УДК 004.855.5

ОПТИМИЗАЦИЯ НЕЙРОСЕТЕВОГО ДЕТЕКТОРА ДВИЖУЩИХСЯ ОБЪЕКТОВ

© **Д. В. Свитов**[1,2], **С. А. Алямкин**[1]

[1] *ООО «Экспасофт»,*
630090, г. Новосибирск, ул. Николаева, 11/1
[2] *Институт автоматики и электрометрии СО РАН,*
630090, г. Новосибирск, просп. Академика Коптюга, 1
E-mail: d.svitov@expasoft.tech

Свёрточные нейронные сети позволяют получать самые высокие значения точности детектирования объектов на изображениях. Но детекторы часто подвержены ложным срабатываниям. В ряде прикладных задач интерес представляет только детектирование движущихся объектов: лица человека, подходящего к домофону, или машины в дорожном трафике. В данной работе предлагается новый подход к детектированию — AmphibianDetector, позволяющий снизить число ложных срабатываний за счёт обработки только движущихся объектов. Такой подход заключается в модификации уже обученной на детекцию свёрточной нейронной сети и может быть применён для повышения точности имеющейся системы путём небольших изменений в ней. Эффективность предлагаемого подхода была продемонстрирована на открытом наборе данных CDNet2014 pedestrian.

Ключевые слова: свёрточные нейронные сети, детектирование, компьютерное зрение, глубокие нейронные сети.

DOI: 10.15372/AUT20210103

Введение. К настоящему моменту детекторы на основе нейронных сетей зарекомендовали себя как наиболее точный инструмент для обнаружения объектов в видеопотоке или на отдельном кадре [1–3]. В последнее время нейросетевые детекторы достигли достаточно высоких скоростей исполнения на маломощных системах [4–6]. Так, нейросетевые детекторы позволяют организовать отслеживание дорожного трафика или систему наружного наблюдения на основе одноплатного компьютера. Снижение нагрузки на вычислитель даёт возможность продлить срок его эксплуатации. В задачах, связанных с детектированием: будь то дорожный трафик или люди для системы наружного наблюдения — кажется нецелесообразным обрабатывать статичные кадры, так как состояние окружения не поменялось. В данной работе предлагается подход, позволяющий эффективно и с высокой точностью отфильтровывать статичные кадры из видеопотока, не добавляя дополнительной вычислительной нагрузки.

Ложные положительные срабатывания детектора могут иметь значительное негативное влияние на работу некоторых систем. Например, в системе распознавания лиц это может привести к недетерминированному поведению, когда система пытается обработать фрагмент фона как лицо. Часто в таких случаях применяют детектор движения и обрабатывают только те кадры, на которых есть движущиеся объекты [7, 8]. Как правило, такие подходы для статичной камеры основываются на построении модели фона и вычислении разницы обрабатываемого кадра с этой моделью. В предлагаемом подходе этапы построения устойчивой модели фона и детектирования объединяются. Для этого используются промежуточные карты признаков детектора как модель фона. Такой подход не требует дополнительных вычислений и даёт возможность получить устойчивую к шуму модель фона.

Для решаемой задачи отсеивания статичных кадров и для отфильтровывания ложных срабатываний детектора нет необходимости в сегментации движения с точностью до пикселя [9, 10], достаточно понимать наличие движения внутри области детекции. Это позволяет оптимизировать процедуру обнаружения движения рассмотрением решётки заданного масштаба, соответствующей карте признаков промежуточного слоя свёрточной нейронной сети (СНС).

Идея использования промежуточной карты признаков для сравнения кадров в видеопотоке близка к идее perceptual loss [11], широко применяемой для обучения генеративных нейронных сетей [12, 13]. В таких подходах для определения схожести генерируемой и целевой картинок берутся карты признаков от некоторой свёрточной сети, такой как VGG-19 [14]. В данной работе предлагается использовать промежуточные карты признаков той же сети детектора, что не увеличивает вычислительную нагрузку.

Биологическим стимулом для данной работы послужила система зрения земноводных. Когда земноводное неподвижно, количество собираемой им зрительной информации гораздо меньше, чем у млекопитающих. Амфибия замечает только движущиеся предметы. Аналогично предлагаемая в представленной работе модель машинного зрения реагирует только на движущиеся объекты, что даёт возможность снизить вычислительную нагрузку. Основной вклад данной работы заключается в том, что в ней предлагается применять промежуточные карты признаков детектора для сравнения кадров и таким образом снижать нагрузку на вычислитель и уменьшать число ложных срабатываний.

Подходы к детектированию.

Детектирование объектов — одна из основных задач машинного зрения. Наибольшее распространение получили модели на основе архитектур R-CNN [1–3], YOLO [4–6], SSD [15].

Faster R-CNN. Модели для детектирования на основе R-CNN используют генератор регионов, предположительно содержащих объект, и СНС для их классификации. Классификация применяется для разделения регионов на содержащие и не содержащие интересующие категории объектов. В [3] предложен подход Faster R-CNN, использующий алгоритм Region Proposal Network. Данная сеть применяет те же карты признаков, что и детектор, и это значительно удешевляет генерацию регионов.

You only look once. В [4] впервые предложен подход к выполнению детектирования одной нейронной сетью. Алгоритм YOLO разделяет входное изображение на решётку, где для каждой ячейки предсказываются ограничивающие объект рамки и вероятности обнаружения классов. Для предсказания координат рамки и вероятностей классов используются полносвязные слои. Такой подход не содержит этапа генерации гипотез и инкапсулирует всю логику внутри одной нейронной сети. Это позволило получить высокую скорость работы данной архитектуры. В YOLOv2 [5] полносвязный слой был заменён якорными ограничивающими рамками (anchor boxes). В YOLOv3 [6] использовалась более точная архитектура нейронной сети для извлечения признаков и было улучшено детектирование объектов разного размера.

SSD: Single Shot MultiBox Detector. В работе [15] предложен другой подход к детектированию объектов на изображении одной нейронной сетью. Алгоритм SSD генерирует шаблонные ограничивающие рамки с различным соотношением сторон и различными масштабами для карты признаков изображения. Затем нейронная сеть предсказывает вероятности обнаружения целевых классов в шаблонных ограничивающих рамках и уточняет их координаты. Основным отличием от YOLO является использование карт признаков с разных слоёв, что повышает точность детектирования объектов разного размера.

Детектирование движения. Задача отделения движущихся объектов от фона давно вызывает интерес у исследователей, занимающихся машинным зрением. Общий подход к обнаружению движения на изображении можно описать двумя основными шагами.

На первом шаге производится построение модели фона — статичной части изображения. На втором шаге обнаруживается движение как разница между текущим кадром и моделью фона. Классические подходы к обнаружению движения использовали в качестве модели фона опорный кадр. Широкое распространение получил алгоритм Gaussian Mixture Model [7]. Такой подход учитывал изменения в модели фона, связанные с динамическим фоном и условиями освещения. В [8] предложен подход SuBSENSE, который использует пространственно-временные бинарные признаки и цветовую информацию для обнаружения изменений, а в [16] — открытый набор тестовых данных для обнаружения движения с точностью до пикселя в различных условиях.

Повысить точность обнаружения движения в видеопотоке позволили методы глубокого машинного обучения. Считается, что в [17] впервые предложена СНС для построения модели фона. Для этого создана модель в градациях серого на основе N кадров изображения, а затем обучена зависящая от рассматриваемой сцены СНС модель сегментации движущихся объектов по модели фона и текущему кадру. Идея применения СНС для детектирования движения на изображении была развита в [9]. Авторы использовали SuBSENSE [8] и Flux Tensor [18] для построения модели фона, далее, подавая на вход СНС модель фона и текущий кадр, с точностью до пикселя обнаруживали движение. В [19] предложен подход FgSegNet, в основе которого лежит СНС на основе триплетов, применяющая различные масштабы изображения для кодирования. Различные масштабы позволили им учитывать контекст, в котором находится фрагмент изображения масштаба. Затем они с помощью декодировщика обнаруживали движение с точностью до пикселя. В работе [10] описан подход FgSegNet-v2. Данный метод расширял модуль объединения признаков FgSegNet введением функций признаков внутрь модели, что дало возможность извлекать признаки различного масштаба из изображения.

Недостатком существующих подходов на основе СНС является необходимость обучать дополнительную нейронную сеть, что увеличивает ресурсы для подготовки и развёртывания такой системы. Использование дополнительной СНС может негативно сказываться на времени обработки кадра. В данной работе предлагается подход, не требующий дополнительных вычислительных ресурсов для обнаружения движения, так как для сравнения кадров применяются карты признаков из промежуточного слоя сети детектора.

Объединение детекторов объектов и движения. Для повышения скорости работы детектора его могут объединять с детектором движения. При таком подходе кадры, не содержащие движущиеся объекты, не будут обрабатываться детектором. В [20] был предложен подход Noscope, который использует архитектурный поиск модели, оптимальной для каскада детекторов, обнаруживающих разницу в кадрах, и специализированных сетей для заданного видеопотока, целевого объекта и опорной нейронной сети. Применение в качестве детектора разницы кадров среднеквадратичной ошибки между опорным кадром и текущим позволило получить высокую скорость обработки видеопотока. Точность полученного каскада при этом совпадает с точностью опорной нейронной сети. Однако такой подход имеет недостатки. Во-первых, он требует переобучения системы для каждой камеры и сцены в отличие от рассматриваемых ранее методов детекции, таких как SSD, YOLO и R-CNN. Во-вторых, обнаружение движения на основе сравнения текущего и опорного кадров неустойчиво к динамическому фону и изменению освещения.

Другой подход к совмещению обнаружения движения и детектирования объектов представлен в методе ReMotENet [21]. Задача состояла в обнаружении движения объекта определённого класса по видеокадрам. Для этого анализируется видеозапись целиком с помощью трёхмерных свёрток, что является недостатком такого подхода, так как ограничивается его применимость для камер наружного наблюдения. Также ReMotENet отвечает только на вопрос о наличии движения объекта интересующего класса на видеозаписи, но не локализует его.

Предложенный подход использует карту признаков, полученную от промежуточных слоёв нейронной сети детектора. Это позволяет производить сравнение кадров на основе сложных иерархий признаков, найденных с помощью СНС. В случае отсутствия движения обработка кадра сетью останавливается на этом этапе.

Предлагаемый метод. В данной работе представлен подход к детектированию движущихся объектов в видеопотоке (рис. 1) — AmphibianDetector. Такой метод позволяет сократить среднее время обработки кадра видеопотока и снизить число ложных срабатываний. Ключевая идея этого метода — использование промежуточных карт признаков из нейронной сети детектора для определения регионов движения в кадре. При этом карта признаков промежуточного слоя рассматривается как набор векторов, где каждый вектор описывает соответствующую область на входном изображении. Сравнивая эти векторы для текущего кадра и векторы, представляющие модель фона, можно оценить изменения на различных участках изображения.

Обозначим через $F_1, \ldots, F_i$ последовательность кадров в видеопотоке, где F_i соответствует i-му кадру. В качестве модели детектирования объектов возьмём SSD [15] как наиболее распространённую архитектуру нейронной сети для обнаружения объектов, которая состоит из извлекателя признаков (feature extractor) из изображения и части сети, предсказывающей координаты ограничивающих рамок. Извлекатель признаков представляет собой свёрточную нейронную сеть такую, как MobileNetV2 [22] или Inception [23]. Она задаётся N свёрточными слоями $L_1, \ldots, L_N$. Для получения карты признаков из промежуточного слоя выбирается некоторое $m \in [1, N]$. В качестве карты признаков предлагается

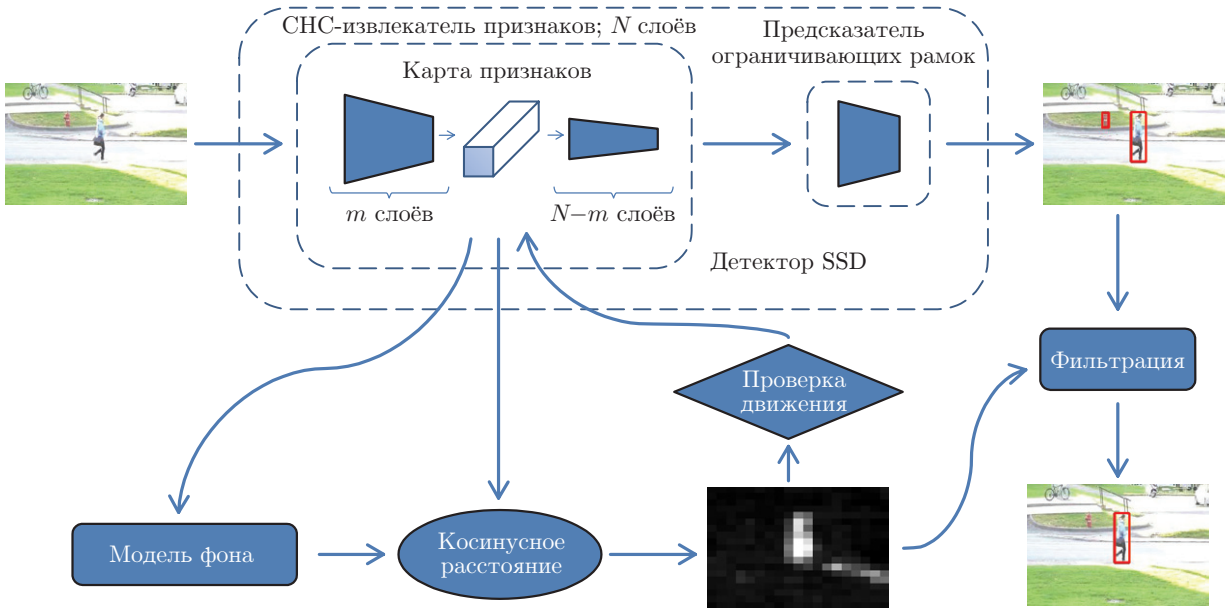

Рис. 1. Схема алгоритма. Входное изображение обрабатывается первыми m слоями извлекателя признаков, входящего в состав модели детектирования объектов. Полученные на этом этапе карты признаков используются для построения модели фона. Карта признаков для текущего кадра сравнивается с моделью фона по косинусной близости. Вычисленная после этого карта движений используется для проверки необходимости окончания выполнения детектирования. Карта движений также применяется для отфильтровывания ограничивающих рамок, чтобы отбраковать ложные срабатывания, соответствующие статичным объектам

использовать выход L_m слоя нейронной сети. Карту признаков для кадра i обозначим через M_i. Получаемая карта имеет размеры $H_m \times W_m \times C_m$, где H_m и W_m соответствуют высоте и ширине карты, C_m задаёт число каналов и может интерпретироваться как размерность вектора в каждой точке карты признаков. От выбора m зависит соответствие каждого вектора в карте некоторой области входного изображения, а также устойчивость векторного представления к артефактам входного изображения. Регулируя параметр m, можно добиваться нужного соотношения между скоростью и точностью работы детектора движений. Выбор оптимального m будет подробно рассмотрен далее.

Размер участка входного изображения, соответствующего одному вектору на карте признаков для заданного m, вычисляется следующим образом. Пусть $H_{in} \times W_{in} \times 3$ — размеры трёхканального входного изображения, где H_{in} и W_{in} — высота и ширина входного изображения соответственно. Тогда для m размеры входной области будут задаваться как $H_{in}/H_m \times W_{in}/W_m$.

Для обнаружения изменений в F_i-кадре построенная для него карта признаков M_i сравнивается с моделью фона M_{bg}, которая инициализируется опорным кадром M_{init}. К опорному кадру предъявляется требование, чтобы он не содержал движущихся объектов, так как их смещение будет вызывать ложные срабатывания.

Для сравнения модели фона M_{bg} и карты признаков кадра M_i вычисляется косинусная близость между представляющими их векторами. Для каждой ячейки решётки, соответствующей слою m с координатами $x \in [0, W_m]$, $y \in [0, H_m]$, косинусная близость вычисляется как

$$\mathrm{Diff}_i\,[x, y] = 1 - \frac{M_{bg}[x, y] \cdot M_i[x, y]}{\|M_{bg}[x, y]\|\,\|M_i[x, y]\|}, \tag{1}$$

где «$\cdot$» — скалярное произведение векторов. Значения карты движения $\mathrm{Diff}_i\,[x, y]$ тем ближе к 1, чем сильнее изменение на участке входного изображения F_i. Как отмечается в [24], использование косинусной близости повышает согласованность сравнения векторных представлений, обученных с помощью функции softmax [25], с евклидовым расстоянием.

Оптимизация скорости работы детектора производится за счёт остановки обработки кадров, на которых не обнаружено движение. Для отбраковки статичных кадров максимальное значение $\max_{x,y}(\mathrm{Diff}_i)$ сравнивается с порогом λ. Кадры F_i, для которых $\max_{x,y}(\mathrm{Diff}_i)$ меньше порога λ, служат для обновления модели фона.

Как отмечено в [9], динамическое обновление модели фона позволяет адаптироваться к изменениям условий освещённости сцены и погодных условий. В рассматриваемом подходе M_{bg} обновляется на основе M_i по следующей формуле:

$$M_{bg} = \begin{cases} M_{bg} \cdot \alpha + M_i \cdot (1 - \alpha), & \text{если } \max_{x,y}(\mathrm{Diff}_i) < \lambda, \\ M_{bg} & \text{иначе.} \end{cases} \tag{2}$$

Управляя параметром α, можно регулировать время нахождения объекта в неподвижном состоянии в кадре, чтобы стать частью модели фона. Для $\alpha = 0$ каждый кадр будет сравниваться с картой признаков предыдущего кадра. Случай для $\alpha = 1$ соответствует сравнению карты признаков i-го кадра с опорным. Для кадров с движущимися объектами после предсказания ограничивающих рамок производится дополнительное отфильтровывание на основе Diff_i. Координаты ограничивающих рамок предсказываются в нормированных значениях от 0 до 1 и затем масштабируются на размер входного изображения F_i ($H_{in} \times W_{in}$) и карту движения Diff_i ($H_m \times W_m$). Обозначим ограничивающие рамки для кадра F_i через $B_{i1}, \ldots, B_{iM}$. Каждая ограничивающая рамка B_{ij} задаётся через её координаты $[\mathrm{left}, \mathrm{top}, \mathrm{right}, \mathrm{bottom}]$, которые соответствуют краевым положениям x и y для

данной рамки на решётке слоя m. Тогда для ограничивающей рамки с индексом k координаты, соответствующие области на входном изображении, будут задаваться как

$$Bin_{ik} = [B_{ik.\text{left}} \cdot W_{in}, B_{ik.\text{top}} \cdot H_{in}, B_{ik.\text{right}} \cdot W_{in}, B_{ik.\text{bottom}} \cdot H_{in}]. \quad (3)$$

Для карты движений Diff_i

$$Bdiff_{ik} = [B_{ik.\text{left}} \cdot W_m, B_{ik.\text{top}} \cdot H_m, B_{ik.\text{right}} \cdot W_m, B_{ik.\text{bottom}} \cdot H_m]. \quad (4)$$

Величина Bin_{ik} удаляется из списка ограничивающих рамок, если $\text{mean}_{x,y}(\text{Diff}_i[\text{Bdiff}_{ik}])$ меньше порога λ. Здесь среднее вычисляется по Diff_i при варьировании x, y в области, заданной $Bdiff_{ik}$.

Предложенный метод позволяет получить ускорение обработки видеопотока за счёт остановки обработки статичных кадров, а также снизить число ложных срабатываний детектора для задач, в которых необходимо детектировать движущиеся объекты.

Эксперименты. В данной работе проведено сравнение предложенного подхода AmphibianDetector с базовым подходом к решению задачи отфильтровывания ложных срабатываний на статичные объекты. Базовый подход решения заключается в попиксельном сравнении кадров по L2 расстоянию при определении движения. Для вычисления модели фона в базовом подходе использовалось задание опорного кадра и обновление модели с коэффициентом α, как описано в предыдущем разделе. В экспериментах оценивались скорость и точность предложенного алгоритма. Для оценки скорости вычислялось среднее время обработки кадра для всех видеофайлов в рассматриваемом наборе данных. Точность оценивалась как метрика mean average precision (MAP) [26]. В качестве набора данных для оценки алгоритма рассматривался поднабор данных с пешеходами — pedestrian из CDNet2014 [16], который содержит 10 видеокадров с движением пешеходов при различном освещении и в различных погодных условиях. Также набор данных содержит видеокадры, снятые внутри и снаружи помещений: всего 26 248 кадров, в среднем 2624 кадра на видеозаписи.

Была проведена симуляция случаев, когда детектор осуществляет ложное срабатывание на статичный объект, схожий с объектом целевого класса. Для этого в исходные кадры было добавлено статичное изображение объекта, напоминающее человека (рис. 2).

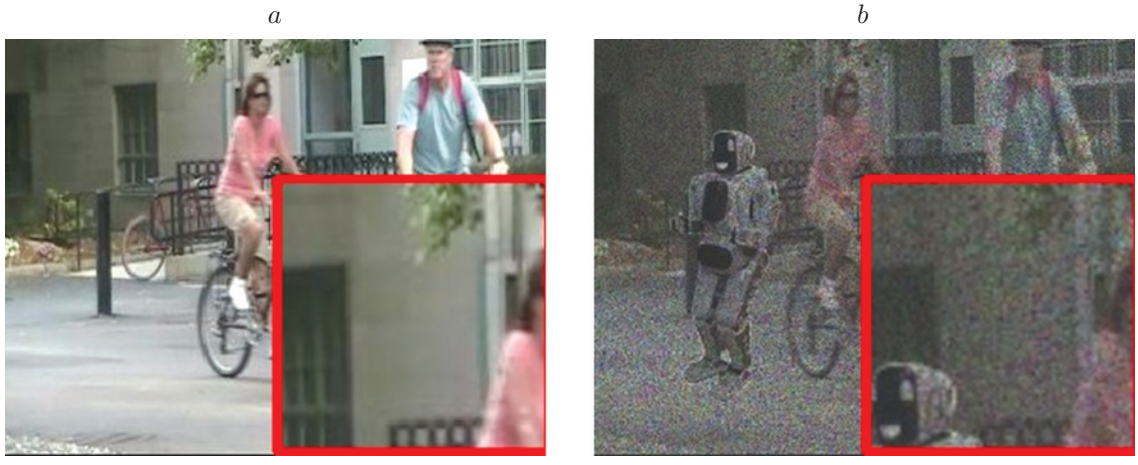

Рис. 2. Модификация видеокадров: *a* — пример исходного кадра из видеопотока; *b* — кадр из видеопотока после добавления статичного объекта, похожего на человека, и умножения на гауссовский шум ($\mu = 0{,}8$, $\sigma = 0{,}2$)

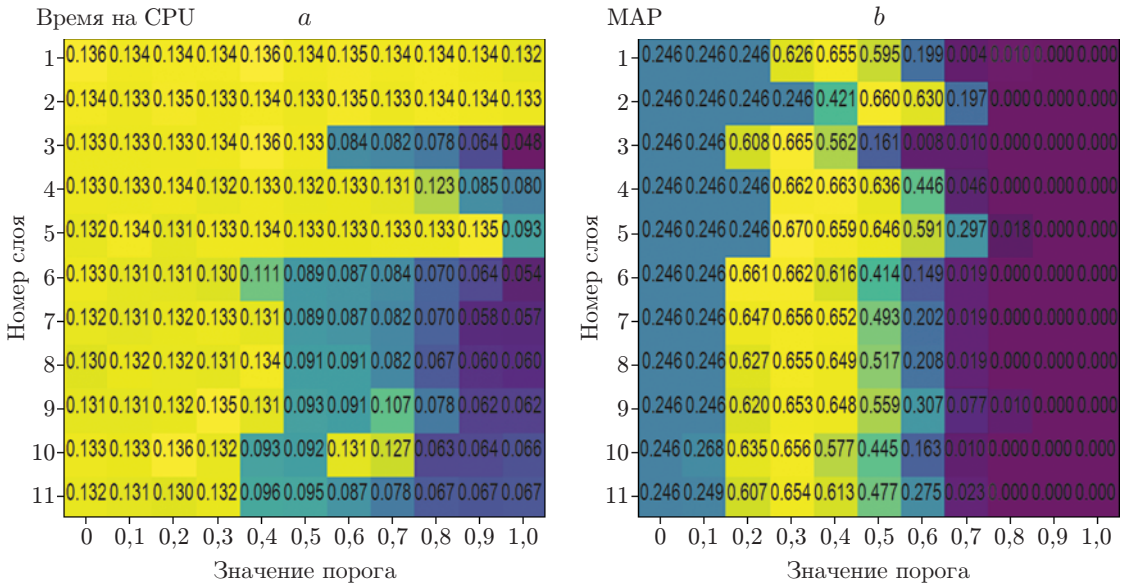

Рис. 3. Сравнение скорости и точности предложенного алгоритма: a — среднее время обработки кадра на Intel Core i5-4210U CPU 1,70GHz×4; b — MAP для различных значений номера слоя m и порога λ

Также видеокадры, получаемые с камер наружного наблюдения, подвержены зашумлению из-за качества матрицы захватывающего устройства и канала передачи видеопотока на сервер. Для того чтобы приблизить набор данных к реальному случаю, каждый кадр модифицировался зашумлением. Исходные кадры умножались на гауссовский шум с $\mu = 0{,}8$ и $\sigma = 0{,}2$, симулируя несовершенства устройства захвата видеокадров.

Для сравнения AmphibianDetector с базовым подходом эксперименты проводились на архитектуре SSD [22] с извлекателем признаков MobileNetV2 [22] со входом размером 300×300 пикселей. Выбор данной архитектуры обусловлен её популярностью для встраиваемых систем. В экспериментах использовалась сеть, обученная на наборе данных COCO [27]. Значения номера слоя m и порога λ выбирались поиском по решётке значений на одном видеопотоке из CDNet2014 pedestrian, состоящем из 2000 кадров. Порог λ варьировался от 0 до 1, так как это минимальное и максимальное значения для косинусной близости. Номер слоя m варьировался между 1 и 11 слоями извлекателя признаков. Из полученных значений (рис. 3) видно, что наибольшее значение MAP, равное 0,67, достигается при $m = 5$ и $\lambda = 0{,}3$. А наилучшее соотношение MAP к среднему времени обработки кадра достигается при параметрах $m = 11$ и $\lambda = 0{,}4$.

Для базового решения значение порога λ варьировалось от 0 до 2500 с шагом 500, так как для большего значения порога значение MAP уменьшалось до 0. Замеры AmphibianDetector с MobileNetV2+SSD проводились для номера слоя $m = 11$, показавшего наилучшее соотношение MAP и времени для поднабора из 2000 кадров. На полученных графиках (рис. 4) видно, что MAP для AmphibianDetector выше MAP для базового решения почти на 15 % при выборе лучшего порога для каждого подхода. Базовое решение достигает 0,405 MAP для порога, равного 500. Метод AmphibianDetector достигает 0,548 для порога, равного 0,4. Следует отметить, что в проведённых экспериментах максимально достижимый MAP ограничен точностью SSD+MobileNetV2 для рассматриваемых данных.

Предложенный подход AmphibianDetector позволяет снизить среднее время обработки кадра. Для вычисления среднего времени обработки кадра были выполнены замеры вре-

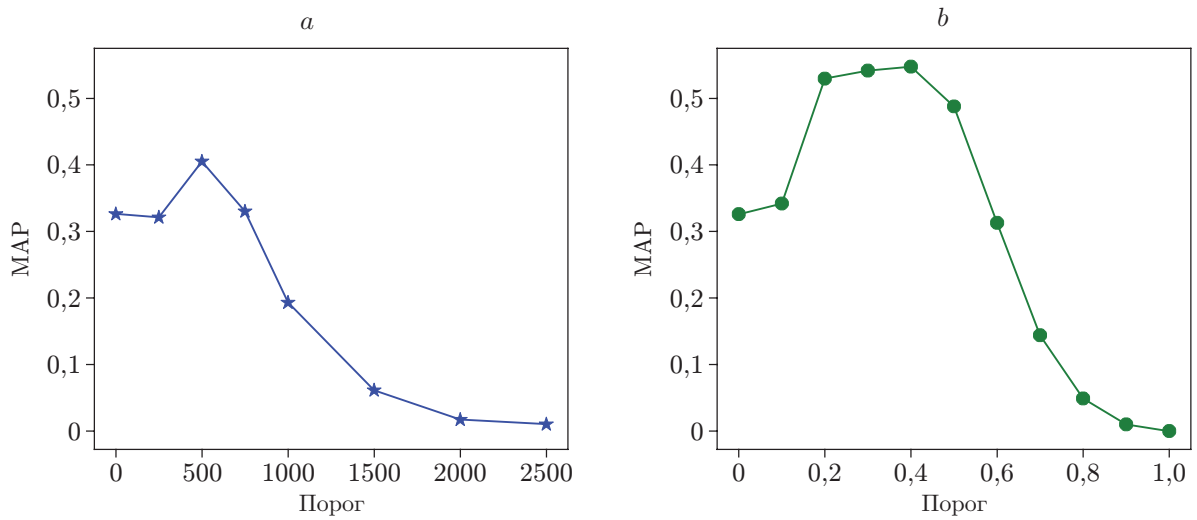

Рис. 4. Сравнение метрики MAP на основе SSD + MobileNetV2: *a* — для базового подхода, максимальное значение метрики достигается при пороге $\lambda = 500$; *b* — для AmphibianDetector, максимальное значение MAP на базе SSD + MobileNetV2 достигается при $\lambda = 0{,}4$

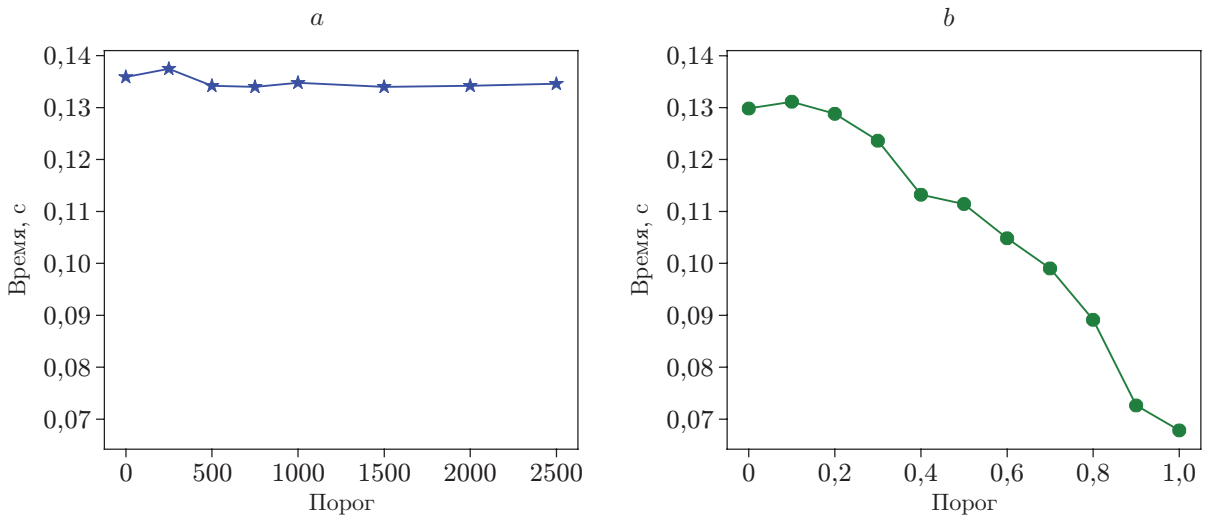

Рис. 5. Сравнение среднего времени обработки кадра из набора данных CDNet2014 pedestrian на Intel Core i5-4210U CPU 1,70GHz×4: *a* — для базового подхода; *b* — AmphibianDetector на основе SSD + MobileNetV2

мени на Intel Core i5-4210U CPU 1,70GHz × 4. Значение порога λ варьировалось в тех же диапазонах, что и замеры для MAP. Из графиков видно (рис. 5), что AmphibianDetector позволяет сократить среднее время на обработку кадра за счёт эффективного отфильтровывания кадров без движения, что достигается использованием для сравнения кадров карт признаков из промежуточного слоя сети.

Было проведено сравнение точности AmphibianDetector на основе SSD+MobileNetV2 со стандартным использованием SSD+MobileNetV2. Значения карт признаков брались из слоя $m = 11$. Пороговое значение λ задавалось равным 0,4. Для сравнения точности сопоставлялись значения MAP для стандартного использования детектора и AmphibianDetector. Подход SSD+MobileNetV2 достигает значения MAP, равного 0,326, применение AmphibianDetector позволяет увеличить MAP до 0,548.

Заключение. В представленной работе предложен подход к ускорению работы детектора за счёт отфильтровывания из видеопотока кадров, не содержащих движущихся объектов. Экспериментально показано, что такой подход уменьшает среднее время обработки кадра по сравнению с базовым подходом на основе попиксельного сравнения кадров. Он позволяет сократить число ложных срабатываний сети, что может быть критично в ряде прикладных задач, например распознавание лиц, где важно не допустить подачи некорректной области изображения в модель, производящую идентификацию. Исходный код с реализацией этого подхода и экспериментами доступен на GitHub [28].

СПИСОК ЛИТЕРАТУРЫ

1. **Girshick R., Donahue J., Darrell T., Malik J.** Rich feature hierarchies for accurate object detection and semantic segmentation // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, USA, 24–27 June 2014. P. 580–587.
2. **Girshick R.** Fast r-cnn // Proc. of the IEEE Intern. Conference on Computer Vision. Santiago, Chile, 11–18 Dec. 2015. P. 1440–1448.
3. **Ren S., He K., Girshick R., Sun J.** Faster r-cnn: Towards real-time object detection with region proposal networks // Proc. of the IEEE Intern. Conference on Advances in Neural Information Processing Systems. Montreal, Canada, 7–12 Dec. 2015. P. 91–99.
4. **Redmon J., Divvala S., Girshick R., Farhadi A.** You only look once: Unified, real-time object detection // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 26 June–1 July 2016. P. 779–788.
5. **Redmon J., Farhadi A.** Yolo9000: Better, faster, stronger // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, USA, 21–26 July 2017. P. 7263–7271.
6. **Redmon J., Farhadi A.** Yolov3: An incremental improvement // Computer Vision and Pattern Recognition. Cornell Univers., 2018. URL: arXiv preprint arXiv (дата обращения: 27.11.2020).
7. **Stauffer C., Grimson W. E. L.** Adaptive background mixture models for real-time tracking // Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Fort Collins, USA, 23–25 June 1999. Vol. 2. P. 246–252.
8. **St-Charles P.-L., Bilodeau G.-A., Bergevin R.** Subsense: A universal change detection method with local adaptive sensitivity // IEEE Transactions on Image Processing. 2014. **24**, N 1. P. 359–373.
9. **Babaee M., Dinh D. T., Rigoll G.** A deep convolutional neural network for video sequence background subtraction // Pattern Recognition. 2018. **76**. P. 635–649.
10. **Lim L. A., Keles H. Y.** Learning multi-scale features for foreground segmentation // Pattern Analysis and Applications. 2020. **23**, N 3. P. 1369–1380.
11. **Johnson J., Alahi A., Fei-Fei L.** Perceptual losses for real-time style transfer and super-resolution // European Conference on Computer Vision. Amsterdam, Netherlands, 8–16 Oct. 2016. P. 694–711.

12. **Wei Y., Gu S., Li Y., Jin L.** Unsupervised real-world image super resolution via domain-distance aware training // Computer Vision and Pattern Recognition. Cornell Univers., 2020. URL: arXiv:2004.01178 (дата обращения: 27.11.2020).
13. **Fritsche M., Gu S., Timofte R.** Frequency separation for real-world super-resolution // Proc. of the IEEE/CVF Intern. Conference on Computer Vision Workshop (ICCVW). Seoul, Korea (South), 27–28 Oct. 2019. P. 3599–3608.
14. **Simonyan K., Zisserman A.** Very deep convolutional networks for large-scale image recognition // Computer Vision and Pattern Recognition. Cornell Univers., 2014. URL: arXiv: 1409.1556 (дата обращения: 27.11.2020).
15. **Liu W., Anguelov D., Erhan D. et al.** Ssd: Single shot multibox detector // Proc. of the Europ. Conference on Computer Vision. Gewerbestrasse, Switzerland, 8–16 Oct. 2016. P. 21–37.
16. **Wang Y., Jodoin P.-M., Porikli F. et al.** Cdnet 2014: An expanded change detection benchmark dataset // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Columbus, USA, 24–27 June 2014. P. 387–394.
17. **Braham M., Van Droogenbroeck M.** Deep background subtraction with scene-specific convolutional neural networks // Proc. of the Intern. Conference on Systems, Signals and Image Processing (IWSSIP). Bratislava, Slovakia, 2–4 June 2016. P. 1–4.
18. **Wang R., Bunyak F., Seetharaman G., Palaniappan K.** Static and moving object detection using flux tensor with split gaussian models // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. Columbus, USA, 24–27 June 2014. P. 414–418.
19. **Lim L. A., Keles H. Y.** Foreground segmentation using convolutional neural networks for multiscale feature encoding // Pattern Recognition Lett. 2018. **112**. P. 256–262.
20. **Kang D., Emmons J., Abuzaid F. et al.** Noscope: Optimizing neural network queries over video at scale // Computer Vision and Pattern Recognition. Cornell Univers., 2017. URL: arXiv:1703.02529 (дата обращения: 27.11.2020).
21. **Yu R., Wang H., Davis L. S.** Remotenet: Efficient relevant motion event detection for large-scale home surveillance videos // Proc. of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). Lake Tahoe, USA, 12–14 March 2018. P. 1642–1651.
22. **Sandler M., Howard A., Zhu M. et al.** Mobilenetv2: Inverted residuals and linear bottlenecks // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA, 18–22 June 2018. P. 4510–4520.
23. **Szegedy C., Liu W., Jia Y. et al.** Going deeper with convolutions // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 7–12 June 2015. P. 1–9.
24. **Wang H., Wang Y., Zhou Z. et al.** Cosface: Large margin cosine loss for deep face recognition // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA, 18–22 June 2018. P. 5265–5274.
25. **Goodfellow I., Bengio Y., Courville A., Bengio Y.** 6.2.2.3 softmax units for multinoulli output distributions // Deep Learning. 2016. **1**. P. 180–184.
26. **Everingham M., Van Gool L., Williams C. K. et al.** The pascal visual object classes (voc) challenge // Intern. Journ. Computer Vision. 2010. **88**, N 2. P. 303–338.
27. **Lin T.-Y., Maire M., Belongie S. et al.** Microsoft coco: Common objects in context // Proc. of the Europ. Conference on Computer Vision. Zurich, Switzerland, June 2014. P. 740–755.
28. **GitHub.** David-svitov/AmphibianDetector. GitHub Inc., 2021. URL: https://github.com/david-svitov/AmphibianDetector (дата обращения: 27.11.2020).

Поступила в редакцию 27.11.2020
После доработки 21.12.2020
Принята к публикации 21.12.2020