

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

ОТЧЁТ
« ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОЦЕНКИ
ТРАФИКА МАГАЗИНА »

Ли Ицзя

3530904.90102

Оглавление

Реферат	3
Введение.....	4
ГЛАВА 1 Обзор литературы	5
Распознавание пешеходов.....	5
Алгоритм распознавания объектов YOLO	7
Отслеживание (Трекинг) людей в виде последовательности.....	8
Алгоритм трекинга множества объектов DeepSORT	9
Повторная идентификация людей.....	11
Алгоритм Fast reid	13
Метод подсчета количества людей в видеопоследовательности.....	15
Основные исследования этой статьи	15
ГЛАВА 2 Обучение модели для отслеживания людей в нескольких видеопотоках.....	16
Обучение модели для повторной идентификации fastreid	16
Подготовка набора данных	16
Принцип обучения	17
Построить модель.....	17
Функции потери и алгоритмы оптимизации	18
Этапы обучения.....	19
Тренировочный процесс	20
Результаты обучения.....	20
Реализация алгоритма DeepSORT с помощью fastreid и yolov5	23
ГЛАВА 3 Система оценки трафика магазина основа на нейронных сетей	28
Цели системы	28
Описание системы	28
Общая архитектура системы	28
Модуль обнаружения пешеходов	29
Модуль извлечения признаков.....	29
Модуль отслеживания пешеходами	29
Модуль подсчета посетителей	30
Модуль хранения.....	31
Среда разработки.....	31
Демонстрация и тестирование системы	32
Алгоритм статистики трафика	32
Тестовая среда	33

Междукамерное отслеживание пешеходов	33
Метрики оценки системы	39
ГЛАВА 4 Анализ полученных результатов	45
Экспериментальная оценка модели fastreid на открытой базе данных	45
Экспериментальная оценка улучшенной модели DeepSORT на открытой базе данных.....	46
ЗАКЛЮЧЕНИЕ.....	48
Список Литературы.....	49

Реферат

На - с., - рисунков, - таблицы, - приложение

KEY WORDS: Computer vision, Neural networks, Object detection, Multiple object tracking, Pedestrian re-identification

КЛЮЧЕВЫЕ СЛОВА: КОМПЬЮТЕРНОЕ ЗРЕНИЕ, НЕЙРОННЫЕ СЕТИ, ОБНАРУЖЕНИЕ ОБЪЕКТОВ, ОТСЛЕЖИВАНИЕ НЕСКОЛЬКИХ ОБЪЕКТОВ, ПЕРЕИДЕНТИФИКАЦИЯ ПЕШЕХОДОВ, YOLO, DeepSORT

Выпускная квалификационная работа на тему: «Применение нейронных сетей для оценки трафика магазина».

Данная статья посвящена реализации алгоритмов нейронных сетей для оценки трафика в магазине. Для решения поставленных задач были использованы методы глубокого обучения и компьютерного зрения.

Задачи, решаемые в ВКР:

1. Для обнаружения пешеходов был использован алгоритм yolov5
2. Для установления связи между персонажами при работе с несколькими камерами был реализован алгоритм повторного распознавания (reid).
3. Для отслеживания локальных зон был реализован алгоритм DeepSORT он основе повторного распознавания.
4. На основе методики, описанных в данной статье, была разработана система подсчета посетителей для супермаркета.

В результате проведенных исследований была разработана система подсчета посетителей для супермаркета, которая демонстрирует точность 81.4%.

Ведение

С развитием розничной торговли и усилением конкуренции многие отечественные торговые центры осознали важность данных о потоках клиентов для принятия бизнес-решений. Поток покупателей – это количество покупателей, входящих и выходящих из торгового центра в единицу времени. Благодаря углубленному анализу пассажиропотока он может обеспечить научную основу для управления всеми аспектами торговых центров. Анализ потока клиентов играет важную роль в повышении научности ежедневных бизнес-решений торговых центров, комфортности торговой среды и рациональности распределения человеческих ресурсов.

Самым ранним методом подсчета является метод искусственного подсчета. Торговый центр отправляет несколько сотрудников на вход и выход и непрерывно подсчитывает посетителей, входящих в торговый центр, визуальным способом в течение определенного периода времени. Преимущество этого метода в том, что он не требует вложений в оборудование для сбора информации, а данные обследования получаются более полными и гибкими, а недостаток в том, что он требует трудовых и материальных ресурсов, подвержен пропускам.

С появлением механического сенсорного оборудования, инфракрасного сенсорного оборудования и оборудования для камер наблюдения метод подсчета людей, в свою очередь, прошел две стадии: статистику физического прикосновения и статистику инфракрасного излучения. Физическая сенсорная статистика в основном используется на входах и выходах, где пешеходы проходят по очереди, но есть недостатки, такие как высокие требования к установке оборудования, неудобное использование, низкая скорость движения и невозможность использования в широком диапазоне; инфракрасная статистика имеет низкие требования к установке и относительно удобна в использовании, но их статистика погрешности велика и не может быть использована в сценариях большой площади, кроме того, эти методы имеют весьма очевидные недостатки: невозможно узнать время входа и выхода каждого покупателя из магазина. Данные, полученные этим методом, могут быть использованы только для качественного понимания и не имеют реальной ценности для статистического анализа. В сегодняшней жесткой экономической конкуренции эти статистические методы совершенно не отвечают потребностям рынка.

Традиционные ручные методы проверки неэффективны и дорогостоящи, поэтому для удовлетворения быстро меняющихся требований рынка необходим более эффективный, точный и автоматизированный метод. Исходя из этого, мы провели данное исследование, целью которого было реализовать алгоритм мониторинга и расчета пассажиропотока супермаркетов с использованием метода нейронных сетей.

Основная цель этой статьи — предложить метод, который сочетает в себе алгоритмы обнаружения пешеходов, отслеживания пешеходов и повторной идентификации пешеходов для получения статистики посещаемости магазина. Внедрив алгоритм yolov5, алгоритм отслеживания DeepSORT и алгоритм повторной идентификации пешеходов, мы выполнили точное отслеживание и подсчет покупателей, входящих и выходящих из магазина под разными углами и в разных условиях освещения, а также разработали систему статистики пассажиропотока для супермаркетов.

С точки зрения анализа осуществимости, мы считаем, что метод, предложенный в этой статье, имеет высокую осуществимость. Прежде всего, используемые алгоритмы являются относительно зрелыми и широко используемыми технологиями в современной области глубокого обучения и полностью проверены экспериментально. Во-вторых, с точки зрения аппаратной и программной среды конфигурация, требуемая в этой статье, относительно невелика и может работать на обычных ПК. Наконец, мы также проводим всестороннюю экспериментальную оценку точности, эффективности и стабильности предложенного метода, демонстрируя его осуществимость.

Для достижения цели этой статьи нам необходимо выполнить следующие задачи:

- 1) Соберите изображения наблюдения с камеры супермаркета и выполните предварительную обработку;
- 2) Разработать и реализовать модуль обнаружения пешеходов на основе алгоритма yolov5;
- 3) Разработать и внедрить алгоритм отслеживания DeepSORT на основе алгоритма повторной идентификации пешеходов, который применяется для отслеживания пешеходов в видеопотоке;
- 4) Разработайте и внедрите алгоритм повторной идентификации пешеходов для ассоциации персонажей с назначенными клиентами под несколькими камерами;
- 5) Хранить и записывать фотографии и время входа и выхода клиентов, вести статистику пассажиропотока.

На этой основе мы разрабатываем практическую систему статистики потоков клиентов, ориентированную на супермаркеты, которая может обеспечить эффективное управление и поддержку принятия решений для супермаркетов.

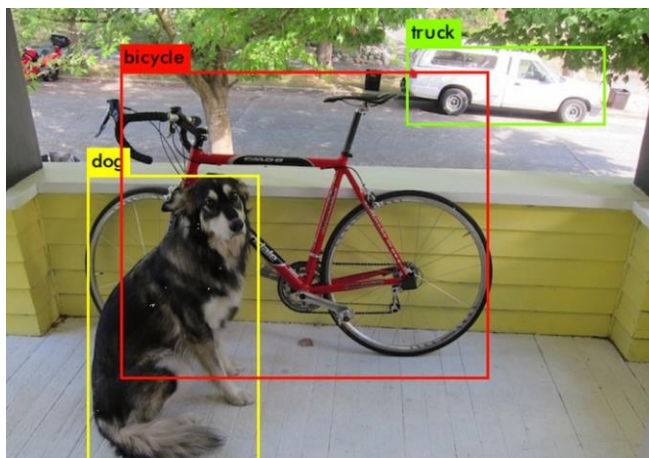
ГЛАВА 1 Обзор литературы

Подсчет людей — это комплексная задача, объединяющая технологии обнаружения, отслеживания и повторной идентификации пешеходов. Ввиду этого в данной статье сначала описывается исследования обнаружения и отслеживания людей, затем вводится алгоритм повторной идентификации пешеходов и, наконец, вводится исследования подсчета людей на видеопоследовательности. Повторная идентификация пешеходов является самой важной технологией для решения проблемы ассоциации траекторий между камерами, а также основой данной исследовательской работы. Поэтому мы сосредоточимся и подробно представим реализованный нами алгоритм повторной идентификации пешеходов: fastreid.

Распознавание пешеходов

Суть обнаружения пешеходов заключается в обнаружении объектов. Обнаружение объектов лежит в основе задач многоцелевого сопровождения, основная цель которого — отметить объект, подлежащий обнаружению, прямоугольной рамкой (bounding box) из статической картинке каждого кадра и получить конкретное положение цели на картинке.

Если при обнаружении объекта происходит пропущенное обнаружение, цель не будет отслеживаться позже, поэтому обнаружение объекта играет очень важную роль в задачах отслеживания нескольких целей.



Появление сверточных нейронных сетей значительно повысило эффективность методов обнаружения объектов. Такие алгоритмы можно условно разделить на следующие две категории. Одна из них представляет собой двухэтапный алгоритм обнаружения, который сначала выбирает области-кандидаты на изображении, а затем выполняет задачи классификации этих областей. Этот тип алгоритма характеризуется высокой точностью, но медленной скоростью. Например, R-CNN, предложенная Россом и др. [5], сначала предварительно определяет области, которые должны быть обнаружены на изображении, с помощью операции выборочного поиска, а затем извлекает признаки этих областей с помощью сверточной нейронной сети и, наконец, классифицирует их. Хотя алгоритм R-CNN значительно повышает точность обнаружения объектов, он имеет проблему вычислительной избыточности. Алгоритм sppnet, предложенный Хе и др. [6], устраняет влияние повторяющегося масштабирования изображений R-CNN путем добавления слоя объединения пространственных пирамид. Кроме того, Гиршик и др. [7] предложили алгоритм Fast R-CNN, который дополнительно оптимизировал алгоритмы R-CNN и sppnet, добавив слой объединения регионов. Рен и др. [8] предложили алгоритм Faster R-CNN, который заменил выборочный поиск сетью предложений регионов и обучил сеть от начала до конца, что значительно повысило скорость детектора. Другой представляет собой одноэтапный алгоритм обнаружения, который выполняет регрессию для вывода положения и категории объекта обнаружения при создании области-кандидата. Репрезентативным алгоритмом является алгоритм серии YOLO [9].

Серия алгоритмов YOLO была впервые предложена Редмоном и др. В 2016. Он отказался от двухэтапной парадигмы обнаружения «обнаружение предложения + проверка» и напрямую предсказал вероятность категории и положение каждой сетки, разделив изображение на несколько сеток, информация.

Из приведенного выше введения мы делаем вывод, что традиционные подходы к обнаружению объектов обычно требуют ручного проектирования функций и классификаторов, которые плохо работают в сложных сценах и требуют большого опыта и времени для настройки параметров. Напротив, технология обнаружения объектов на основе глубокого обучения использует глубокие нейронные сети для автоматического изучения функций и классификаторов, что обеспечивает более высокую точность и надежность и подходит для различных сложных сцен. Обнаружение объектов является предпосылкой отслеживания целей. Мы выбрали yolov5 в качестве детектора объектов для исследовательского проекта. Основная причина в том, что yolov5 отличается высокой

точностью, высокой эффективностью, простотой в использовании и широкими возможностями настройки.

Исходя из вышеперечисленных преимуществ, yolov5 больше подходит для наших сценариев использования.

(卷积神经网络的出现, 使得物体检测技术的性能获得了显著的提高。此类算法大致可分为以下两类, 一类是两阶段检测算法, 先在图像中选择候选区域, 然后再完成这些区域的分类任务, 该类算法的特点是精度高但速度慢。如 Ross 等人[5]所提出的 R-CNN, 首先以选择性搜索操作预先定位图片中的待检测区域, 然后通过卷积神经网络提取这些区域上的特征, 最后进行分类。虽然 R-CNN 算法大大提高了物体检测精度, 但其存在计算冗余的问题。He 等人[6]提出的 sppnet 算法, 通过加入空间金字塔池化层解决了 R-CNN 反复缩放图像造成的影响。此外, Girshick 等人[7]提出 Fast R-CNN 算法, 通过加入区域池化层进一步优化了 R-CNN 与 sppnet 算法。Ren 等人[8]提出 Faster R-CNN 算法, 将选择性搜索用区域提案网络代替, 对网络进行端到端地训练, 极大地提高了检测器的速度。另一类是一阶段检测算法, 在产生候选区域的同时进行回归输出检测对象的位置和类别, 其代表性算法就是 YOLO 系列算法[9]。

YOLO 系列算法首先由 Redmon 等人于 2016 年提出, 其抛弃了“提案检测+验证”的两阶段检测范式, 通过将图像分割成多个网格, 直接预测每个网格所属的类别概率和位置信息。

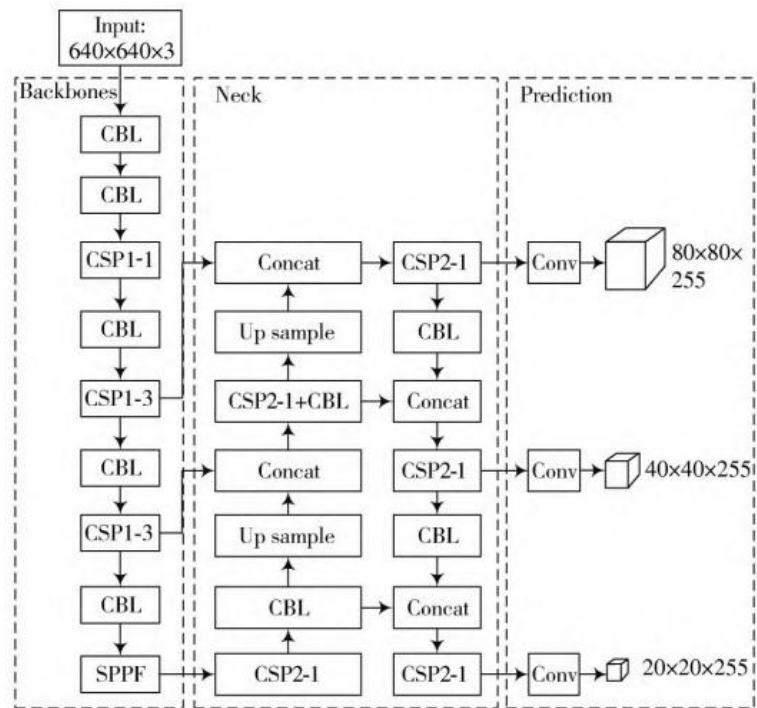
从上面的介绍我们得出结论: 传统方法的物体检测通常需要手动设计特征和分类器, 这些方法在处理复杂的场景时表现不佳, 并且需要大量的专业知识和时间来调整参数。相比之下, 基于深度学习的物体检测技术利用深度神经网络自动学习特征和分类器, 具有更高的准确性和鲁棒性, 适用于各种复杂场景。物体检测是目标跟踪的前提, 我们选择 yolov5 作为研究项目的物体检测器, 主要原因是 yolov5 具有高精度, 高效率, 简单易用, 而且可定制性强。

综合以上优点, yolov5 比较适合我们的使用场景。)

Алгоритм распознавания объектов YOLO

Yolov5 был предложен автором Glenn Jocher. Существует четыре размера моделей, а именно yolov5s, yolov5m, yolov5l и yolov5x. Среди них yolov5s имеет наименьшее количество сверточных слоев, самую быструю скорость обнаружения и самую низкую точность обнаружения, для остальных трех моделей количество сверточных слоев расположено в порядке возрастания. По мере увеличения сложности модели обнаружение становится быстрее, а точность обнаружения постепенно увеличивается.

Структура yolov5 состоит из четырех частей: Input, Backbones, Neck и Prediction, как показано на рисунке.

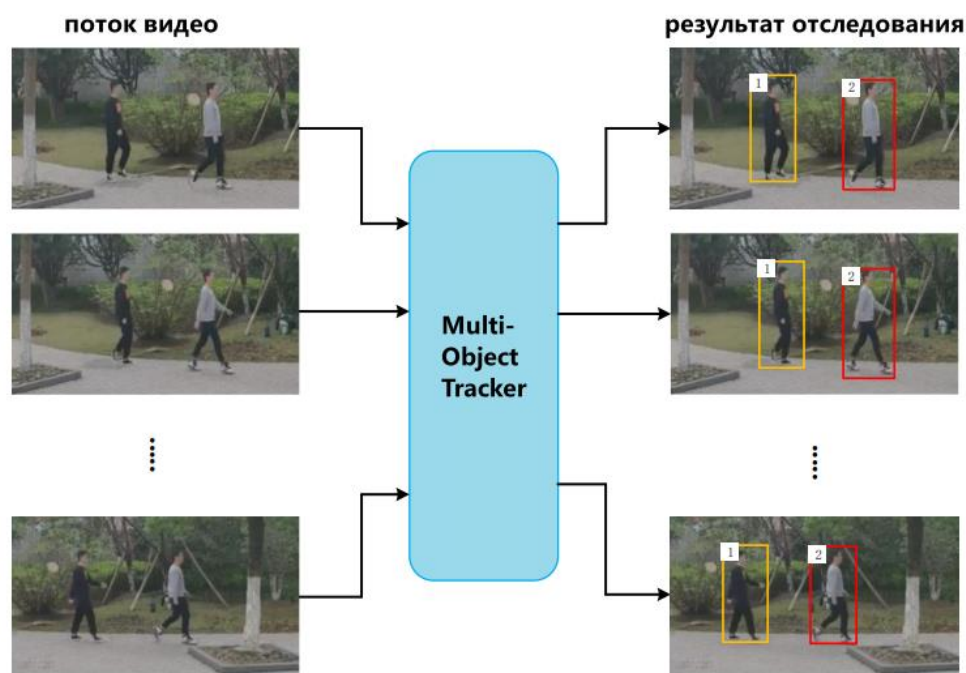


Сторона ввода включает в себя улучшение Mosaic, динамическую опорную рамку, адаптивную обработку изображений и т. Д.; Магистралы включают в себя кросс-этапную локальную сеть CSP, пространственный пирамидный пул SPPF, первый помогает сократить объем вычислений, а второй повышает точность обнаружения; Neck использует FPN+ Структура PAN, понижающая дискретизация улучшает семантическую информацию, повышающая дискретизация улучшает информацию о местоположении; Прогноз является конечным результатом обнаружения.

Чтобы добиться более высокой скорости, мы выбираем предварительно обученную модель yolov5s, выпущенную веб-сайтом с открытым исходным кодом yolov5 [Примечания].

Отслеживание (Трекинг) людей в виде последовательности

Задача трекинга объектов в видеопотоке заключается в сопоставлении детекция объекта на последовательности кадров видеопотока треку объекта. Задача трекинга множества объектов (Multi-Object Tracking, MOT) заключается в трекинге нескольких различных объектов. **【ОТСЛЕДОВАНИЕ ЛЮДЕЙ ПО ВИДЕО ПОСЛЕДОВАТЕЛЬНОСТИ Н.С. Захаров】** В частности, алгоритм трекинга множества объектов в основном решает проблему обнаружения объектов, которые мы хотим отслеживать, в каждом кадре видео, получения позиции на кадре и присвоения идентификатора каждому объекту. В процедуру трекинга идентификатор каждого объекта должен оставаться неизменным.



Как показано на рисунке, после ввода последовательности кадров, собранных камерой, в алгоритм MOT выводятся результаты отслеживания нескольких целей в кадре, включая положение цели на изображении и соответствующую идентификационную информацию.

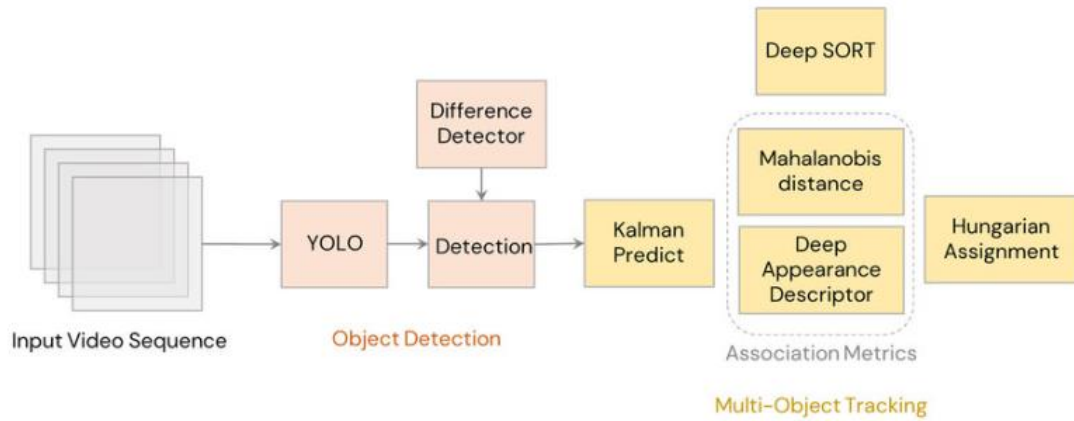
По сравнению с отслеживанием одной цели задачи отслеживания нескольких целей должны не только сталкиваться с такими проблемами, как изменение освещения, окклюзия и размытие движения, но также решать такие проблемы, как неопределенное количество целей и прерывание траекторий целей во время процесса отслеживания. С непрерывным ростом исследований многоцелевого отслеживания в последние годы, связанные с ними методы появляются бесконечно. Далее рассматривается популярный алгоритм MOT – DeepSORT

Алгоритм трекинга множества объектов DeepSORT

DeepSORT является усовершенствованием алгоритма отслеживания SORT. Основная цель DeepSORT — генерировать непрерывные траектории движения множества персонажей в видеопотоке, а общий рабочий процесс показан на рисунке ниже. Чтобы уменьшить количество переключателей идентификатора цели отслеживания (переключение идентификатора, ID switch), DeepSORT вводит функции внешнего вида и каскадное сопоставление. Алгоритм DeepSORT принимает координаты кадра обнаружения и достоверность результатов обнаружения сети обнаружения объектов в качестве входных данных, предсказывает положение пешеходов в следующем кадре с помощью алгоритма

фильтра Калмана, выполняет каскадное сопоставление, а затем использует венгерский алгоритм для ассоциации данных. В конце, фильтр Калмана обновлен.

(DeepSORT 是对 SORT 追踪算法的改进。DeepSORT 的主要目的是生成视频流中多个人物运动的连续轨迹，其总工作流程如下图。为了降低跟踪目标身份编号切换（ID switch）次数，DeepSORT 引入了外观特征和级联匹配。DeepSORT 算法以物体检测网络检测结果的检测框坐标和置信度作为输入，通过卡尔曼滤波算法对下一帧行人的位置进行预测，再进行级联匹配，然后利用匈牙利算法进行数据关联。最后进行卡尔曼滤波更新。)



DeepSORT вычисляет сходство, используя информацию о движении и внешнем виде цели. Для информации о движении расстояние Махаланобиса используется для оценки корреляции между прогнозируемой целью и обнаруженной целью. Выражение расстояния Махаланобиса:

$$d^{(1)}(i, j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i)$$

Где d_j – позиция bounding box j , y_i – позиция, предсказанное фильтром Калмана, S_i – ковариационная матрица обнаруженных и предсказанных позиций.

Когда цель закрыта в течение длительного времени или угол обзора колеблется, необходимо ввести информацию о внешнем виде и использовать косинусное расстояние для решения проблемы переключения идентичности, вызванного окклюзией.

Выражение косинусного расстояния:

$$d^{(2)}(i, j) = \text{Min}\{1 - r_j^T r_k^{(i)} \mid r_k^{(i)} \in R_i\}$$

Где r_j — собственный вектор bounding box d_j ; $r_k^{(i)}$ — набор собственных векторов, соответствующих последним 100 кадрам трекера i ; R_i — набор векторов признаков внешнего вида.

Чтобы в полной мере использовать два вида информации, для суммирования используется линейно-взвешенный метод:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j)$$

В формуле: λ является гиперпараметром тогда и только тогда, когда значение измерения $c_{i,j}$ Существует между $d^{(1)}(i,j)$ И $d^{(2)}(i,j)$, только в этом случае он считается относящимся к цели.

Стоит заметить, что, поскольку алгоритм фильтра Калмана основан на линейном равномерном движении, алгоритм DeepSORT может предсказывать состояние цели только в линейной среде и может быть не в состоянии хорошо предсказать, когда траектория пешехода нелинейна.

Повторная идентификация людей

Повторная идентификация множества людей (Person Re-identification, также известная как Ре-идентификация человека или межкамерное сопровождение) — это новая технология, появившаяся в области интеллектуального анализа видео в последние годы. Ре-идентификация множества людей может быть определена как задача присвоения одного и того же имени или индекса всем образам одного и того же человека, получаемым с пространственно-разнесенных камер, области видимости которых не пересекаются друг с другом, на основе выделения и анализа признаков его изображений.

【СОПРОВОЖДЕНИЕ И ПОВТОРНАЯ ИДЕНТИФИКАЦИЯ ЛЮДЕЙ Р. П. Богуш】 В видеонаблюдении из-за разрешения камеры и угла съемки обычно невозможно получить очень качественные изображения лиц. Когда распознавание лиц не работает, reid становится очень важной замещающей технологией. Очень важной особенностью reid является кросскамерность, поэтому при оценке производительности в научных работах необходимо получать одни и те же изображения пешеходов с разных камер.

Такие задачи характеризуется высокой сложностью реализации и требуют точной локализации людей в кадрах и правильной идентификации на текущем кадре или на кадре другой видеокamеры относительно предыдущих. **Одной из основных проблем является выбор дескриптора, описывающего человека.** 【Ye, S. Person Tracking and Re-Identification in Video for Indoor Multi-Camera Surveillance Systems / S. Ye, R. Bohush, C. Chen, I. Zakharava, S. Ablameyko // Pattern Recognition and Image Analysis, 2020. - Vol. 30, №4 – P. 827-837】 Для ее решения необходимо выявить отличительные признаки и путем сопоставления изображений людей из разных кадров или выполнения запроса сравнить их между собой или с признаками из имеющейся выборки изображений множества людей (галереи для ре-идентификации). Поиск и выделение набора наиболее отличительных особенностей объектов на изображениях, в том числе и людей, не формализован. Следовательно, требуется эмпирический поиск признаков, который в большинстве случаев является долгим и трудоемким процессом. Для сопровождения и повторной идентификации людей, из-за неоднозначности внешнего вида с разных ракурсов, вариаций освещения, различных разрешений камер, окклюзий такой подход требует нерационально большое количество времени. Поэтому долгое время значимые результаты для указанных задач не достигались. Совершенствование средств вычислительной техники и открытия в области глубокого обучения, в частности, развитие сверточных нейронных сетей (СНС) позволили автоматизировать процесс извлечения признаков изображений людей и обеспечить значительное увеличение точности повторной идентификации, однако в полной мере решение не получено в настоящее время. 【СОПРОВОЖДЕНИЕ И ПОВТОРНАЯ ИДЕНТИФИКАЦИЯ ЛЮДЕЙ Р. П. Богуш】

Наиболее часто используемая концепция повторной идентификации пешеходов:

При выполнении задачи повторной идентификации пешеходов мы обычно разделяем все изображения пешеходов на две группы: набор запросов (Query) и набор галерей (Gallery).

Набор запросов (Query): коллекция изображений, которые мы хотим использовать для поиска подобных изображений в наборе галерей.

Набор галерей (Gallery): коллекция всех доступных изображений для поиска.

Probe (или query image): одно изображение или группа связанных изображений для поиска. В нашем случае это изображение человека.

Обычно мы используем "probe" для представления каждого изображения в наборе запросов, которое является целью нашего поиска, а "галерея" представляет собой набор изображений, в которых мы ищем соответствующие изображения. Цель повторной идентификации пешеходов заключается в том, чтобы найти все изображения пешеходов в наборе галереи, которые похожи или совпадают с изображениями в наборе запроса, и могут быть точно аутентифицированы или идентифицированы.



Задача повторной идентификации пешеходов в основном состоит из двух этапов: извлечение признаков и измерение сходства. Извлечение признаков - это обучение особенностей, способных справляться с изменениями пешеходов под разными камерами. Обучение измерения - это отображение обученных признаков в пространство более высокой размерности, таким образом, чтобы одинаковые люди находились ближе друг к другу, а разные люди - дальше.

Традиционный метод заключается в ручном извлечении признаков изображения, например, цвета, HOG (гистограмма ориентированных градиентов)[4], SIFT (масштабно-инвариантное преобразование функций)[5], LOMO (локальный максимальный возникновение) и т. Д. Затем используется XQDA (кросс-видовой квадратичный дискриминантный анализ)[6] или KISSME (простое и прямолинейное изучение метрики)[7] для изучения лучшего измерения сходства. Однако традиционные методы ручного описания особенностей имеют ограниченные возможности и трудно применяются к задачам с большим объемом данных в сложных сценах. Кроме того, при обработке больших объемов данных традиционные методы обучения измерений также становятся очень сложными.

В последние годы глубокое обучение, представленное сверточными нейронными сетями, достигло больших успехов в области компьютерного зрения, победив традиционные методы в многих задачах и даже в какой-то мере превзойдя уровень человека [8-9]. В случае повторной идентификации пешеходов методы на основе глубокого обучения могут автоматически извлекать сложные описания признаков и использовать простые евклидовы расстояния для измерения сходства, что дает хорошие результаты. Другими

словами, глубокое обучение может реализовать задачу повторной идентификации пешеходов полностью, что делает задачу более простой. В настоящее время методы повторной идентификации пешеходов на основе глубокого обучения значительно превосходят традиционные методы по производительности. Эти преимущества делают глубокое обучение популярным в области повторной идентификации пешеходов.

Мы выбрали алгоритм fastreid, основанный на метрическом обучении, в качестве метода повторной идентификации пешеходов исследовательского проекта, и эксперименты показали, что она сыграла очень хорошую роль в нашем проекте. Ниже мы представим его подробно.

Алгоритм Fast reid

Благодаря построению нейронной сети повторной идентификации WRN, DeepSORT способен эффективно решать проблему перекрытия. WRN состоит главным образом из остаточных блоков, имеет много параметров и требует больших вычислительных затрат. Чтобы улучшить скорость и способность идентификации повторной идентификации сети, мы вводим модель повторной идентификации человека fastreid взамен WRN. Это позволяет снизить вычислительную сложность, увеличить экспрессивность и достичь более быстрой скорости и лучшего эффекта идентификации. Fastreid в целом состоит из четырех частей: предварительная обработка изображения, основная сеть, интеграция функций и головные сети. Общая архитектура показана на следующей рисунке:

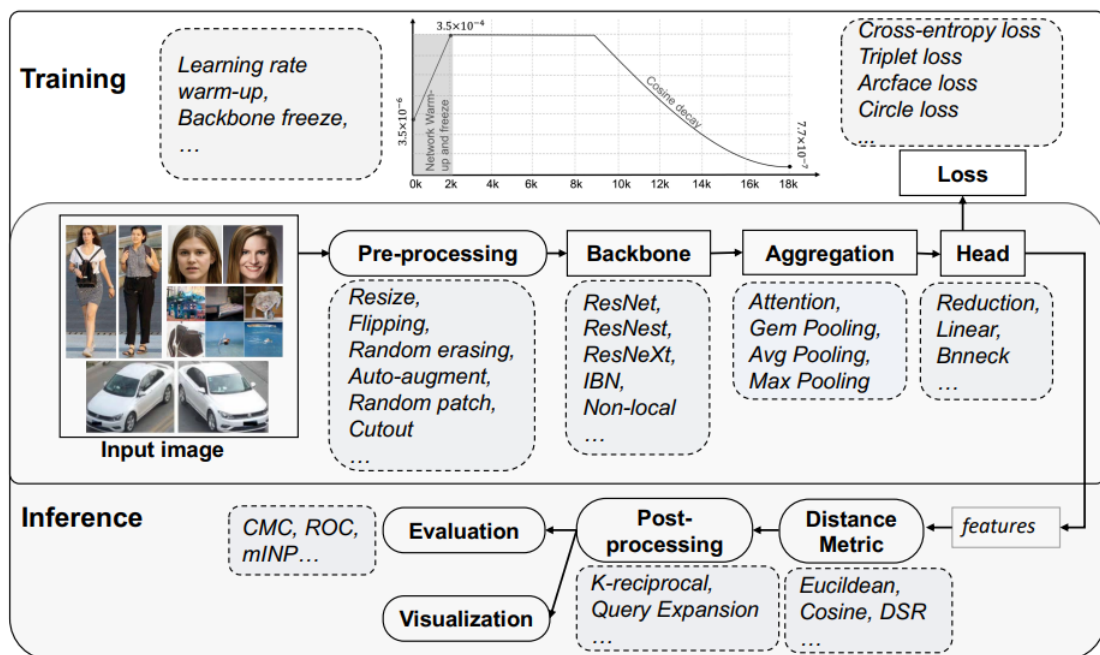


Figure 1. The Pipeline of FastReID library.

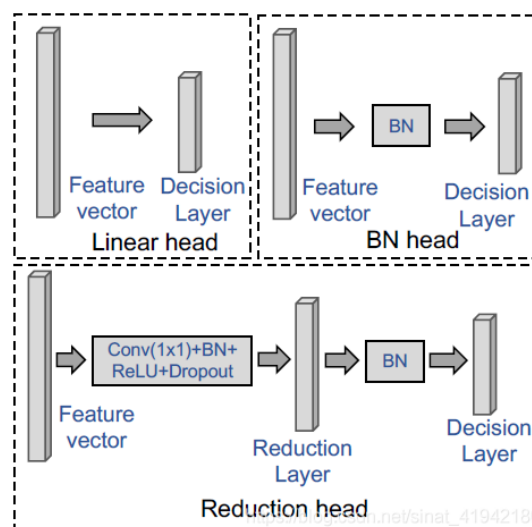
Training

Модуль предварительной обработки данных (Pre-processing) : В модуле предварительной обработки данных fastreid используются различные методы увеличения данных, такие как Resize, Flipping, Random erasing, Auto-augment, Random patch, Cutout и т.д.; эти приемы повышают устойчивость функций.

Модуль основной сети (Backbone) Внутри модуля основной сети реализована модель извлечения черт resnet. Мы также добавили модуль instance batch normalization (IBN) в основную часть, чтобы изучить более устойчивые функции.

Модуль интеграции функций (Aggregation) Модуль интеграции характеристик используется для агрегации функций, сгенерированных основной сетью, в глобальную характеристику. Реализованы четыре операции пулинга: максимальное пулингирование, среднее пулингирование, Gem пулингирование и Attention пулингирование.

Модуль заголовков (Head) Модуль заголовка - это часть, которая обрабатывает глобальный вектор, созданный модулем агрегации, включая BN Head (batch normalization head), Linear head и Reduction head. Три типа заголовков показаны на рисунке 3: линейный заголовок содержит только один слой decision, BN заголовок содержит слой BN и слой decision, а упрощенный заголовок содержит операции conv+BN+relu+dropout, слой reduction и слой decision.



Batch Normalization применяется для решения проблемы внутреннего сдвига ковариации, так как модели с насыщенной нелинейностью трудно обучать. Упрощенный слой (Reduction layer) направлен на уменьшение размерности высокомерных характеристик до 512 из 2048. Слой решений (Decision layer) выводит вероятности различных классов, отличает различные классы, используется для последующего обучения модели.

Функции потерь: Авторы реализовали четыре функции потерь, включая:

- Потерю кросс-энтропии (Cross-entropy loss)
- Тройную потерю (Triplet loss)
- Arcface loss
- Circle loss, предложенная Megvii в 2020 году, считается лучшей в настоящее время для всех видов задач машинного обучения.

Inference

Модуль расстояний (Distance Metric):

После обработки изображений модулями pre-processing, backbone, aggregation и head получается характеристический вектор размерности 512. Модуль расстояний выполняет сравнение вектора функций и галерейного векторного хранилища и выдает результат поиска: набор изображений, похожих на оригинальное изображение.

Модуль последующей обработки (Post-processing): Послеобработка относится к обработке результатов поиска, включая два способа повторной классификации (re-rank): k-reciprocal coding [11] и Query Expansion [12].

Модуль оценки производительности (Evaluation)

Для оценки эффективности мы использовали стандартные показатели, которые можно найти в большей части литературы по повторной идентификации личности, а именно кумулятивную кривую соответствия (СМС) и среднюю среднюю точность (map). Кроме того, мы добавляем две метрики: кривую рабочих характеристик приемника (ROC) и средний обратный отрицательный штраф (mnr).

Модуль визуализации (Visualization)

Предоставляется инструмент рейтингового списка для поиска результатов, который полезен для отображения результатов повторной идентификации человека.

Метод подсчета количества людей в видеопоследовательности

Под визуальным подсчетом людей понимается возможность оценить количество людей в видео наблюдения с помощью определенной обработки. Визуальная статистика потоков людей широко используется во многих областях, в то же время широкое внедрение камер наблюдения и постоянное расширение сценариев применения статистики потоков людей привлекли большое внимание исследователей к проблеме статистики потоков людей. .

Многие компании в мире провели множество исследований и экспериментов с прикладными продуктами для подсчета людей, например, серия продуктов для подсчета людей Wise Count от компании Fei Ruisi имеет статистическую точность более 95% и может работать непрерывно в течение всего времени. День.В практических приложениях Он имеет высокую точность в реальном времени и статистическую точность.

Что касается статистики трафика покупателей, мы изучили алгоритм, основанный на сверточной нейронной сети, который может регистрировать конкретное время входа и выхода каждого покупателя из магазина для принятия бизнес-решений и анализа.

Основные исследования этой статьи

Основное содержание исследования этой статьи заключается в разработке метода, сочетающего алгоритм yolov5, алгоритм отслеживания DeepSORT и алгоритм повторной идентификации пешеходов fastreid для реализации статистики пассажиропотока магазина. Путем оптимизации алгоритмов обнаружения, отслеживания и повторной идентификации пешеходов мы добились точного отслеживания и подсчета покупателей под разными

углами и в различных условиях освещения, а также разработали систему статистики потока покупателей, ориентированную на магазины.

ГЛАВА 2 Обучение модели для отслеживания людей в нескольких видеопотоках

Обучение модели для повторной идентификации fastreid Подготовка набора данных

Набор данных MSMT17

На конференции CVPR2018 был предложен новый крупномасштабный набор данных MSMT17, который ближе к реальной сцене, а именно Multi-Scene Multi-Time, охватывающий несколько сцен и несколько периодов времени.

Набор данных MSMT17 использует сеть из 15 камер видеонаблюдения в кампусе, включая 12 наружных камер и 3 внутренние камеры. Для сбора необработанного видео наблюдения было выбрано 4 дня в месяц с разными погодными условиями. Каждый день собирается 3 часа видео, охватывающих три временных периода: утро, полдень и день. Таким образом, общая продолжительность необработанного видео составляет 180 часов.



Три комментатора-человека потратили два месяца на просмотр обнаруженных ограничивающих прямоугольников и пометку пешеходов. Наконец, получается 126 441 ограничительная рамка 4 101 пешехода. Сравнение и статистическая информация с другими наборами данных показаны на рисунке ниже

Dataset	<i>MSMT17</i>	<i>Duke</i> [41, 27]	<i>Market</i> [39]	<i>CUHK03</i> [20]	<i>CUHK01</i> [19]	<i>VIPeR</i> [8]	<i>PRID</i> [10]	<i>CAVIAR</i> [3]
BBoxes	126,441	36,411	32,668	28,192	3,884	1,264	1,134	610
Identities	4,101	1,812	1,501	1,467	971	632	934	72
Cameras	15	8	6	2	10	2	2	2
Detector	Faster RCNN	hand	DPM	DPM, hand	hand	hand	hand	hand
Scene	outdoor, indoor	outdoor	outdoor	indoor	indoor	outdoor	outdoor	indoor

Преимущества MSMT17 по сравнению с другими наборами данных заключаются в следующем:

- (1) Больше пешеходов, ограничивающих рамок и камер;

- (2) Сложные сцены и фоны;
- (3) охват нескольких периодов времени, поэтому есть сложные изменения освещения;
- (4) Лучший детектор пешеходов (Faster- RCNN), ограничительная рамка более точная.

Протокол оценки

Набор данных случайным образом делится в соотношении обучение-тестирование 1:3, а не поровну, как другие наборы данных. Целью этого является поощрение эффективных стратегий обучения, поскольку маркировка данных очень дорога в реальных приложениях.

Наконец, обучающий набор содержит 1041 пешехода с 32621 ограничивающей рамкой, а тестовый набор содержит 3060 пешеходов с 93820 ограничивающими рамками. Для тестового набора 11659 ограничивающих рамок выбираются случайным образом в качестве запроса, а остальные 82161 ограничивающие рамки используются в качестве галереи.

Показатели теста - кривая СМС и mAP. Для каждого запроса может быть несколько положительных совпадений.

Принцип обучения

Принцип обучения модели Рейда заключается в следующем:

1. Предварительная обработка данных: сначала изображения в наборе данных предварительно обрабатываются с вероятностью 0,5 обрезки, масштабирования, симметрии и улучшения данных, чтобы повысить устойчивость модели к изображениям людей в разных позах и условиях освещения.

2. Извлечение признаков: fastreid использует обученную сверточную нейронную сеть серии resnet в качестве экстрактора признаков для извлечения 512-мерного вектора признаков из входного изображения. В то же время fastreid также вводит некоторые новые методы извлечения признаков, такие как выравнивание с учетом масштаба (SAA) и случайное стирание, для дальнейшего улучшения способности выражения признаков.

3. Метричное обучение: вектор признаков сопоставляется с низкоразмерным пространством посредством метрического обучения, а функции потерь триплетов и перекрестной энтропийной потери используются для оптимизации модели, так что образцы с одной и той же идентичностью находятся ближе в пространстве признаков, а разные идентичности находятся дальше друг от друга в пространстве признаков.

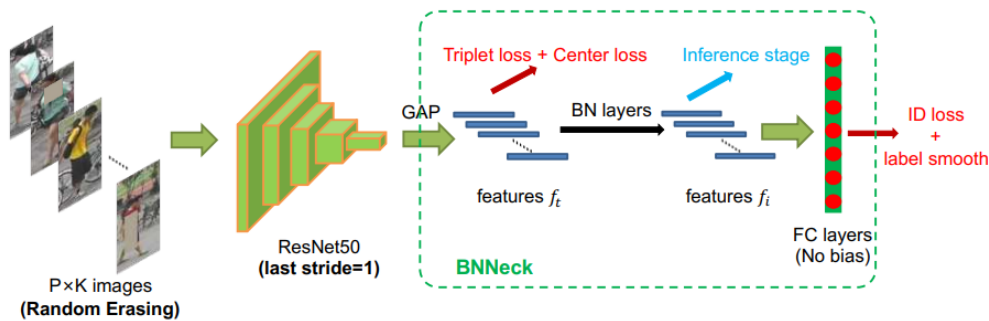
4. Оптимизация модели: используйте оптимизатор Adam для оптимизации параметров модели, чтобы повысить эффективность обучения и точность модели.

Построить модель

Чтобы ускорить сходимость моделей и повысить производительность модели, мы используем resnet50, модель сверточной нейронной сети, предварительно обученную на наборе данных imagenet. Imagenet — это широко используемый набор данных для классификации изображений, который содержит 1,4 миллиона изображений в более чем 1000 категориях. Модель resnet50 использует глубокую остаточную сетевую структуру, которая может эффективно справляться с крупномасштабными задачами классификации

изображений. После завершения обучения веса модифицированной модели можно использовать в других задачах компьютерного зрения для повышения производительности и эффективности задачи, например, в нашей повторной идентификации пешеходов.

Размерность выходных признаков предварительно обученной модели resnet50 составляет 2048. Мы продолжаем вставлять средний слой пула и линейный классификатор после магистральной сети. Выходная размерность линейного классификатора равна N. N представляет собой идентификационный номер обучающих данных.



Функции потерь и алгоритмы оптимизации

Функция потерь:

Наша модель Ре-идентификация дает два результата: признак f и предсказанные логиты p . Функции f используются для расчета триплетных потерь, а логиты p используются для расчета энтропии перекрестных потерь.

Cross Entropy Loss

Crossentropyloss — это широко используемая функция потерь классификации, которая обычно используется для обучения глубоких нейронных сетей задачам классификации изображений. В частности, для входной выборки x и ее истинной метки y формула расчета перекрестной энтропийной потери выглядит следующим образом:

$$\text{Cross Entropy Loss} = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

Где C — количество категорий, y_i Представляет вероятность i -й категории (то есть истинной метки), а \hat{y}_i Представляет вероятность того, что модель предсказывает, что x принадлежит i -й категории.

Цель crossentropyloss — минимизировать разрыв между предсказанием и реальной меткой, чтобы модель могла более точно предсказывать категорию, к которой относится каждая выборка (каждый человек в обучающей выборке рассматривается как отдельная категория).

Triplet Loss

Triplet Loss — одна из функций потерь, широко используемых в области распознавания лиц, назначение которой — приблизить векторы встраивания изображений (embedding) одного и того же человека и отдалить векторы встраивания изображений разных людей. В

частности, Triplet Loss изучит вектор встраивания для каждого образца, что может сделать изображения одного и того же человека ближе, а изображения разных людей — дальше.

Для триплета, включающего три изображения привязки, положительное и отрицательное, формула расчета tripletloss выглядит следующим образом:

$$\text{Triplet Loss} = \text{Max}(0, d(a, p) - d(a, n) + m)$$

Где $d(a, p)$ представляет собой евклидово расстояние между привязкой и положительным значением, $d(a, n)$ представляет собой евклидово расстояние между привязкой и отрицательным значением, а m представляет собой отступ, который является предустановленным гиперпараметром A , обычно положительным числом. Смысл этой формулы таков: если расстояние между текущим якорем и плюсом минус расстояние между якорем и минусом плюс маржа меньше или равно 0, это означает, что текущий вектор встраивания достаточно хорош и не нуждается в оптимизации. ; в противном случае необходимо обновить параметры модели для лучшего встраивания векторов.

Triplet Loss обычно используется в сочетании с Batch Hard Triplet Mining, то есть для обучения в каждой партии выбирается самый сложный триплет. Конкретный метод заключается в выборе изображений пешеходов, очень похожих на якорь, но разных категорий, в качестве отрицательных образцов, и выборе изображений пешеходов, которые сильно отличаются от якоря, но принадлежат к той же категории, что и положительные образцы, чтобы улучшить производительность модели.

Этапы обучения

1. Используйте resnet50 (инициализированные веса взяты из предварительно обученной модели imagenet), а затем измените ее полносвязный слой на N . N представляет собой идентификационный номер обучающих данных.

2. Мы случайным образом отбираем P идентификаторов и собираем K изображений для каждого идентификатора, а размер последней партии $B = P * K$. В этой статье мы устанавливаем $P = 16$, $K = 4$.

3. Мы изменили размер каждого изображения на 256×128 и заполнили 10 пикселей значениями 0, а затем повторно обрезаем изображение размером 256×128 , используя случайное кадрирование.

4. Каждое изображение случайным образом переворачивается по горизонтали с вероятностью 0,5.

5. Каждое изображение декодируется как 32-битное необработанное значение пикселя с плавающей запятой в $[0,1]$, а затем мы нормализуем каналы RGB, вычитая 0,485, 0,456, 0,406 и разделяя на 0,229, 0,224, 0,225.

6. Предсказать логиты p в соответствии с функциями $\text{reid } f$, выводимыми моделью.

7. Функции $\text{reid } f$ используются для расчета триплетных потерь, логиты p используются для расчета энтропии перекрестных потерь, а запас m триплетных потерь устанавливается равным 0,3.

8. Для оптимизации модели используется метод Адама. Начальная скорость обучения установлена на 0,00035 и уменьшается на 0,1 в 40-ю и 70-ю эпохи соответственно. Всего 120 эпох.

Тренировочный процесс

```
Terminal Local Command Prompt
[05/08 02:55:20 fastreid.utils.events]: eta: 12:07:38 epoch/iter: 22/10999 total_loss: 1.474 loss_cls: 1.344 loss_triplet: 0.2238 time: 0.9364 data_time: 0.0088 lr: 3.50e-04 max_mem: 2539M
[05/08 03:10:33 fastreid.utils.events]: eta: 12:04:28 epoch/iter: 22/10199 total_loss: 1.474 loss_cls: 1.37 loss_triplet: 0.122 time: 0.9264 data_time: 0.0012 lr: 3.50e-04 max_mem: 2539M
[05/08 03:21:29 fastreid.utils.events]: eta: 12:01:20 epoch/iter: 22/10383 total_loss: 1.442 loss_cls: 1.331 loss_triplet: 0.1098 time: 0.9363 data_time: 0.0009 lr: 3.50e-04 max_mem: 2539M
[05/08 03:32:40 fastreid.utils.events]: eta: 12:01:04 epoch/iter: 22/10399 total_loss: 1.441 loss_cls: 1.328 loss_triplet: 0.1125 time: 0.9363 data_time: 0.0009 lr: 3.50e-04 max_mem: 2539M
[05/08 03:43:07 fastreid.utils.events]: eta: 11:57:53 epoch/iter: 22/10599 total_loss: 1.433 loss_cls: 1.334 loss_triplet: 0.1054 time: 0.9363 data_time: 0.0013 lr: 3.50e-04 max_mem: 2539M
[05/08 03:53:26 fastreid.utils.events]: eta: 11:54:37 epoch/iter: 22/10799 total_loss: 1.449 loss_cls: 1.341 loss_triplet: 0.1108 time: 0.9363 data_time: 0.0015 lr: 3.50e-04 max_mem: 2539M
[05/08 04:03:46 fastreid.utils.events]: eta: 11:51:41 epoch/iter: 22/10855 total_loss: 1.445 loss_cls: 1.332 loss_triplet: 0.1148 time: 0.9363 data_time: 0.0008 lr: 3.50e-04 max_mem: 2539M
[05/08 04:13:01 fastreid.utils.events]: eta: 11:51:16 epoch/iter: 22/10999 total_loss: 1.453 loss_cls: 1.33 loss_triplet: 0.1201 time: 0.9362 data_time: 0.0013 lr: 3.50e-04 max_mem: 2539M
[05/08 04:24:08 fastreid.utils.events]: eta: 11:47:51 epoch/iter: 23/11199 total_loss: 1.448 loss_cls: 1.34 loss_triplet: 0.1049 time: 0.9362 data_time: 0.0010 lr: 3.50e-04 max_mem: 2539M
[05/08 04:34:07 fastreid.utils.events]: eta: 11:45:45 epoch/iter: 23/11227 total_loss: 1.432 loss_cls: 1.318 loss_triplet: 0.115 time: 0.9362 data_time: 0.0008 lr: 3.50e-04 max_mem: 2539M
[05/08 04:44:22 fastreid.utils.events]: eta: 11:44:12 epoch/iter: 24/11399 total_loss: 1.434 loss_cls: 1.316 loss_triplet: 0.1223 time: 0.9362 data_time: 0.0014 lr: 3.50e-04 max_mem: 2539M
[05/08 04:54:16 fastreid.utils.events]: eta: 11:41:16 epoch/iter: 24/11599 total_loss: 1.453 loss_cls: 1.342 loss_triplet: 0.1033 time: 0.9361 data_time: 0.0010 lr: 3.50e-04 max_mem: 2539M
[05/08 05:04:28 fastreid.utils.events]: eta: 11:37:56 epoch/iter: 24/11799 total_loss: 1.433 loss_cls: 1.296 loss_triplet: 0.1203 time: 0.9361 data_time: 0.0018 lr: 3.50e-04 max_mem: 2539M
[05/08 05:14:30 fastreid.utils.events]: eta: 11:37:50 epoch/iter: 24/11799 total_loss: 1.433 loss_cls: 1.296 loss_triplet: 0.1203 time: 0.9361 data_time: 0.0018 lr: 3.50e-04 max_mem: 2539M
[05/08 05:24:38 fastreid.utils.events]: eta: 11:34:54 epoch/iter: 24/11999 total_loss: 1.462 loss_cls: 1.352 loss_triplet: 0.1079 time: 0.9361 data_time: 0.0012 lr: 3.50e-04 max_mem: 2539M
[05/08 05:34:42 fastreid.utils.events]: eta: 11:31:56 epoch/iter: 25/12199 total_loss: 1.434 loss_cls: 1.399 loss_triplet: 0.1105 time: 0.9360 data_time: 0.0008 lr: 3.50e-04 max_mem: 2539M
[05/08 05:44:49 fastreid.utils.events]: eta: 11:30:28 epoch/iter: 25/12271 total_loss: 1.438 loss_cls: 1.398 loss_triplet: 0.1127 time: 0.9360 data_time: 0.0014 lr: 3.50e-04 max_mem: 2539M
[05/08 05:54:59 fastreid.utils.events]: eta: 11:28:26 epoch/iter: 26/12399 total_loss: 1.454 loss_cls: 1.341 loss_triplet: 0.09971 time: 0.9360 data_time: 0.0010 lr: 3.50e-04 max_mem: 2539M
[05/08 06:05:10 fastreid.utils.events]: eta: 11:25:01 epoch/iter: 26/12599 total_loss: 1.42 loss_cls: 1.331 loss_triplet: 0.1034 time: 0.9359 data_time: 0.0009 lr: 3.50e-04 max_mem: 2539M
[05/08 06:15:09 fastreid.utils.events]: eta: 11:22:45 epoch/iter: 26/12743 total_loss: 1.424 loss_cls: 1.313 loss_triplet: 0.1172 time: 0.9359 data_time: 0.0014 lr: 3.50e-04 max_mem: 2539M
[05/08 06:25:02 fastreid.utils.events]: eta: 11:21:51 epoch/iter: 27/12799 total_loss: 1.434 loss_cls: 1.331 loss_triplet: 0.1091 time: 0.9359 data_time: 0.0012 lr: 3.50e-04 max_mem: 2539M
[05/08 06:34:58 fastreid.utils.events]: eta: 11:18:22 epoch/iter: 27/12999 total_loss: 1.410 loss_cls: 1.31 loss_triplet: 0.1054 time: 0.9358 data_time: 0.0009 lr: 3.50e-04 max_mem: 2539M
[05/08 06:44:34 fastreid.utils.events]: eta: 11:15:09 epoch/iter: 27/13199 total_loss: 1.444 loss_cls: 1.317 loss_triplet: 0.1161 time: 0.9358 data_time: 0.0014 lr: 3.50e-04 max_mem: 2539M
[05/08 06:54:29 fastreid.utils.events]: eta: 11:14:54 epoch/iter: 27/13215 total_loss: 1.440 loss_cls: 1.331 loss_triplet: 0.1151 time: 0.9358 data_time: 0.0012 lr: 3.50e-04 max_mem: 2539M
[05/08 07:04:21 fastreid.utils.events]: eta: 11:11:45 epoch/iter: 28/13399 total_loss: 1.414 loss_cls: 1.316 loss_triplet: 0.1003 time: 0.9357 data_time: 0.0010 lr: 3.50e-04 max_mem: 2539M
[05/08 07:14:27 fastreid.utils.events]: eta: 11:08:35 epoch/iter: 28/13599 total_loss: 1.428 loss_cls: 1.307 loss_triplet: 0.112 time: 0.9356 data_time: 0.0013 lr: 3.50e-04 max_mem: 2539M
[05/08 07:24:26 fastreid.utils.events]: eta: 11:07:06 epoch/iter: 28/13687 total_loss: 1.438 loss_cls: 1.331 loss_triplet: 0.1019 time: 0.9356 data_time: 0.0012 lr: 3.50e-04 max_mem: 2539M
[05/08 07:34:34 fastreid.utils.events]: eta: 11:05:19 epoch/iter: 29/13799 total_loss: 1.414 loss_cls: 1.319 loss_triplet: 0.09249 time: 0.9356 data_time: 0.0010 lr: 3.50e-04 max_mem: 2539M
[05/08 07:44:37 fastreid.utils.events]: eta: 11:02:15 epoch/iter: 29/13999 total_loss: 1.407 loss_cls: 1.291 loss_triplet: 0.107 time: 0.9355 data_time: 0.0014 lr: 3.50e-04 max_mem: 2539M
[05/08 07:54:39 fastreid.data.datasets.hoi3d]: == Loaded HOI37 in csv format:
| dataset | # imgs | # images | # cameras |
|-----|-----|-----|-----|
| query | 3040 | 11659 | 35 |
| gallery | 3040 | 12161 | 35 |
[05/08 08:04:32 fastreid.evaluation.evaluator]: Start inference on 93620 images
[05/08 08:10:46 fastreid.evaluation.evaluator]: Inference done 11/733. 0.6099 s / batch. ETA=0:08:04
[05/08 08:19:16 fastreid.evaluation.evaluator]: Inference done 56/733. 0.6714 s / batch. ETA=0:07:35
[05/08 08:29:47 fastreid.evaluation.evaluator]: Inference done 101/733. 0.4745 s / batch. ETA=0:07:07
[05/08 08:40:16 fastreid.evaluation.evaluator]: Inference done 146/733. 0.4764 s / batch. ETA=0:06:37
[05/08 08:49:48 fastreid.evaluation.evaluator]: Inference done 191/733. 0.4759 s / batch. ETA=0:06:06
[05/08 08:59:18 fastreid.evaluation.evaluator]: Inference done 236/733. 0.4793 s / batch. ETA=0:05:36
[05/08 09:08:49 fastreid.evaluation.evaluator]: Inference done 281/733. 0.4757 s / batch. ETA=0:05:06
[05/08 09:18:19 fastreid.evaluation.evaluator]: Inference done 325/733. 0.4766 s / batch. ETA=0:04:36
[05/08 09:27:50 fastreid.evaluation.evaluator]: Inference done 369/733. 0.4780 s / batch. ETA=0:04:07
[05/08 09:37:20 fastreid.evaluation.evaluator]: Inference done 413/733. 0.4782 s / batch. ETA=0:03:37
[05/08 09:46:50 fastreid.evaluation.evaluator]: Inference done 457/733. 0.4795 s / batch. ETA=0:03:08
[05/08 09:56:21 fastreid.evaluation.evaluator]: Inference done 501/733. 0.4808 s / batch. ETA=0:02:38
[05/08 10:05:51 fastreid.evaluation.evaluator]: Inference done 546/733. 0.4820 s / batch. ETA=0:02:09
[05/08 10:15:21 fastreid.evaluation.evaluator]: Inference done 591/733. 0.4830 s / batch. ETA=0:01:40
[05/08 10:24:52 fastreid.evaluation.evaluator]: Inference done 637/733. 0.4843 s / batch. ETA=0:01:10
[05/08 10:34:22 fastreid.evaluation.evaluator]: Inference done 675/733. 0.4845 s / batch. ETA=0:00:39
[05/08 10:44:53 fastreid.evaluation.evaluator]: Inference done 719/733. 0.4851 s / batch. ETA=0:00:09
[05/08 09:30:20 fastreid.evaluation.evaluator]: Total inference time: 0:08:49.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:26 fastreid.evaluation.evaluator]: Total inference time: 0:08:45.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:28 fastreid.evaluation.evaluator]: Total inference pure compute time: 0:08:44 (0.720107 s / batch per device, on 1 devices)
C:\source\511\hoi3d\fast-reid\fastreid\evaluation\rank.py:14: UserWarning: Cython rank evaluation (very fast so highly recommended) is unavailable, now use python evaluation.
warnings.warn
[05/08 10:10:06 fastreid.engine.default]: Evaluation results for HOI37 in csv format:
[05/08 10:10:06 fastreid.evaluation.testing]: Evaluation results in csv format:
| dataset | Rank-1 | Rank-5 | Rank-10 | AP | mAP | metric |
|-----|-----|-----|-----|-----|-----|-----|
| HOI37 | 84.19 | 98.95 | 99.18 | 62.89 | 14.94 | 65.84 |
[05/08 10:10:06 fastreid.engine.default]: Prepare testing set
```

```
Terminal Local Command Prompt
[05/08 08:38:22 fastreid.data.datasets.hoi3d]: == Loaded HOI37 in csv format:
| dataset | # imgs | # images | # cameras |
|-----|-----|-----|-----|
| query | 3040 | 11659 | 35 |
| gallery | 3040 | 12161 | 35 |
[05/08 08:48:32 fastreid.evaluation.evaluator]: Start inference on 93620 images
[05/08 08:58:46 fastreid.evaluation.evaluator]: Inference done 11/733. 0.6099 s / batch. ETA=0:08:04
[05/08 09:09:16 fastreid.evaluation.evaluator]: Inference done 56/733. 0.6714 s / batch. ETA=0:07:35
[05/08 09:19:47 fastreid.evaluation.evaluator]: Inference done 101/733. 0.4745 s / batch. ETA=0:07:07
[05/08 09:30:16 fastreid.evaluation.evaluator]: Inference done 146/733. 0.4764 s / batch. ETA=0:06:37
[05/08 09:40:48 fastreid.evaluation.evaluator]: Inference done 191/733. 0.4759 s / batch. ETA=0:06:06
[05/08 09:50:18 fastreid.evaluation.evaluator]: Inference done 236/733. 0.4793 s / batch. ETA=0:05:36
[05/08 10:00:49 fastreid.evaluation.evaluator]: Inference done 281/733. 0.4757 s / batch. ETA=0:05:06
[05/08 10:10:19 fastreid.evaluation.evaluator]: Inference done 325/733. 0.4766 s / batch. ETA=0:04:36
[05/08 10:20:50 fastreid.evaluation.evaluator]: Inference done 369/733. 0.4780 s / batch. ETA=0:04:07
[05/08 10:30:20 fastreid.evaluation.evaluator]: Inference done 413/733. 0.4782 s / batch. ETA=0:03:37
[05/08 10:40:50 fastreid.evaluation.evaluator]: Inference done 457/733. 0.4795 s / batch. ETA=0:03:08
[05/08 10:50:21 fastreid.evaluation.evaluator]: Inference done 501/733. 0.4808 s / batch. ETA=0:02:38
[05/08 11:00:51 fastreid.evaluation.evaluator]: Inference done 546/733. 0.4820 s / batch. ETA=0:02:09
[05/08 11:10:21 fastreid.evaluation.evaluator]: Inference done 591/733. 0.4830 s / batch. ETA=0:01:40
[05/08 11:20:52 fastreid.evaluation.evaluator]: Inference done 637/733. 0.4843 s / batch. ETA=0:01:10
[05/08 11:30:22 fastreid.evaluation.evaluator]: Inference done 675/733. 0.4845 s / batch. ETA=0:00:39
[05/08 11:40:53 fastreid.evaluation.evaluator]: Inference done 719/733. 0.4851 s / batch. ETA=0:00:09
[05/08 09:30:20 fastreid.evaluation.evaluator]: Total inference time: 0:08:49.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:26 fastreid.evaluation.evaluator]: Total inference time: 0:08:45.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:28 fastreid.evaluation.evaluator]: Total inference pure compute time: 0:08:44 (0.720107 s / batch per device, on 1 devices)
C:\source\511\hoi3d\fast-reid\fastreid\evaluation\rank.py:14: UserWarning: Cython rank evaluation (very fast so highly recommended) is unavailable, now use python evaluation.
warnings.warn
[05/08 10:10:06 fastreid.engine.default]: Evaluation results for HOI37 in csv format:
[05/08 10:10:06 fastreid.evaluation.testing]: Evaluation results in csv format:
| dataset | Rank-1 | Rank-5 | Rank-10 | AP | mAP | metric |
|-----|-----|-----|-----|-----|-----|-----|
| HOI37 | 84.19 | 98.95 | 99.18 | 62.89 | 14.94 | 65.84 |
[05/08 10:10:06 fastreid.engine.default]: Prepare testing set
```

```
Terminal Local Command Prompt
[05/08 09:30:20 fastreid.evaluation.evaluator]: Total inference time: 0:08:49.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:26 fastreid.evaluation.evaluator]: Total inference time: 0:08:45.339901 (0.721615 s / batch per device, on 1 devices)
[05/08 09:30:28 fastreid.evaluation.evaluator]: Total inference pure compute time: 0:08:44 (0.720107 s / batch per device, on 1 devices)
C:\source\511\hoi3d\fast-reid\fastreid\evaluation\rank.py:14: UserWarning: Cython rank evaluation (very fast so highly recommended) is unavailable, now use python evaluation.
warnings.warn
[05/08 10:10:06 fastreid.engine.default]: Evaluation results for HOI37 in csv format:
[05/08 10:10:06 fastreid.evaluation.testing]: Evaluation results in csv format:
| dataset | Rank-1 | Rank-5 | Rank-10 | AP | mAP | metric |
|-----|-----|-----|-----|-----|-----|-----|
| HOI37 | 84.19 | 98.95 | 99.18 | 62.89 | 14.94 | 65.84 |
[05/08 10:10:06 fastreid.engine.default]: Prepare testing set
```

Результаты обучения

Метрики оценки эффективности модели

В исследованиях повторной идентификации пешеходов для оценки эффективности модели в основном используются два показателя оценки: кумулятивные характеристики соответствия (Cumulative MatchingCharacteristics, CMC) и средняя средняя точность (Mean Average Precision, map).

Кумулятивная кривая соответствия (CMC)

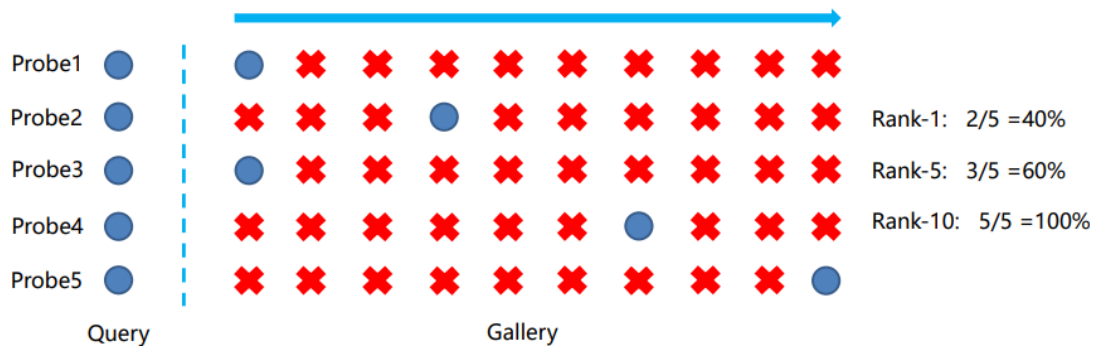
Кумулятивная кривая соответствия представлена как k-я частота совпадения (ранг-k), которая конкретно относится к вероятности нахождения одного и того же изображения пешехода на k изображениях, наиболее похожих на изображение, которое необходимо

получить *probe* в библиотеке изображений-кандидатов *G*. Выражение выглядит следующим образом:

$$Rank - k = \frac{\sum_{probe \in Q} f(k, index_{probe})}{m}$$

$$f(k, index_{probe}) = \begin{cases} 0, & k \geq index_{probe} \\ 1, & k < index_{probe} \end{cases}$$

Где $index_{probe}$ Указывает местоположение первого изображения *probe*, принадлежащего тому же пешеходу, что и зонд изображения, который должен быть запрошен в результатах поиска, а *m* указывает количество изображений, которые должны быть извлечены в базе данных *Q* для извлечения. Вообще говоря, Ранг-1 является наиболее важным показателем для оценки эффективности модели повторной идентификации человека, который может быть выражен как вероятность того, что первое изображение в полученном отсортированном списке является тем же пешеходом, что и изображение, которое нужно получить. В реальных ситуациях, помимо использования ранга-1, другие часто используемые ранг-*k* включают ранг-5, ранг-10, ранг-20 и т. Д.



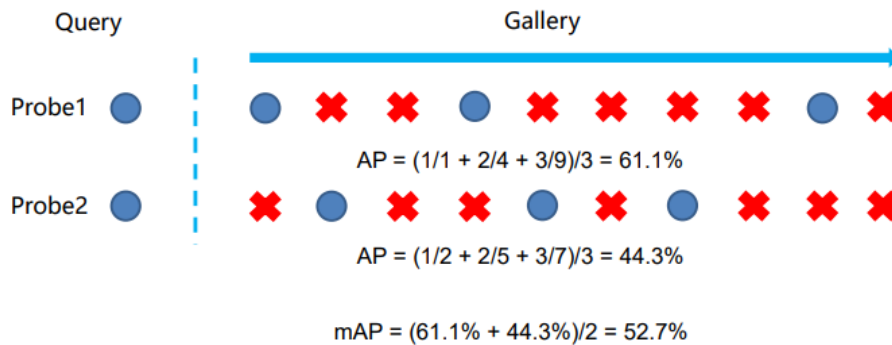
Средняя средняя точность (mAP)

В ранних наборах данных повторной идентификации пешеходов в библиотеке изображений-кандидатов есть только одно изображение пешехода *probe*, которое совпадает с извлеченным изображением *probe*, но с введением больших наборов данных, таких как Market-1501 и dukemtmc-reid, изображение *probe* обычно можно найти несколько совпадающих изображений с одинаковым идентификатором в библиотеке изображений-кандидатов *G*, и трудно оценить влияние труднодоступных образцов на производительность модели, используя только индекс оценки СМС. Следовательно, для более полной оценки эффективности модели повторной идентификации человека к индексу оценки модели повторной идентификации человека добавляется средняя точность. Средняя точность — это индекс, который может оценить результаты ранжирования всех положительных образцов. Только когда все изображения извлеченного человека ранжируются вверху в библиотеке кандидатов, индекс mAP будет высоким, поэтому он может более полно отражать пешеходов. • Производительность моделей повторной идентификации. При расчете mAP сначала вычисляется средняя точность (Average Precision, AP), соответствующая каждому извлекаемому изображению *probe*, которая используется для измерения точности распознавания модели на одном образце запроса. Процесс расчета показан в формуле:

$$AP(q) = \frac{\sum_{k \in k_1, k_2, \dots, k_S} \frac{k_r}{k}}{S}$$

Где S указывает количество положительных образцов, соответствующих полученному изображению *probe* в библиотеке кандидатов G , $\{k_1, k_2, \dots, k_s\}$ — позиция индекса S положительных образцов в результатах сортировки, а k_r указывает количество положительных образцов в первых k результатах. Наконец, после вычисления средней точности всех изображений пешеходов в библиотеке Q , содержащей m изображений, можно использовать среднее значение значений AP всех выборок для получения mAP . Процесс расчета показан в формуле:

$$mAP = \frac{\sum_{q_i \in Q} AP(q_i)}{m}$$

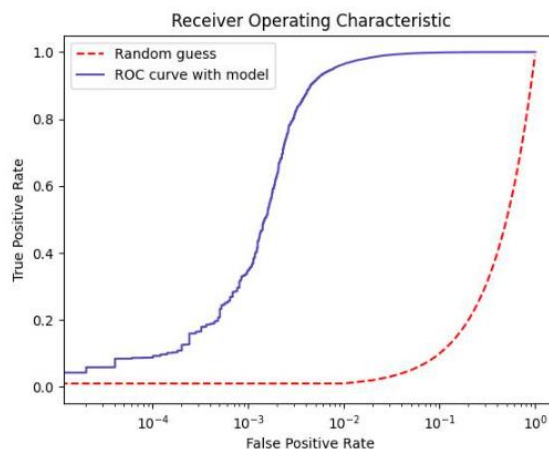


Среднее обратное отрицательное наказание ($minp$): указывает стоимость поиска всех правильных совпадений, чем ниже, тем лучше.

Результаты обучения на наборе данных MSMT17

Мы используем 1/3 данных в MSMT17 в качестве образцов для обучения модели Рейда, а затем используем оставшиеся 2/3 данных в качестве проверочного набора для проверки возможностей модели, и результаты следующие:

Dataset	Rank-1↑	Rank-5↑	Rank-10↑	Map↑	Minp↓
MSMT17	84.19	90.95	95.10	62.89	14.94



По приведенным данным видно, что модель *fastreid* добилась хороших результатов на наборе данных MSMT17. В частности, она достиг 84,19% точности на 1-м ранге, 90,95% и

95,10% на 5-м и 10-м рангах соответственно. При этом средняя точность (map) также достигла 62,89%, что означает, что производительность модели на всем наборе данных относительно стабильна. Следует отметить, что модель составляет всего 14,94% с точки зрения обратного среднего коэффициента штрафа за отрицательную выборку (minp), Следует отметить, что с точки зрения обратного среднего отрицательного штрафа за выборку (minp) модель составляет 14,94%, что означает, что в некоторых трудно идентифицируемых случаях производительность модели может снижаться. В целом эти результаты показывают, что модель fastreid хорошо работает с набором данных MSMT17, но все еще есть возможности для улучшения при применении к особенно сложным сценариям распознавания.

Сравним производительность других моделей Reid в этом наборе данных:

Methods	MSMT17	
	Rank-1	Map
Ianet(IVPR'19)	75.7	45.8
Auto-reid(ICCV'19)	78.2	52.5
Osnet(ICCV'19)	78.7	52.9
Abdnet(ICCV'19)	82.3	60.8
Circle Loss(CVPR'20)	76.9	52.1
Ours	84.19	62.89

Через сравнение мы знаем, что точность нашей модели намного выше, чем у других моделей, выпущенных в то же время, и показатели точности являются лучшими.

Реализация алгоритма DeepSORT с помощью fastreid и yolov5

Устраивание модели

1. Используйте DeepSORT в качестве сети отслеживания, чтобы отслеживать каждого человека и сопоставлять его с предыдущими траекториями.
2. Используйте yolov5s в качестве части обнаружения DeepSORT для обнаружения людей в каждом кадре изображения и вывода кадра обнаружения.
3. Используйте fastreid в качестве экстрактора признаков для извлечения признаков внешнего вида пешеходов, чтобы DeepSORT мог различать разных людей.
4. Определить гиперпараметры модели. Параметры обучающей модели следующие:

DEEPSORT:

```

REID_CKPT: "./fast-reid/checkpoint/model-final.pth"
MAX_DIST: 0.2
MIN_CONFIDENCE: 0.3
NMS_MAX_OVERLAP: 0.5
MAX_IOU_DISTANCE: 0.7
MAX_AGE: 140
N_INIT: 3
NN_BUDGET: 100

```

Набор тестовых данных MOT-16

Мы используем общедоступный набор данных о пешеходах MOT-16 для тестирования нашей модели отслеживания нескольких объектов (DeepSORT после улучшения модуля повторной идентификации пешеходов). Этот набор данных является общим набором данных, используемым для оценки производительности алгоритмов отслеживания нескольких целей. Существуют различные пешеходные сцены, в том числе 14 видеопоследовательностей, а 7 видеопоследовательностей помечены подробно с идентификаторами пешеходов и позициями кадров, которые используются для обучения многоцелевых пешеходов Алгоритм слежения, еще 7 видеофрагментов служат тестовой выборкой.



图 4-1 MOT 16 数据集示意图

Критерии оценки, использованные в эксперименте: точность сопровождения (MOTA), точность сопровождения (MOTP), количество правильно сопровождаемых траекторий цели выше 80% (MT), количество правильно сопровождаемых траекторий цели ниже 20% (ML), преобразование идентификатора цели Время (IDSW), точность отслеживания и точность отслеживания рассчитываются следующим образом:

$$MOTA = 1 - \frac{\sum_t (m_t + n_t + s_t)}{\sum_t g_t}$$

$$MOTP = \frac{\sum_{i,t} d_i^t}{\sum_t c_t}$$

Где t представляет t -й кадр, m_t представляет количество пропущенных целей обнаружения в t -кадре; n_t представляет количество ложных целей обнаружения в t -кадре; s_t представляет количество переключений идентичности в t -кадре; g_t представляет общее количество целей в кадре t ; расстояние между прогнозируемой позицией цели кадра i и реальной позицией; c_t представляет количество успешно сопоставленных целей в кадре t .

Все эксперименты проводились на видеопоследовательности тестового набора MOT-16. Чтобы лучше подчеркнуть производительность алгоритма в этой статье, для сравнения были разработаны два разных эксперимента.

Эксперимент 1: Сравнительное тестирование нашего улучшенного алгоритма DeepSORT (yolov5+fastreid+DeepSORT) и оригинального алгоритма DeepSORT. Протестируйте оба варианта на тестовом видео, чтобы проанализировать плюсы и минусы улучшенного алгоритма DeepSORT.

Эксперимент 2. Анализ надежности улучшенного алгоритма в различных сценариях. Сравните улучшенный алгоритм с несколькими основными алгоритмами и проанализируйте преимущества и недостатки улучшенного алгоритма.

Эксперимент 1: Сравнительное тестирование улучшенного алгоритма DeepSORT и исходного алгоритма DeepSORT.

Процедура эксперимента

Мы протестируем все видеопоследовательности в наборе данных MOT16, в которых идентификаторы пешеходов и положения границ отмечены вручную (gt.txt), всего 7. Это: MOT16-02, MOT16-04, MOT16-05, MOT16-09, MOT16-10, MOT16-11, MOT16-13. Здесь в качестве примера взят MOT16-13, а операции для других видео аналогичны.

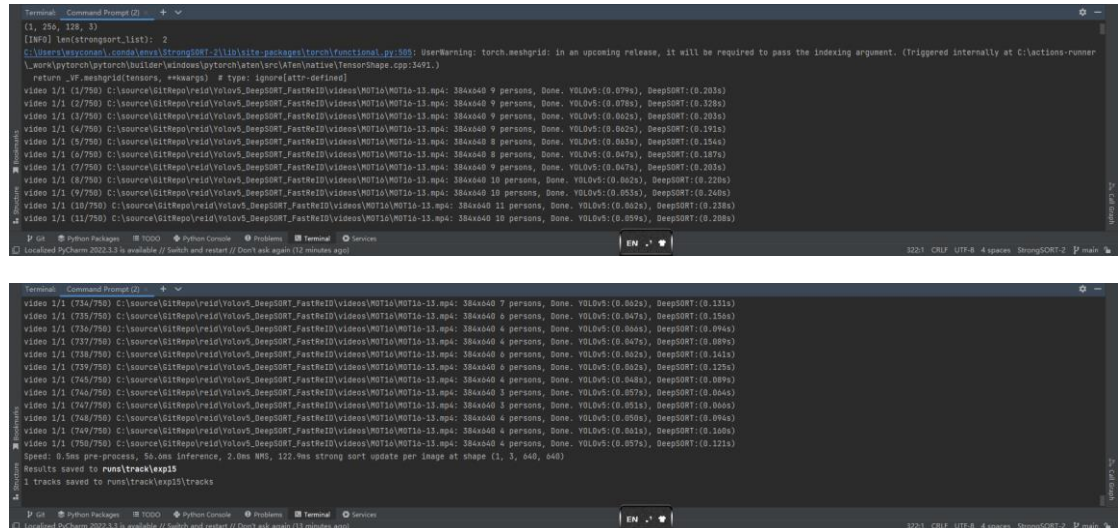
1. Загрузите набор данных MOT16, получите видеофрагменты и соответствующие им файлы gt.txt. Файл gt.txt в MOT16-13 выглядит следующим образом (частично), а параметры разделены запятыми:

1,1,1376,485,37,28,0,11,1
2,1,1379,486,37,28,0,11,1
3,1,1382,487,38,29,0,11,1
4,1,1386,488,38,29,0,11,1
5,1,1389,490,38,29,0,11,1

Файл gt.txt представляет собой текстовый файл CSV, каждая строка содержит объект, описывающий отслеживаемый объект в одном из фреймов, с 9 значениями, разделенными запятыми. TracEval использует только первые 6, номер кадра, идентификатор цели и 4 координаты кадра отслеживания, а последние 3 (надежность цели, категория цели, видимость) не участвуют в расчетах и могут быть проигнорированы следующим образом:

<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <class>, <visibility>

2. Запустите программу track.py в созданной нами модели DeepSORT.



```
Terminal - Command Prompt (C:)
[Info] len(strongsort_list): 2
C:\Users\asysonal\cmd\envs\strongsort_2\lib\site-packages\torch\functional.py:505: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered internally at C:\actions-runner\_work\python\python\lib\site-packages\torch\tensor\src\aten\native\TensorShape.cpp:3491.)
return VF.meshgrid(tensors, **kwargs) # Type: ignore[attr-defined]
video 1/1 (1/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 9 persons, Done. VLOvS:(0.079s), DeepSORT:(0.203s)
video 1/1 (2/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 9 persons, Done. VLOvS:(0.078s), DeepSORT:(0.326s)
video 1/1 (3/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 9 persons, Done. VLOvS:(0.062s), DeepSORT:(0.203s)
video 1/1 (4/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 9 persons, Done. VLOvS:(0.062s), DeepSORT:(0.321s)
video 1/1 (5/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 8 persons, Done. VLOvS:(0.065s), DeepSORT:(0.354s)
video 1/1 (6/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 8 persons, Done. VLOvS:(0.047s), DeepSORT:(0.187s)
video 1/1 (7/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 9 persons, Done. VLOvS:(0.047s), DeepSORT:(0.203s)
video 1/1 (8/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 10 persons, Done. VLOvS:(0.062s), DeepSORT:(0.220s)
video 1/1 (9/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 10 persons, Done. VLOvS:(0.053s), DeepSORT:(0.240s)
video 1/1 (10/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 11 persons, Done. VLOvS:(0.062s), DeepSORT:(0.288s)
video 1/1 (11/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 10 persons, Done. VLOvS:(0.059s), DeepSORT:(0.288s)

P Git Python Packages TQDM Python Console Problems Terminal Services
Localized PyCharm 2022.3.1 is available // Switch and restart // Don't ask again (12 minutes ago) 32x1 CRUF UTF-8 4 spaces StrongSORT-2 main
```

```
Terminal - Command Prompt (C:)
video 1/1 (12/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 7 persons, Done. VLOvS:(0.062s), DeepSORT:(0.111s)
video 1/1 (13/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 8 persons, Done. VLOvS:(0.047s), DeepSORT:(0.156s)
video 1/1 (14/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.066s), DeepSORT:(0.094s)
video 1/1 (15/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.047s), DeepSORT:(0.089s)
video 1/1 (16/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 8 persons, Done. VLOvS:(0.062s), DeepSORT:(0.141s)
video 1/1 (17/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 8 persons, Done. VLOvS:(0.062s), DeepSORT:(0.125s)
video 1/1 (18/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.048s), DeepSORT:(0.089s)
video 1/1 (19/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 3 persons, Done. VLOvS:(0.057s), DeepSORT:(0.066s)
video 1/1 (20/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 3 persons, Done. VLOvS:(0.051s), DeepSORT:(0.066s)
video 1/1 (21/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.080s), DeepSORT:(0.094s)
video 1/1 (22/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.061s), DeepSORT:(0.166s)
video 1/1 (23/750) C:\source\GitRepo\reid\VolovS_DeepSORT_FastReID\videos\MOT16\MOT16-13.mp4: 384x640 4 persons, Done. VLOvS:(0.057s), DeepSORT:(0.121s)

Speed: 0.0s pre-process, 56.0s inference, 2.0s NMS, 122.9s strong sort update per image at shape (1, 3, 640, 640)
Results saved to runs\track\exp15
1 traces saved to runs\track\exp15\traces

P Git Python Packages TQDM Python Console Problems Terminal Services
Localized PyCharm 2022.3.1 is available // Switch and restart // Don't ask again (13 minutes ago) 32x1 CRUF UTF-8 4 spaces StrongSORT-2 main
```

Получите файл данных отслеживания MOT16-13.txt, совместимый с форматом MOT16. Файл MOT16-13.txt:

```
3 7 726 247 12 40 -1 -1 -1 0
3 8 269 276 16 45 -1 -1 -1 0
3 9 744 256 13 37 -1 -1 -1 0
4 1 1 341 13 93 -1 -1 -1 0
4 2 773 279 22 57 -1 -1 -1 0
```

Формат MOT16-13.txt немного отличается от gt.txt, заявляя, что он «совместим» с форматом MOT16. Формат «совместимость» имеет всего 10 значений, а его параметры разделены пробелами. Первые 6 параметров, участвующих в расчете MOT16, такие же, как и gt.txt, а последние 4 параметра в расчете индикатора не участвуют, все они равны -1. Формат следующий:

<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>

3. Запускаем программу оценки и получаем результат:

All sequences for deepsort finished in 0.91 seconds																
NOTA: deepsort-pedestrian		MOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	ONTA	HOTA(0)	LocA(0)	HOTALocA(0)			
MOT16-13		57.313	59.474	56.029	66.611	73.942	63.201	75.428	81.504	60.957	74.316	76.452	56.816			
COMBINED		57.313	59.474	56.029	66.611	73.942	63.201	75.428	81.504	60.957	74.316	76.452	56.816			
CLEAR: deepsort-pedestrian		MOTA	MOTP	MODA	CLR Re	CLR Pr	MTR	PTR	MLR	sMOTA	CLR TP	CLR FN	CLR FP	IDSW	MT	PT
MOT16-13		72.154	78.316	72.338	81.212	90.149	50	31.25	18.75	54.544	2654	614	290	6	8	5
COMBINED		72.154	78.316	72.338	81.212	90.149	50	31.25	18.75	54.544	2654	614	290	6	8	5
Count: deepsort-pedestrian		Dets	GT Dets	IDs	GT IDs											
MOT16-13		2944	3268	18	16											
COMBINED		2944	3268	18	16											

4. Выполните ту же операцию для MOT16-02, MOT16-04, MOT16-05, MOT16-09, MOT16-10, MOT16-11 соответственно и просмотрите весь набор данных MOT16. Получите окончательный результат на этом наборе данных MOT16.

Результаты эксперимента

Сравните производительность исходного алгоритма DeepSORT, см. Таблицу ниже:

Algorithm	MOTA ↑	MOTP ↑	MT / % ↑	ML / % ↓	Ids / % ↓
Original DeepSORT	61.4	79.1	32.8	18.2	781
Ours	66.2	80.8	35.3	17.6	760

Видно, что после улучшения алгоритма DeepSORT все показатели немного улучшаются. Точность сопровождения (MOTA) увеличилась на 5,2%, точность сопровождения (MOTP) увеличилась на 1,7%, количество правильно сопровождаемых траекторий целей более 80% (MT) увеличилось на 3,5%, а количество правильно сопровождаемых траекторий целей ниже 20%. (ML) сократилось на 0,6%, а общее количество целевых переключателей идентификации (ID) сократилось в 21 раз. Видно, что улучшенный DeepSORT может уменьшить количество пешеходных переключений.

Эксперимент 2: сравнение улучшенного алгоритма DeepSORT с несколькими основными алгоритмами.

Результаты эксперимента 2 представлены в таблице.

Algorithm	MOTA ↑	MOTP ↑	MT / % ↑	ML / % ↓	Ids / % ↓	FPS / Hz ↑
SORT	59.8	79.6	25.4	22.7	1423	8.6
Original DeepSORT	61.4	79.1	32.8	18.2	781	6.4
JDE	64.4	-	35.4	20	1544	18.5
Ours	66.2	80.8	35.3	17.6	760	5.8

Мы усовершенствовали алгоритм повторной идентификации пешеходов (fastreid) на основе DeepSORT, благодаря чему была повышена точность, точность и возможность сохранения идентификатора пешехода модели. По сравнению с другими моделями наша модель имеет комплексные преимущества: точность сопровождения (MOTA) и точность сопровождения (MOTP) самые высокие, а количество правильно сопровождаемых траекторий целей более чем на 80% (MT) практически равно JDE и делят первое место. Количество правильно отслеженных траекторий целей ниже 20% (ML) и общее количество переключений идентификации целей (ID) являются самыми низкими.

Кроме того, замечено, что производительность в реальном времени немного хуже, чем у алгоритма JDE, потому что JDE является одноэтапным алгоритмом отслеживания, а отслеживание в реальном времени относительно высоко, но переключение идентификаторов относительно частые, что обусловлено взаимным перекрытием мишеней. В розничной среде часто бывают сцены с большим количеством плотных пешеходов, поэтому наш алгоритм является подходящим выбором в это время.

ГЛАВА 3 Система оценки трафика магазина основа на нейтронных сетей

Цели системы

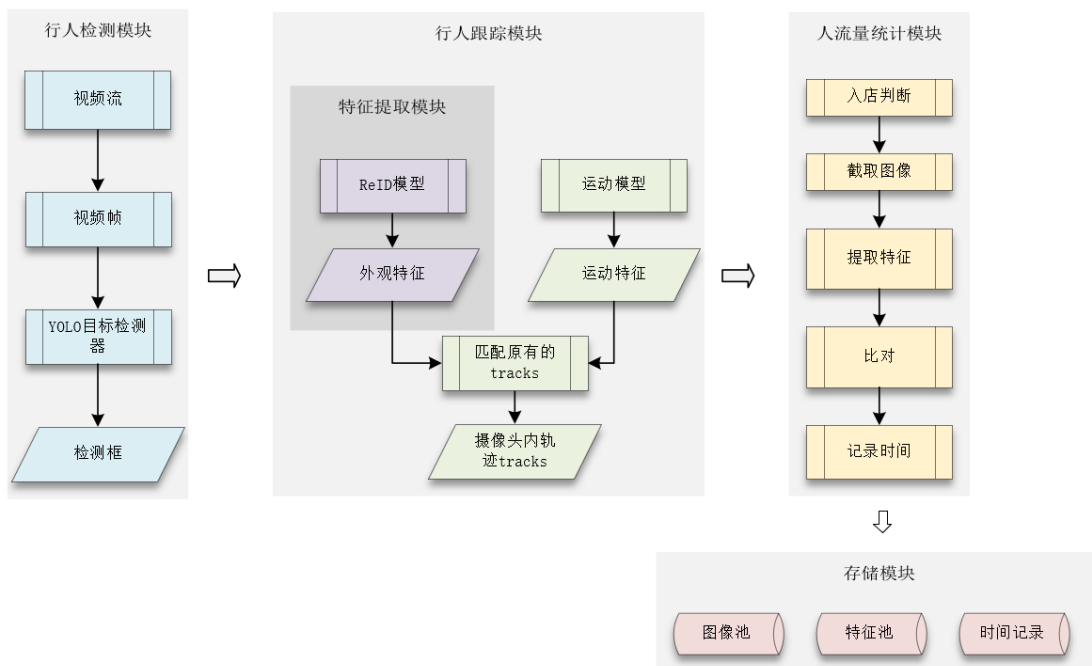
Подсчет людей на основе видеонаблюдения является основой для сбора бизнес-информации и выполнения задач интеллектуального управления, а также важным функциональным компонентом современного интеллектуального видеонаблюдения. В этом проекте будут использоваться передовые технологии компьютерного зрения и глубокого обучения, такие как yolov5, DeepSORT и Fastreid, для реализации автоматической идентификации, отслеживания и анализа покупателей в магазине. В частности, система будет получать видеопоток в магазине через камеру, использовать yolov5 для автоматического обнаружения и идентификации клиентов, появляющихся на видео, затем использовать DeepSORT для отслеживания цели покупателя и, наконец, использовать fastreid для идентификации и анализа поведение покупателя, чтобы реализовать автоматическую статистику и анализ посещаемости магазина.

Благодаря исследованиям и практике этого проекта он может помочь продавцам более точно понять привычки и предпочтения поведения клиентов, сформулировать более научные и эффективные стратегии продаж, а также повысить прибыльность и рыночную конкурентоспособность магазинов. В то же время система также может предоставлять продавцам функции мониторинга и прогнозирования потоков клиентов в режиме реального времени, помогая продавцам еще больше оптимизировать управление операциями, повышать уровень обслуживания и удобство работы пользователей.

Описание системы

Общая архитектура системы

На основе fastreid и DeepSORT в этой статье разрабатывается система статистики потока покупателей, ориентированная на магазины. Система состоит из четырех модулей: модуль обнаружения пешеходов, модуль слежения за пешеходами, модуль статистики потоков людей и модуль хранения. Общая структура системы показана на рисунке.



Модуль обнаружения пешеходов

На вход модуля обнаружения пешеходов подается видеопоток с одной камеры, в процессе реализации в данной работе используется алгоритм обнаружения целей YOLO. В частности, в части обнаружения пешеходов алгоритм обнаружения целей YOLO используется для обнаружения пешеходов в каждом кадре видео, и пешеходы помечаются прямоугольным кадром из статического изображения каждого кадра, а конкретное положение пешехода в возвращается картинка, определение которой приведено в формуле:

$$b = [x, y, w, h]$$

Где (x, y) — пиксельные координаты верхнего левого угла кадра обнаружения, а (w, h) — ширина и высота кадра обнаружения.

Модуль извлечения признаков

Модуль извлечения признаков предназначен для извлечения признаков внешнего вида пешеходов. Здесь мы будем использовать алгоритм fastreid и алгоритм фильтра Калмана. Входными данными модуля извлечения признаков является кадр обнаружения пешеходов, выдаваемый модулем обнаружения пешеходов. После модели fastreid и фильтра Калмана получаются признаки внешнего вида и признаки информации о движении пешеходов. Информация об этих двух функциях будет служить ориентиром для последующего модуля отслеживания пешеходов, который будет соответствовать траектории и кадру обнаружения.

Модуль отслеживания пешеходами

Модуль отслеживания пешеходов использует модель DeepSORT, и его входными данными является кадр обнаружения, предоставленный модулем обнаружения пешеходов, и функции пешеходов, извлеченные модулем извлечения функций. Цель состоит в том, чтобы связать кадр обнаружения пешеходов в текущем кадре с существующим движением. Траектория в предыдущем кадре. После получения кадра обнаружения, предоставленного модулем обнаружения пешеходов, сначала используйте модуль извлечения признаков и модуль извлечения информации о движении, чтобы извлечь признаки внешнего вида и признаки движения текущего кадра обнаружения кадра и существующей дорожки движения, а затем вычислить матрицу расстояний между ними. , и, наконец, использовать венгерский алгоритм. Алгоритм связывает кадры-кандидаты обнаружения в текущем кадре с существующими траекториями движения в предыдущем кадре, чтобы построить траектории всех пешеходов в видео.

Модуль подсчета посетителей

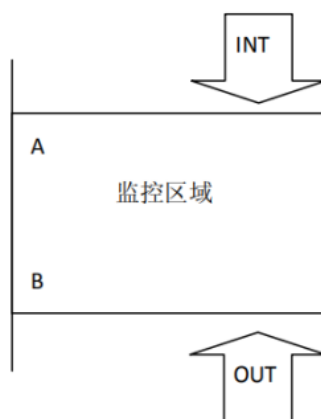
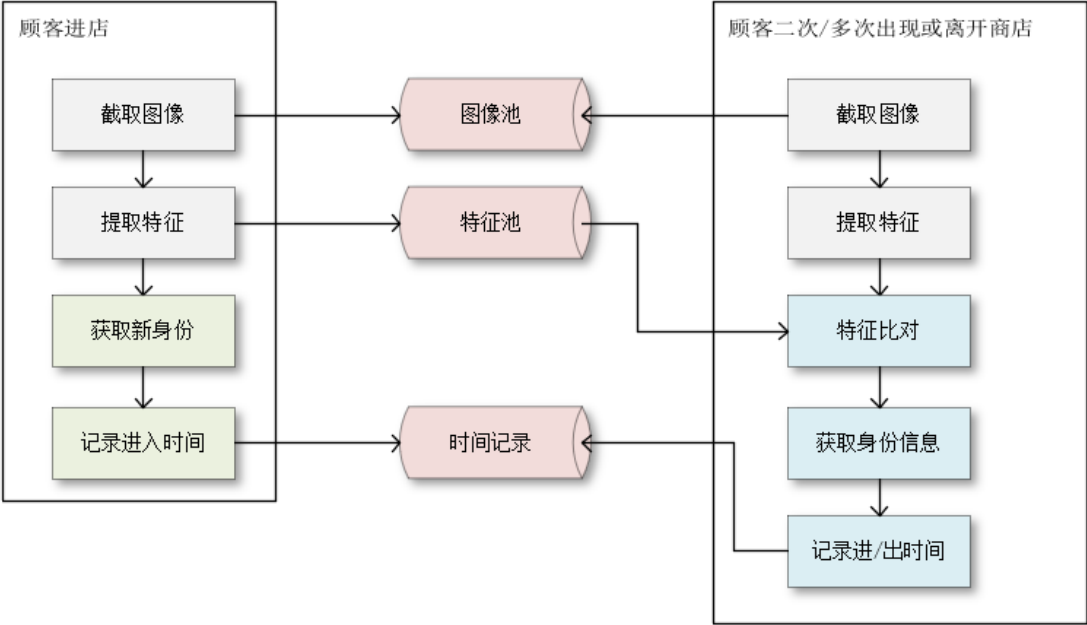


图 3-5 计数区域

Вход модуля подсчета посетителей — это траектория движения всех людей под одной камерой, и его цель — точно зафиксировать количество людей, входящих и выходящих из магазина. После рассмотрения реалистичных требований к задаче в этом исследовании используется «метод двух линий» для завершения подсчета людей. Конкретный принцип двухлинейного метода заключается в следующем: в магазине устанавливается стационарная камера и получается видеопоток наблюдения, причем в видеопотоке отмечаются две счетные линии, которые записываются как A и B соответственно. На основе траектории движения тела на пересечении траектории и линии указывается конкретное количество проходящих людей.

В качестве примера предположим, что магазин находится ниже линии B. Если мы предположим, что траектория входа клиента будет сначала соприкасаться с линией A, а затем с контактом с B, это называется поведением при входе в дверь. И наоборот, если траектория ухода пешехода должна сначала коснуться линии B, а затем линии A, это называется выходом. Нам нужно только записать, касается ли траектория назначенного пешехода сначала линии A или линии B, а затем мы можем определить, входит ли пешеход в магазин или выходит из него.

Когда происходит вход или выход, модули обнаружения пешеходов, отслеживания пешеходов и извлечения функций будут работать последовательно, чтобы получить снимки экрана и функции клиента, а также записать отметку времени в журнал. Когда клиент уходит, мы сравниваем его характеристики с набором характеристик в базе данных, чтобы получить информацию о личности клиента, и на этой основе фиксируем время ухода. Таким образом, мы можем точно фиксировать время входа и выхода покупателей в супермаркете.



Модуль хранения

Модуль хранения отвечает за регистрацию времени, когда покупатели заходят в магазин, появляются под камерой и выходят из магазина.

Каждый раз, когда кто-то входит в зону подсчета, записывайте время появления человека и в то же время перехватывайте изображение соответствующего человека и извлекайте вектор признаков на основе изображения. Если камера является камерой в магазине или на выходе, личность пешехода также учитывается в соответствии с моделью повторной идентификации. Наконец, личность человека, в каком объективе он появляется, время появления и изображение человека сохраняются на диске для последующего статистического анализа.

Среда разработки

Аппаратные среды, использованные для разработки, показаны в Таблице 1:

Таблица 1

Предмет	Технические характеристики	Комментарий
---------	----------------------------	-------------

Чип	AMD Ryzen 5 5600H with Radeon Graphics 3.30 ghz	
Памяти	16.0 GB	ПО использует до 2.1 GB
Камер	1080P	720P минимум
Операционная системы	Windows 10 Professional	

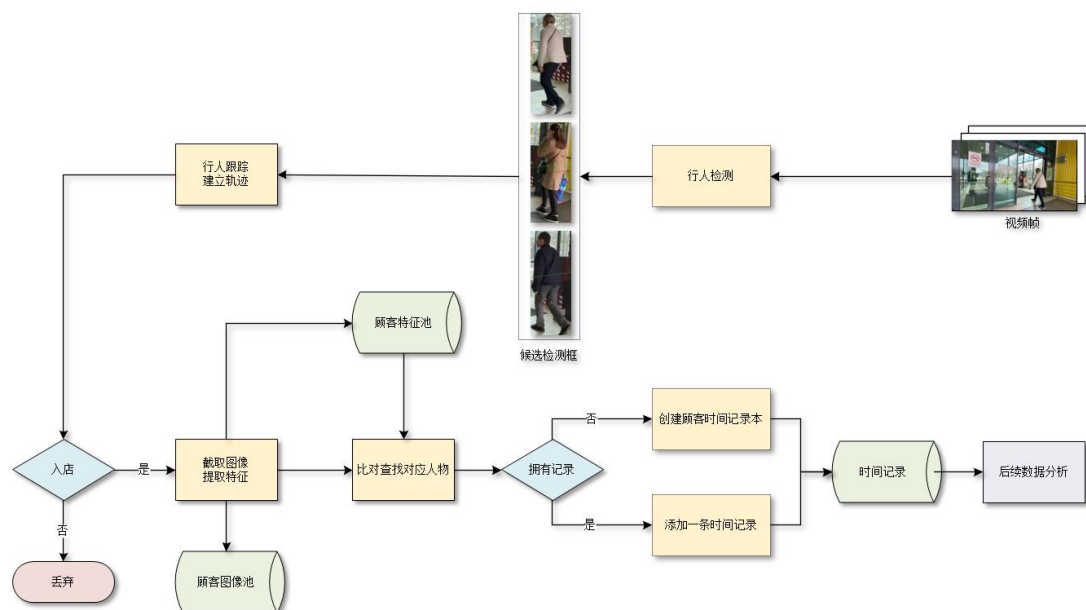
Программные среды, использованные в разработке, показаны в Таблице 2:

Таблица 2

Предмет	Версия	Комментарий
Python	3.6	Данная или выше
Pycharm	2022.1.2 (Educational Edition)	Данная или выше
Opencv	4.5.3	Данная или выше
Pytorch	1.3.0	Данная или выше

Демонстрация и тестирование системы

Алгоритм статистики трафика



Тестовая среда



Для проверки работоспособности различных функций в этой системе статистики пассажиропотока тест в этой главе выбран в крупном супермаркете ОКЕЙ. Мы развернули 3 фиксированные камеры по траектории пешеходов, входящих в супермаркет, для получения видео. План этажа супермаркета и расположение 3-х камер показаны на рисунке выше.

Междукамерное отслеживание пешеходов

Отслеживание людей под одной камерой

Следующие три изображения выбираются соответственно из изображений 300-го, 330-го, 360-го и 390-го кадра под камерой 3, а временной интервал выделения составляет около 1,5 секунды. На рисунке ниже в основном показано отслеживание человека 881.

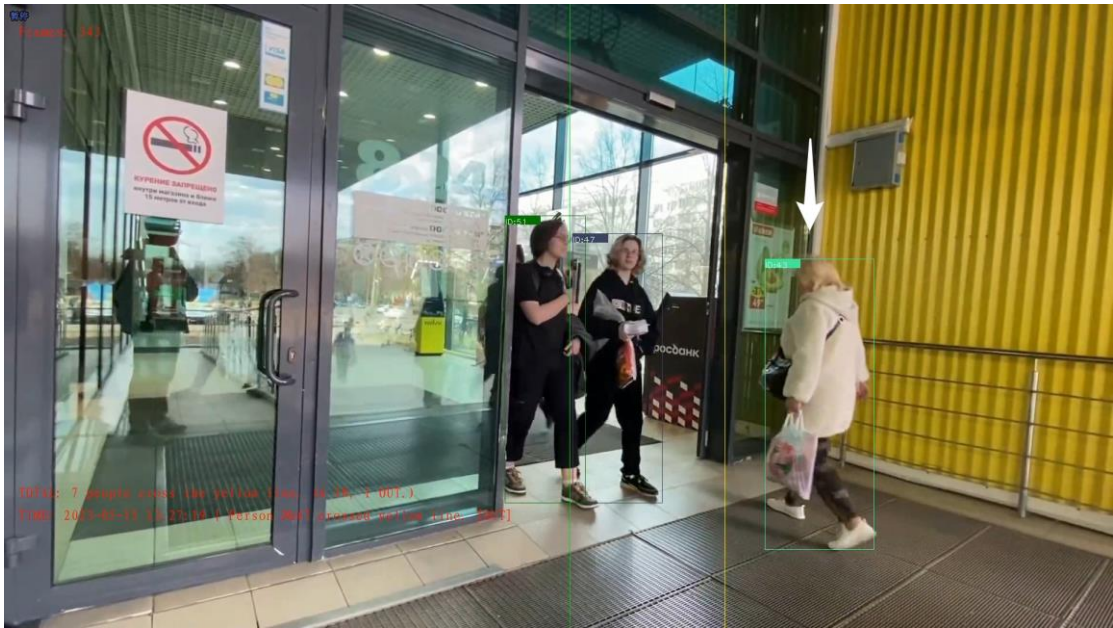


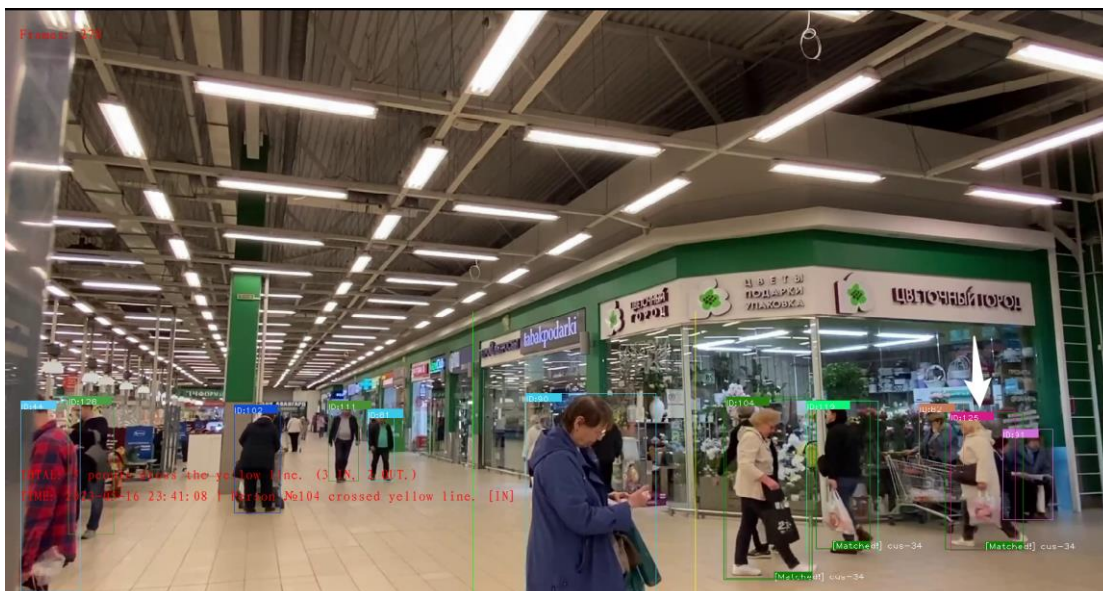


Видно, что детекция человека у № 881 всегда хорошая в пределах 100 кадров, а идентификатор всегда сохраняется без переключения. Даже если окклюзия возникает в третьем кадре, идентификатор пешехода остается равным 881, когда окклюзия исчезает в четвертом кадре. Это роль ассоциации признаков внешнего вида, предоставляемая fastreid. Кроме того, другие символы, отличные от 881, такие как 803, 819, 869 и т. Д., также хорошо отслеживаются.

Результаты испытаний доказывают, что система может непрерывно отслеживать нескольких пешеходов под одной камерой.

Отслеживание людей под несколькими камерами





На трех вышеприведенных снимках показаны результаты отслеживания пешехода cus-43, проходящего через три камеры в хронологическом порядке. Из результатов теста видно, что при первом появлении пешехода в зоне наблюдения камеры 1 система выдает ему идентификатор: cus-43, который является глобально уникальным. Когда пешеход cus-43 исчез из поля зрения камеры 1 и вошел в камеру 2, система извлекла черты его внешности через fastreid для сравнения, узнала идентификационную информацию пешехода cus-43 и восстановила отслеживание и повторно идентифицировала личность. Отображается информация в правом нижнем углу окна обнаружения. Точно так же, когда она вышла из камеры 2 и вошла в камеру 3, система также успешно идентифицировала ее в зоне наблюдения камеры 3.

Когда пешеходы проходят статистическую линию, в левом нижнем углу экрана печатаются подсказки времени прохождения, личности входящего персонала и общего количества входящих и выходящих.



Запись времени входа и выхода покупателей

PERSON-ID	CAMERA-ID	TIME	CAMERA-ID	TIME	CAMERA-ID	TIME
cus-1	cam-1	2023-05-10 03: 31: 27	cam-2	2023-05-10 03: 37: 59	cam-3	2023-05-10 03: 46: 04
cus-5	cam-1	2023-05-10 03: 31: 37	cam-2	2023-05-10 03: 50: 03		
cus-35	cam-1	2023-05-10 03: 32: 03	cam-2	2023-05-10 03: 38: 10	cam-3	2023-05-10 03: 46: 19
cus-71	cam-1	2023-05-10 03: 32: 11	cam-2	2023-05-10 03: 38: 15	cam-3	2023-05-10 03: 46: 23
cus-82	cam-1	2023-05-10 03: 32: 37	cam-2	2023-05-10 03: 38: 26	cam-3	2023-05-10 03: 46: 35
cus-94	cam-1	2023-05-10 03: 32: 48	cam-2	2023-05-10 03: 38: 32	cam-3	2023-05-10 03: 46: 42
cus-101	cam-1	2023-05-10 03: 32: 53	cam-2	2023-05-10 03: 41: 28		
cus-103	cam-1	2023-05-10 03: 32: 53	cam-2	2023-05-10 03: 41: 39	cam-3	2023-05-10 03: 50: 03
cus-108	cam-1	2023-05-10 03: 33: 00	cam-2	2023-05-10 03: 41: 02	cam-3	2023-05-10 03: 49: 20
cus-120	cam-1	2023-05-10 03: 33: 10	cam-2	2023-05-10 03: 42: 06	cam-3	2023-05-10 03: 50: 23
cus-134	cam-1	2023-05-10 03: 33: 14	cam-2	2023-05-10 03: 42: 30	cam-3	
cus-147	cam-1	2023-05-10 03: 33: 21	cam-2	2023-05-10 03: 42: 57	cam-3	2023-05-10 03: 51: 18
cus-203	cam-1	2023-05-10 03: 34: 15	cam-2	2023-05-10 03: 42: 59	cam-3	2023-05-10 03: 51: 16
cus-253	cam-1	2023-05-10 03: 35: 21	cam-2		cam-3	
cus-248	cam-1	2023-05-10 03: 35: 24	cam-2		cam-3	
cus-257	cam-1	2023-05-10 03: 35: 28	cam-2		cam-3	
cus-306	cam-1	2023-05-10 03: 36: 17	cam-2		cam-3	
cus-314	cam-1	2023-05-10 03: 36: 30	cam-2		cam-3	
cus-331	cam-1	2023-05-10 03: 36: 53	cam-2		cam-3	
cus-335	cam-1	2023-05-10 03: 36: 56	cam-2		cam-3	

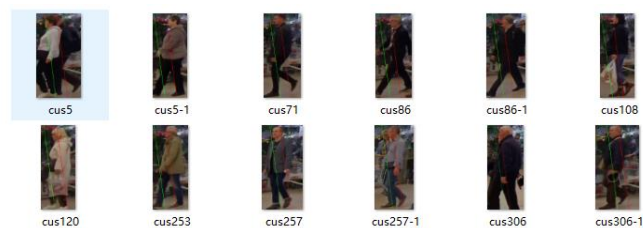
Мы используем структуру данных в Python для записи входных и выходных записей всех клиентов за определенный период времени. Распечатайте его, как показано выше.

Вырезанные изображения покупателей

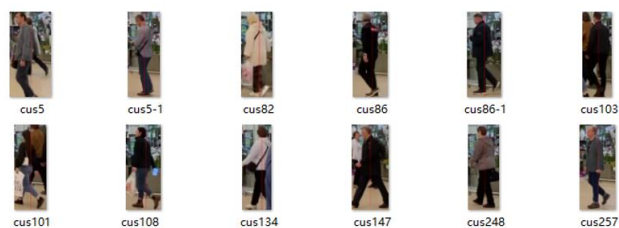
Вырезанные изображения покупателей из 1-ой камерой (частично):



Вырезанные изображения покупателей из 2-ой камерой (частично):



Вырезанные изображения покупателей из 3-ой камерой (частично):



Метрики оценки системы

Для оценки эффекта статистического метода в данной работе используются 3 камеры для выборки видео крупного супермаркета ОКЕЙ. Каждое из этих видео длится примерно 2 минуты. Результаты статистического теста представлены в таблице ниже:

Модуль подсчета входов и выходов пешеходов

Камера	真实值		统计值		Точность
	Направление	Количество посетителей	Направление	Количество посетителей	
Cam1	Вход	30	Вход	30	100.00%
	Выход	16	Выход	15	93.75%
	Всего	46	Всего	45	97.83%
Cam2	Вход	30	Вход	28	93.33%
	Выход	23	Выход	20	86.96%
	Всего	53	Всего	48	90.57%
Cam3	Вход	28	Вход	27	96.43%
	Выход	25	Выход	25	100.00%
	Всего	53	Всего	52	98.11%
总计				Вход	96.59%
				出	93.57%
				总共	95.50%

Эти данные получают путем подсчета потока людей двухстрочным методом, включающим реальные и статистические значения трех камер. Видно, что каждая камера имеет разную точность в направлении входа и выхода, но в целом средняя точность выше, 93,33% (вход), 93,57% (выход) и 93,59% (вход и выход всего).

При детальном анализе положения каждой камеры можно обнаружить, что точность обнаружения камеры 2 в направлении выхода низкая, всего 86,96%. Это связано с настройками камеры, окружающей средой и другими факторами. Потому что в in2 область для обнаружения уходящих людей меньше. Более того, в течение некоторого времени большое количество людей внезапно вошло и вышло из магазина одновременно (10 человек в течение 5 секунд), поэтому между персоналом произошло серьезное совпадение, и было 3 пропущенных инспекционных персонала, которые понизил среднюю точность.

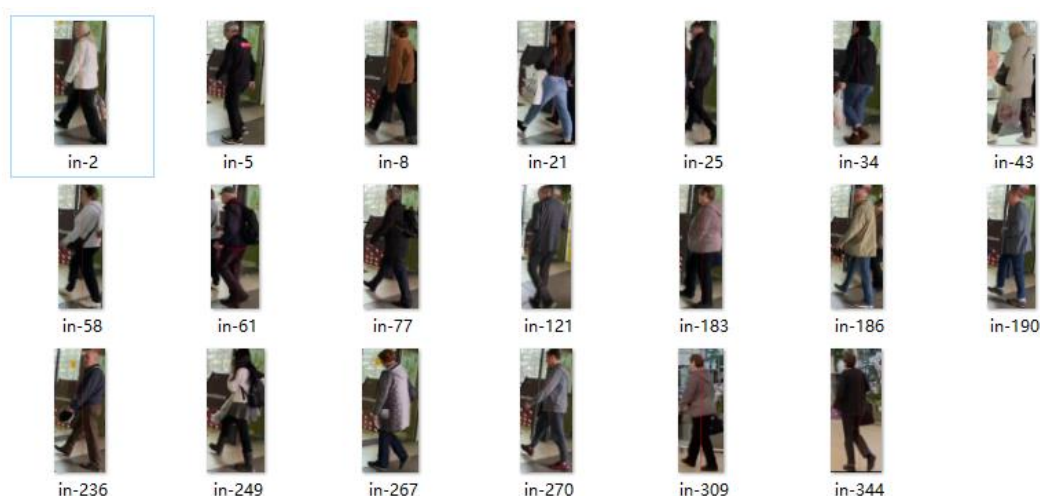
Точность кулачка 1 и кулачка 3 относительно высока, выше 97%. Это связано с тем, что в этих двух сценариях почти нет больших толп людей, входящих и выходящих из магазина.

В целом, несмотря на то, что еще есть возможности для повышения точности, общая производительность относительно стабильна и точна, что может помочь руководителям объектов лучше понимать поток людей.

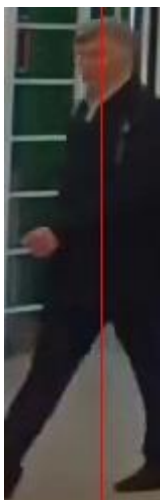
Модуль межкамерной Ре-идентификации пешеходов

Всего в этом раунде мы оценили 20 человек. Эти 20 человек идут от камеры 1 к камере 2 через камеру 2, то есть имеют полную траекторию, пересекающую камеру.

В cam1 20 человек, формируют 20 скриншотов клиентов, на данный момент в пуле фич 20 векторов внешности.



В cam2 по очереди прошло 20 человек, а скриншотов сформировалось всего 19, потому что было два пешехода, идущих параллельно и блокирующих друг друга, поэтому два кадра детектирования не могли быть правильно сформированы. Таким образом, происходит пропущенное обнаружение. Среди 19 изображений, подлежащих повторной идентификации, 18 человек получили правильную идентичность после повторной идентификации, а 1 человек не смог получить идентичность (внешнее сходство с исходной идентичностью было слишком низким, поэтому образец был отфильтрован как отрицательный). Следовательно, точность повторной идентификации по кулачку-2 составляет $18/20=90,0\%$.



В камере-3 последовательно прошли 20 человек, из которых сформировалось только 19 скриншотов, потому что рядом шли два пешехода, блокируя друг друга, и два кадра детектирования не могли быть корректно сформированы. Таким образом, происходит пропущенное обнаружение. Среди 19 изображений, подлежащих повторной идентификации, 19 получили правильную идентичность после повторной идентификации, а 0 получили неправильную идентичность. Следовательно, точность повторной идентификации по кулачку-3 составляет $19/20=95,0\%$.



Полный эксперимент приводит к следующей таблице:

ID чел	Cam-1	Cam-2	Cam-3
2	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
5	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
8	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Скрыт и не может быть повторно идентифицирован
21	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
25	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
34	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
43	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
58	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность

61	Успешно вошли в статистику	Скрыт и не может быть повторно идентифицирован	Успешно повторно идентифицирован и получил правильную личность
77	Успешно вошли в статистику	Ошибка повторной идентификации, получить неверную идентификацию	Успешно повторно идентифицирован и получил правильную личность
123	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
183	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
186	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
188	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
190	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
236	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
249	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
250	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
267	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность
270	Успешно вошли в статистику	Успешно повторно идентифицирован и получил правильную личность	Успешно повторно идентифицирован и получил правильную личность


И записали времени, когда они входят в магазин (виртуальное время в экспериментальной среде):

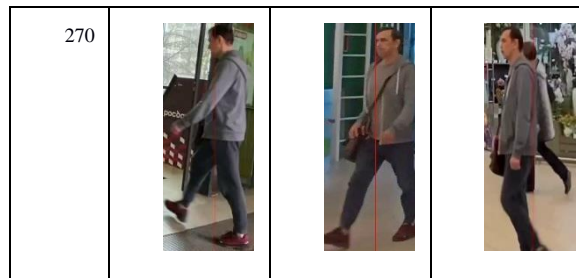
```

1  }, 'customer-2': {'cam-1': '2023-05-15 13: 26: 36', 'cam-2': '2023-05-15 13: 33: 20', 'cam-3': '2023-05-15 13: 40: 24'
2  }, 'customer-5': {'cam-1': '2023-05-15 13: 26: 45', 'cam-2': '2023-05-15 13: 33: 43', 'cam-3': '2023-05-15 13: 42: 01'
3  }, 'customer-8': {'cam-1': '2023-05-15 13: 26: 55', 'cam-2': '2023-05-15 13: 33: 31'
4  }, 'customer-21': {'cam-1': '2023-05-15 13: 26: 59', 'cam-2': '2023-05-15 13: 36: 31', 'cam-3': '2023-05-15 13: 45: 21'
5  }, 'customer-25': {'cam-1': '2023-05-15 13: 27: 00', 'cam-2': '2023-05-15 13: 33: 37', 'cam-3': '2023-05-15 13: 40: 38'
6  }, 'customer-34': {'cam-1': '2023-05-15 13: 27: 11', 'cam-2': '2023-05-15 13: 33: 47', 'cam-3': '2023-05-15 13: 40: 55'
7  }, 'customer-43': {'cam-1': '2023-05-15 13: 27: 25', 'cam-2': '2023-05-15 13: 33: 56', 'cam-3': '2023-05-15 13: 41: 01'
8  }, 'customer-58': {'cam-1': '2023-05-15 13: 27: 28', 'cam-2': '2023-05-15 13: 34: 10', 'cam-3': '2023-05-15 13: 44: 10'
9  }, 'customer-61': {'cam-1': '2023-05-15 13: 27: 30', 'cam-3': '2023-05-15 13: 41: 49'
10 }, 'customer-77': {'cam-1': '2023-05-15 13: 27: 39', 'cam-3': '2023-05-15 13: 41: 28'
11 }, 'customer-123': {'cam-1': '2023-05-15 13: 28: 39', 'cam-2': '2023-05-15 13: 39: 09', 'cam-3': '2023-05-15 13: 42: 44'
12 }, 'customer-183': {'cam-1': '2023-05-15 13: 30: 16', 'cam-2': '2023-05-15 13: 37: 35', 'cam-3': '2023-05-15 13: 45: 21'
13 }, 'customer-186': {'cam-1': '2023-05-15 13: 30: 19', 'cam-2': '2023-05-15 13: 37: 51', 'cam-3': '2023-05-15 13: 44: 04'
14 }, 'customer-188': {'cam-1': '2023-05-15 13: 30: 40', 'cam-2': '2023-05-15 13: 38: 04', 'cam-3': '2023-05-15 13: 43: 25'
15 }, 'customer-190': {'cam-1': '2023-05-15 13: 30: 26', 'cam-2': '2023-05-15 13: 37: 00', 'cam-3': '2023-05-15 13: 43: 09'
16 }, 'customer-236': {'cam-1': '2023-05-15 13: 31: 37', 'cam-2': '2023-05-15 13: 38: 30', 'cam-3': '2023-05-15 13: 44: 41'
17 }, 'customer-249': {'cam-1': '2023-05-15 13: 31: 56', 'cam-2': '2023-05-15 13: 39: 02', 'cam-3': '2023-05-15 13: 47: 47'
18 }, 'customer-267': {'cam-1': '2023-05-15 13: 32: 27', 'cam-2': '2023-05-15 13: 45: 59', 'cam-3': '2023-05-15 13: 48: 49'
19 }, 'customer-270': {'cam-1': '2023-05-15 13: 32: 32', 'cam-2': '2023-05-15 13: 45: 57', 'cam-3': '2023-05-15 13: 53: 03'
20 }
21 }

```

Всего было успешно повторно идентифицировано 17 человек из 20(частичная таблица):

ID	Cam-1	Cam-2	Cam-3
2			
5			
21			
25			



Экспериментальное явление: наблюдалось вышеописанное явление: 20 пешеходов прошли через камеру-1, камеру-2 и камеру-3 за определенный промежуток времени, 17 из них были полностью отслежены и имели записи с 3-х камер, и 2 пешехода прошли мимо. Другие Пешеходы закрыты и не могут быть обнаружены, а один человек не может получить первоначальную личность после повторной идентификации, поэтому у них есть только неполные записи времени под двумя камерами, которые не включены в полное отслеживание.

Экспериментальный вывод: точность отслеживания нашей системы статистики трафика достигает примерно $17/20=85,0\%$.

ГЛАВА 4 Анализ полученных результатов

Экспериментальная оценка модели fastreid на открытой базе данных

После завершения обучения модели мы решили протестировать производительность нашей модели на другом наборе данных Market1501 и DukeMTMC и сравнили ее с другими моделями ReID, появившимися в тот же период:

Methods	Market1501		DukeMTMC	
	Rank-1	mAP	Rank-1	mAP
IANet(IVPR'19)	94.4	83.1	87.1	73.4
Auto-ReID(ICCV'19)	94.5	85.1	-	-
OSNet(ICCV'19)	94.8	84.9	88.6	73.5
ABDNet(ICCV'19)	95.6	88.3	89.0	78.6
Circle Loss(CVPR'20)	96.1	87.4	89.0	79.6
ours	95.7	88.4	90.1	81.3

Согласно приведенным выше данным, мы видим, что модель FastReID хорошо работает как с наборами данных Market1501, так и с наборами данных DukeMTMC. Его

точность Rank-1 в наборе данных Market1501 составляет 95,7%, mAP — 88,4%, а его точность Rank-1 в наборе данных DukeMTMC — 90,1%, mAP — 81,3%.

По сравнению с другими алгоритмами FastReID показал очень хорошую производительность. Например, в наборе данных Market1501 точность Rank-1 FastReID и mAP превзошли такие алгоритмы, как IANet, Auto-ReID и OSNet соответственно; даже в наборе данных DukeMTMC его точность Rank-1 и mAP превзошли все другие алгоритмы. Это показывает, что FastReID обладает высокой надежностью и способностью к обобщению в задачах повторной идентификации личности.

В заключение, основываясь на этих оценочных метриках, мы можем считать FastReID эффективной моделью повторной идентификации личности, которую предполагается применять в практических сценариях.

Экспериментальная оценка улучшенной модели DeepSORT на открытой базе данных

Чтобы проверить эффективность улучшенного метода многоцелевого отслеживания DeepSORT на основе FastReID, мы провели сравнительный эксперимент на более сложном новом наборе данных MOT20.

Этот набор данных аннотирует 8 сложных видеопоследовательностей (4 для обучения и 4 для тестирования) в неограниченных условиях, снятых с помощью статических и движущихся камер. По сравнению с предыдущей серией наборов данных MOT Challenge, MOT20 фокусируется на сценах с плотным скоплением людей, а его видео могут охватывать до 246 человек в одном кадре. Таким образом, отслеживание становится более сложным. В нем сравнивается способность модели слежения за несколькими целями отслеживать небольшие цели.



Мы протестировали алгоритм многоцелевого отслеживания на четырех подмножествах набора данных MOT20 и сравнили результаты с результатами 10 лучших победителей всемирного конкурса MOT Challenge (в 2020 году). Результаты следующие:

Traker	MOTA ↑	IDF1↑	MT/% ↑	ML/% ↓	IDs ↓	FPS ↑
ByteTrack	67	70.2	47.7	21.2	680	17.7
OUTrack	65.4	65.1	49.5	13.3	2,885	5.1
UTM	64.4	65.9	65	8.7	2,592	22.4
RFTracker	62.4	53	48.7	15.5	3,804	0
SUSHI	61.6	71.6	47.6	19.2	1,053	5.3
TransCenter	61	49.8	48.4	15.5	4,493	1
MPTC	60.6	59.7	51.1	16.7	4,533	0.7
TMOH	60.1	61.2	46.7	17.8	2,342	0.6
OCSORT	59.9	67	38.5	26.6	554	27.6
MFI	59.3	59.1	41.1	17.3	191	0.5
ours	60.4	60.9	35.2	17.4	2021	8.6

Во-первых, следует отметить, что в наборе данных MOT20 производительность трекера измеряется такими показателями, как MOTA, IDF1, MT и ML. Среди них MOTA является основной оценочной метрикой для отслеживания нескольких объектов, которая сочетает в себе точность и полноту обнаружения, сопоставления и отслеживания. IDF1 - показатель, уделяющий больше внимания точности следящего устройства. MT и ML соответственно представляют собой долю количества правильно сопровождаемых целей и долю количества пропущенных сопровождаемых целей в общем количестве обработанных кадров.

Из данных видно, что лучше всего показал себя ByteTrack, за ним следуют OUTrack и UTM, а MOTA этих трех трекеров превысил 64. Другие трекары работали относительно плохо, но были и трекары, которые преуспели по определенным показателям. Например, SUSHI и OCSORT показывают более высокие результаты по IDF1 и MOTA, но хуже по ML, в то время как TMOH лучше по IDF1.

Наш трекаер имеет следующие сильные и слабые стороны по сравнению с другими 10 мировыми победителями:

преимущество:

1. Трекаер хорошо работает по индикатору IDF1 с оценкой 60,4, что указывает на то, что он в основном может выполнять многоцелевые задачи отслеживания.

2. Трекаер также относительно хорошо показывает FPS, достигая 8,6 кадров/с, что означает, что он может обрабатывать высокоскоростные динамические сцены.

недостаток:

1. По индикатору MOTA трекер набрал всего 60,4 балла, что указывает на то, что его производительность в наборе данных MOT20 относительно слабая.

2. Индекс ML/% относительно высок, 17,4, что показывает, что он не имеет хорошего контроля над пропорцией количества пропущенных целей к общему количеству.

3. Производительность этого трекера по двум индикаторам идентификаторов не так хороша, как у некоторых других трекеров. Идентификаторы 2021, а MT/% составляет 35,2, что показывает, что у него есть некоторые проблемы, такие как ложное обнаружение, пропущенные обнаружение и ложное отслеживание.

Хотя наш трекер хорошо работает по показателям FPS и IDF1, и имеет определенную производительность в реальном времени, он относительно плохо работает по отношению количества правильно сопровождаемых целей к общему количеству и отношению количества пропущенных сопровождаемых целей к общему числу. После анализа я думаю, что низкая производительность связана с алгоритмом обнаружения целей YOLO. Наш алгоритм обнаружения YOLO представляет собой одноэтапный алгоритм обнаружения целей, преимуществом которого является быстрая работа и экономия вычислительных ресурсов. Недостатки также весьма очевидны: точность позиционирования низкая, а обнаружение мелких целевых объектов и плотных объектов, таких как группа птиц и большая группа плотных толп, не является хорошей. В топ-10, кроме OCSORT, ByteTrack и UTM, все они используют двухэтапный алгоритм обнаружения целей, что обеспечивает высокое качество обнаружения целей и для мелких целей.

Поэтому я думаю, что если вы хотите улучшить способность модели отслеживать небольшие цели в будущем, вы можете начать с YOLO со следующими стратегиями:

- Настройте структуру модели: вы можете рассмотреть возможность использования более глубокой сетевой структуры, чтобы улучшить способность распознавания модели.
- Введение новой функции потерь: за счет введения функции Focal Loss вес мелких объектов, которые трудно классифицировать, увеличивается, поэтому модель уделяет больше внимания небольшим объектам, которые трудно классифицировать. Это может эффективно улучшить способность обнаружения небольших целей.
- Многомасштабное обнаружение: стратегия многомасштабного обнаружения может быть разработана для идентификации и обнаружения целей разных размеров и повышения точности обнаружения модели. Например, структура FPN (Feature Pyramid Networks) добавляется к шейке сети YOLO, а функции верхнего и нижнего уровня объединяются для улучшения способности сети обнаруживать цели разных масштабов.
- Добавить модуль внимания: добавьте механизм пространственного внимания и механизм внимания канала к сверточному слою YOLO, чтобы модель могла уделять больше внимания важным каналам функций в процессе свертки. Это может помочь модели лучше изучить характеристики мелких объектов и улучшить способность обнаружения.

ЗАКЛЮЧЕНИЕ

Список Литературы