Fast Tracking-by-Detection of Bus Passengers with Siamese CNNs
Claire Labit-Bonis, Jérôme Thomas, Frédéric Lerasle, Francisco Madrigal

To cite this version:
Claire Labit-Bonis, Jérôme Thomas, Frédéric Lerasle, Francisco Madrigal. Fast Tracking-by-Detection of Bus Passengers with Siamese CNNs. 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Sep 2019, Taipei, Taiwan. pp.1-8, 10.1109/AVSS.2019.8909843 . hal-02401951

HAL Id: hal-02401951
https://hal.laas.fr/hal-02401951
Submitted on 10 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast Tracking-by-Detection of Bus Passengers with Siamese CNNs
Claire Labit-Bonis, Jérôme Thomas, Frédéric Lerasle, Francisco Madrigal

To cite this version:
Claire Labit-Bonis, Jérôme Thomas, Frédéric Lerasle, Francisco Madrigal. Fast Tracking-by-Detection of Bus Passengers with Siamese CNNs. 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Sep 2019, Taipei, Taiwan. pp.1-8, 10.1109/AVSS.2019.8909843 . hal-02401951

HAL Id: hal-02401951
https://hal.laas.fr/hal-02401951
Submitted on 10 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast tracking-by-detection of bus passengers with Siamese CNNs

Claire Labit-Bonis
LAAS-CNRS
ACTIA Automotive
Toulouse, France
clabitbo@laas.fr

Jérôme Thomas
ACTIA Automotive
Toulouse, France
jerome.thomas@actia.fr

Frédéric Lerasle
LAAS-CNRS
Université UPS
Toulouse, France
lerasle@laas.fr

Francisco Madrigal
LAAS-CNRS
Toulouse, France
jfmadrig@laas.fr

Abstract

Knowing the exact number of passengers among the city bus fleets allows public transport operators to optimally distribute their vehicles into the traffic. However, interpreting overcrowded scenarios, at rush hour, with day/night illumination changes can be tricky. Based on the visual tracking-by-detection paradigm, we benefit from video stream information provided by cameras placed above doors to infer people trajectories and deduce the number of enterings/leavings at every bus stop. In this way a person detector estimates the location of the passengers in each image, a tracker matches detections between successive frames based on different cues such as appearance or motion, and infers trajectories over time. This paper proposes a fast and embeddable framework that performs person detection using relevant state-of-the-art CNN detectors, and couple the best one (in our applicative context) with a newly designed Siamese network for real-time tracking/data association purposes. Evaluations on our own large scale in-situ dataset are very promising in terms of performances and real-time constraint expected for on-board processing.

1. Introduction

Multiple-people tracking-by-detection methods are widely investigated in the literature and find many applications. Within public transport, more specifically in city buses, we can study bus lines occupancy rate by counting passengers and help the operator to optimally distribute its fleet of vehicles. Counting passengers independently from the ticket validation systems can also provide good estimates for frauds. Several market studies show real needs for such systems in public transport [14]. Hence, there are two main challenges: (i) dealing with highly cluttered scenes during rush hour and relatively sparse at off-peak hour both with illumination changes, and (ii) providing online, real-time and high accuracy performances using embedded hardware with limited CPU resources.

Several bus passenger counting systems already exist, e.g. electronic ticket validators or floor-located sensors; we focus on passenger detection based on embedded exteroceptive sensors placed above the doors. Many sensor technologies showed good performance, e.g. radar [9], active camera [19], or laser scanner [3]. Their cost and recent advances in computer vision motivated our choice to use conventional and cheap perspective cameras for our application.

With a full view of inside the bus, counting could be summed up as the detection of all passengers at once when the bus leaves a stop. In this setting, too many cameras are needed to cover the whole vehicle which can be from 10 to 18m long. Then, we must settle for fewer sensors placed above each door: estimating the number of people going in/out at each stop, in each view, will give insight into the global passenger flow. Figure 1 provides examples of such views, taken from our dataset.

To overcome frame-by-frame detector limitations, we
perform this task through tracking-by-detection. Dealing with overcrowded scenes, thus lots of trajectories to manage at the same time, becomes one of the main challenges.

Counting passengers using tracking-by-detection implies three steps: (i) detect people on a frame-by-frame basis, (ii) associate detections between frames to reconstruct consistent trajectories, (iii) analyse trajectories directions to determine whether passengers went indoors/outdoors, and finally count them. We focus on the second step, i.e. associating detections in the video stream using the outputs of the best Convolutional Neural Network (CNN) based detector prototyped for our application.

Multiple-people tracking most often deals with front views of pedestrians [17, 6]. Due to buses narrowness, taking such videos from inside the vehicle would lead to drastic occlusions and thus bad performances. This drawback can be avoided with zenithal viewpoint cameras placed above the doors. It then differs from classic applications, e.g. for pedestrian detection/tracking, as it can be seen in plethora of public benchmarks/challenges in the literature.

CNN offline learning based methods have proved to be effective both in terms of object detection and data association, but require specific training on goal-related datasets. No public labelled ones exist in the context of public transport and zenithal indoor point of view, therefore no comparison to public benchmarks can be led and specific trainings have to be performed. To this end, we acquired and labelled our own videos on a sufficiently long period of time to get all the variability, clutter and illumination changes we need to be challenged with: acquisitions were done and monitored in a city center bus during daylight and night hours, with a few as well as huge amount of passengers.

Given that the application has to run in real time – or at least between two bus stops, we must meet the requirement of on-board processing with low computing capacity hardware, close to Jetson TX2-like configuration. It is then necessary to use compact and high-speed neural networks while ensuring enough performances on the tasks to perform, i.e. detection along with tracking.

As a recap, passenger counting in city buses implies dealing with complex rush hour cluttered scenes involving the management of many trajectories at the same time, processed on platforms with limited CPU resources. Handling such constraints, especially with the use of greedy, yet very effective deep learning methods, requires the optimization of the neural networks used for object detection as well as data association. In this context, our contributions relate to the design of nearly real-time embedded tracking-by-detection within our applicative context of public transport and zenithal view, thanks to:

- the comparison of state-of-the-art CNN detectors,
- the prototyping of a fast siamese CNN for detection association in the video stream in order to address real-time constraints: tracking a huge number of targets in highly cluttered scenes with deep learning methods remains a major challenge,
- the evaluation of such techniques on our large scale bus passengers dataset, showing state-of-the-art tracking performance along with higher frame rates.

2. Related work

2.1. Deep object detection

Visual object detection consists in locating all instances of particular classes in an image. It has been extremely well-studied in the literature and deep learning based methods outperform by far classic approaches using handcrafted feature extraction. Among them, two types of methods stand out from the rest:

(i) **Region-based** methods generate and classify candidate areas in the image. The best known detectors of this kind result from successive improvements of R-CNN [7], which extracts \( \approx 2k \) region proposals from the original image, applies a CNN on each region to get 4096-d feature vectors, and jointly performs SVM classification and bounding box regression relatively to each region. Faster R-CNN [22] inverts R-CNN feature extraction and region proposal, and replaces the latter with a CNN that predicts regions to pass along to the classification/regression part. From R-CNN to Faster R-CNN, test-time speed jumps from 49 to 0.2s per image and gains in precision, making it suitable for nearly real-time applications.

(ii) **One-shot** methods do not rely on region classification but only process the whole image once. Since its second version, YOLO [20, 21] passes the original image into a CNN, reducing input dimensions from \((N \times N \times 3)\) to a \((S \times S \times (B \times 5 + C))\) output volume. Each cell of the \((S \times S)\) grid is thus described by \((B \times 5 + C)\) channels, i.e. \(B\) bounding boxes \(*5\) parameters each: \([x, y, w, h]\) coordinates relative to the bounds of the \(B\) designed-by-hand cell anchor boxes, confidence score for the presence of an object, and the probability distribution over the \(C\) classes to predict. SSD [16] also provides an end-to-end detection framework taking the whole image as input. It applies a CNN as feature extractor and adds several layers to softly reduce dimensions. Each added layer feeds two more ones: (i) a bounding box prediction layer which outputs a \((S \times S \times (B \times 4))\) volume and (ii) a class prediction layer which outputs a \((S \times S \times (B \times C))\) volume. Both detectors seem appropriate choices for
real-time applications as they can respectively run at 81 and 46fps for \( \approx (300 \times 300) \) input images, with their initially designed feature extractors. A comparative study \cite{12} demonstrates the benefits of replacing SSD feature extractor with a newer and more powerful MobileNet \cite{11} architecture to take a leap forward in terms of speed, at lower detection performance costs.

Faster R-CNN, YOLO and SSD show great ability to detect people on the Pascal VOC \cite{5} dataset with 79.6, 79.4 and 81.3\% of average precision for the person class respectively, depending on the base network used for feature extraction. Such performances and execution speeds legitimize their comparison in our applicative context.

### 2.2. Detection association and similarity function

Matching targets over successive frames is known as data association. It is commonly represented as a graph optimization problem \cite{1, 31} where each node is a detection and each edge is a potential link between them. Solving the assignment problem consists in minimizing the total association cost, i.e. the sum of all the edges of associated nodes, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.

In the domain of pedestrian similarity matching and re-identification, learning such features with convolutional based methods outperformed handcrafted ones \cite{15, 28} and is now widely used. Because it offers good performance and an online, real-time approach for multi-object tracking (MOT) based on CNN embedding for similarity learning, the work of Wojke et al. is in complete agreement with our applicative context. They propose a cosine metric learning framework \cite{28}, trained on a re-identification dataset and integrate it in their tracker \cite{29} to compute distinctive features for each new detection. More specifically, they train a network with as many output neurons as there are distinct identities, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.

In the domain of pedestrian similarity matching and re-identification, learning such features with convolutional based methods outperformed handcrafted ones \cite{15, 28} and is now widely used. Because it offers good performance and an online, real-time approach for multi-object tracking (MOT) based on CNN embedding for similarity learning, the work of Wojke et al. is in complete agreement with our applicative context. They propose a cosine metric learning framework \cite{28}, trained on a re-identification dataset and integrate it in their tracker \cite{29} to compute distinctive features for each new detection. More specifically, they train a network with as many output neurons as there are distinct identities, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.

In the domain of pedestrian similarity matching and re-identification, learning such features with convolutional based methods outperformed handcrafted ones \cite{15, 28} and is now widely used. Because it offers good performance and an online, real-time approach for multi-object tracking (MOT) based on CNN embedding for similarity learning, the work of Wojke et al. is in complete agreement with our applicative context. They propose a cosine metric learning framework \cite{28}, trained on a re-identification dataset and integrate it in their tracker \cite{29} to compute distinctive features for each new detection. More specifically, they train a network with as many output neurons as there are distinct identities, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.

In the domain of pedestrian similarity matching and re-identification, learning such features with convolutional based methods outperformed handcrafted ones \cite{15, 28} and is now widely used. Because it offers good performance and an online, real-time approach for multi-object tracking (MOT) based on CNN embedding for similarity learning, the work of Wojke et al. is in complete agreement with our applicative context. They propose a cosine metric learning framework \cite{28}, trained on a re-identification dataset and integrate it in their tracker \cite{29} to compute distinctive features for each new detection. More specifically, they train a network with as many output neurons as there are distinct identities, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.

In the domain of pedestrian similarity matching and re-identification, learning such features with convolutional based methods outperformed handcrafted ones \cite{15, 28} and is now widely used. Because it offers good performance and an online, real-time approach for multi-object tracking (MOT) based on CNN embedding for similarity learning, the work of Wojke et al. is in complete agreement with our applicative context. They propose a cosine metric learning framework \cite{28}, trained on a re-identification dataset and integrate it in their tracker \cite{29} to compute distinctive features for each new detection. More specifically, they train a network with as many output neurons as there are distinct identities, which can be done with classic approaches like the Hungarian algorithm \cite{13}. The challenge is to define a similarity function able to generate discriminative representations, in order to have the lowest cost value between two similar detections and the highest value between two different ones.
Table 1: Training / validation sets are used for training. We present our results on two test sets: sparsely ("min_clutter") and highly ("max_clutter") populated scenes, containing from 0 to ≈ 10 people at the same time. IDs indicates the number of distinct targets, Clutter describes the scene congestion (Weak < Medium < High < Strong) – s indicates that the bus seats are also occupied. Illum. describes the illumination variation of the scene (hour of the day, with/without artificial lighting), Variability enumerates few cases taken from the dataset.

<table>
<thead>
<tr>
<th>Name</th>
<th>Images</th>
<th>IDs</th>
<th>Clutter</th>
<th>Illum.</th>
<th>Variability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>video_1</td>
<td>15,382</td>
<td>46</td>
<td>Medium (max 4)</td>
<td>Daylight</td>
<td></td>
</tr>
<tr>
<td>video_2</td>
<td>18,427</td>
<td>79</td>
<td>Medium (max 3 + s)</td>
<td>Late + artificial</td>
<td></td>
</tr>
<tr>
<td>video_3</td>
<td>29,889</td>
<td>95</td>
<td>High (max 6 + s)</td>
<td>Daylight</td>
<td></td>
</tr>
<tr>
<td>Val.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>video_4</td>
<td>9,751</td>
<td>44</td>
<td>Weak (max 3)</td>
<td>Daylight</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min_clutter</td>
<td>11,576</td>
<td>37</td>
<td>Weak (max 2)</td>
<td>Night artificial</td>
<td></td>
</tr>
<tr>
<td>max_clutter</td>
<td>20,353</td>
<td>43</td>
<td>Strong (≈ 10 + s)</td>
<td>Night artificial</td>
<td></td>
</tr>
<tr>
<td>Total (≈ 1h)</td>
<td>105,378</td>
<td>345</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: MobileNetv1 consists in stacking a convolutional block followed by multiple depthwise blocks. The purple "BatchNorm" block was removed in our modified version.

3.3. Detection association with siamese CNNs

Our goal is to propose a study of siamese networks addressing detection association for MOT purposes within highly cluttered scenes, while remaining consistent with our real-time constraint. Our contribution is two-fold: we compare (i) two different feature extraction structures combined with (ii) two approaches to produce the feature map em-
Table 2: Modified version of MobileNetv1 feature extractor. \(\alpha\), is the width multiplier factor from the original paper [11], which increases/decreases the number of filters for each layer. Different values were tested and 0.66 appeared to be a reasonable speed/accuracy trade-off on our dataset.

<table>
<thead>
<tr>
<th>Layer type</th>
<th># filters/stride</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input image</td>
<td>-</td>
<td>(360 \times 480 \times 3)</td>
</tr>
<tr>
<td>Conv block</td>
<td>(\alpha \cdot 2)</td>
<td>(180 \times 240 \times 21)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 64)</td>
<td>(180 \times 240 \times 42)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 128)</td>
<td>(90 \times 120 \times 84)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 128)</td>
<td>(90 \times 120 \times 84)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 256)</td>
<td>(45 \times 60 \times 168)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 256)</td>
<td>(45 \times 60 \times 168)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 512)</td>
<td>(23 \times 30 \times 337)</td>
</tr>
<tr>
<td>5 x Dw conv block</td>
<td>(\alpha \cdot 512)</td>
<td>(23 \times 30 \times 337)</td>
</tr>
<tr>
<td>Dw conv block</td>
<td>(\alpha \cdot 1024)</td>
<td>(23 \times 30 \times 675)</td>
</tr>
</tbody>
</table>

beddings. Therefore, we propose a fast siamese framework capable of computing discriminative representations with state-of-the-art performance and less computational time.

Based on the work of Wojke et al. [29], we re-train their re-identification network for similarity learning purposes and compare it against a faster MobileNetv1-based network. In the paper, Wojke et al. apply convolutions on an input image, flatten the last (11 × 11 × 128) feature map, and apply a fully connected layer trained for multi-classification, where the number of outputs is the number of distinct identities to classify. We cut the network just before the flattening layer, thus retrieve the last (11 × 11 × 128) output volume. Similarly, we cut MobileNetv1 structure after the fifth depthwise convolutional block in order to obtain a (10 × 10 × 128) volume. For speed purposes, we also compare two MobileNet1 versions, i.e. with/without BN layers in depthwise blocks (cf. section 3.2). Left side of figure 4 illustrates this feature extractors comparison.

In the original contrastive loss paper [4], the distance \(d\) from equation 1 is the euclidean distance between vectors, but other related works show interest in learning a cosine similarity measure in the context of visual object tracking [27][13]. similarly, DeepSORT computes the cost matrix of data association using the cosine distance between vectors. Based on cosine similarity, we use the angular distance between vectors as our distance \(d\): two closely related vectors in the embedding space should have an angular distance near 0, 1 otherwise. The angular distance between two normalized vectors is retrieved from their scalar product.

In order to reduce dimensions of this configuration into a single feature vector, we compare two techniques: (i) we flatten the feature map generated by the feature extractor, e.g. going from a (10 × 10 × 128) feature map in the case of MobileNet to a single 12800-d feature vector, and apply a fully connected layer to project this vector into the 128-d embedding space; (ii) we drastically reduce the number of parameters and execution speed of the first technique by replacing fully connected layers with global average pooling and 1x1 convolutions after the feature extractor last feature map, as described in figure 3.

During training, the siamese CNNs are fed with pairs of \((85 \times 85 \times 3)\) image patches – the average size of a person in our training set. These patches are the resized outputs of SSD on the training set. Based on the overlap between patches and ground truth, we can determine the real identity label of a detection and thus construct positive/negative pairs as inputs to the siamese network. Training the latter with real detections instead of ground truth patches allows the network to see realistic and not perfectly centered cases. To equally optimize the distance between positive/negative pairs, it is necessary to produce them in equal amount. Thus, as a lot of frames in our training set do not necessarily contain two or more distinct targets, constructing positive/negative pairs for siamese learning has to be done artificially. Positive pairs are generated within a temporal horizon of \([0-300]\) frames between first and second detection, with \(X \sim N(\mu = 1, \sigma^2 = 60)\). In this way, we simulate non-detections and appearance changes. For each positive pair, a negative one is produced: the first patch is the same as in the positive pair, the second patch is randomly picked among all other identities. 5% of the time, we pick a randomly selected SSD false positive output as the second negative patch of the pair. Figure 4 shows examples of positive and negative pairs used during training.

4. Evaluations and discussion

Hereafter, we justify the choice of SSD along with a customized MobileNetv1 feature extractor as input to data association, and demonstrate performance in line with the literature, at higher frame rates, for our siamese architecture.

4.1. Evaluation metrics

As it is common for evaluating object detectors, we use the Pascal VOC [5] Mean Average Precision (mAP) metric which gives intuition on detectors global performance. This metric computes the average precision for different recall values from 0 to 1, over all classes, and thus integrates information about both false positive/negative errors.

To evaluate our siamese network against other architectures, we integrate them into the DeepSORT tracker and compute the Multi Object Tracking Accuracy (MOTA) [2]. As it relies on many types of errors (false positives, misses, mismatches), this metric gives an overall idea of the tracker ability to compute consistent trajectories over time.
4.2. Object detection

Figure 5 shows a comparison between Faster R-CNN, YOLO and SSD: we can highlight the correlation between speed and precision. The more precise the detector, the slower it is. But even if the gain in precision is approximately 10% from SSD to Faster R-CNN with ResNet50, SSD still achieves about 75% of mAP while being 14.5× faster, which makes it a reasonable candidate for our embedded application. YOLO is also capable of notable frame rates with good mAP in its original version, but is still far behind SSD in terms of speed. It should be noted that because big architectures like Faster R-CNN do not fit into the Jetson TX2, fps are given on Titan Xp for all detectors. This comparison corroborates the comparative study led by Huang et al. on CNN object detectors [12], at least for Faster R-CNN and SSD, and the execution time of the latter motivated our choice to use it as input to data association.

Table 3 gets into the details of MobileNet feature extractors comparison for SSD and shows the detector mAP on the training and validation sets with the three different tested architectures. Removing the second-to-last layer of MobileNet feature extractor shows slightly better performance both on training and validation. It can be due to the fact that this layer is usually applied with a stride of 2 and makes the prediction for close objects more difficult (the feature map
Table 3: Mean Average Precision (mAP) and frames per second (fps) of SSD feature extraction architectures, with width multiplier $\alpha = 0.66$. Results are presented on training and validation sets so as not to skew the final SSD evaluation against other detectors. Fps are produced on Jetson TX2. Best results are in bold and underlined.

<table>
<thead>
<tr>
<th>Feature extractor</th>
<th>Train mAP</th>
<th>Val mAP</th>
<th>Fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full MobileNet</td>
<td>88.35</td>
<td>83.12</td>
<td>29.95</td>
</tr>
<tr>
<td>Custom w/ BN</td>
<td>90.06</td>
<td>87.06</td>
<td>29.72</td>
</tr>
<tr>
<td>Custom w/o BN</td>
<td>89.87</td>
<td>86.26</td>
<td>30.95</td>
</tr>
</tbody>
</table>

is two times smaller than in the modified version).

A slight gain in speed on the target platform can also be observed with the removal of BN layers, but as mentioned in section 3.2, the major time consumption of SSD is located in its regression layers. For further experiments in our embedded context, effort should be put on this part.

4.3. Siamese CNNs for data association

As described in table 4, fully connected layers alone account for the major portion of the whole networks parameters: 2 millions for DeepSORT base network, 1.6 million for MobileNets (respectively 75 and 97% of the parameters). Using 1x1 convolutions drastically reduces this proportion (respectively 8 and 64% for DeepSORT and MobileNets), scaling memory usage of full structures down by a factor of 1.5 to 2.7 depending on the base network, and thus leading to faster execution speed. Once integrated to DeepSORT tracking framework, our siamese networks show equivalent MOTA performance to the original architecture trained on our dataset as a re-identification task. However, the architecture we put forward is $1.7 \times$ faster.

The estimated time for computing worst case scenario feature vectors shows that our siamese 1x1 architecture with customized MobileNet w/o BN feature extractor is the closest to real-time processing. Such results are very promising and comfort us in the fact that we can embed a full real-time system, or at least process with short delay.

We can notice a difference between min_clutter and max_clutter: it can be partially explained by slightly lower performance of SSD on the first set, due to a larger proportion of false positives. Moreover, the confidence score threshold used to determine which detections to track was arbitrarily fixed to 0.5 for all sets and configurations. It would be worth choosing a threshold based on the training set tracker performance to get more balanced results and reduce the gap between test sets. This would imply to fine-tune this threshold over multiple values, across all training sets, with all similarity learning architectures.

5. Conclusion

When it comes to online multi-object tracking-by-detection, especially with huge number of objects to track within highly cluttered scenes, deep learning techniques hardly address real-time constraints. In this paper, we propose a nearly real-time and embeddable architecture applied to bus passengers counting with zenithal cameras. To this end, we first compare state-of-the-art CNN detectors and optimize the best one to meet our on-board requirements. Then, to associate detections over successive frames, we also propose a fast and new siamese architecture showing state-of-the-art performance on our challenging dataset. Several tracks can be explored: based on our evaluations, we may focus on the time-consuming detector regression layers but also get a system-wide overview of passenger counting to have better insight into achievable speed.

6. Acknowledgements

This work is partially supported by the french National Association for Research and Technology (ANRT). We would also like to show our gratitude to TISSEO partners who provided us with access to a bus for video recordings.

References


<table>
<thead>
<tr>
<th>Base</th>
<th>Type</th>
<th>Params</th>
<th>MB</th>
<th>Fps</th>
<th>Estim.</th>
<th>MOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReID (DeepSORT)</td>
<td>FC</td>
<td>2,689,888</td>
<td>22.48</td>
<td>105</td>
<td>2.38s</td>
<td>43.9</td>
</tr>
<tr>
<td>Siamese (DeepSORT)</td>
<td>FC</td>
<td>2,689,888</td>
<td>22.48</td>
<td>105</td>
<td>2.38s</td>
<td>43.7</td>
</tr>
<tr>
<td>Siamese (Mob w/ BN)</td>
<td>FC</td>
<td>1,676,384</td>
<td>9.94</td>
<td>140</td>
<td>1.78s</td>
<td>43.3</td>
</tr>
<tr>
<td>Siamese (Mob w/o BN)</td>
<td>FC</td>
<td>1,675,168</td>
<td>9.59</td>
<td>157</td>
<td>1.59s</td>
<td>43.7</td>
</tr>
<tr>
<td>Siamese (DeepSORT)</td>
<td>1x1</td>
<td>773,728</td>
<td>15.12</td>
<td>118</td>
<td>2.11s</td>
<td>43.7</td>
</tr>
<tr>
<td>Siamese (Mob w/ BN)</td>
<td>1x1</td>
<td>104,288</td>
<td>3.89</td>
<td>161</td>
<td>1.55s</td>
<td>43.7</td>
</tr>
<tr>
<td>Siamese (Mob w/o BN)</td>
<td>1x1</td>
<td>103,072</td>
<td>3.55</td>
<td>182</td>
<td>1.37s</td>
<td>43.7</td>
</tr>
</tbody>
</table>

Table 4: The three siamese feature extractors combined either with fully connected layers ("FC") or with global average pooling plus 1x1 convolutions ("1x1"). Params accounts for the total number of parameters, i.e. feature extractor combined with last layers. We show the impact of such structural choices on memory usage in megabytes (MB) along with the average number of frames per second (Fps) and the estimated time to compute feature vectors in worst case scenarios (Estim.), i.e. 10 detections per frame throughout 25 frames, on Jetson-TX2. Final tracking performances are evaluated both on min_clutter and max_clutter test scenes in terms of Multi Object Tracking Accuracy (MOTA). Best results are in bold and underlined.