NAPC: A Neural Algorithm for Automated Passenger Counting in Public Transport on a Privacy-Friendly Dataset

Robert Seidel*, Nico Jahn*, Sambu Seo, Thomas Goerttler†, Klaus Obermayer

Real-time load information in public transport is of high importance for both passengers and service providers. Neural algorithms have shown a high performance on various object counting tasks and play a continually growing methodological role in developing automated passenger counting systems. However, the publication of public-space video footage is often contradicted by legal and ethical considerations to protect the passengers' privacy. This work proposes an end-to-end Long Short-Term Memory network with a problem-adapted cost function that learned to count boarding and alighting passengers on a publicly available, comprehensive dataset of approx. 13,000 manually annotated low-resolution 3D LiDAR video recordings (depth information only) from the doorways of a regional train. These depth recordings do not allow the identification of single individuals. For each door opening phase, the trained models predict the correct passenger count (ranging from 0 to 67) in approx. 96% of boarding and alighting, respectively. Repeated training with different training and validation sets confirms the independence of this result from a specific test set.

***Index Terms*—Intelligent transportation, Long Short-Term Memory (LSTM), neural network, boarding and alighting passenger counting, privacy, range imaging, LiDAR**

I. Introduction

THE day-to-day operational management of transport systems relies on large networks of sensors, actuators, and software to provide passengers with safe, reliable, and affordable means of transportation. Improving on these goals is not only of scholarly concern but the provision of accessible and sustainable public transport systems by 2030 is one of the targets within the United Nations' *Sustainable Development Goals* framework [1]. Indeed, the efficiency and quality of service in public transportation are related to the quality of the living conditions of many citizens.

An essential task in the management of transport systems is the monitoring of per-vehicle load information. It provides a valuable service for passengers to select a comfortable itinerary, service providers to schedule their vehicle fleet or share revenues within a transportation network operated by several independent companies, and policymakers to identify structurally relevant transportation routes.

The history of automated passenger counting (APC) goes back to the mid-1970's [2] and links to other disciplines such as people detection, crowd density estimation, and people flow counting. The fast development of hardware and machine learning techniques made it possible for neural networks to be successfully applied in various tasks related to passenger counting. Stewart *et al.* [3] introduced an end-to-end people detection algorithm in images of crowded scenes using a recurrent neural network with Long Short-Term Memory (LSTM) units [4]. In this algorithm, a convolutional neural network (CNN) first encodes each image as a $15 \times 20 \times 1024$ dimensional high-level descriptor matrix, which is subsequently decoded by an LSTM network into a variable-length sequence of bounding boxes and corresponding confidence values that a previously undetected person is present within the respective image region. A confidence threshold is used to terminate the detection process [3].

For RGB-D based human detection the multi-glimpse LSTM in [5] and an asymmetric adaptive fusion two-stream network (AAFTS-net, [6]) were proposed. For real-time people detection in top-view depth images from video surveillance systems, WatchNet and its extension WatchNet++ were presented in [7] and [8]. WatchNet consists of a feature extraction module and a series of prediction stages that sequentially refine the prediction maps for human body landmarks (head and shoulders) and is trained with artificial and real depth data for people detection. The neural object detection method YOLO (You Look Only Once, [9]) was used among other applications for person detection from an overhead view [10], for pedestrian detection [11], and for person detection in thermal images [12]. The neural network based object detection method SSD (Single Shot multi-box Detector, [13]) is deployed for top view people detection and counting [14].

Wang *et al.* introduced an end-to-end deep CNN regression model for counting people in extremely dense crowds [15]. To reduce false positives, they intentionally included potentially confusing negative samples such as lush trees, buildings, and some natural scene images in the training data. Following the work of [15], CNNs have been widely applied in human detection or feature extraction for people counting research [16]. In the context of detecting humans, CNNs were deployed for head-shoulder detection in crowd images [17], for predicting density maps on a given crowd image [18], for human detection in nighttime images obtained by a visible light camera [19], and for passenger recognition in passenger flow monitoring systems [20]. CNNs were also used for feature extraction. Gao *et al.* combined CNN-based feature extraction

Manuscript received August 30, 2021; revised November 12, 2021; accepted December 20, 2021. This work was funded by a grant from the European Regional Development Fund (grant number 10167006) to K. Obermayer. We acknowledge support by the German Research Foundation and the Open Access Publication Fund of TU Berlin.

Asterisk indicates that the first two authors contributed equally to this work. The dagger symbol indicates the corresponding author (thomas.goerttler@tu-berlin.de). All authors are with the Technische Universität Berlin, Institute of Software Engineering and Theoretical Computer Science, Neural Information Processing Group, Straße des 17. Juni 135, 10623 Berlin, Germany.

The authors wish to thank Michael Siebert and Jan Sablatnig (Interautomation Deutschland GmbH, Berlin) for providing domain expertise, tools, and the 3D LiDAR (ToF) recordings. The authors wish to thank Dimitra Zarafeta and the many labeling assistants for their help viewing and annotating the video material.

with Adaboost in an algorithm for counting people in crowded surveillance environments based on head detection. In contrast, [22] used CNN-autoencoder feature extraction in an algorithm estimating passenger occupancy in crowds of passengers on a bus. A region-based CNN (R-CNN [23]) was deployed in a variety of human detection tasks, e.g. in a crowd [24], in complex scenes [25], and in drone imagery [26].

Wilie *et al.* introduced an end-to-end people counting algorithm from 2D crowd images by using a pre-trained network, the Xception (Extreme Inception, [28]) network, and adding a fully connected network on the top of the pre-trained network [27]. A class of end-to-end architectures called long-term recurrent convolutional neural network (LRCN) is proposed by Donahue *et al.* for visual recognition and description in video data [29]. LRCN is constructed by combining a CNN and an LSTM network using variable-length inputs and generating variable-length representations. Massa *et al.* proposed a regression model called LRCN-RetailNet for counting people in videos captured by low-cost surveillance cameras in retail stores [30]. The model is trained on sequences of a fixed number of continuous images in time, where the number of images is a hyperparameter (in their experiments, the three values 5, 9, 12 are used). Each sequence is annotated with the number of people in the last image of the sequence. LRCN-RetailNet predicts the number of people at the time of the last frame of the video [30].

Recently, neural networks were also applied to boarding and alighting passenger counting on video data in public buses. Liu *et al.* used a CNN and the spatio-temporal context model for passenger detection and passenger tracking, respectively, and achieved a passenger counting accuracy of 93%. This APC counted 108 of 116 passengers in the bus transportation scene [31]. In [32], a two-class SSD and a Kalman filter are used for detecting passengers and tracking their movements. They conducted experiments using a 7-segment bus monitoring video where the segments have different characteristics, such as dark/strong outside light, crowded while getting on/off, including passengers carrying children, including passengers with babies and children. Their APC counted all 28 boarding passengers and 79 of 81 alighting passengers, resulting in an accuracy of 98%. Furthermore, they conducted some pedestrian statistical experiments in the laboratory, including single walking, two people walking together, 5 to 6 people walking, cross walking and squatting, etc. Their APC counted 64 of 68 "in" passengers and 54 of 61 "out" passengers for the experiments, resulting in an accuracy of 91%.

Sun *et al.* introduced a depth video stream generating the method from RGB-D videos obtained by a camera mounted on top of the door area of three different buses. They propose a boarding and alighting passenger counting method combining a two-step (generating and refining head proposal) head detection with a tracking algorithm for the generated depth video samples [33]. Different from [31], and [32], in [33] the APC was tested on a large data set with 2000 videos with four different sub-categories, which were defined based on the noise level (strong/mild sunlight) and crowdedness (crowded/uncrowded) of the scene. The performance for the different sub-categories ranged from 72.3% to 85.4% for boarding and from 91.3% to 93.7% for alighting passengers.

The three approaches [31]–[33] for counting boarding and alighting passengers from image sequences perform person detection and tracking with two different modules and subsequently combine both results for the counting. In our approach, we consider the architecturally more straightforward approach where a single neural network solves the detection, tracking, and counting problems *simultaneously* in an end-to-end learning fashion.

The development of information technology to collect data through cameras and sensors gives rise to significant privacy risks [34], [35]. Privacy issues become more and more critical by collecting and analyzing vast amounts of images and video sequences. Various privacy protection solutions for visual recognition were proposed, e.g., by ad hoc de-identifying face images [36], by different methods to hide distinguishing facial information [37], by regions of interest (ROI) based transform-domain or codestream-domain scrambling [38], by face morphing [39], by distortion-based visual privacy filters [40], by a degradation transform for the original video inputs [41], or by a video face anonymizer [42].

To assure a high level of passenger anonymity when counting people in sequences of color images, Skrabanek *et al.* used a camera set-up for capturing images from a top-down (orthogonal) view [20]. Low-resolution depth images obtained by RGB-D sensors are used for privacy-preserving human pose estimation in [43] and for head detection in the task of counting boarding and alighting passengers [33]. Top-view depth images from a video surveillance system are employed for detecting people [7], as well as people committing attacks and intrusions [8].

A. Contributions

Our work introduces an end-to-end algorithm for a real-time automated boarding and alighting passenger counting system called "Neural Automated Passenger Counter" (NAPC) based on an LSTM recurrent neural network. We present an LSTM network with a tailored cost function, which is trained on a large dataset of 3D LiDAR video recordings of individual door openings (hereafter called 'sequences', see Fig. 1) to automate passenger counting (Section III). It achieves high performance in a series of counting experiments (Section IV): On average, the algorithm obtains an *exact* count in 96 out of 100 sequences for both boarding and alighting passengers. The magnitude of the error made in the miscounted sequences is small. The algorithm fails to count only 1.46% of boarding and only 1.11% of alighting passengers (see Section IV). This performance is superior to the performance reported for three recently proposed methods for counting boarding and alighting passengers [31]–[33] (Section V).

Counting experiments were conducted on a large set of manually labeled videos recorded with 3D LiDAR cameras that are installed over the doorways of an eight-door German regional train with a top-down/high-angle perspective (Section II). At least three independent labeling assistants manually annotated the events (that is, boarding and alighting passengers) in each sequence using additional grayscale video

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

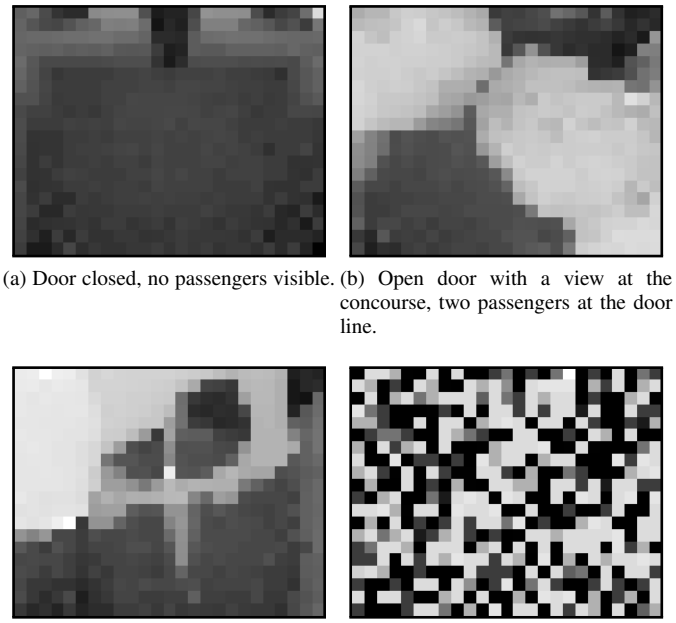
recordings (320×240 pixels at 10 frames per second). If no consensus upon the correct annotation (that is, upon the total number of events, and upon their timestamps up to an intra-labeler standard variation of 2 seconds) was reached, the number of viewers was increased up to seven before an administrators' decision upon the correct annotation of that sequence was made. Even if their annotation was difficult for human viewers, many challenging sequences could be retained from rejection by this approach.

The NAPC system accepts sequences with hundreds of frames, whose lengths are variable and only determined by the duration of a realistic door opening phase and the recording frame rate. It reliably predicts passenger counts from low-resolution depth information with just 20×25 pixels (see Section IV), making our NAPC a privacy-aware passenger counting system. Learning is accomplished in an end-to-end manner. No background modeling, head detection, or trajectory tracking is required. Counting is realized through a single neural network architecture. A similar approach to our method is LRCN-RetailNet [30] for counting people in a retail store. Both approaches, NAPC and LRCN-RetailNet, consider the people counting task as a regression problem. While LRCN-RetailNet takes a fixed number (5, 9, or 12 frames) of the RGBP video sequences obtained by combining color information and extracted foreground (people) information as input, NAPC takes a variable length of video sequences recorded with 3D LiDAR cameras as input. Whereas LRCN-RetailNet predicts the occupancy at the store, our method predicts the number of boarding and alighting passengers during a door opening phase. To our best knowledge, our method is the first end-to-end recurrent neural learning algorithm for a boarding and alighting passenger counting system from 3D LiDAR video recordings.

A tailored cost function and data augmentation strategies (such as mirroring or backward-playing of video, see Section III) is used to maximize the information extracted from a given training set such that the number of required training videos can be minimized. Approx. 2,000 sequences are already sufficient to train an NAPC network from random initialization to high accuracy (see Section IV-G). In our setting, this amounts to six days of data collection which underpins the practical significance of our approach.

II. The *Berlin-APC* Dataset

For the evaluation of our system, we employ a large-scale dataset of APC-relevant image sequences (*Berlin-APC* Dataset, [44]). It consists of 12,956 sequences with a shape of $t \times 20 \times 25$, where t denotes each sequence's variable number of frames. Note that only 3D LiDAR (but no RGB) information is captured, resulting in one channel per pixel. (The tradeoff between this approach and other sensor types was not subject of this research.) This mode of recording does not allow the identification of individual passengers (of Fig. 1) but preserves enough information to give an accurate algorithmic passenger count (see Section IV). The video sequences were recorded in 2017 by 3D LiDAR cameras mounted above the doors of a regional train under regular operation in the Berlin metropolitan area. Every sequence is annotated by the number of boarding and alighting passengers (excluding children) as a label.



(a) Door closed, no passengers visible. (b) Open door with a view at the concourse, two passengers at the door line.

(c) Passenger with bicycle. (d) Noise distorted frame.

Fig. 1. Sample frames from the depth sensor. The door is located at the top of the frame's field of view. Each sequence begins with a door opening (a), then shows zero or more passengers boarding and/or alighting (b,c), and ends with a door closing (a). There may be noise distorted frames at the end of a stream after the door closing (d).

The recordings were made at 40 frames per second but were later reduced to 10 frames per second. Each pixel takes floating-point values between 0 and 1, where 0 is closest to the sensor and 1 is 4 m away from the sensor. After reducing the framerate, the number of frames per sequence ranges from 56 to 3275 (avg. ~190, see Table I). Each sequence shows one entire door opening phase in a top-down perspective, including the physical opening and closing of the pictured sliding door (see Fig. 1 for sample frames).

Each sequence was initially given to three human labelers who independently annotated the timestamp and direction (boarding or alighting) of every adult passenger they discerned in the sequence using a specialized labeling software.[1] An additional grayscale video recording with 320×240 pixels at 10 frames per second of every scene was made available to the labeling assistants to enhance clarity and comprehensibility. Rare events such as dense crowds or large objects could be annotated in a free-text field. After annotation, the labelers were asked to mark every sequence as decidable or undecidable. Sequences marked undecidable by at least two labelers (primarily due to sensor errors) were rejected.

It was then checked whether the three initial labelers agreed upon the total counts per category and direction. If there were no simple majority (1,270 out of 12,956 sequences, 9.80%), the sequences would be re-examined by up to four

[1]We used the video-based annotation and counting tool *VisualCount*, developed by Interautomation Deutschland GmbH, Berlin, Germany.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

TABLE I
STATISTICS OF THE BERLIN-APC DATASET. LEFT: DISTRIBUTION OF THE NUMBER OF FRAMES PER SEQUENCE. RIGHT: DISTRIBUTION OF THE NUMBER OF BOARDING AND ALIGHTING EVENTS PER SEQUENCE.

Number of Frames		**Number of Events**		
			no. of sequences	
range	no. of sequences	range	boarding	alighting
0 – 99	87	0	4551	6437
100 – 199	9633	1	4030	2537
200 – 299	2100	2 – 4	2905	2276
300 – 599	971	5 – 9	957	1072
600 – 1199	141	10 – 19	395	481
1200 – 1799	14	20 – 49	117	151
1800 – 2399	7	50 – 99	1	2
2400 – 2999	2			
3000 – 3599	1			
min.	56	**min.**	0	0
avg.	190	**avg.**	2.03	2.02
max.	3275	**max.**	55	67

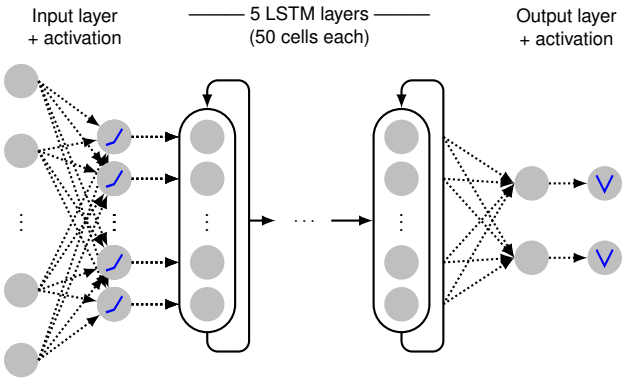


Fig. 2. The proposed neural network architecture. The values of each frame are passed through a fully connected layer with the output size of 50, followed by a Leaky Rectified Linear Unit (LReLU) [49] with the slope parameter α set to 0.3. The 5 subsequent hidden layers are LSTMs with 50 cells each. Their output is reduced again with a fully connected layer. Followed by a LReLU with the slope parameter α set to -1. The final outputs are the predictions for our two output classes.

additional human annotators. If still no majority upon a correct label was reached (396 out of 12,956 sequences, 3.0%), an administrators' decision finally determined the label. The per-sequence totals of boarding and alighting adults were then stored. This iterative approach ensured that dense crowds and sequences with many passengers obtain consistent labels after careful manual inspection. Though comparatively rare, their sequences are a cornerstone for training a neural network and evaluating its predictive performance.

This procedure yielded a dataset of 12,956 sequences with 26,243 boarding and 26,164 alighting events; only 382 sequences were unusable. The detailed statistics are summarized in Table I.

III. THE NAPC-APPROACH

A. The Network Architecture

There are two main approaches in deep learning which are capable of modeling temporal dependencies. Autoregressive neural networks like WaveNet [45] condition their new prediction on previous ones. Plain recurrent neural networks (RNN) [46], gated recurrent units (GRU) [47] or LSTMs maintain past information in hidden states throughout the sequence. Plain RNNs can not maintain information throughout long sequences, and GRUs are less capable of solving counting problems compared to LSTMs [48]. Thus, LSTMs are chosen.

The input data is the above mentioned *Berlin-APC Dataset* (see Section II). Every frame is represented as a 500-dimensional vector by concatenating pixel rows. The fully connected input layer reduces the input vector to a smaller representation. This reduced representation is then propagated through the LSTM layers. The output is combined with a second fully connected layer into the final two output classes, namely the counted boarding and alighting passengers.

We propose the network architecture shown in Fig. 2. Finally, the two major hyperparameters of the network's structure are the depth and the height of the LSTM core. Those were optimized using standard hyperparameter selection procedure (for details, see Section IV-F).

B. Data Augmentation

We apply the following data augmentation to each sequence independently at every epoch: The neural network decides whether a passenger is passing through the door, which is centered on the top of each frame (see Fig. 1). This property must not be changed. Thus, the sequence labels remain valid when performing a left-right mirroring of each frame, keeping the door position fixed at the top center of the view. (Mirroring all frames upside-down invalidates the sequence as the door would flip to the bottom, and thus change the region of interest every time.) Due to the low-resolution data, reversing the sequence does potentially distort the view on the objects but does not change the position of the door. When reversing the sequence, previously boarding passengers are now leaving the vehicle and vice versa, i.e. swapping the boarding and alighting labels yields a valid label for the reversed sequence. Both augmentations (left-right mirroring and reversing) are applied independently with a probability of 0.5.

C. Simple Loss

Each sequence is paired with only one label per class (the two accumulated counts of the boarding and alighting passengers). The network processes the whole sequence at once, and the prediction of the last frame is compared to the label. The same approach was used in [30] and also works for the NAPC. Employing more control over the network's loss without acquiring more information about the sequences themselves leads to event-precise predictions. Therefore, the error is calculated for each frame. This may improve the gradient flow through time.

Valid predictions are bounded by zero from below and the respective labels from above, as shown in Fig. 3. We refer to them as the lower and upper bounds.

The error is calculated as follows: Network predictions of boarding or alighting passengers are updated at every frame of the sequence. Let k be the index of a sequence $X_k \in \mathbb{R}^{t_k \times 500}$ with t_k frames, and let $Y_k \in \mathbb{N}^2$ be the corresponding labels,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

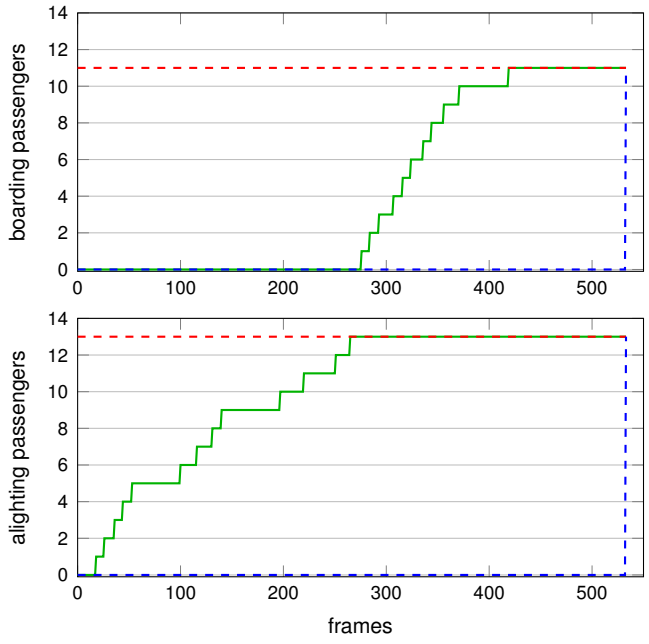


Fig. 3. Illustration of the upper and lower bounds for an example stream. The top and bottom panels show the actual accumulated number for boarding (top) and alighting (bottom) passengers (jagged green line). The red dashed line denotes the upper bound, which is always equal to the total number of boarding or alighting passengers. The dashed blue line denotes the lower bound, which is zero except for the last frame.

with the two output classes. Then, the upper bound $U_k \in \mathbb{N}^{t_k \times 2}$ for that sequence is given by

$$U_{kij} = Y_{kj}, \quad \text{for all } i \in \{1, \dots, t_k\}, j \in \{1, 2\} \tag{1}$$

and the lower bound $L_k \in \mathbb{N}^{t_k \times 2}$ is given by

$$L_{kij} = \begin{cases} 0 & \text{if } i < t_k, \\ Y_{kj} & \text{if } i = t_k, \end{cases} \quad \text{for all } i \in \{1, \dots, t_k\}, j \in \{1, 2\}. \tag{2}$$

Intuitively speaking, we require the network to count *at most* as many events as the label prescribes; as well as to count *at least* zero events, except for the last frame. For the last frame we require the network to exactly predict the label of that sequence. Let $\widehat{Y}_k \in \mathbb{R}^{t_k \times 2}$ denote the NAPC's prediction of that sequence, which is always greater or equal to zero due to the activation function (see Fig. 2). Then the error $E_k \in \mathbb{R}^{t_k \times 2}$ is given by how much the bounds are violated,

$$E_k = \underbrace{\max\left(0, \widehat{Y}_k - U_k\right)}_{\text{overcount}} - \underbrace{\min\left(0, \widehat{Y}_k - L_k\right)}_{\text{undercount}}, \tag{3}$$

where the minimum and maximum operate elementwise.

Minimizing the simple loss results in networks which only predict zeros. This is due to two reasons: First being a significant fraction of sequences not containing any events in either class (see Table I). However, the second and more influencial one being the loss function in combination with the labels. To clarify the problem, assume for simplicity no boarding and alighting events for a sequence X_k. The according labels are both zero. Placing it in Eq. 1 and Eq. 2 results in U_{kij} and L_{kij} being zero for every $i \in \{1, \dots, t_k\}$ and $j \in \{1, 2\}$. Substituting U_k and L_k in Eq. 3 with zero, the resulting error E_k is calculated as:

$$\begin{aligned} E_k &= \max\left(0, \widehat{Y}_k\right) - \min\left(0, \widehat{Y}_k\right) \\ \Rightarrow E_k &= \widehat{Y}_k. \end{aligned}$$

Which holds because of the final activation function of the network. Thus, every prediction which is not *exactly zero* produces an error. When this error is minimized due to a large amount of no events, the only leftover error is produced for the last frame for any label bigger than zero, which is negligible in sequences with up to several thousands of frames.

D. Refined Loss

To overcome the aforementioned problem, we concatenate sequences to create longer sequences with intermediate counting ground truth. Thus counts of the concatenated sequences are accumulated. The bounding boxes of the concatenated sequences (see Fig. 4) now stack on top of each other but moved along the time axis. The new loss function is defined as follows: Let k and l be the indices of the sequences $X_k \in \mathbb{R}^{t_k \times 500}$ and $X_l \in \mathbb{R}^{t_l \times 500}$. The concatenation of these two sequences

$$X^* = \begin{bmatrix} X_k & X_l \end{bmatrix} \in \mathbb{R}^{(t_k + t_l) \times 500}$$

contains two successive door opening phases, and the upper and lower bounds $U^*, L^* \in \mathbb{R}^{(t_k + t_l) \times 2}$ of the concatenated sequences are given by the sum of the two individual bounds, i.e.

$$U^*_{ij} = \begin{cases} Y_{kj} & \text{if } i \leq t_k, \\ Y_{kj} + Y_{lj} & \text{if } i > t_k, \end{cases}$$

and

$$L^*_{ij} = \begin{cases} 0 & \text{if } i < t_k, \\ Y_{kj} & \text{if } t_k \leq i < t_k + t_l, \\ Y_{kj} + Y_{lj} & \text{if } i = t_k + t_l, \end{cases}$$

where $i \in \{1, \dots, t_k + t_l\}$ and $j \in \{1, 2\}$.

Given a neural prediction $\widehat{Y}^* \in \mathbb{R}^{(t_k + t_l) \times 2}$ the loss function now reads as

$$E^* = \max\left(0, \widehat{Y}^* - U^*\right) - \min\left(0, \widehat{Y}^* - L^*\right).$$

The number of concatenated sequences is a hyperparameter of the learning procedure. It was fixed to a length of five for all experiments.

E. Learning Procedure

We use the Adam optimizer [50] with a fixed learning rate of 0.001 when training an NAPC model. It runs for a fixed number of 5,000 epochs, where the prediction accuracy is determined after every 10th epoch on the validation set. The model with the highest validation accuracy is selected for testing (see Section IV and Fig. 5). Unless noted otherwise, all experiments use 5 LSTM layers with 50 LSTM cells each. The sequences are concatenated as described previously and are batched together to 32 concatenated sequences. Thus, every batch consists of 160 (= 5×32) randomly drawn sequences until the training data is exhausted and a new epoch begins.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

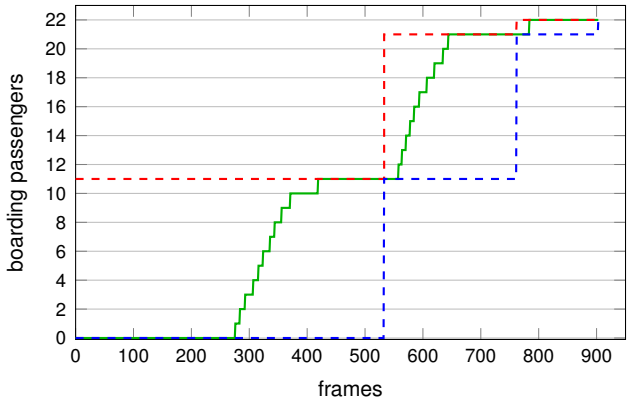


Fig. 4. Illustration of the upper and lower bounds for an example stream created by concatenating three streams (boarding passengers only). Similar to Fig. 3 the red dashed line marks the upper bound, and the blue dashed line the lower bound. The bounding box of the second stream is raised by 11 according to the label of the first stream. Therefore, maintaining states between consecutive streams and accumulating passenger counts (jagged green line) over time.

IV. Results for the *Berlin-APC* Dataset

We next present a series of experiments that demonstrate the high passenger counting accuracy of NAPC on the previously introduced dataset. To that end, we first formulate the passenger counting task in terms of a classification and a regression problem, and compute the respective performance indices. We then explored how prediction performance depends on the hyperparameter selection and the required amount of available training samples. To increase the robustness of our experiments, the splitting of our dataset into a training, validation, and test subset is done along the lines of the original 2017 recording days. That is, if sequences are randomly sampled, always entire recording days are drawn rather than individual sequences (a similar approach is chosen in [33]). To that end, the sequences recorded at 6 out of 31 recording days (approx. 20%) were chosen at random (with random seed $\mathcal{S}$) to build a test set, $\mathcal{H}_\mathcal{S}$, of size $H_\mathcal{S} \in \mathbb{N}$ denoted by

$$\mathcal{H}_\mathcal{S} := \left\{ X_{\mathcal{S},k} \in \mathbb{R}^{t_k \times 500} : k = 1, ..., H_\mathcal{S} \right\},$$

where t_k denotes the number of frames of the k-th sequence. The remaining 25 recording days are split into a training, $\mathcal{T}_\mathcal{S}$, and validation, $\mathcal{V}_\mathcal{S}$, subset in a 3:1 ratio. To simplify the notation, we drop the random seed index $\mathcal{S}$ in the following. After a randomly initialized NAPC instance was trained using $\mathcal{T}$ and model selection was done using $\mathcal{V}$, we computed for each sequence $X_k \in \mathcal{H}$ the prediction tensor $\widehat{Y}_k \in \mathbb{R}^{t_k \times 2}$ ($k = 1, ..., H$). The prediction of the last frame $\widehat{Y}_k[t_k, :] \in \mathbb{R}^2$ is the final algorithmic count, where $[\cdot, \cdot]$ denotes standard array indexing. We denote by

$$\widehat{Y}_k^{(b)} := \text{round}\left(\widehat{Y}_k[t_k, 1]\right) \quad \text{and} \quad \widehat{Y}_k^{(a)} := \text{round}\left(\widehat{Y}_k[t_k, 2]\right)$$

the final boarding and alighting count, respectively, where round denotes standard integer rounding. Intuitively speaking, $\widehat{Y}_k[:, 1]$ contains the prediction of the entire boarding time-series (that is, the neural prediction to the given input sequence X_k), and $\widehat{Y}_k^{(b)}$ contains the predicted boarding count at the last frame, rounded to the next integer. The value for $\widehat{Y}_k^{(a)}$ is similarly extracted from the last frame of $\widehat{Y}_k[:, 2]$. The labels for the boarding and alighting passengers are likewise defined as $Y_k^{(b)}$ and $Y_k^{(a)}$, respectively, as Y_k is the count until the last frame.

The results presented were obtained by the following scheme: First, we used eight different random seeds $\mathcal{S}$ to sample independent random partitionings of the entire dataset into the training, validation, and holdout sets $\mathcal{T}$, $\mathcal{V}$, and $\mathcal{H}$ as described above to estimate the performance of our approach independent of the specific choice of the training and holdout sets. For each of these different partitionings, we then trained four randomly initialized NAPC networks each to test the robustness of the performance results against different influences of model initializations and against different data samples used for training and testing. That is, 32 models were trained in total.

A. Classification Performance

We first looked at the passenger counting problem in terms of a classification problem, that is, every sequence has a discrete class-label in $\mathbb{N} \times \mathbb{N}$. We denote the boarding accuracy of an NAPC network by

$$\text{ACC}_b := \frac{\left| \left\{ k = 1, ..., H : \widehat{Y}_k^{(b)} = Y_k^{(b)} \right\} \right|}{H},$$

that is, the share of correctly classified sequences relative to the total amount of sequences in the test set $\mathcal{H}$. The alighting accuracy ACC_a is defined analogously.

Our approach achieved a boarding accuracy of 0.9615 (average over all models, min: 0.9522, median: 0.9613, max: 0.9711), i.e., approx. 96% of all sequences in the test set were classified correctly. The accuracy only varied by 2 percentage points from training to training. That is, the accuracy was independent of the choice of the test set, and this held for both the boarding and the alighting direction, see Table II.

B. Regression Performance

The passenger counting problem can also be understood as regression task which allows for the quantification of counting errors. To that end, the mean absolute and mean absolute percentage errors in boarding direction (MAE_b and MAPE_b) are given by

$$\text{MAE}_b := \frac{1}{H} \sum_{k=1}^{H} \left| \widehat{Y}_k^{(b)} - Y_k^{(b)} \right|,$$

$$\text{MAPE}_b := \frac{1}{H} \sum_{\substack{k=1 \\ Y_k^{(b)} \neq 0}}^{H} \left| \frac{\widehat{Y}_k^{(b)} - Y_k^{(b)}}{Y_k^{(b)}} \right|.$$

The MAE and MAPE in the alighting direction (MAE_a and MAPE_a) are defined similarly. Note that the MAPE formula excludes sequences with zero passengers to avoid division by zero. Since the models summarized here classify a sequence with zero boarding passengers correctly with a probability >99%, this exclusion does not strongly affect the MAPE's

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

significance. In line with [33], we also report the mean absolute percentage error in the boarding direction relative to the *average* number of boarding passengers (including sequences with zero passengers),

$$\overline{\mathrm{MAPE}}_b := \frac{1}{H}\sum_{k=1}^{H}\frac{\left|\widehat{Y}_k^{(b)} - Y_k^{(b)}\right|}{\overline{Y}^{(b)}},$$

where $\overline{Y}^{(b)} := \frac{1}{H}\sum_{k=1}^{H} Y_k^{(b)} = 2.03$. In the alighting direction, $\overline{\mathrm{MAPE}}_a$ is defined similarly with $\overline{Y}^{(a)} = 2.02$.

Our models achieved a boarding MAE_b of 0.0595 (average over all models, min: 0.0337, median: 0.0535, max: 0.0944), see Table II, i.e. a typical trained NAPC network over- or undercounted approx. 6 boarding passengers per 100 door opening phases. The relative error indicated by the boarding MAPE_b was on average 0.96% (min: 0.76%, median: 0.94%, max: 1.21%), i.e., a trained NAPC network over- or undercounted approx. 1% of boarding passengers in the median of all door opening phases. The relative error indicated by the boarding $\overline{\mathrm{MAPE}}_b$ was on average 2.76% (min: 1.80%, median: 2.78%, max: 3.91%), hinting at larger errors in sequences with more passengers. The results for the alighting direction were similar, see Table II.

TABLE II
THE RANDOMLY INITIALIZED NAPC NETWORKS WERE TRAINED AND EVALUATED ON EIGHT DIFFERENT PARTITIONINGS OF THE AVAILABLE DATASET INTO A TRAINING, VALIDATION AND TEST SET. THE REPORTED VALUES ARE THE TEST SET METRICS.

Statistics of Repeated Independent Trainings

Metric	Boarding			Alighting		
	Min.	Avg.	Max.	Min.	Avg.	Max.
ACC	95.22%	96.15%	97.11%	88.14%	95.64%	97.01%
MAE	0.0337	0.0595	0.0944	0.0337	0.0553	0.1398
MAPE	0.76%	0.96%	1.21%	0.65%	0.95%	1.45%
$\overline{\mathrm{MAPE}}$	1.80%	2.76%	3.91%	1.80%	2.59%	6.57%
$\Delta^{(\mathrm{global})}$	−2.79%	−1.46%	−0.15%	−2.18%	−1.11%	1.92%
$\mathrm{CI}^{(\mathrm{lower})}$	−4.01%	−2.24%	−0.57%	−2.93%	−1.70%	1.06%
$\mathrm{CI}^{(\mathrm{upper})}$	−1.57%	−0.68%	0.28%	−1.43%	−0.53%	2.78%

C. Counting Metrics

We further propose to employ the following two metrics tailored to the passenger counting task. First, the global relative bias in boarding direction $\Delta_b^{(\mathrm{global})}$ is defined by

$$\Delta_b^{(\mathrm{global})} := \frac{\sum_{k=1}^{H}\widehat{Y}_k^{(b)}}{\sum_{k=1}^{H} Y_k^{(b)}} - 1.$$

The global relative bias in alighting direction $\Delta_a^{(\mathrm{global})}$ is defined analogously.

Our models achieved a boarding bias $\Delta_b^{(\mathrm{global})}$ of -1.46% (average over all models, min: -2.79%, median: -1.49%, max: -0.15%), i.e. a trained NAPC network counted in the median 985 passengers boarding for every 1,000 manually counted boarding passengers (similar results for the alighting direction).

Second, we computed the 95% confidence interval (CI) of a t-test-induced equivalence test [51] with the alternative hypotheses

$$H_0 : |\mu| \geq m, \quad \text{i.e., a systematic boarding count error}$$
$$H_1 : |\mu| < m, \quad \text{i.e., no systematic boarding count error}$$

where $m > 0$ is the required equivalence margin, and μ is the (unknown theoretical) MAPE_b if the test set contained an infinite number of sequences (i.e., $H \to \infty$; analogously defined for MAPE_a). Then, if the confidence interval is fully contained in the interval $[-m, m]$, the hypothesis H_1 holds with a probability of 95%.

In our experiments, we achieved a lower bound on the confidence interval $\mathrm{CI}^{(\mathrm{lower})}$ for the boarding direction of -2.24% (average of over all models, min: -4.01%, median: -2.14%, max: -0.57%) and an upper bound $\mathrm{CI}^{(\mathrm{upper})}$ of -0.68% (average of over all models, min: -1.57%, median: -0.65%, max: 0.28%). Note that the zero (that is, a theoretically unbiased system) was not contained within the median confidence interval, but the equivalence test suggested an undercounting bias. The results for the alighting direction were similar. A typical fully trained NAPC network would not have passed the equivalence test for the boarding direction with an equivalence margin of $m = 0.01$ as suggested in [52] since the 95% CI is not fully contained in the interval $[-m, m]$.

D. Challenging Sequences and Rare Events

Next, we posed the question of how well a trained NAPC network handles sequences that contain challenging events such as large objects, crowds, bicycles, or lingerers. To answer this question, we further examined a trained NAPC network from the previous section. First, we plotted its training and validation accuracy together with the average (taken over all 32 previously trained NAPC networks) of the accuracy on the training set and the accuracy on the validation set along with ± 2 times their standard deviation in Fig. 5. We further marked the epoch with the highest attained validation accuracy after the training was stopped. Our results show that all training repetitions had a steep increase in validation accuracy up to approx. 500 epochs, and only minor improvements after approx. 4,000 epochs.

We further plotted the error distribution, that is, the distribution of the differences between the manual and the automated count, as histograms in Fig. 6. Over the entire test set, we observed that average differences are close to zero with a tendency to undercount. Notably, the fully trained NAPC instance never missed more than six passengers—even for challenging sequences. (Similar results were obtained for the alighting direction and when analyzing the other models summarized in Table II.) If a door opening phase featured a dense passenger sequence, a passenger jam, or more than 20 boarding passengers, the accuracy decreased to merely 55% due to undercounted (that is, overlooked) passengers. This is also observed in the confusion matrix (not reported here). The distribution for lingerers is right-tailed, which indicates that lingerers were often double-counted. Bicycles or other large objects had a slight adversarial effect on counting performance, decreasing the accuracy by approx. 7 percentage points.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

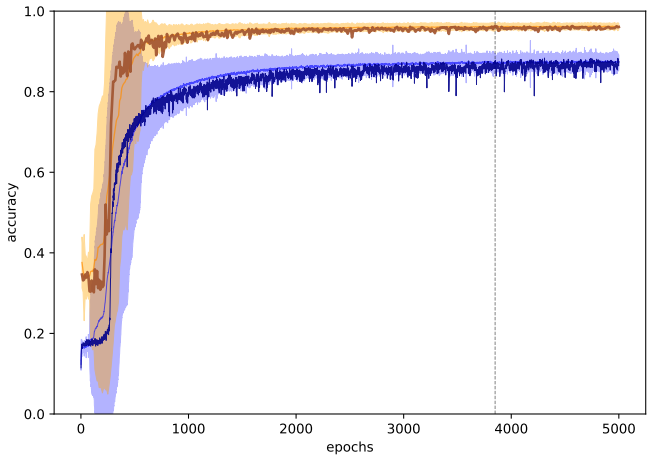


Fig. 5. Learning curves for several NAPC instances trained on eight different partitions of the dataset into training, validation, and testing subsets and four repetitions per partitioning (total of 32 models). The figure shows the epoch-wise average of the validation accuracy (mean of boarding and alighting direction, thin orange line), ± 2 times the epoch-wise standard deviation (orange shaded area), and the analogously plotted training accuracy (blue). The thick lines visualize the validation (dark orange) and training (dark blue) accuracy of a randomly selected single NAPC instance. The vertical line indicates the epoch with the highest attained validation accuracy after the training was stopped; the corresponding model was selected for testing. Note that the training accuracy is determined using concatenations of five sequences (see Section III) and is, therefore, lower than the validation accuracy.

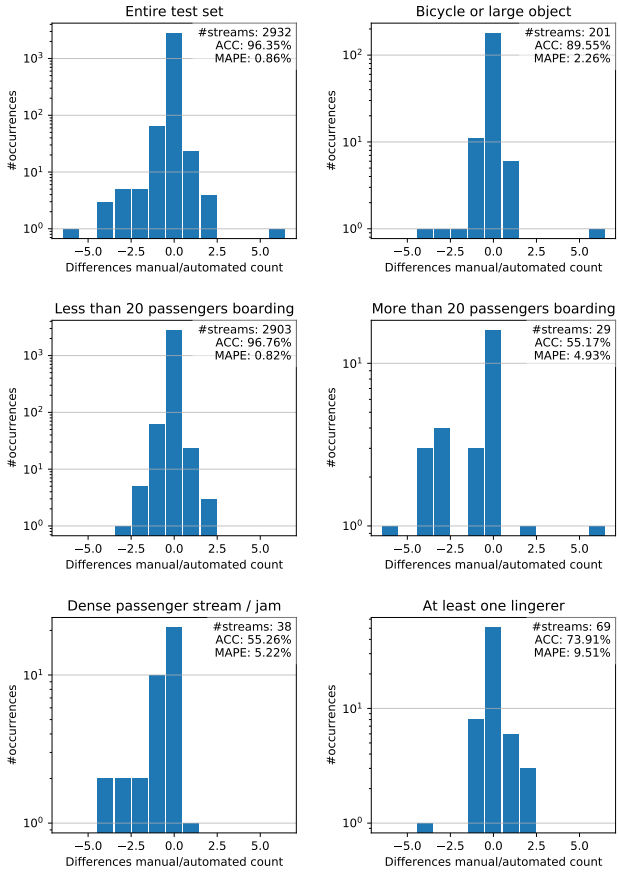


Fig. 6. Histogram of boarding errors for a fully trained NAPC instance. The panels show the distributions for the entire test set (top left), for sequences with at least one bicycle, or at least one large object (top right), for sequences with less than 20 passengers boarding (middle left), for sequences with more than or equal 20 passengers boarding (middle right), for 'dense' passenger flows with concourse (i.e., there is no visible gap between people in at least one frame, bottom left), and for at least one lingerer (i.e., a person that abides on the door line, bottom right). Note that the vertical axis has a logarithmic scale.

E. Resolution Tradeoff

To determine the tradeoff between the spatial resolution of the depth video and the performance of our counting system, we artificially decreased the resolution of the input data by downsampling (bi-linear spline interpolation with anti-aliasing) simulating different sensor resolutions (15×18, 10×12, 5×6, and 2×3 pixels). The data was then upsampled to the original resolution of 20×25 pixels (using the same method) in order to keep the network architecture constant over the entire experiment. For each of the four datasets, a randomly initialized NAPC instance was trained. This experiment was repeated five times with different random seeds $\mathcal{S}$, i.e., a total of 20 additional NAPC instances were trained. The results are summarized in Table III. For convenience, we also state the relevant metrics for the full-resolution experiment (without downsampling) from Sections IV-A–IV-C as a reference.

Counting performance remains stable for lower resolutions down to a resolution of 5×6 pixels, for which the performance slightly decreased in all metrics accompanied by an increase of the standard deviation (i.e., an increased dependence of the performance on the chosen holdout set). Still, more than 95% of all sequences in the holdout set are counted correctly. Only when the resolution is reduced to 2×3 pixels, counting performance drops significantly, and correct counts are achieved for 45% of all sequences only. While we consider a resolution of 20×25 pixels to be low enough to prevent the identification of single individuals, even lower resolutions down to 10 x 12 pixels can be employed without sacrificing counting performance.

TABLE III
SELECTED TEST SET METRICS FOR DOWNSAMPLED SEQUENCES (AVERAGE OVER REPEATED INDEPENDENT TRAININGS, STANDARD DEVIATION IN BRACKETS)

	Downsampling Resolution (in pixels)				
	20×25	15×18	10×12	5×6	2×3
Boarding					
ACC	0.961 (0.005)	0.961 (0.006)	0.959 (0.010)	0.955 (0.012)	0.474 (0.434)
MAE	0.060 (0.017)	0.056 (0.018)	0.059 (0.021)	0.070 (0.030)	1.687 (1.589)
$\Delta^{(\text{global})}$	−0.015 (0.007)	−0.013 (0.008)	−0.013 (0.009)	−0.019 (0.011)	0.241 (0.622)
Alighting					
ACC	0.956 (0.015)	0.959 (0.004)	0.957 (0.007)	0.951 (0.017)	0.435 (0.459)
MAE	0.055 (0.019)	0.053 (0.009)	0.057 (0.013)	0.067 (0.031)	1.972 (1.878)
$\Delta^{(\text{global})}$	−0.011 (0.007)	−0.009 (0.002)	−0.012 (0.003)	−0.017 (0.012)	0.285 (0.694)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2021.3139393, IEEE Open Journal of Intelligent Transportation Systems

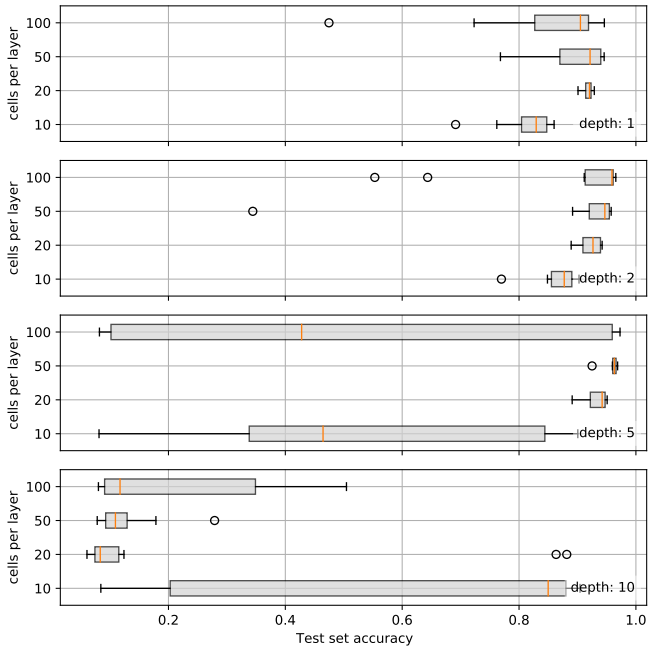


Fig. 7. Boxplots of the test set accuracies jointly plotted for the boarding and alighting direction after training with different combinations of hyperparameters (see axis labels). For every tuple, the orange line denotes the median accuracy; the grey box extends from the lower quartile accuracy $q_{0.25}$ to the upper quartile accuracy $q_{0.75}$. The left (right) whisker denotes the lowest (highest) attained accuracy above $q_{0.25} - 1.5(q_{0.75} - q_{0.25})$ (below $q_{0.75} + 1.5(q_{0.75} - q_{0.25})$). The attained accuracy values that do not fall between the whiskers are plotted as flier points. The networks with 5 LSTM layers with 50 cells each show both the smallest variation (smallest interquartile range) and the highest accuracy.

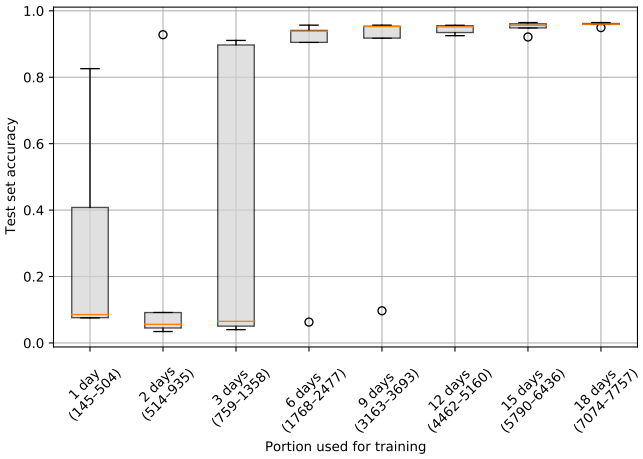


Fig. 8. Boxplots of the test set accuracy as a function of the training set size, jointly plotted for the boarding and alighting direction. The training set included 1 up to 18 recording days. The corresponding number of sequences is provided in brackets. (This number varies because different recording days are used in the respective trainings, depending on the seed.) For details of the boxplot representation, see Fig. 7.

F. Hyperparameter Validation

Next, we validated the choice of the LSTM network's depth and height. To that end, we fixed a training, validation and test partitioning of the entire dataset and trained a total of five NAPC instances each for the hyperparameters $\mathit{depth} \in \{1, 2, 5, 10\}$ and $\mathit{height} \in \{10, 20, 50, 100\}$, i.e. a total 80 NAPC instances. Thereby, we obtained for each tuple $(\mathit{depth}, \mathit{height})$ a total of ten values for the performance of a trained model in terms of its boarding and alighting accuracies. We jointly visualized the distribution of these ten accuracy values for each hyperparameter configuration in Fig. 7. The choice of 5 LSTM layers with 50 cells each reached the highest accuracy and exhibited the smallest variation (in terms of interquartile range). These findings are confirmed by the analogous plot of the hyperparameter validation in the global relative bias $\Delta^{(\text{global})}$ (not reported here).

G. Minimal Required Train Set Size

Finally, we decreased the number of recording days provided to the training procedure while keeping all other parameters constant. More precisely, we drew a fixed validation and test set and varied the number of recording days used for training. Unused sequences were discarded. This procedure was validated with five different seeds $\mathcal{S}$, that is, with five different training, validation and holdout sets. Similar to Subsection IV-F, we jointly visualized the boarding and alighting accuracy as a function of the train set size, see Fig. 8. Our results show that training can reach an accuracy above 90% in both directions with as few as approx. 1,000 sequences, but is not robust as it depends on the choice of the test set. Training the model with 2,000 sequences already had a high chance of reaching accuracy values of 90% and above, and only failed to do so once. When using around 5,000 sequences, none of the trained models had an accuracy below 90%. In the *Berlin-APC* dataset, this amounts to around two weeks of in-vehicle video material collection.

V. Conclusion

We introduced a real-time Neural Automated Passenger Counting system (NAPC), which is based on an end-to-end LSTM recurrent neural network. NAPC counts the number of boarding and alighting passengers from 3D LiDAR videos obtained by a top-down sensor during door opening phases.

A direct quantitative comparison with other counting algorithms is not possible because they were tested on datasets different from ours in the context of different real-world scenarios, distinguished, e.g., by location (urban/rural), type of vehicle (bus/train/tram), sensor (RGB/RGB-D/depth-only, with various resolutions), number of samples to test, or viewpoints (tow-down/oblique-view). However, it is possible to test whether our approach yields a performance comparable to the performance of other counting algorithms whereby each algorithm is tested in the scenario it was designed for. NAPC counted on average approx. 99% of the boarding and alighting passengers in our test data. This performance is superior to counting 108 of 116 passengers (i.e., approx. 93%, jointly for both directions) reported in the scenario of [31], and counting 107 from 109 passengers (i.e., approx. 98%, jointly for both directions) reported in the scenario of [32]. Note that the authors of [32] report an accuracy of 100% for the boarding direction; however, their test data consists of only 28 boarding

passengers (ours: ranging 4555–5841 boarding passengers, i.e., a roughly 200 times bigger test set). Looking at the performance measure of [33], NAPC obtained on average an absolute relative error of approx. 3% for both boarding and alighting passenger counting, which is superior to all four sub-categories reported in the scenario of [33] (15% absolute relative error for boarding, and 6% for alighting passengers, respectively in the best-performing sub-categories). As we have pointed out, these results must be interpreted with caution as better performance indices could also be explained by other, scenario-dependent factors. However, it can be concluded that the use of deep learning on low-resolution depth-data leads to results competitive with other APC algorithms.

Besides looking at performance metrics, we stress the following advantage of NAPC compared to other deep learning and/or classical methods: Unlike three recently proposed boarding and alighting passenger counting methods [31]–[33], NAPC is based on an end-to-end architecture. No separate background modeling, head detection, or trajectory tracking is required. Our experiments were conducted on a large-scale dataset of approx. 13,000 depth-only recordings with a resolution of 20×25 (or less) pixels. This demonstrates the possibilities to build a privacy-friendly APC system with competitive counting performance based on a deep-learning approach, avoiding unnecessary data collection in the public sphere.

Source Code / Dataset Availability

The source code of the TensorFlow implementation of NAPC is available at https://github.com/nicojahn/open-neural-apc.

The Berlin-APC Dataset [44] can be downloaded at https://dx.doi.org/10.14279/depositonce-12205.3.

References

[1] United Nations General Assembly, *Transforming our world: The 2030 agenda for sustainable development*, A/RES/70/1, 2015.

[2] C. C. Hodges, "Automatic Passenger Counter Systems: The State of the Practice," U.S. Department of Transportation, Tech. Rep., 1985. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/398.

[3] R. Stewart, M. Andriluka, and A. Y. Ng, "End-to-End People Detection in Crowded Scenes," in *2016 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 2325–2333.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comp.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] H. Li, J. Liu, G. Zhang, Y. Gao, and Y. Wu, "Multi-glimpse LSTM with color-depth feature fusion for human detection," in *2017 IEEE Int. Conf. on Image Process. (ICIP)*, IEEE, 2017, pp. 905–909.

[6] W. Zhang, X. Guo, J. Wang, N. Wang, and K. Chen, "Asymmetric adaptive fusion in a two-stream network for RGB-D human detection," *Sensors*, vol. 21, no. 3, p. 916, 2021.

[7] M. Villamizar, A. Martínez-González, O. Canévet, and J.-M. Odobez, "Watchnet: Efficient and depth-based network for people detection in video surveillance systems," in *2018 15th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2018, pp. 1–6.

[8] M. Villamizar, A. Martínez-González, O. Canévet, and J.-M. Odobez, "WatchNet++: efficient and accurate depth-based network for detecting people attacks and intrusion," *Machine Vision and Appl.*, vol. 31, no. 6, article number: 41, 2020.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[10] M. Ahmad, I. Ahmed, and A. Adnan, "Overhead view person detection using YOLO," in *2019 IEEE 10th Annu. Ubiquitous Comput., Electronics Mobile Commun. Conf. (UEMCON)*, 2019, pp. 627–633.

[11] W. Lan, J. Dang, Y. Wang, and S. Wang, "Pedestrian detection based on yolo network model," in *2018 IEEE Int. Conf. on Mechatronics and Automation (ICMA)*, 2018, pp. 1547–1551.

[12] M. Ivašić-Kos, M. Krišto, and M. Pobar, "Human detection in thermal imaging using yolo," in *5th Int. Conf. Comput. and Technol. Appl. (ICCTA 2019)*, 2019, pp. 20–24.

[13] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., ser. Lecture Notes in Computer Science. Cham: Springer, 2016, vol. 9905, pp. 21–37.

[14] M. Ahmad, I. Ahmed, K. Ullah, and M. Ahmad, "A deep neural network approach for top view people detection and counting," in *2019 IEEE 10th Annu. Ubiquitous Comput., Electronics Mobile Commun. Conf. (UEMCON)*, 2019, pp. 1082–1088.

[15] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep people counting in extremely dense crowds," in *23rd ACM Int. Conf. on Multimedia (MM)*, 2015, pp. 1299–1302.

[16] V. A. Sindagi and V. M. Patel, "A survey of recent advances in cnn-based single image crowd counting and density estimation," *Pattern Recognition Letters*, vol. 107, pp. 3–16, 2018.

[17] Z. Li, L. Zhang, Y. Fang, *et al.*, "Deep people counting with faster R-CNN and correlation tracking," in *Proc. Int. Conf. Internet Multimedia Computing and Service (ICIMCS)*, 2016, pp. 57–60.

[18] L. Boominathan, S. Kruthiventi, and R. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," in *24th ACM Int. Conf. on Multimedia (MM)*, 2016, pp. 640–644.

[19] J. H. Kim, H. Hong, and K. Park, "Convolutional neural network-based human detection in nighttime images using visible light camera sensors," *Sensors*, vol. 17, no. 5, p. 1065, 2017.

[20] P. Skrabanek, P. Dolezel, Z. Nemec, and D. Stursa, "Person detection for an orthogonally placed monocular camera," *J. Adv. Transp.*, vol. 2020, pp. 1–13, 2020.

[21] C. Gao, P. Li, Y. Zhang, J. Liu, and L. Wang, "People counting based on head detection combining Adaboost and CNN in crowded surveillance environment," *Neurocomputing*, vol. 208, pp. 108–116, 2016.

[22] Y.-W. Hsu, Y.-W. Chen, and J.-W. Perng, "Estimation of the number of passengers in a bus using deep learning," *Sensors*, vol. 20, no. 8, p. 2178, 2020.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2014, pp. 1–8.

[24] K. Zhang, F. Xiong, P. Sun, L. Hu, B. Li, and G. Yu, "Double anchor R-CNN for human detection in a crowd," 2019. arXiv: 1909.09998.

[25] Y. Wang, J. Wu, and H. Li, "Human detection based on improved mask R-CNN," *J. Physics: Conf. Series*, vol. 1575, p. 12 067, 2020.

[26] S. Sambolek and M. Ivašić-Kos, "Person detection in drone imagery," in *5th Int. Conf. on Smart and Sustainable Technol. (SpliTech)*, 2020, pp. 1–6.

[27] B. Wilie, S. Cahyawijaya, and W. Adiprawita, "CountNet: End to end deep learning for crowd counting," in *2018 5th Int. Conf. on Elect. Eng., Comput. Science and Informatics (EECSI)*, 2018, pp. 128–132.

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807.

[29] J. Donahue, L. Hendricks, M. Rohrbach, *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[30] L. Massa, A. Barbosa, K. Oliveira, and T. Vieira, "LRCN-RetailNet: A recurrent neural network architecture for accurate people counting," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5517–5537, 2021.

[31] G. Liu, Z. Yin, Y. Jia, and Y. Xie, "Passenger flow estimation based on convolutional neural network in public transportation system," *Knowledge-Based Syst.*, vol. 123, pp. 102–115, 2017.

[32] Y. Zhang, W. Tu, K. Chen, *et al.*, "Bus passenger flow statistics algorithm based on deep learning," *Multimedia Tools and Applications*, vol. 79, pp. 28 785–28 806, 2020.

[33] S. Sun, N. Akhtar, H. Song, C. Zhang, J. Li, and A. Mian, "Benchmark data and method for real-time people counting in cluttered scenes using depth sensors," *IEEE Trans. Intelligent Transp. Syst.*, vol. 20, no. 10, pp. 3599–3612, 2019.

[34] C. Slobogin, "Public privacy: Camera surveillance of public places and the right to anonymity," *SSRN Electronic J.*, 2003.

[35] I. Y. Jung, "A review of privacy-preserving human and human activity recognition," *Int. J. on Smart Sensing and Intelligent Syst.*, vol. 13, pp. 1–13, 2020.

[36] E. Newton, L. Sweeney, and B. Malin, "Preserving privacy by de-identifying face images," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 2, pp. 232–243, 2005.

[37] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Trans. Circuits and Syst. for Video Technology*, vol. 18, pp. 1168–1174, 2008.

[38] F. Dufaux and T. Ebrahimi, "A framework for the validation of privacy protection solutions in video surveillance," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2010, pp. 66–71.

[39] P. Korshunov and T. Ebrahimi, "Using face morphing to protect privacy," in *2013 10th IEEE Int. Conf. on Adv. Video and Signal Based Surveillance (AVSS)*, 2013, pp. 208–213.

[40] P. Korshunov and T. Ebrahimi, "Towards optimal distortion-based visual privacy filters," in *2014 IEEE Int. Conf. on Image Process. (ICIP)*, 2014, pp. 6051–6055.

[41] Z. Wu, Z. Wang, Z. Wang, and H. Jin, "Towards privacy-preserving visual recognition via adversarial training: A pilot study," in *2018 European Conf. on Comput. Vision (ECCV)*, 2018.

[42] Z. Ren, Y. J. Lee, and M. S. Ryoo, "Learning to anonymize faces for privacy preserving action detection," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science. Cham: Springer, 2018, vol. 11205, pp. 639–655.

[43] V. Srivastav, A. Gangi, and N. Padoy, "Human Pose Estimation on Privacy-Preserving Low-Resolution Depth Images," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, D. Shen, T. Liu, T. M. Peters, *et al.*, Eds., ser. Lecture Notes in Computer Science. Cham: Springer, 2019, vol. 11768, pp. 583–591.

[44] R. Seidel, D. Zarafeta, M. Siebert, *et al.* "Manual to Berlin-APC: A Privacy-Friendly Dataset for Automated Passenger Counting in Public Transport," Technische Universität Berlin, Interautomation Deutschland GmbH. (2021), [Online]. Available: http://dx.doi.org/10.14279/depositonce-12205.3.

[45] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," 2016. arXiv: 1609.03499.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[47] K. Cho, B. V. Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014. arXiv: 1406.1078.

[48] G. Weiss, Y. Goldberg, and E. Yahav, "On the practical computational power of finite precision RNNs for language recognition," 2018. arXiv: 1805.04908.

[49] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings 30th Int. Conf. Machine Learning*, 2013.

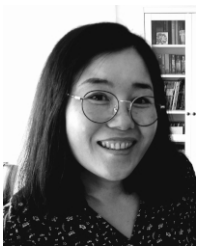
[50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. on Learning Representations*, 2015.

[51] M. Siebert and D. Ellenberger, "Validation of automatic passenger counting: introducing the t-test-induced equivalence test," *Transportation*, vol. 47, no. 6, pp. 3031–3045, 2020.

[52] VDV: Die Verkehrsunternehmen, *Automatische Fahrgastzählsysteme*, 2018. [Online]. Available: https://dms.vdv.de/mitglieder/Documents/457__V2.1_SDS.pdf.

Robert Seidel received his B.S. degrees (2015) in Mathematics and Mathematical Finance from the University of Konstanz, and his M.S. degree (2019) in Mathematics from Technische Universität (TU) Berlin, Germany. From 2019 to 2020, he worked as research assistant at the Neural Information Processing Group at TU Berlin. His interests include mathematical signal processing, compressed sensing, and machine learning. He is currently pursuing a M.A. degree in Theory and History of Science at TU Berlin with a focus on ethics of technology.

Nico Jahn received his B.S. degree in Computer Science from Technische Universität Berlin, Germany in 2019, where he is currently pursuing his M.S. degree in Computer Science with a focus on machine learning. Since August 2019, he has been working as a software engineer for Interautomation Deutschland GmbH in Berlin. His interests include machine learning methods, DevOps, and MLOps.



Sambu Seo holds a B.A. degree in Sociology from Ewha Woman University in South Korea and a M.S. degree in Computer Science from Technische Universität (TU) Berlin, Germany. She received her Ph.D. degree from the Department of Computer Science and Electrical Engineering at TU Berlin in 2005. From 2000 to 2001, she worked as a researcher for the Cortologic AG in Berlin. From 2002 to 2018, she was a member of the Neural Information Processing Group at TU Berlin. From May 2019 to May 2021 she worked as a data scientist for Interautomation Deutschland GmbH in Berlin. Currently, she is a visiting researcher at TU Berlin. Her research interests include machine learning methods and applications, e.g. to biomedical data, medical imaging data, and public transport data.

Thomas Goerttler received his B.S. degree (2015) in IT-Systems Engineering from the Hasso-Plattner Institute in Potsdam, Germany, and his M.S. degree (2017) in Statistics jointly from the Humboldt-Universität zu Berlin, Freie Universität, and Technische Universität (TU) Berlin, Germany. From 2017 to 2018, he worked as a research assistant at the Hasso-Plattner-Institute, dealing with missing values in data mining problems. Since 2018, he has been part of the Neural Information Processing Group at TU Berlin. He is currently pursuing a Ph.D. doing research on the quantification of deep neural networks.

Klaus Obermayer received the Diplom degree in Physics in 1987 from the University of Stuttgart, Stuttgart, Germany, and the Dr. rer. nat. degree from the Department of Physics, Technical University of Munich, Munich, Germany, in 1992. From 1992 and 1993, he was a Postdoctoral Fellow with the Rockefeller University, New York, USA, and the Salk Institute for Biological Studies, La Jolla, USA. From 1994 to 1995, he was member of the Technische Fakultät, University of Bielefeld, Germany. He became Associate Professor in 1995 and Full Professor in 2001 with the Department of Electrical Engineering and Computer Science, Technische Universität Berlin, Germany. He is head of the Neural Information Processing Group and member of the Steering Committee of the Bernstein Center for Computational Neuroscience Berlin. He was a member of the Governing Board of the International Neural Network Society from 2004 to 2012 and was Vice-President of the Organisation for Computational Neuroscience from 2008 to 2011. From 1999 to 2003, he was one of the Directors of the European Advanced Course of Computational Neuroscience. He has co-authored more than 300 scientific publications. His current research interests include computational neuroscience, artificial neural networks and machine learning, and the analysis of neural data.