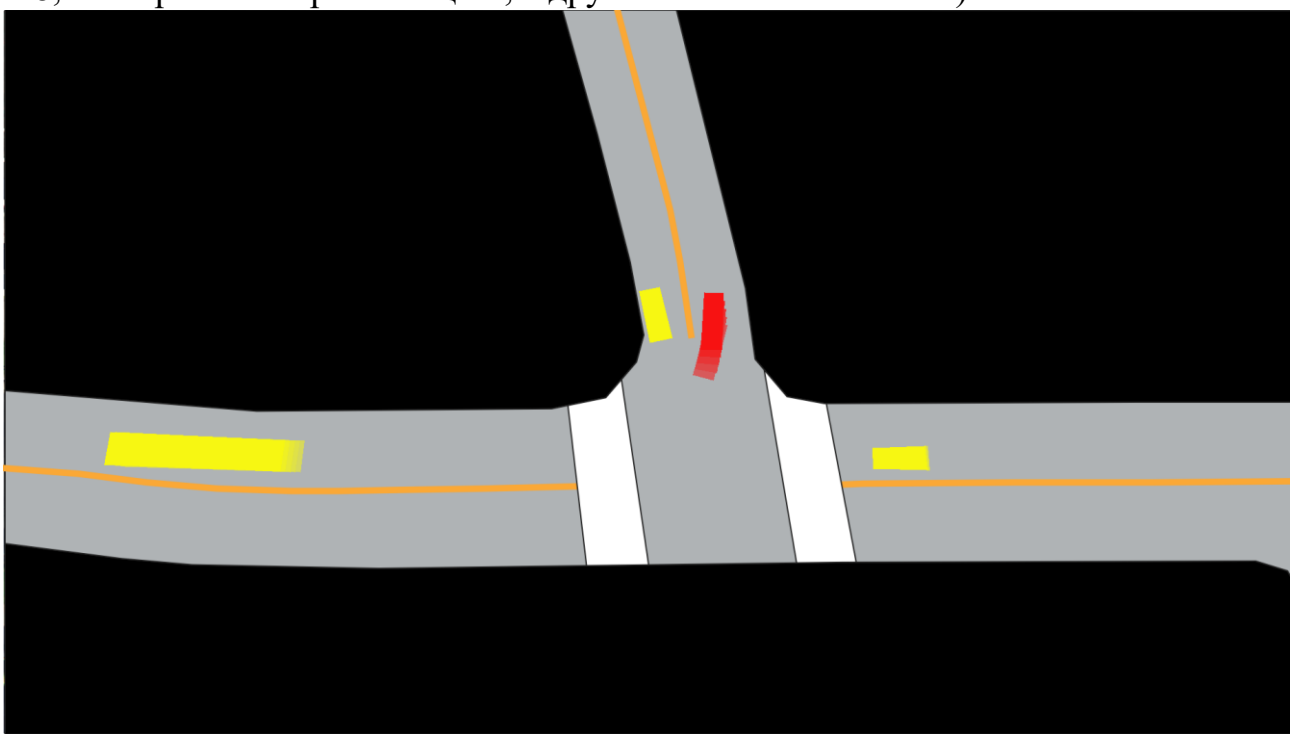


Рисунок 2.8 Шаг №6. Нанесение главного (интересного) агента красным цветом, остальных – желтым. Затухание отрисовывается по предыдущим 5 кадрам.

Для того, чтобы отразить историю состояний агентов, их bounding box в последовательные моменты времени $[t_{j-k+1}, \dots, t_j]$ растеризируются поверх всех векторных слоев. Каждый полигон истории агента растеризируется либо желтым, либо красным цветом, но с собственным уровнем яркости, отражающимся

в эффекте затухания. Яркость в момент времени t_{j-k} равна $max(0, 1 - k\delta)$, где $k = 0,1, \dots, K-1$; $\delta = 0.1$; $K = 5$.

2.5 Вывод функции потерь

Для вычисления аналитического выражения для функции потерь нейронной сети сперва введем ошибку отклонения для i агента в момент времени t_j и горизонта предсказания $h \in \{1, \dots, H\}$:

$$d_{i(j+h)} = \left(\left(x_{i(j+h)} - \hat{x}_{i(j+h)}(S_j, M, \theta)\right)^2 + \left(y_{i(j+h)} - \hat{y}_{i(j+h)}(S_j, M, \theta)\right)^2\right)^{\frac{1}{2}} \quad (2.1)$$

определенное как евклидово расстояние между предсказанной и реальной позициями. Здесь, θ обозначает параметры модели, а $\hat{x}_{i(j+h)}(S_j, M, \theta)$ и $\hat{y}_{i(j+h)}(S_j, M, \theta)$ означают предсказания, полученные из модели после подачи на вход состояния S_j и карты M. Тогда полная функция потерь, вычисляемая на основе предсказанных траекторий для полного горизонта предсказания есть среднеквадратичное отклонение точек траектории:

$$L_{ij} = \frac{1}{H}\sum_{h=1}^{H} d_{i(j+h)}^2 \quad (2.2)$$

Оптимизируя по всем агентам и моментам времени, мы имеем следующее выражение для оптимальных параметров модели, минимизируя полную функцию потерь на обучении:

$$\theta^* = argmin_\theta L = argmin_\theta \sum_{j=1}^{T}\sum_{i=1}^{N_j} L_{ij} \quad (2.3)$$

Однако, следует учесть, что задача предсказания довольно шумная по своей природе и скорее неточная в смысле существования распределения траекторий, а не конкретно одной, поэтому полезно учесть неопределенность задачи и оптимизировать ее в том числе. Допустим, что ошибка отклонения распределена по полунормальному закону:

$$d_{i(j+h)} \sim N(0, \hat{\sigma}_{i(j+h)}(S_j, M, \theta)^2) \quad (2.4)$$

где дисперсия рассчитывается самой моделью. Тогда перепишем нашу полную функцию потерь как отрицательный логарифм правдоподобия:

$$L_{ij} = \sum_{h=1}^{H}\left(\frac{d_{i(j+h)}^2}{2\hat{\sigma}_{i(j+h)}(S_j, M, \theta)^2} + \log\left(\hat{\sigma}_{i(j+h)}(S_j, M, \theta)\right)\right) \quad (2.5)$$

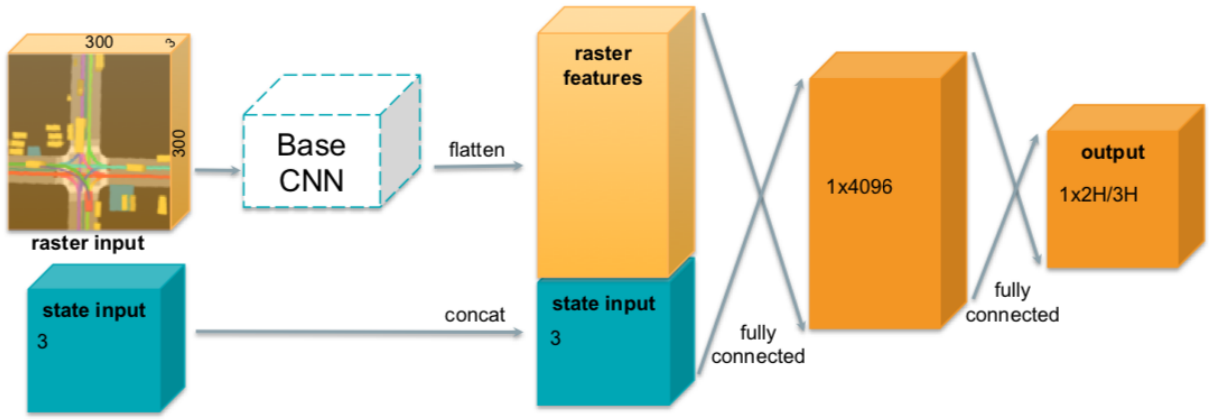


Рисунок 2.9. Архитектура оптимизируемой сети

Теперь выход нашей модели есть вектор размерности $3H$, представляющий координаты предсказанных местоположений вместе со стандартным отклонением для H точек траектории.

3.5.1 Учет мультимодальности распределения траекторий

Несмотря на то, что оптимизируемый функционал, описанный выше, является теоретически корректным и даже дает положительные результаты на практике, у него есть один существенный недостаток – он не учитывает потенциальную мультимодальность распределения будущих траекторий, что является критичным для точного долговременного прогнозирования.

В области задач машинного обучения проблема мультимодальности не нова. Например, сеть смешанных плотностей (Mixture Density Network, MDN) является общепринятым универсальным подходом для решения мультимодальных задач регрессии, выучивая параметры смеси нормальных распределений. Однако, на практике сложно хорошо обучить такую сеть из-за численной неустойчивости, вызванной большой размерностью пространства. Дабы обойти эту проблему, вместо нее обучаете ансамбль сетей или одна сеть, но выдающая M различных выходов для M различных гипотез, используя функцию потерь, которая вычисляется по наиболее «близкому» к истинной разметке выходу. Так как эмпирически доказано, что последний подход дает хорошие результаты относительно затраченных ресурсов, то именно он и будет использован для улучшения нашей функции потерь.

Перепишем унимодальную функцию потерь из пункта 2.5 как функцию потерь для моды m из M-модального распределения:

$$L_{ij}^{m} = \sum_{h=1}^{H} \left(\frac{d_{im(j+h)}^{2}}{2\hat{\sigma}_{im(j+h)}(S_j, M, \theta)^2} + \log\left(\hat{\sigma}_{im(j+h)}(S_j, M, \theta)\right) \right) \quad (2.6)$$

Объединить такие функции потерь в одну мультимодальную можно достаточно прямолинейно методов смеси экспертов (Mixture-of-Experts, ME):

$$L_{ij}^{ME} = \sum_{m=1}^{M} p_{im} L_{ij}^{m} \qquad (2.7)$$

что соответствует математическому ожиданию унимодальной функции потерь.

Однако как показывает практика, данный способ уязвим к схлопыванию предсказываемых мод, поэтому не является пригодным для нашей задачи. Дабы побороть это, явно смоделируем нужную нам функцию потерь (Multiple Trajectory Prediction (MTP) Loss).

Для i-агента в момент времени t_j вычислим выходы нейронной сети τ_{imj} (траекторию агента согласна моде m) прямым проходом, получив тем самым M траекторий. Далее определим моду m^*, которая наиболее близка к истинной траектории согласно некоторой произвольной функции расстояния $dist(\tau_{ij}, \hat{\tau}_{imj})$:

$$m^* = argmin_{m \in \{1,\dots,M\}} dist(\tau_{ij}, \hat{\tau}_{imj}) \qquad (2.8)$$

Теперь финальная функция потерь может быть определена как:

$$L_{ij}^{MTP} = -\sum_{m=1}^{M} I_{m=m^*} \log(p_{im}) + \alpha \sum_{m=1}^{M} I_{m=m^*} L_{ij}^{m}(\tau_{ij}, \hat{\tau}_{imj}) \qquad (2.9)$$

где I это предикат, равный 1, если условие истинно, и 0 иначе, а α – гиперпараметр для регулирования баланса между двумя компонентами функции потерь. Первая компонента заставляет модель сделать вероятность моды m^* как можно ближе к 1, а всех остальных – к 0. Вторая компонента непосредственно штрафует моду m^* за отклонение от истинной траектории. Таким образом во время обучения выходы, отвечающие за положение агента, обновляются только для одной моды, а вероятности мод – для всех мод, что позволяет каждой моде специализироваться под один конкретный класс поведения агента (движение прямо или поворот).

Для практического применения выведенной функции потерь не хватает только выбора функции $dist$. В ходе экспериментов с выборов среднего отклонения в качестве такой функции было обнаружено, что моды часто смешиваются и неправильно классифицируются перед проездом перекрестков – например, если предсказать медленное движение прямо, то среднее отклонение от траектории, поворачивающей направо, будем меньше, чем от траектории проезда прямо на высокой скорости (истинной траектории), что приведет к неправильной классификации моды. Дабы побороть эту проблему будет использоваться мера угла

между векторами, соединяющими текущее положение агента и последние точки траекторий.

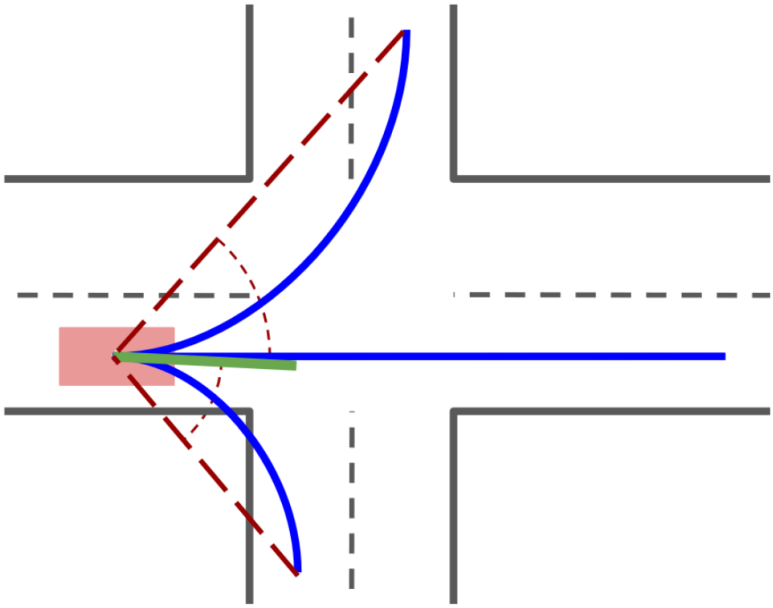


Рисунок 2.10. Метод классификации мод (выделены синим): истинная траектории (зеленая) классифицируется как поворот направо при использовании среднего отклонения, но как езда прямо при использовании угла.

2.6 Датасет

2.6.1 Проблематика

Для решения нашей задачи необходим большой и робастный датасет записи проездов SDV с видом от третьего лица (желательно иметь ортогональную проекцию bird-eye view или top-down view), состоящий из нескольких сотен часов и собранный на локациях с хорошей и простой дорожной разметкой.

Данные такого рода, к сожалению, не существуют в публичном доступе, поэтому перед нами встает важная подзадача – создание и разметка нужных нам данных. Исходя из того, что данных необходимо действительно много и использовать ручную разметку нецелесообразно, мы решили обучить нейронную сеть для автоматического решения задачи обнаружения объектов и постпроцессинга, которых высчитывал бы данные в их конечном формате.

2.6.2. Анализ источника сырых данных

Специфика задачи заключается в том, что не существует способа получить реальные данные проездов кроме как действительно, используя автомобиль и лидар, их получить. Однако данный способ является очень время- и финансозатратным.

В своем подходе мы использовали компьютерную игру GTA V в качестве симулятора и источника таких данных. В самом деле, GTA V является игрой с открытым и независимым от игрока миром, который существует и развивается

сам по себе, подчиняясь законам физики и правилам дорожного движения. К слову, в игре существует городской траффик с хорошими алгоритмами проезда перекрестков с учетом светофоров, пешеходов и разметки, что делает ее отличным инструментом для сбора данных в тематике SDV. Действительно, существуют даже решения, например DeepGTA, которые упрощают передачу команд из программы напрямую в автомобиль для его управления, используемые не в одной научной статье, однако нам это не нужно, нам достаточно только записи происходящего в мире.

2.6.3 Подготовка среды

Во-первых, по умолчанию игра не позволяет наблюдать за миром с высоты птичьего полета ортогонально поверхности. Однако, сама игра написана с использованием игрового движка RAGE, который поддерживает пользовательские расширения в виде динамически загружаемых библиотек, написанных на C++. Нами была написана DLL, позволяющую переходить от вида по-умолчанию к виду «сверху-вниз», а также регулировать масштаб.

Во-вторых, мы настраиваем интенсивность трафика, делая его реалистичным с помощью собственной модификации, вдохновленной модом Real-Traffic. Например, утром и вечером трафик будет иметь повышенную интенсивность, а остальное время пониженную, что позволяет создать довольно разнородный и многообразный датасет.

В-третьих, дабы повысить качество данных и упростить процесс сходимости обучения, искусственно увеличим длину дня до нескольких реальных астрономических часов, а ночь сократить до нескольких реальных астрономических минут. Также установим постоянные хорошие метеоусловия, дабы избежать тумана и облачности, затрудняющих обзор.

2.6.4 Сбор данных

Будем автоматически записывать отдельные клипы с частотой кадров 30fps с помощью свободного программного обеспечения Open Broadcast Software. Одним

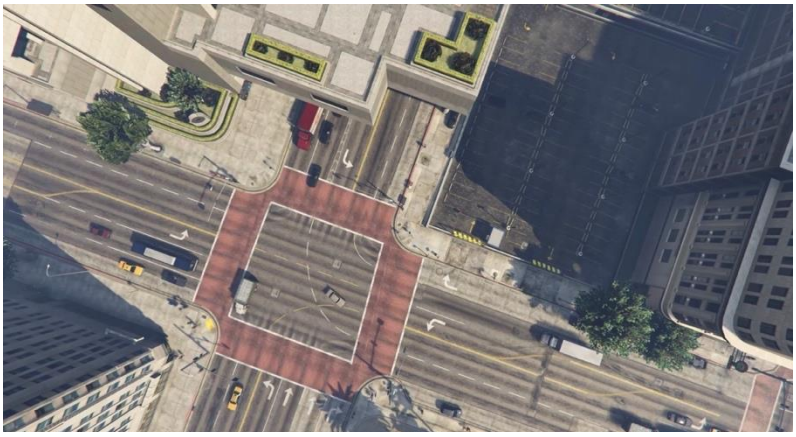


Рисунок 3.10 Пример получаемого кадра

из приятных бонусов является поддерживаемый рендер в mp4 формате и отсутствия пользовательского интерфейса на получаемых записях, что было бы неизбежно в случае записи картинки самого экрана.

На основе значений яркости пикселей картинки строим простой бинарный классификатор «день/ночь», чтобы вручную не перебирать и фильтровать кадры с ночным временем суток.

В заключении, дискретизируем видеоряд в набор картинок разрешения 1920*1080 с частотой 10Hz.

2.6.5 Разметка данных

В действительности, возможностей динамически загружаемых библиотек и нативных функций внутриигрового движка достаточно, чтобы напрямую выгрузить точные координаты всех наблюдаемых объектов. Естественно, идеальное знание входных данных положительно скажется на итоговом результате, и может сложиться ложное ощущение, что именно таким способом и нужно воспользоваться. Однако, такое решение приведет к сильному отрыву данных от наблюдаемых в реальной жизни, ведь в естественных условиях можно наблюдать зашумление датасета: ошибки трэкера, пропадания детекций, колебания оценок местоположения агентов, слепые зоны и т.д.

Дабы отразить перечисленные выше особенности данных и приблизить их к реальным, выгруженные точные координаты будут использоваться только для разметки объектов на изображениях, используемых для обучения сети, задача которой состоит в обнаружении агентов на изображении и выдаче их координат. Такая сеть даст хорошую точность координат вместе с некоторым природным шумом и аномалиями, свойственными реальным данным сенсоров в реальной среде.

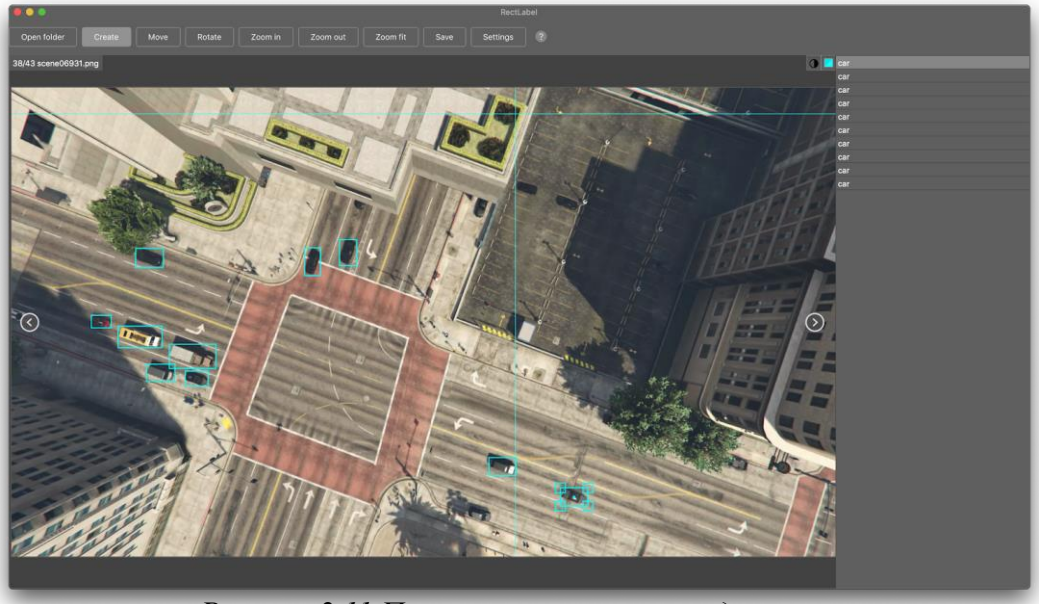


Рисунок 2.11 Пример размеченного кадра

Размечаем данные с помощью bounding boxes со сторонами параллельными сторонам картинки, используя приложение RectLabel. В действительности, нам нужны bounding boxes с некоторым вращением, однако обучение по таким данным сильно сложнее, поэтому в качестве базовой реализации решено было использовать данный вариант, а в будущем

Рисунок 2.12 Результаты обучения в виде предсказанных bounding boxes

воспользоваться масками автомобилей для определения курса. Разметка производится и использованием техники Extreme Points, что позволяет существенно ускорить процесс.

2.6.6 Обнаружение объектов

Обучим модель на наших данных, произведя fine-tuning модели Faster ResNet-101, предобученную на SDV датасете KITTY с видом на машины от первого лица. Реализовывать обучение будем, используя фреймворк Tensorflow, оптимизатор Adam и TFRecord формат входных данных.

В результате обучения имеем готовую систему автоматической детекции автомобилей на видео с метриками:

- $AR@10 - 0.839$
- $AR@100 - 0.913$
- $mAP@10 - 0.88$
- $mAP@0.75{:}\,IoU - 0.98$

2.7 Результаты обучения

Суммарно был собран датасет, состоящий из 4 различных перекрестков американского типа (2 X-образных и 2 T-образных), каждый из которых непрерывно

записывался и логировался на протяжении двух игровых дней (около 40 реальных минут).

Препроцессинг данных и растеризация самих кадров осуществлялись на языке программирования Python 3.5 с использованием библиотеки Pillow и собственной реализации B-сплайнов для отрисовки кривых. В виду наличия большого количества данных вычисления были распараллелены между 24 процессами (ядрами) с помощью библиотеки joblib, что позволило значительно уменьшить время подготовки датасета. В ходе предобработки изображения уменьшались с размера 1920x1080 до 1280x720 с сохранением соотношения сторон для ускорения вычислений без потери в качестве.

Статическая картографическая подложка (слои дорожного полотна, пешеходных переходов, разметки) были отрисованы единожды вручную для каждого перекрестка в специализированной программе для работы с геоданными QGis 3.6.

Все использованные архитектуры нейронных сетей были реализованы с использованием open-source фреймворка PyTorch, разрабатываемого и поддерживаемого компанией Facebook, версии 1.1.

Непосредственное обучение производилось на GPU Nvidia GeForce 1080 Ti с использованием CUDA версии 9.0 батчами по 64 изображения на протяжении 24 часов с использованием ручного линейного понижения learning rate с 10^{-3} до 10^{-4} в 3 этапа.

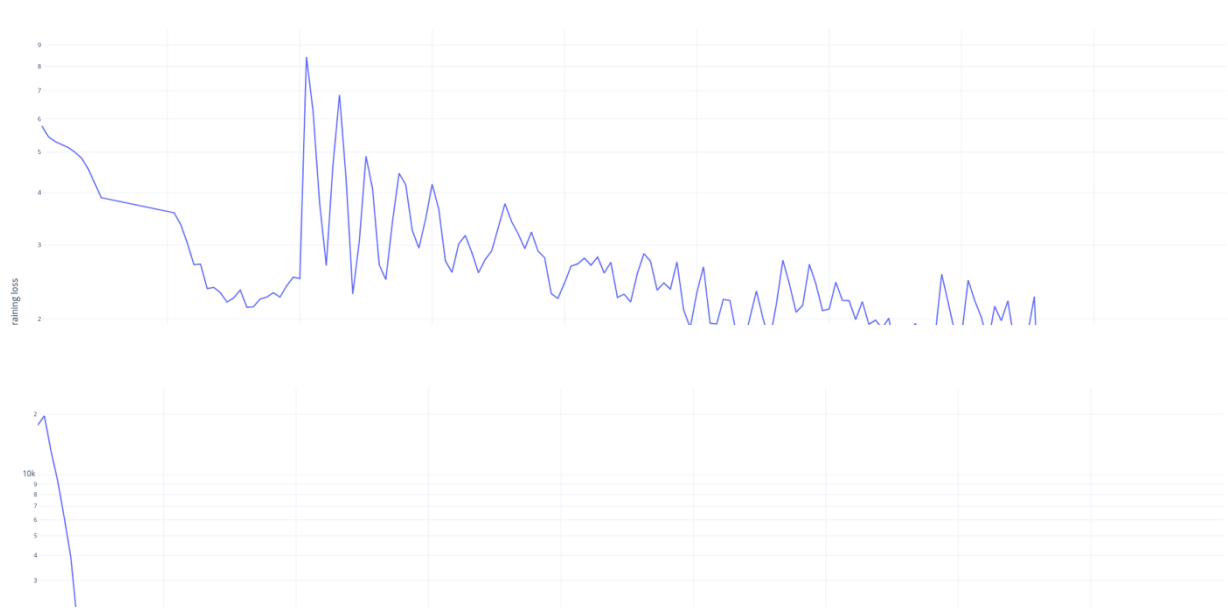


Рисунок 2.13 Функция потерь в обучающей выборке в зависимости от эпохи обучения

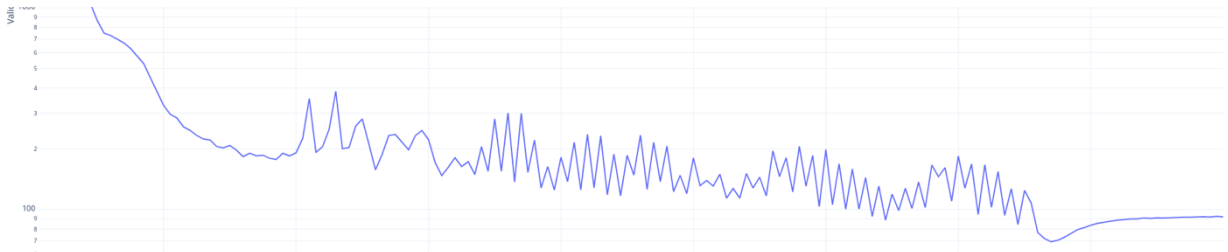


Рисунок 2.14 Функция потерь в валидационной выборке в зависимости от эпохи обучения

2.7.1 Влияние размера рецептивного поля

В силу того, что предсказание поведения на дороге часто требует знания контекста на большом удалении от агента, размер рецептивного поля сверточной нейронной сети может быть важным для финального качества предсказания.

В таблице 3.1 приведено сравнение метрик качества ($DE@t$ – отклонение через t секунд и ADE – среднее отклонение) для различных размеров фильтров. Сравнивая размеры фильтров 3, 5 и 7 на разрешении 400×400, можно заметить, что более крупные фильтры дает небольшой прирост качества. Однако это также ведет к квадратичному росту вычислительной сложности, поэтому важно соблюдать баланс между этими двумя параметрами.

Разрешение	Фильтр	$DE@1$с	$DE@2$с	$DE@3$с	ADE
400x400	3x3	0.55	0.82	1.19	0.73
400x400	5x5	0.54	**0.81**	1.17	**0.72**
400x400	7x7	**0.53**	**0.81**	**1.16**	**0.72**

Таблица 2.1 Влияние размера рецептивного поля на метрики отклонения и среднего отклонения

Далее приведены результаты эксперимента с варьируемым разрешением сцены, проведенный с целью выяснить как это влияет на качество по метрикам и вычислительную сложность. Были протестированы три различных разрешения: 400×400 (0.25 метра на пиксель), 200×200 (0.5 метра на пиксель) и 100×100 (1 метра на пиксель). По таблице 3.2 можно видеть, что в целом качество предсказания растет при увеличении разрешения сцены.

Разрешение	Фильтр	$DE@1$с	$DE@2$с	$DE@3$с	ADE
100x100	3x3	0.63	0.94	1.32	0.82
200x200	3x3	0.57	0.86	1.21	0.75
400x400	3x3	**0.55**	**0.82**	**1.19**	**0.73**

Таблица 2.2 Влияние разрешения сцены на метрики отклонения и среднего отклонения

Модель/ Метод	*Растеризация*	*Вектор состояния*	*Отклонение*	*Вдоль полосы*	*Поперек полосы*
Фильтр Калмана	-	нет	1.46	1.21	0.57
AlexNet	без затухания	нет	3.14	3.11	0.35
AlexNet	с затуханием	нет	1.24	1.23	0.22
AlexNet	без затухания	да	0.97	0.94	0.21
AlexNet	с затуханием	да	0.86	0.83	0.20
VGG19	с затуханием	да	0.77	0.75	0.19
ResNet-50	с затуханием	да	0.76	0.74	0.18
MobileNet-v2	с затуханием	да	0.73	0.70	0.18
MobileNet-v2	с затуханием	да	**0.71**	**0.68**	**0.18**

Таблица 2.3 Средние отклонения предсказанных траекторий от референсных по итогу обучения

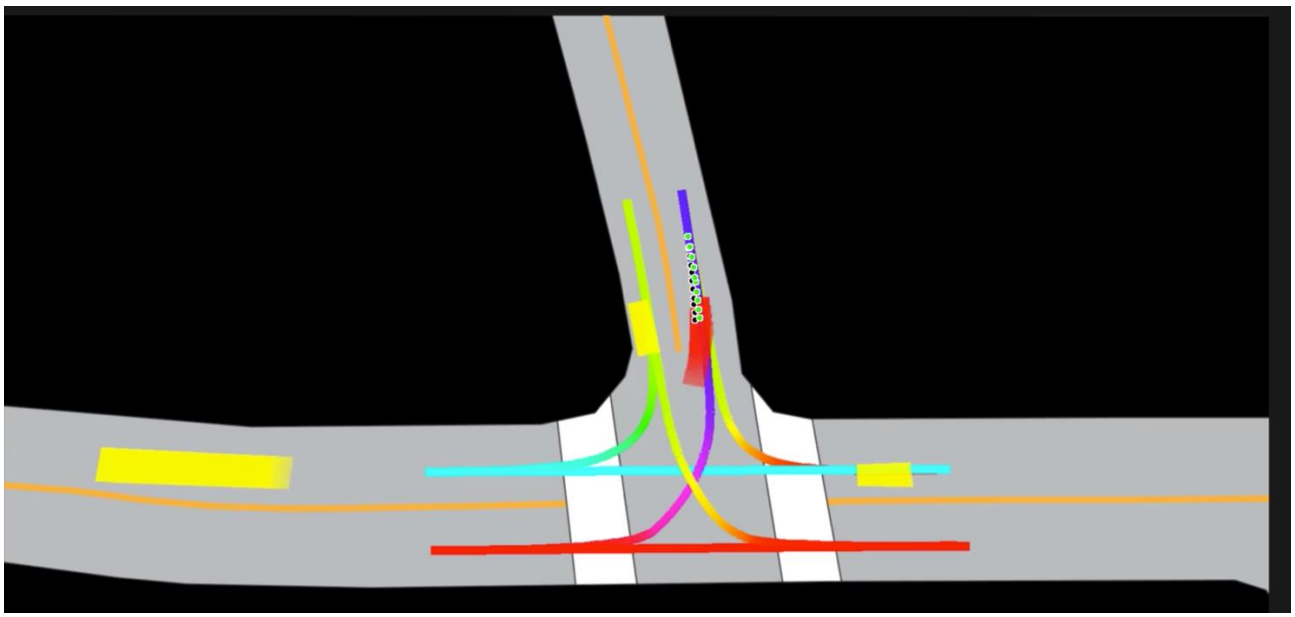


Рисунок 2.12 Результат работы модели для красной машины. Черные точки – предсказанная траектория, зеленые точки - референсная

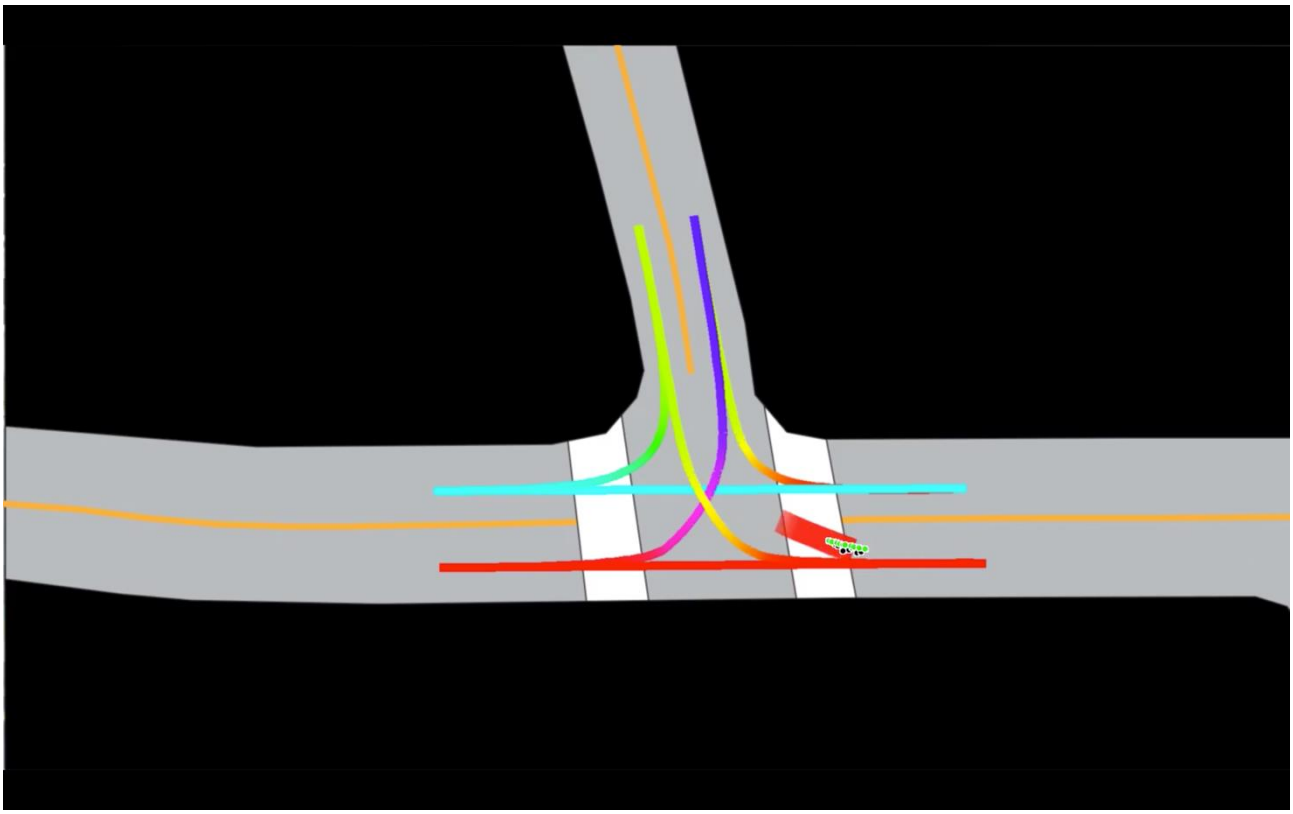


Рисунок 2.15 Результат работы модели для красной машины. Черные точки – предсказанная траектория, зеленые точки - референсная

ГЛАВА 3

ВЕКТОРНЫЙ ПОДХОД

3.1 Введение

Одно из ключевых качеств хорошего водителя — это способность предвидеть и предсказывать, что могут сделать другие участники дорожного движения на дороге. Например, какова вероятность того, что другая машина перестроиться на нашу полосу движения или велосипедист перед нами свернет налево? Способность точно предугадывать вероятные намерения других участников дорожного движения позволяет водителю принимать наиболее безопасные решения.

Предсказать поведение других людей на дороге сложно: часто требуется целостное понимание сцены и ее контекста, включая ширину дорожных полос, правила четырехсторонних перекрестков, светофоры и знаки. В сочетании с информацией, поступающей от датчиков в режиме реального времени, специально разработанные карты обеспечивают этот важный семантический контекст для водителя; однако для прогнозов нам нужно нечто большее, чем данные датчиков и карты. Поведение других участников дорожного движения часто сложно и трудно уловить с помощью набора правил дорожного движения, основанных на карте, потому что модели вождения различаются в разных местах, и другие участники дорожного движения могут нарушать эти правила. Машинное обучение - распространенный инструмент, используемый для моделирования и снижения этой сложности, позволяя системе изучать новые типы поведения.

Традиционные методы прогнозирования поведения основаны на правилах, при этом несколько гипотез из разных мод распределения поведения генерируются на основе ограничений из дорожных карт. В последнее время было предложено множество подходов, которые предлагают преимущество в виде вероятностных интерпретаций различных поведенческих гипотез, но требуют построения специального репрезентативного представления для кодирования информации о карте и траектории.

Интересно, что хоть по своей сути карты сильно структурированы, организованы как объекты с указанием местоположения (например, полосы движения) и атрибутов (например, управляющие секции светофора), большинство из существующих подходов выбирают представление карт в виде атрибутов с цветовой кодировкой (см. главу 2), которое требует ручных спецификаций и настроек, а затем кодируют контекстную информацию сцены с помощью сверточных нейронных сетей, которые имеют ограниченные поля восприятия. В связи с этим возникает вопрос: можем ли мы научиться

осмысленному представлению контекста непосредственно из структурированных карт?

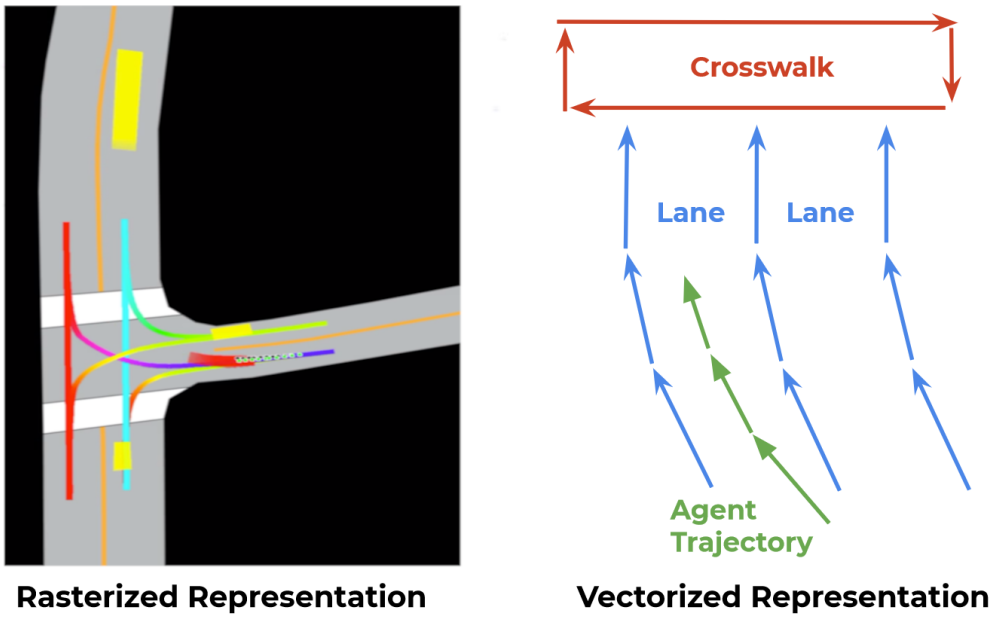


Рисунок 3.1 Иллюстрация подходов растеризированного рендера (слева) и векторного (справа) для представления карты и траекторий агентов

Далее предлагается исследовать унифицированное представление для мультиагентной динамики взаимодействия автомобилей и структурированного контекста сцены непосредственно из векторизованной формы (рисунок 3.17, справа). Гео-представление интересующих нас признаков дорожной обстановки может быть точкой, многоугольником или кривой в некоторых географических координатах. Например, граница полосы движения содержит несколько контрольных точек, образующих сплайн; пешеходный переход — это многоугольник, определяемый несколькими точками; знак остановки представлен одной точкой. Все эти географические объекты могут быть точно аппроксимированы как полилинии, определяемые несколькими контрольными точками вместе с их атрибутами. Точно так же динамика движущихся агентов также может быть аппроксимирована полилиниями на основе их траекторий движения. Все эти полилинии далее могут быть представлены в виде наборов векторов.

Затем можно использовать графовые нейронные сети (GNN, Graph Neural Network), чтобы агрегировать эти наборы векторов и извлечь полезную информацию. В графе каждый вектор рассматривается как вершина и определяются свойства вершины - начальная и конечная точка каждого вектора, наряду с некоторыми другими атрибутами, такими как идентификатор группы полилиний и различные семантические метки. Контекстная информация из карт вместе с траекториями других движущихся агентов проталкивается на целевую

вершину агента силами GNN. Затем мы можем использовать свойства целевой вершины, соответствующее целевому агенту, чтобы декодировать его будущие траектории.

3.2 Описание процесса векторизации

Большинство объектов карты представляют из себя либо сплайны (например, полосы), либо замкнутые формы (например, область перекрестка) либо точки (например, светофоры) с дополнительной атрибутивной информацией, такой как например семантика объекта или его текущее состояние (например, текущий сигнал светофора или скоростное ограничение полосы). Для агентов дорожного движения траектории являются направленными сплайнами в координатно-временном пространстве.

Все эти элементы могут быть аппроксимированы как последовательность векторов: для карточных объектов мы берем их стартовую точку и направление, равномерно сэмплируем ключевые точки сплайнов на равном расстоянии и последовательно соединяем точки в вектор; для траекторий, мы равномерно сэмплируем ключевые точки с фиксированным временным интервалом и соединяем их в вектор. С учетом выбора достаточно малого шага по расстоянию или времени, полученные в результате полилинии служат хорошей и точной аппроксимацией оригинальной карты и траекторий.

Процесс векторизации представляет собой биективное отображение между непрерывными траекториями, элементами карты и множеством векторов, хоть последний и неупорядочен. Это позволяет нам использовать графовое представление, построенное на основе множества векторов, которое далее может быть закодировано графовой нейронной сетью. Формально, мы определяем каждый вектор v_i, принадлежащий полилинии P_j, как вершину в графе с набором признаков:

$$v_i = [d_i^s, d_i^e, a_i, j], \qquad (3.1)$$

где d_i^s и d_i^e – координаты начала и конца точек вектора, в виде пары (x, y) координат; a_i отвечает за вектор атрибутивных признаков, как например тип объекта, время для траектории, ограничение скорости для полос; j есть целочисленный идентификатор P_j, означающий что $v_i \in P_j$.

Для того, чтобы входные признаки были инвариантны к положению предсказываемых агентов, дополнительно происходит нормализация координат всех векторов с центром координат в последнем наблюдаемом положении агента и соответствующим поворотом продольной оси сцены по направлению ориентации агента.

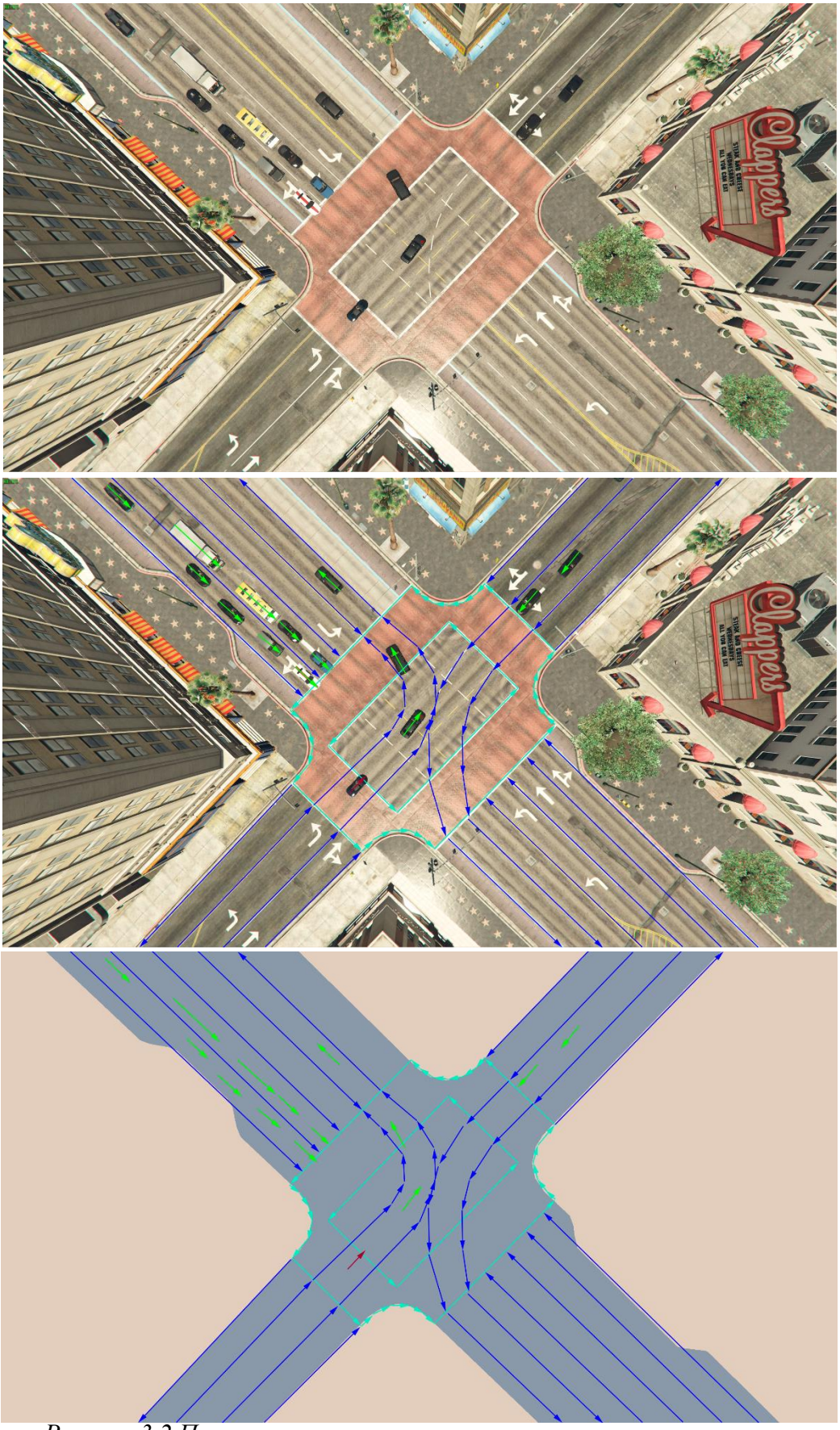


Рисунок 3.2 Поэтапная иллюстрация процесса векторизации сцены

3.3 Построение векторных подграфов для полилиний

Дабы захватить пространственную и семантическую связность между вершинами, воспользуемся иерархическим подходов для создания модели нейронной сети. На первом уровне (векторном) построим серию связных подграфов, в которой все векторные вершины, принадлежащие одной и той же полилинии, будут соединены ребром.

Рассмотрим полилинию P и ее образующие вершины $\{v_1, v_2, \dots, v_N\}$ и определим нейронный слой подграфа (оператор распространения графовой нейронной сети):

$$\mathrm{v}_i^{(l+1)} = \varphi_{rel}\left(g_{enc}\left(v_i^{(l)}\right), \varphi_{agg}\left(\left\{g_{enc}\left(v_j^{(l)}\right)\right\}\right)\right), \qquad (3.2)$$

где $v_i^{(l)}$ – признаки вершины v_i на l-ой итерации, соответственно $v_i^{(0)}$ – входные признаки. Функция $g_{enc}(\cdot)$ преобразует индивидуальные признаки вершины (кодирует в скрытое представление), $\varphi_{agg}(\cdot)$ агрегирует признаки от соседних вершин, а $\varphi_{rel}(\cdot)$ представляет собой оператор отношения между вершиной v_i и ее соседями.

На практике, $g_{enc}(\cdot)$ представляет из себя многослойный линейный перцептрон, чьи веса являются общими для всех вершин, а именно один полносвязный слой, за которым следует нормализация и активационная функцию ReLU. $\varphi_{agg}(\cdot)$ реализуется как операция пулинга максимумом, а $\varphi_{rel}(\cdot)$ как операция конкатенации.

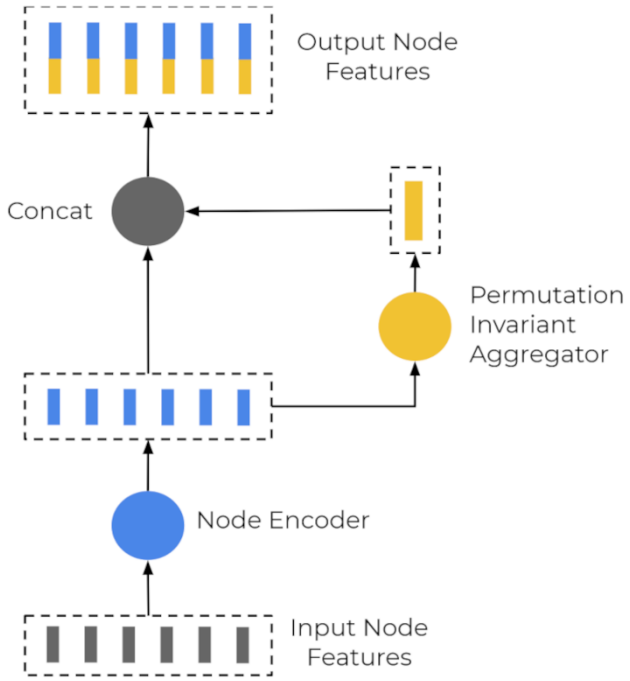


Рисунок 3.3 Иллюстрация подходов растеризированного рендера (слева) и векторного (справа) для представления карты и траекторий агентов

Далее, путем последовательно соединения множества таких слоев с разными между собой весами $g_{enc}(\cdot)$, получаем итоговый нейронный граф, на выходе которого получаем признаки сцены уже на втором иерархическом уровне, уровне полилиний, вычисляя:

$$\mathrm{p} = \varphi_{agg}\left(\{v_i^L\}\right), \qquad (3.3)$$

где L – количество последовательно соединенных слоев.

Хороший и внимательный специалист по трехмерному компьютерному зрению может заметить, что предложенный подход архитектурно очень похож на обобщение PointNet, нейронной сети для классификации трехмерных облаков точек. Действительно, если положить $\mathrm{d}^s = \mathrm{d}^e$, a пустым и избавиться от целочисленного идентификатора в формуле 3.6, то можно увидеть, что рассматриваемая архитектура имеет такой вход и поток вычислений, как и PointNet.

Однако, благодаря кодированию в векторах порядковой информации, ограничению связности подграфов на основе группировки в полилинии и кодирования признаков в графовых вершинах, рассмотренный метод куда лучше подходит для кодирования структурированных карточных объектов и траекторий, чем PointNet в своем первозданном виде.

3.4 Верхнеуровневый граф для взаимодействий высокого порядка

Теперь, находясь на втором иерархическом уровне, уровне полилиний и их множеством признаков $\{p_1, p_1, \dots, p_M\}$, можно приступить к моделированию взаимодействий между ними с помощью графа взаимодействий высокого порядка:

$$\left\{p_i^{(l+1)}\right\} = \mathrm{GNN}\left(\left\{p_i^{(l)}\right\}, A\right), \quad (3.4)$$

где $\left\{p_i^{(l)}\right\}$ – множество признаков полилиний, $\mathrm{GNN}(\cdot)$ обозначает один слой графовой нейронной сети, а A соответствует матрица смежности множества полилиний, которая может быть построена, например, по эвристике на основе пространственного расстояния между полилиниями. Для простоты далее положим, что A есть полносвязный граф, то есть все полилинии могут взаимодействовать между собой, то есть отдадим нейронной сети на откуп разобраться какие взаимодействия могут быть, а какие нет.

Графовый слой GNN на практике представляет из себя self-attention оператор:

$$\mathrm{GNN}(P) = \mathrm{softmax}\left(P_Q P_K^T\right) P_V, \qquad (3.5)$$

где P – матрица признаков вершин, а P_Q, P_K, P_V – ее линейные проекции.

Будущие траектории декодируется из вершин, отвечающих за движущихся агентов:

$$\mathrm{v}_i^{future} = \varphi_{traj}\left(p_i^{(L_t)}\right), \qquad (3.6)$$

где L_t – это количество слоев GNN, $\varphi_{traj}(\cdot)$ - декодер траектории.

На практике в качестве декодера используется обычный многослойный линейный перцептрон, однако для более разнообразных траекторий можно использовать декодеры посложнее, такие, как например MultiPath или вариационные RNN.

Также в реализации будет использоваться однослойная GNN, что позволит в режиме применения вычислять только те вершины, которые относятся к интересующим нас агентом, и соответственно сильно сэкономить ресурсы и добиться меньшей задержки между слепком данных с сенсоров и получением предсказанной траектории. Однако можно и использовать несколько слоев для моделирования еще более сложных взаимодействий, если ресурсы это позволяют.

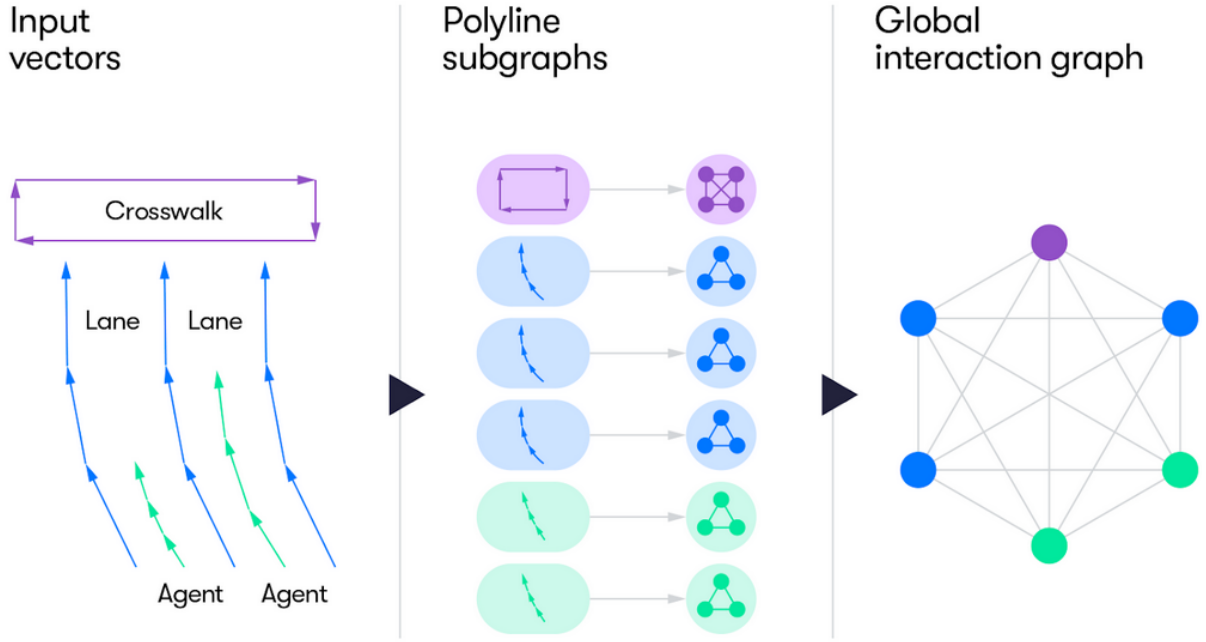


Рисунок 3.4 Иллюстрация пайплайна кодирования в векторном представлении и потока данных в итоговой нейронной графовой сети

3.5 Общий фреймворк

После построения архитектуры иерархической графовой сети, мы можем определить функцию потерь, которую мы будем оптимизировать, по аналогии с оной для растеризационного подхода (см. формулу 3.5) как отрицательное гауссово логарифмическое правдоподобие. Из данного определения функции

потерь можно догадаться, что результирующая траектория, как и в предыдущем подходе, рассматривается как элемент некоторого нормального распределения, представляемый в виде набора координат и их дисперсий.

Для того, чтобы уменьшить вероятность тривиальных решений, дополнительно понижаем магнитуду признаков вершин используя L2-нормализацию перед их подачей в GNN.

Предсказываемые траектории параметризуются координатами смещения от продольной оси сцены, на которой мы в последний раз наблюдали агента.

3.6 Результаты обучения

Сперва представим результаты эксперимента, суть которого была выяснить пользу от совместного использования признаков карты и траекторий агентов. Таблица 4.1 отражает использование в предсказании только прошлых траекторий предсказываемого агента («агент»), только карточных признаков («карта») и того и другого вместе («карта + агент»). Можно заметить, что добавление информации о карте значительно улучшает качество предсказания по сравнению с референсом. Совместный учет признаков обеих категорий также дополнительно улучшает результат.

Признаки	$DE@1c$	$DE@2c$	$DE@3c$	ADE
агент	0.77	0.99	1.29	0.92
карта	0.55	0.78	1.07	0.7
карта + агент	0.53	0.74	1.00	0.66

Таблица 3.1 Влияние признаков различных категорий на метрики предсказания

В следующем эксперименте было изучено влияние архитектуры графовой нейронной сети, а именно ее глубины и ширины, на качество предсказания. Из таблицы 4.2 видно, что наилучшее качество дает использование трех слоев в сети для полилиний и одного слоя в глобальном графе. Использование более широкого внутреннего представления сцены не приводит к улучшению метрик и даже немного их ухудшает, что может быть связано с недостаточно большим датасетом и проблемой переобучения.

В заключительном эксперименте (таблица 4.3) было произведено сравнение векторного подхода и растеризационного. Оба подхода имеют примерно паритетные метрики при использовании сети ResNet в качестве кодировщика, однако векторный подход имеет куда более экономичен в контексте потребляемых ресурсов, что означает меньшую задержку в предсказании, которая является критичным параметром в прикладном применении, т.к. определяет скорость реакция на изменяющуюся обстановку. Также стоит отметить, что растеризационный подход хорошо себя показывает в достаточно тривиальных случаях, например при большом количестве стационарных машин,

т.к. легко выучивать локальные паттерны, однако в натуральном распределении дорожных сцен присутствуют краевые сложные случаи, которые очень важны для рассматриваемой задачи, т.к. важно понимать качество и в худшем случае – в этом месте доминирует векторный подход с большим отрывом, предположительно за счет своей способности улавливать далекий контекст и дальние взаимодействия благодаря иерархической графовой сети.

Глубина	*Ширина*	*Глобальная глубина*	*Глобальная ширина*	$DE@3c$
1	64	1	64	1.09
3	64	1	64	1.00
3	128	1	64	1.00
3	64	2	64	**0.99**
3	64	2	256	1.02

Таблица 3.2 Влияние признаков различных категорий на метрики предсказания

Модель	*FLOPs*	*Кол-во параметров*	$DE@3c$
R18-k3-r100	0.66G	246K	1.32
R18-k3-r200	2.64G	246K	1.21
R18-k3-r400	10.56G	246K	1.16
R18-k5-r400	15.81G	509K	1.16
R18-k7-r400	23.67G	902K	1.16
Векторный подход	0.041G	72K	1.05

Таблица 3.3 Сравнение растеризационного и векторного подхода. R18-kM-rS означает ResNet-18 с размером фильтров $M \times M$ и разрешением сцены $S \times S$. FLOPs означает количество операций с плавающей точкой.

ЗАКЛЮЧЕНИЕ

В первой главе диссертации рассмотрены SOTA подходы к построению нейронных сетей и особенности их обучения в контексте долгосрочных предсказаний.

Во второй главе рассмотрены SOTA подходы к краткосрочным и долгосрочным предсказаниям движения агентов в индустрии и науке. Затем был предложен новый подход к предсказанию траекторий с использованием метода растеризации окружающей среды и сверточных нейронных сетей.

В ходе анализа была формализована решаемая задача, описан алгоритм и детали процедуры растеризации, выведена формула для полной функции потерь, учитывающей неопределенность (распределение) входных и выходных данных, представлена архитектура сверточной сети.

В процессе была сформулирована важная подзадача в виде создания системы автоматической разметки данных и их генерации с использованием GTA V в качестве симулятора, модифицированной с помощью как собственных, так и сторонних плагинов. Приведен анализ архитектуры SOTA подходов к задаче обнаружения объектов. Обучена сеть Faster ResNet 101, получены желаемые результаты с хорошей точностью.

Выведена формула функции потерь, учитывающая мультимодальность распределения траекторий, что является критичным для точного долговременного предсказания траекторий и позволяет закрепить за каждой предсказываемой модой свой некоторый маневр (езда прямо, поворот направо и т.д.).

Полученный синтетический набор данных был использован для обучения собственной реализации сети, успешно протестирован.

Во второй части рассмотрен двойственный к растеризационному подход – векторный. За счет использования особого кодирования и графовой нейронной сети данный метод позволяет получить более легковесную сцену по сравнению с предыдущим без потери качества, что положительно отразится на задержке в предсказании в реальных условиях. Дополнительно благодаря иерархической концепции нейронной сети модель лучше выучивает сложные шаблоны взаимодействия между агентами. Для обоих подходов проведены эксперименты по подбору и влиянию гиперпараметров и сравнительный анализ. Полученные результаты говорят о перспективности второго подхода и необходимости его развития, а также возможности его интеграции в существующие SDV

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Машинное обучение: курс лекций / К. В. Воронцов // Школа анализа данных [Электронный ресурс] – 2016-2017. – Режим доступа: https://wiki.school.yandex.ru/shad/MachineLearning – Дата доступа: 06.02.2018.
2. A. Alahi, “Social LSTM: Human Trajectory Prediction in Crowded Spaces.”, A. Alahi, K. Goel, et al. IEEE, Jun 2016. DOI: 10.1109/CVPR.2016.110
3. M. Bojarski, “End to end learning for self-driving cars”, M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., arXiv preprint arXiv:1604.07316, 2016.
4. S. Chen, “Kalman filter for robot vision: a survey,” IEEE Transactions on Industrial Electronics, vol. 59, no. 11, p. 4409–4420, 2012.
5. N. Djuric, “Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks”, N. Djuric,V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider, arXiv preprint arXiv:1808.05819, 2018.
6. K. Fragkiadaki, “Learning visual predictive models of physics for playing billiards” K. Fragkiadaki, P.Agrawal, International Conference on Learning Representations (ICLR), 2016.
7. C. Gong, “A methodology for automated trajectory prediction analysis”, C. Gong, D. McNally, Navigation, Control Conference and Exhibit, 2004.
8. I. Goodfellow “Deep Learning.” Goodfellow, Y. Bengio, and A. Courville, MIT Press, 2016.
9. B. Lakshminarayanan, “Simple and scalable predictive uncertainty estimation using deep ensembles,” B. Lakshminarayanan, A. Pritzel, and C. Blundell, in Advances in Neural Information Processing Systems, 2017.
10. P. Ondrúška, “Deep tracking: Seeing beyond seeing using recurrent neural networks,” P. Ondrúška and I. Posner, in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, 2016, p. 3361– 3367.
11. K. He, “Deep residual learning for image recognition”, X. Zhang, S. Ren, and J. Sun, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, p. 770–778.
12. J. Wiest, “Statistical long-term motion prediction.”, arXiv: 1705.04532, 2017.
13. M. D. Zeiler, “Visualizing and understanding convolutional networks”, M. D. Zeiler, R. Fergus, European conference on computer vision. Springer, 2014, p. 818–833.
14. NCHS, “Health, United States, 2016: With chartbook on long-term trends in health,” National Center for Health Statistics, Tech. Rep. 1232, May 2017.