University of Amsterdam
Informatics Institute

MSc Artificial Intelligence
Master Thesis

Crossing-Line Counting with Deep Spatial-Temporal Feature Learning

by
Jan Erik van Woerden
11033711

April 28, 2021

48 European Credits
February 2020 - January 2020

External Supervisor:
Thomas W.D. Jongstra

Internal Supervisor:
Zenglin Shi

Assessor:
Dr. Thomas E.J. Mensink

Abstract

This thesis presents a new method to count objects crossing a line in videos. Specifically, we decompose the raw counting task into two sub-tasks, supervised density map estimation, and unsupervised optical flow estimation. We predict the optical flow to obtain the velocity in an unsupervised way, avoiding the time-consuming labeling work. To obtain the count more accurately, we propose to estimate the density map with a spatial-temporal feature warping approach. We introduce a network architecture that links the flow and density estimation. When optimized jointly, both tasks learn from each other.

Additionally, we outline the misalignment problem, an aligning problem between the unsupervised-trained velocity map and the supervised-trained density map. We propose to solve this problem by applying a maxing filter to increase the velocity map area of each moving object.

We introduce three datasets by relabeling three public datasets with our developed labeling tool to evaluate the proposed methods. The experimental results show that our deep spatial-temporal feature learning method outperforms the existing methods. Leveraging temporal information improves the counting of temporarily occluded objects, and solving the spatial differences by warping improves the counting of moving objects. Lastly, a small study is done and shows the usability of our approach in real-world systems.

A Github repository is provided with code and instructions: `https://github.com/ijanerik/Thesis`

Acknowledgements

The process of creating and writing my thesis was long and hard. I would thank all the people who surrounded me during this process for all the support. I want to thank Thomas Jongstra for all the weekly meetings and for giving structure to this long process. I want to thank Zenglin Shi for all the support in steering the thesis to good scientific research and giving me the opportunity to publish a paper. I want to thank the City of Amsterdam for providing me a nice internship and showcasing the real-world applications of the researched methods. Lastly, I want to thank the Robolab for providing a server to run all experiments and a space to work and brainstorm with other students.

Contents

Chapter 1

Introduction

1.1 Background and related work

In recent years the amount of surveillance cameras has increased immensely to the point that it is hard to supervise them all manually by humans. Therefore, research has been done on developing approaches to automatically extract interesting information from those cameras [Sreenu and Saleem Durai, 2019]. In this thesis, we are interested in counting the numbers of objects crossing a line in the surveillance cameras.

Counting the number of objects in the images, namely, object counting, has been widely studied [Wang et al., 2020, Li et al., 2018, Fang et al., 2019a, Liu et al., 2019c]. While object counting only focuses on counting the exact number of objects in a frame, it does not consider the number of objects passing by overtime. This is no issue when cameras are present in the whole area of interest. However, with large areas, the number of required cameras increases quickly, which can become an issue. For example, it would be much more convenient to count the number of customers inside a shop by only tracking the customers walking inside and outside the shop, rather than having cameras in the whole shop.

Therefore, crossing-line counting is introduced, which counts the objects crossing a line in videos [Zhao et al., 2016, Zheng et al., 2019]. The traditional crossing-line counting methods focus on temporal-slicing using traditional feature extractors and regression models [Ma and Chan, 2013, Ma and Chan,]. Temporal-slicing reduces the temporal information to a single image, which reduces the complexity of the data. However, due to different directions and speeds of the objects, artifacts are introduced in the sliced-image, which makes it harder to track objects [Zhao et al., 2016]. To solve the disadvantages of temporal-slicing-based methods, recent works [Zhao et al., 2016, Zheng et al., 2019] propose a direct frame-by-frame prediction method in which both location and direction of objects are obtained separately before counting the objects crossing lines. However, the frame-by-frame prediction methods couldn't leverage the useful temporal information well, leading to an inferior counting performance in some scenarios like occlusion.

1.2 Contributions

This thesis presents a new method to count objects crossing a line in videos. Specifically, we decompose the raw counting task into two sub-tasks, supervised density map estimation and unsupervised optical flow estimation. We predict the optical flow to obtain the velocity in an unsupervised way, avoiding the time-consuming labeling work. To obtain the count more accurately, we propose to estimate the density map with a spatial-temporal feature warping approach. We introduce a network architecture that links the flow and density estimation. When optimized jointly, both tasks learn from each other. Additionally, we outline the misalignment problem, an aligning problem between the unsupervised-trained

velocity map and the supervised-trained density map. We propose to solve this problem by applying a maxing filter to increase the velocity map area of each moving object.

To evaluate the proposed methods, we introduce three datasets by relabeling three public datasets with our developed labeling tool. The experimental results show that our deep spatial-temporal feature learning method outperforms the existing methods. Leveraging temporal information improves the counting of temporarily occluded objects, and solving the spatial differences by warping improves the counting of moving objects. Lastly, a small study is done and shows the usability of our approach in real-world systems.

1.3 Thesis outline

The rest of this thesis is divided into the next chapters:

- **Background and related work**, explains several fields and related works to understand the starting point for this thesis.
- **Method**, presents the method of the proposed solution.
- **Experiments and results**, presents the used datasets and used approach to label the proposed new datasets, shows the hyperparameters and evaluation methods, and discusses the results of the experiments.
- **Conclusion**, wraps it up and summarizes what we can conclude.

Chapter 2

Background and Related work

In this chapter, we introduce some background knowledge to give a basis to understand the rest of this thesis.

2.1 Deep Convolutional Neural Network

Deep convolutional neural networks have become one of the most successful neural networks, widely used in computer vision, image processing, natural language processing, and speech processing. Compared to classical neural networks using fully connected layers, convolutional neural networks consists of convolutional layers and pooling layers.

A convolutional layer contains several convolution filters that each take the previous layer's output as input and stacks the convolution's features maps as input for the next layer. A convolution filter is convolved over the input. This results in an abstract feature map concerning the respective filter over the input. For every pixel in the input, the local pixels are multiplied with the filter, and the sum is taken. These filters share the same weights for convolving over different pixels within an input feature map. Convolutional neural works have fewer parameters with convolution and weight sharing and achieve the translation variance compared to a fully connected neural network.

Each convolutional layer is followed by a non-linear activation function such as a Sigmoid, ReLU, or LeakyReLU [Xu et al., 2015]. After the activation function, a pooling layer is used. A pooling layer reduces the dimensions of the feature maps by summarizing small parts of the feature maps, which can be done by, for example, max pooling or average pooling.

To optimize deep convolutional neural networks, a backpropagation algorithm is employed. Backpropagation minimizes a loss function, which penalizes the layers when an incorrect prediction is made. Every training iteration, the network's weights get updated until a minimum in the loss function is reached or till otherwise.

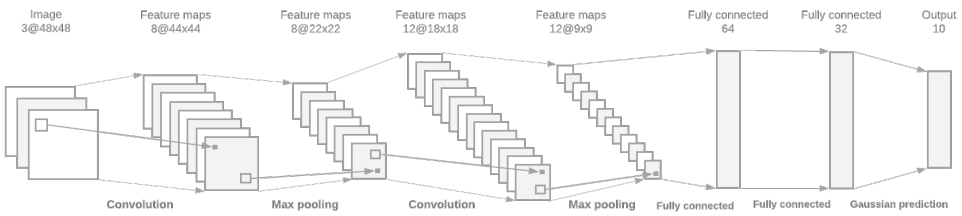


Figure 2.1: Sample of a convolutional neural network with convolutional layers for feature extraction and fully connected layers for prediction

2.2 Region of Interest Counting

Region of Interest (ROI) counting is a widely studied counting problem. Its goal is to estimate the number of pedestrians in a single image or video.

In the early days of ROI counting, several *detection-based* methods have been proposed to estimate the number of pedestrians [Dalal and Triggs, 2005, Dollár et al., 2012]. Some methods tried to detect only the head [Subburaman et al., 2012], and others tried to focus on general part detection [Wu and Nevatia, 2007, Lin and Davis, 2010]. These methods rely on individually detecting the pedestrians. This becomes much harder when the occlusion of the pedestrians starts to happen. This is why these methods' performance starts to degrade when the density of the pedestrians in an image starts to increase.

Later works introduced a *regression-based* solution, which tries to predict the number of pedestrians in crowd blobs [Chan and Vasconcelos, 2009, Idrees et al., 2013, Zheng et al., 2019]. The count inside crowd blobs is predicted using SVM or other regressor methods, and several features such as the number of foreground pixels of the blob and detected key points. Regression-based solutions improved over the detection methods but still lack the capabilities to estimate pedestrian counts in highly occluded areas.

Figure 2.2: Example of generated density map on the right side, for the left image

Instead of directly regressing the total count, Lempitsky *et al.* [Lempitsky and Zisserman, 2010] proposed to regress a density map with point annotations. A *density map* (Figure 2.2) is a map which represents the density of pedestrians of each pixel. The density map is generated by taking the locations of each pedestrian ($p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$ in equation 2.1) and place those locations on the the density map. Individual dots are very hard for the regression model to detect correctly and are prone to errors. To circumvent this, a Gaussian-shaped circle is created around the labeled location, still with a sum of 1. The amount of pedestrians in the frame can be extracted from the density map by taking the sum over all the pixels of the density map (Equation 2.2, where $D_t(p)$ is the density for location $p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$ for trainings frame t).

$$D_t(p) = \frac{1}{2\pi\sigma_p^2} \sum_{p \in P} e^{\frac{(x_p - x)^2 + (y_p - x)^2}{-2\sigma_p^2}} \tag{2.1}$$

$$C_t = \sum_{p \in P} D_t(p) \tag{2.2}$$

Early deep network-based methods for density estimation were mostly multi-column models [Zhang et al., 2016], which use different filter sizes to detect pedestrians of different sizes. With CSRNet [Li et al., 2018] dilation filters were introduced. Dilation filters enlarge the filter area without increasing the number of parameters. With dilation filters,

the necessity of using multi-column models was gone. Other works focused on improving the performance using extra information [Shi et al., 2018, Shi et al., 2019, Liu et al., 2019c], such as global density [Shi et al., 2019] and variable Gaussians for density map generation [Zhang et al., 2016, Li et al., 2018, Wan and Chan, 2019].

Several papers focused on video-based counting [Xiong et al., 2017, Zhang et al., 2017, Fang et al., 2019b, Shi et al., 2020] and take temporal information into account to better handle occlusion. This was done using several methods such as LSTM [Xiong et al., 2017], Transformer [Fang et al., 2019b] and warping [Shi et al., 2020].

2.3 Line of Interest Counting

Figure 2.3: Example of LOI, orange: Line of Interest, blue: located pedestrians

Line of Interest (LOI) counting is another counting task, close to (ROI) counting. Where ROI counting aims to count the number of people inside a specified region. LOI counting aims to count the number of pedestrians crossing a specified line during a certain time.

Early LOI works focuses on temporal-slicing using traditional feature extractors [Ma and Chan, 2013, Ma and Chan,] or a Neural Network [Cao et al., 2015, Denman et al., 2018]. Using temporal-slicing, a single image is obtained by taking a set of consecutive frames, slicing a small area around the LOI, and stitching the slices together. Temporal-slicing reduces the temporal information to a single image, which reduces the complexity of the data. However, due to different directions and speeds of the pedestrians, artifacts are introduced in the sliced-image, which makes it harder to track pedestrians [Zhao et al., 2016].

To solve the disadvantages of temporal-slicing-based methods, recent works [Zhao et al., 2016, Zheng et al., 2019], propose a direct frame-by-frame prediction method, in which both location and direction of pedestrians are obtained separately before counting the actual LOI crossings. [Zhao et al., 2016] introduces a unified network, predicting both density map and velocity map and obtaining the LOI countings by a pixel-level multiplication.

2.4 Optical Flow Estimation

Optical flow estimation tries to capture the optical flow of objects, which is the movement of objects in a sequence of frames, such as a video. It is one of the building blocks of many applications [Menze and Geiger, 2015, Simonyan and Zisserman, 2014, Shi et al., 2020] and widely researched.

Since Horn and Schunck [Horn and Schunck, 1981], a lot of papers [Mémin and Pérez, 1998, Bruhn et al., 2005] are published on estimating flow. Sparse optical flow estimation was traditionally more popular [Horn and Schunck, 1981, Mémin and Pérez, 1998], whereas dense

optical flow estimation was computationally more expensive and less researched. Early dense optical flow estimation was often extrapolated from key points, which reduces accuracy on borders of objects. With the introduction of learnable approaches [Pock et al., 2008, Wedel et al., 2009], dense optical flow is more studied due to the direct supervision for learnable parameters.

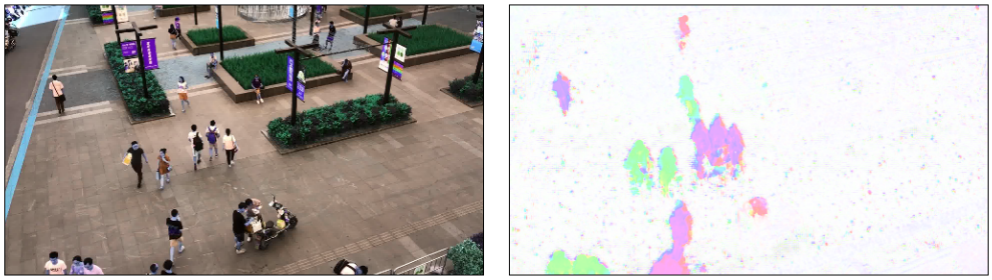


Figure 2.4: Example of generated velocity map on the right side, for the left image

Recent works introduce convolutional network-based models [Dosovitskiy et al., 2015, Ilg et al., 2016, Ranjan and Black, 2017, Hui et al., 2018] to directly predict a dense optical flow map (*Velocity Map*) as in figure 2.4. Due to the complexity of optical flow estimation, several papers show that instead of fully relying on convolutional neural networks, intermediate processing improves results, such as adding correlation [Dosovitskiy et al., 2015] and warping [Sun et al.,] (Equation 2.3).

$$F_{t+1} = Warp(F_t, V_t) \tag{2.3}$$

Due to the dense supervision of these end-to-end models, labeling optical Flow datasets is very hard. It is almost impossible to be created using real-world footage without introducing errors [Dosovitskiy et al., 2015]. Therefore, most of the flow estimation benchmarks are generated videos by computer 3D-engines, making it possible to generate pixel-perfect dense flows based on the engine's generated videos. Due to the large gap in domain and scene between the generated datasets and real-world applications [Liu et al., 2008], supervised papers [Dosovitskiy et al., 2015, Sun et al.,] tend to overfit on these datasets and therefore perform rather poor on real-world applications. One promising direction is unsupervised learning [Yu et al., 2016, Janai et al., 2018, Liu et al., 2019a, Liu et al., 2019b]. Early papers only predicted non-occluded pixels [Yu et al., 2016, Janai et al., 2018], but recent papers use methods to estimate occluded pixels as well [Liu et al., 2019a, Liu et al., 2019b]. These papers tend to use existing supervised flow estimation models with a specialized loss function [Yu et al., 2016, Janai et al., 2018, Liu et al., 2019a] which use a photometric loss to optimize for equation 2.3.

Chapter 3

Method

In this section, we introduce our method for crossing-line counting. We decompose the raw counting task into two sub-tasks, density map estimation, and optical flow estimation. We predict the density map to obtain the count and predict the optical flow to obtain the velocity. To avoid time-consuming labeling work, we propose to estimate optical flow in an unsupervised way. To more accurately estimate the density map, we propose a spatial-temporal feature warping approach. We introduce a network architecture that links the flow and density estimation, as illustrated in Figure 3.1. When optimized jointly, both tasks learn from each other.

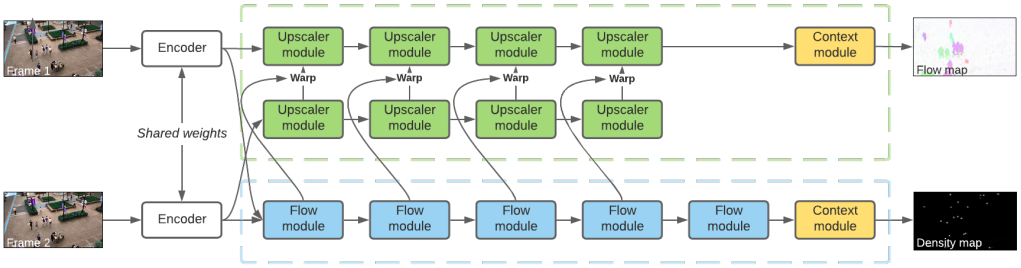


Figure 3.1: Proposed model with spatial-temporal warping

Let $\{I^{(1)}, I^{(2)}, ..., I^{(T)}\}$ be a sequence of frames, where $I^{(t)}$ is the frame for timestamp $t \in [1..T]$. Let the LOI be defined as the line between p_1 and p_2, which is constant for all $I^{(t)}$, and let v_{perp} be the normalized directional vector perpendicular to the LOI, as in $(p_1 - p_2 \perp v_{perp}) \wedge \|v_{perp}\| = 1$. Let M_s be the area around each side of the LOI (Figure 3.2), with d as the width of the area, where s defines $1, 2$ for each side of the crossing. Let $\{\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, ..., \mathcal{D}^{(T)}\}$ be the obtained density maps and $V^{(1)}, V^{(2)}, ..., V^{(T-1)}\}$ the obtained velocity maps. Let $Q^{(t)}$ be a projection of $V^{(t)}$ on v_{perp}, as in equation 3.1.

$$Q^{(T)}(p) = V^{(T)}(p) \cdot v_{perp}. \tag{3.1}$$

The line-crossings can be obtained by first calculating the per-frame crossings as $\{c_s^{(1)}, c_s^{(2)}, ..., c_s^{(T-1)}\}$, which can be obtained via equation 3.2, and finally sum per side, $C_s = \sum_{t=1}^{T} c_s^{(t)}$. $c^{(t)}$.

$$\begin{aligned} c_1^{(t)} &= \sum_{\{p \in M_1 | Q^{(t)}(p) > 0\}} \mathcal{D}^{(t)}(p) \cdot \frac{Q^{(t)}(p)}{d} \\ c_2^{(t)} &= \sum_{\{p \in M_2 | Q^{(t)}(p) < 0\}} \mathcal{D}^{(t)}(p) \cdot \frac{-Q^{(t)}(p)}{d} \end{aligned} \tag{3.2}$$

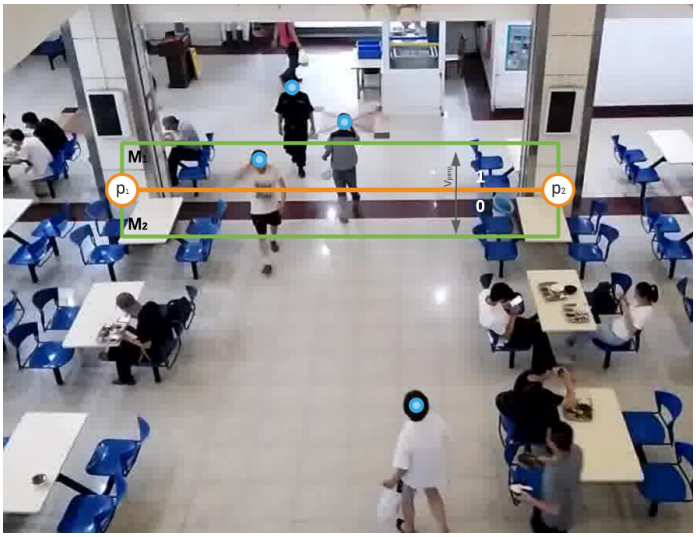


Figure 3.2: Example of LOI, orange: LOI line by p_1 and p_2, green: LOI areas by M_1 and M_2, grey: v_{perp}

3.1 Unsupervised Optical Flow Estimation

We estimate the optical flow estimation in an unsupervised manner. Although the supervised optical flow estimation approaches have achieved remarkable performance, they still require significant labeling, and few datasets have been labeled. By using the unsupervised optical flow estimation method, labeling is no longer required.

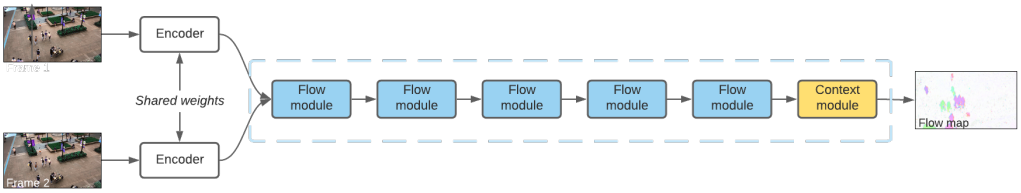


Figure 3.3: PWCNet architecture

3.1.1 Network architecture

We use the PWCNet to estimate the optical flow (Figure 3.3). The PWCNet network is a small two-frame-input flow estimator, which at the time of publication was the best performing supervised flow estimator and, to the best of our knowledge, still the best performing two-frame model currently published. At the time of publication, the PWCNet had one of the fastest inferences compared to other models. PWCNet uses a u-shaped network, which utilizes skip connections between the encoder and decoder.

Flow module

Each flow module (Figure 3.4), $F_{(l)}$, uses warping, correlation and a cost volume to refine the intermediate estimated flow estimation. The 5 convolutional layers (with LeakyReLU) are a dense architecture [Huang et al., 2017], where each layer gets the outputs of each of the earlier layers in the module, resulting in the cost volume.

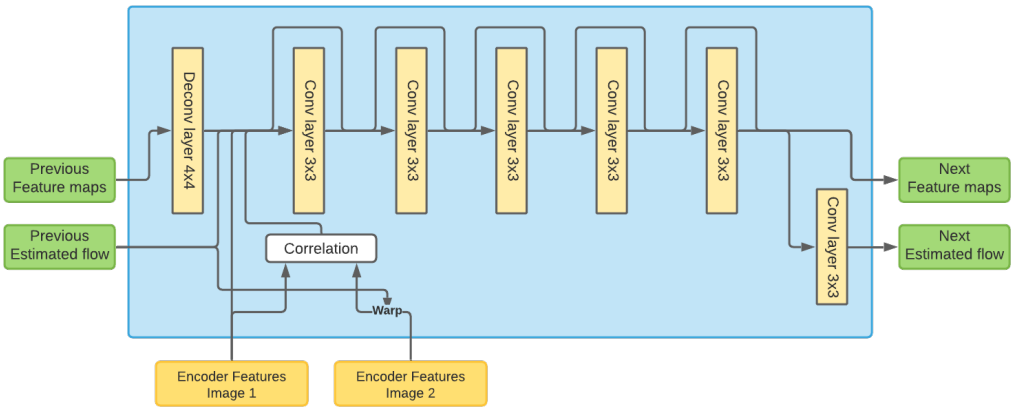


Figure 3.4: Flow module

Spatial Context module

The context module (Figure 3.5) contains of 6 convolutional layers (with LeakyReLU) and a final convolutional layer to predict the velocity/density map. Four of the layers use higher dilation for better spatial awareness.

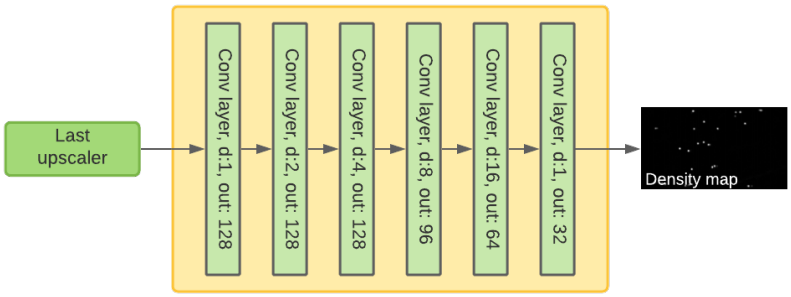


Figure 3.5: Spatial context module

Encoder module

For each image a simple encoder is used for feature extraction (Figure 3.6). Each encoder contains 18 convolutional layers (with LeakyReLU) separated in blocks of 3 convolutional layers. Each block downsamples the features by applying a stride of 2. After each block, a skip connection to the decoder is applied.

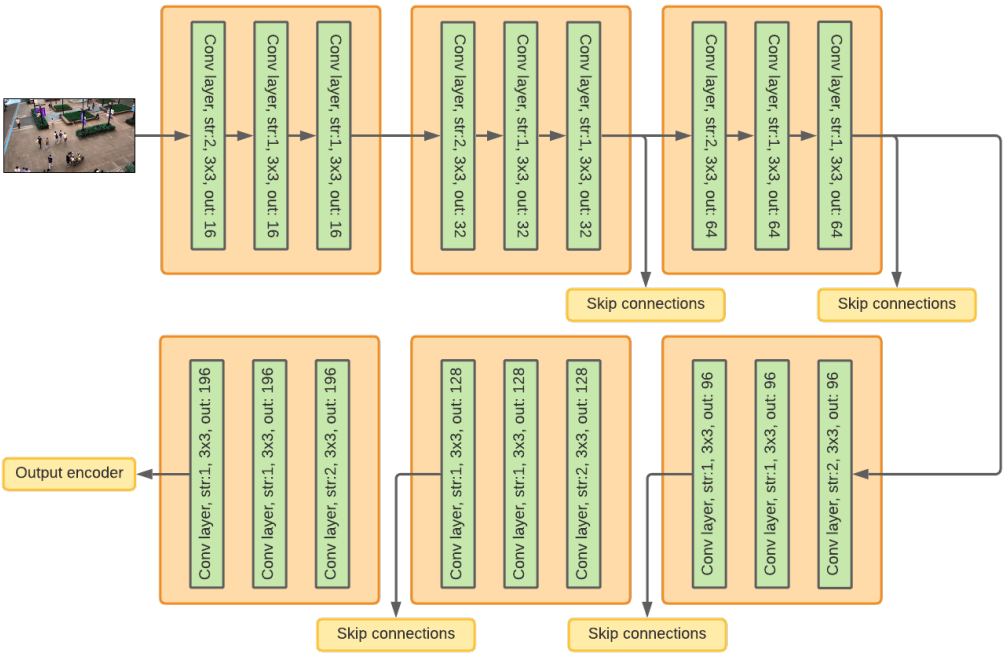


Figure 3.6: Encoder

3.1.2 Loss function

The photometric loss [Yu et al., 2016, Janai et al., 2018] as in equation 3.3 is used for unsupervised learning.

$$L_p = \sum_t \psi\left(I^{(t)} - I_{bw}^{(t+1)}\right) + \sum_t \psi\left(I^{(t+1)} - I_{fw}^{(t)}\right) \tag{3.3}$$

For L_p in equation 3.3, let $I_{bw}^{(t+1)}$ and $I_{fw}^{(t)}$ be two warped images based on estimated velocity. For $I_{bw}^{(t+1)}$ the velocity map of $I^{(t)} \rightarrow I^{(t+1)}$ is calculated $(V_{fw}^{(t)})$ and then used to backwarp $I^{(t+1)}$ to $I^{(t)}$. Same, for $I_{fw}^{(t)}$ the velocity map of $I^{(t+1)} \rightarrow I^{(t)}$ is calculated $(V_{bw}^{(t)})$ and then used to backwarp $I^{(t)}$ to $I^{(t+1)}$. Let $\psi(x)$ be a robust loss function: $\psi(x) = (|x|+\epsilon)^q$.

3.2 Density map estimation with spatial-temporal warping

In high-density frames, pedestrians can be temporarily occluded by other pedestrians in $I^{(t)}$ and visible again in $I^{(t+1)}$. Therefore both $I^{(t)}$ and $I^{(t+1)}$ are provided for temporal context in the proposed decoder. However, due to the temporal difference, there is a spatial difference as well between both frames. This may confuse the density estimator and could count pedestrians with a spatial difference incorrectly. These spatial differences can be solved by warping based on the estimated Optical Flow.

3.2.1 Network architecture

The density map estimation network has an encoder-decoder architecture. The encoder of the flow estimation network is shared between both tasks. On top of the encoder, a new

decoder network branch for density estimation is added. The added density decoder contains several upscaler modules, several warping modules, and a final context module as in figure 3.1. Next, the two new types of modules are shortly introduced. The module details are provided in supplementary materials.

Upscaler module

Each upscaler module (Figure 3.7) contains a deconvolutional layer and two convolutional layers, all followed by a LeakyReLU activation. After the deconvolutional layer, the context features are added. The number of filters used inside the module is based on the upscaler level, 196, 128, 96, 64, respective.

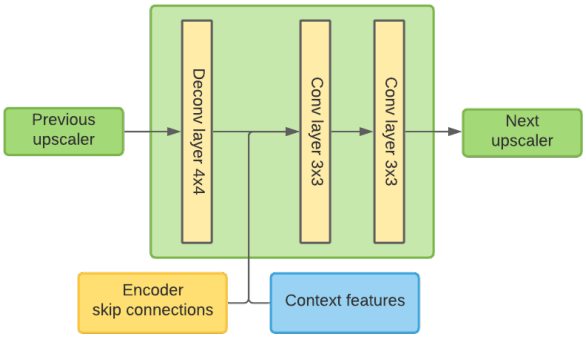


Figure 3.7: Density upsampler module

Warping module

Let $I^{(t)}, I^{(t+1)}$ be two consecutive RGB frames and let $V^{(t)}$ be the optical flow map from $I^{(t)}$ to $I^{(t+1)}$. Let $F(I^{(t)})$ and $F(I^{(t+1)})$ be the spatial feature maps of $I^{(t)}$ and $I^{(t+1)}$ for estimating their density map. We warp features of frame $I^{(t+1)}$, , $F(I^{(t+1)})$, toward the frame $I^{(t)}$ using the flow map $V^{(t)}$:

$$\hat{F}(I^{(t+1)}) = Warp(F(I^{(t+1)}), V^{(t)}), \tag{3.4}$$

Where $\hat{F}$ denotes the warped features. $V^{(t)}$ is obtained from the flow module in the decoder of optical flow estimation. We use bi-linear interpolation to implement the warping operation as [Gadde et al., 2017, Sun et al., 2018]. Then we combine the warped features $\hat{F}(I^{(t+1)})$ and spatial features $F(I^{(t)})$ by concatenation for the density estimation of frame $I^{(t)}$.

3.2.2 Loss function

For the density estimation, the $L2$ loss is a common choice, but it is also known to be sensitive to outliers, which hampers generalization [Belagiannis et al., 2015]. We prefer to learn the density estimation branch by jointly optimizing the $L2$ and $L1$ loss, which adds robustness to outliers:

$$L_c = \sum \parallel \mathcal{D} - Y \parallel_2^2 + \sum \parallel \mathcal{D} - Y \parallel_1, \tag{3.5}$$

where $\mathcal{D}$ denotes the predicted density map, and Y denotes the ground truth density map, which can be obtained by convolving the annotated points with a Gaussian kernel $\mathcal{N}(p|\mu{=}P, \sigma_P^2)$. Here p denotes a pixel location, P denotes a single point annotation. σ_P^2 is the isotropic covariance of the Gaussian kernel.

A unified loss is used to train both decoders in parallel (eq. 3.6). For the unsupervised flow estimation, the photometric loss (eq. 3.3), L_v, is used. For the density map estimation, an L2 loss, L_c, is used. λ is a scalar for equal training.

$$L_t = L_v + \lambda \cdot L_c \tag{3.6}$$

3.3 Maxing filter for misalignment handling

A maxing filter is a filter that takes the maximum in a local area. Applied on a velocity map, the area of velocity of moving objects is enlarged to solve the misalignment handling as in figure 3.8.

In [Zhao et al., 2016], a supervised method of aligning the velocity map and density map is used. However, due to the unsupervised Flow Estimation, it is not possible to use this alignment. As shown in figure 3.8, this misalignment is an issue when multiplying the velocity map and density map in a pixel-wise matter. The blobs predicted in the density map are often much bigger than the flow of the pedestrian predicted. Additionally, for several Crowd datasets, the pedestrian's tagging is done on the head, which is why the blobs are often predicted over the pedestrian's head. The maxing filter can solve these misalignments. The maxing filter is described as the equation 3.7, where R_p is a small region around location p.

$$V^{(t)}(p) = \max_{i \in R_p}(V^{(t)}(i)) \tag{3.7}$$

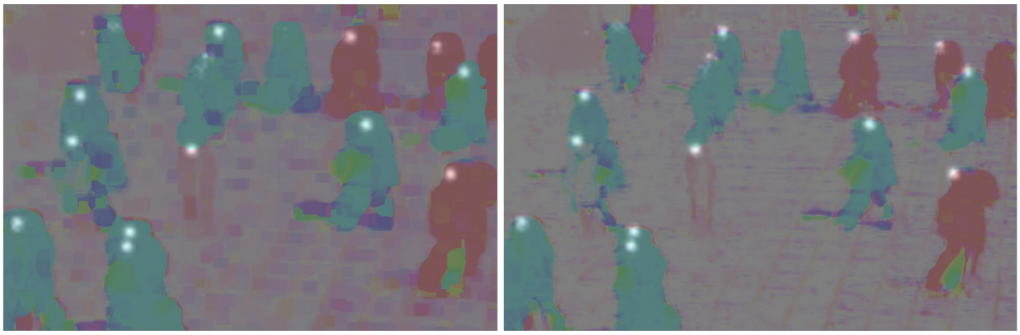


Figure 3.8: Left: Velocity map with maxing filter, Right: Velocity map without maxing filter applied

Chapter 4

Experiments and Results

4.1 Datasets

In this section, the used datasets are explained in detail. Three public datasets are used in our experiments. For both training and evaluation, each pedestrians' location is required to generate the ground-truth density maps, $\mathcal{D}_{gt}^{(t)}$. The ground-truth line-crossing counts, $C_{gt,s}$, are required as well for evaluation. Some of these datasets lack the line-crossing counts. To compensate for the lack of some required labeling, a specialized tool is developed for labeling.

4.1.1 Labeling software

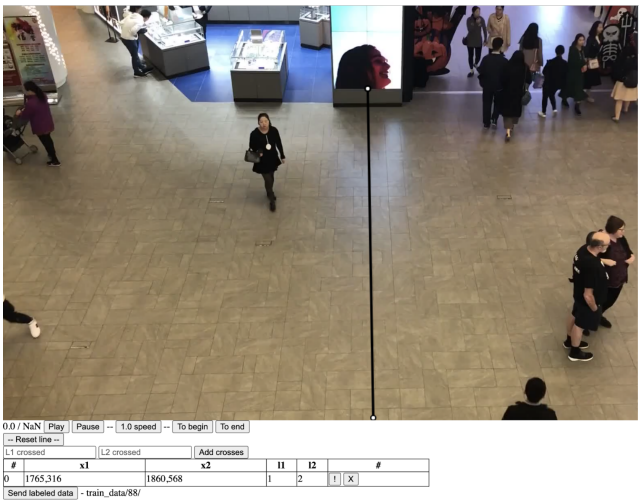


Figure 4.1: User interface of the labeling software

Due to the lack of trajectories and line-crossing counts in the datasets, a special tool is created to label the test videos (Interface is shown in figure 4.1). The labeling software loads an unlabeled video and allows the labeler to draw multiple lines in the video. Both p_1 and p_2 of the line can be defined by clicking in the video; per line, the line-crossing counts can be labeled. The user has several options for manipulating the video, such as scrolling through the video at several speeds.

4.1.2 Fudan-ShanghaiTech

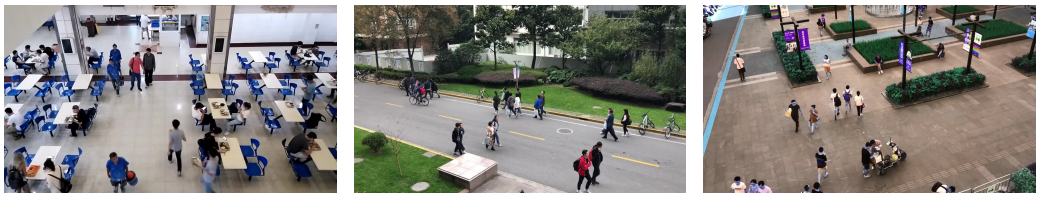


Figure 4.2: Several samples of the Fudan-ShanghaiTech dataset

The Fudan ShanghaiTech dataset [Fang et al., 2019a] is a public dataset with 100 videos of 13 different scenes. Each video contains 6 seconds of footage at 25 fps with a resolution of 1920x1080 or 1280x720. The scenes have between 20-100 pedestrians per frame. For each frame, the pedestrians are labeled with a head-annotation for density map generation. The dataset contains 60 training videos and 40 testing videos.

No line-crossing counts are provided; therefore, all test videos are labeled by the labeling tool. For each video, 1-4 lines are labeled with their respective line-crossing counts.

4.1.3 CrowdFlow

Figure 4.3: Several samples of the CrowdFlow dataset

CrowdFlow is a synthetic dataset generated using the Unreal game engine. The dataset contains 10 videos, which is reduced to 4 videos for training and testing due to errors and moving cameras. Each sequence has a duration of 12.5 seconds at an FPS of 25. The CrowdFlow dataset sequences contain a large number of pedestrians ranging between 200-1000. For each pedestrian, the position and trajectory are labeled.

For each sequence, 3 lines are drawn with line-crossing counts ranging between 20-150 pedestrians per line. Due to the data's sparsity, only 20% of the data (10% at the start and 10% at the end of the sequence) is used for training, and the rest is used as testing.

4.1.4 AI City Challenge

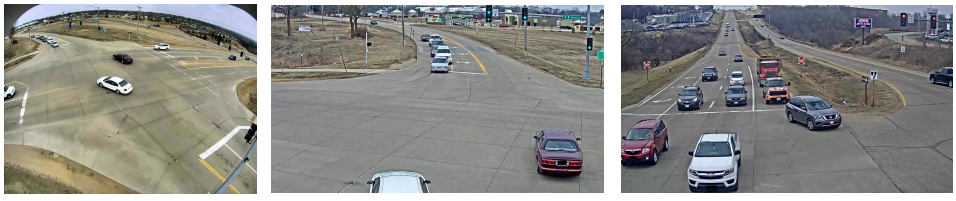


Figure 4.4: Several samples of the AI City Challenge dataset

The AI City Challenge dataset is a large dataset of camera footage along roads with passing cars and trucks. The total dataset contains 20 different scenes, but only the four busiest

scenes are used for training and testing. Each scene contains 3.5 minutes of footage in 10FPS, where the first half of the scene is used as training and the second half for testing. Each vehicle is labeled as a bounding box, which can be used for the center-point. Each vehicle's trajectory is provided to obtain the line-crossing counts for a specified line automatically. For each scene, 1-2 lines are labeled.

4.2 Implementation details

4.2.1 Method and Environment

Optimizer

For all the experiments, the Adam optimizer [Kingma and Ba, 2015] is used. For the models, a start learning rate of $2 \cdot 10^{-4}$ is used, and every 35 epochs the learning rate halves, and regularization of 10^{-4} is used. Also, a maximum of 400 epochs is applied. Due to exploding loss in CSRNet, a start learning rate of $2 \cdot 10^{-5}$ is used, and a maximum of 1000 epochs is applied.

Unified loss

During training, the unified loss (Equation 3.6) is used for training using a λ of 50. At the start, it focuses mainly on the density map loss and, after some initial training, produces an equal loss distribution between the velocity map loss and density map loss.

Augmentation

Several augmentations are applied to the training samples, similar to the augmentations used in [Li et al., 2018]. A crop between $1/3$ and $1/6$ of the total image size is made and resized to $1/4$ of the original image. Half of the time, it is horizontally flipped as well.

Density kernel

To obtain the ground truth density map from the provided point annotations, we use a Gaussian kernel to smooth the annotation maps. For all the different datasets, a fixed kernel size is used of $\sigma = 8$.

Line Crossing

In the equation 3.2 all the parameters for the Line Crossed are shown. The width of the line (parameter d) is the last parameter that is not defined. In preliminary results, the performance difference with changing the parameter is minimal. Therefore during all the experiments, a width of 20px is used.

Maxing filter

During experiments, the maxing filter is optimized per dataset. For the Fudan-ShanghaiTech dataset, a maxing filter is applied with a distance of 16px, optimized in implementation. For CrowdFlow, a distance of 6px is applied, and for AICity, a distance of 12px.

Training Server

For training of all the models, the Robolab in the UvA provided me with a workstation. This workstation contains a powerful NVidia RTX 2080Ti with 11GB of VRAM, an Intel i9-9900 CPU with 8 cores and 16 threads, and a total of 64GB of RAM.

4.2.2 Metrics

For both the ROI and the LOI, the Mean Absolute Error and the Root Mean Squared Error are used. Additionally, the LOI uses the Mean Absolute Percentage Error. The MAE gives a good indicator of the model's performance, whereas the RMSE indicates the number of outliers comparing the MAE. MAPE is more robust for differences in the ground-truth. The performance in crowded and non-crowded areas is normalized.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|C_i - P_c^{(i)}| \tag{4.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(C_i - P_c^{(i)})^2} \tag{4.2}$$

For the ROI the MAE and the rMSE are defined as equation 4.1 and 4.2. Where $P_c^{(i)}$ is the predicted density map for the given frame.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|G_l^{(i)} - P_l^{(i)}| + |G_r^{(i)} - P_r^{(i)}| \tag{4.3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(G_l^{(i)} - P_l^{(i)})^2 + (G_r^{(i)} - P_r^{(i)})^2} \tag{4.4}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|G_l^{(i)} - P_l^{(i)}| + |G_r^{(i)} - P_r^{(i)}|}{G_l^{(i)} + G_r^{(i)}} \tag{4.5}$$

For the LOI, the MAE and RMSE are defined as quation 4.3 and 4.4. The MAPE is simply defined as in equation 4.5. Where $G_l^{(i)}$ is the ground truth for sample i for side left-to-right. And $P_r^{(i)}$ is the predicted value for right-to-left.

4.3 Results

4.3.1 Baseline models

In the experiments, four baselines are compared with the proposed model and several modifications with extra context for the proposed model.

Baseline 1

The first model is proposed in [Zhang et al., 2016]. Because this model assumes a supervised Flow Estimation, parts of the proposed method cannot be used. Therefore, the model is used as a baseline. The proposed model uses FlowNet with only an extra Conv-layer to predict the density map and the velocity map. For the baseline, the FlowNet model is replaced with the PWCNet model. Due to the network's full shared nature, the assumption is that the network will perform poorly when predicting both the density map and the unsupervised flow map. Therefore we propose a second baseline as well.

Baseline 2

This second baseline is a combination of two independent models (Figure 4.5). CSRNet [Li et al., 2018] for predicting the density map and PWCNet [Sun et al.,] for flow estimation. Both models perform very well in comparing to papers in their respective fields. Therefore, we assume that this baseline is a very powerful baseline to compare to. The models are

trained independently of each other with no shared loss function to optimize their individual performance.

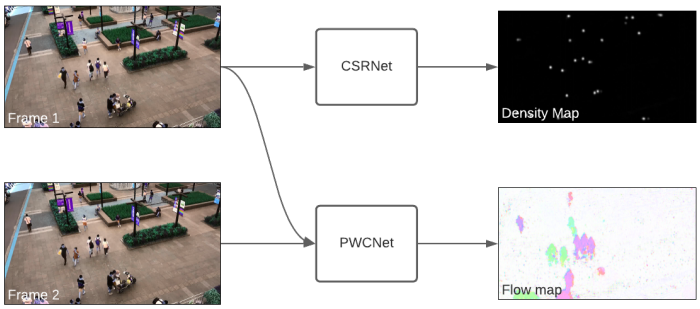


Figure 4.5: CSRNet+PWCNet baseline

Baseline 3

This third baseline is the architecture propose in the method, however without any positional context (figure 4.6). Only $I^{(t)}$ is used for density map prediction. No second stream of upscalers is used.

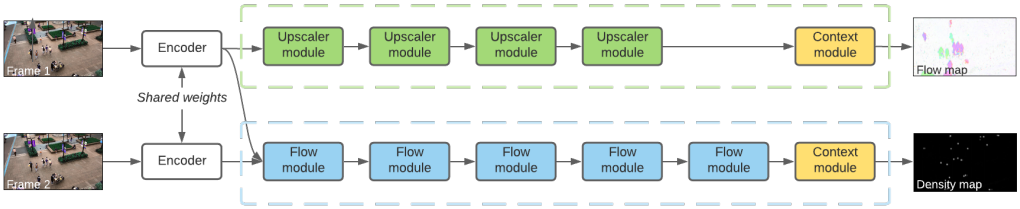


Figure 4.6: No positional context

Baseline 4

For positional context the encoded information of the second frame is used (Figure 4.7). The positional context uses it's own upscaler modules and concatenates the output of the upscaler with the main upscaler pipeline.

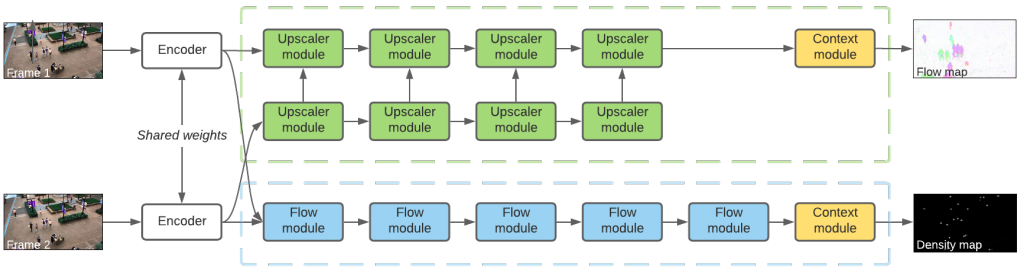


Figure 4.7: Add extra positional context to the decoder

4.3.2 Ablation study

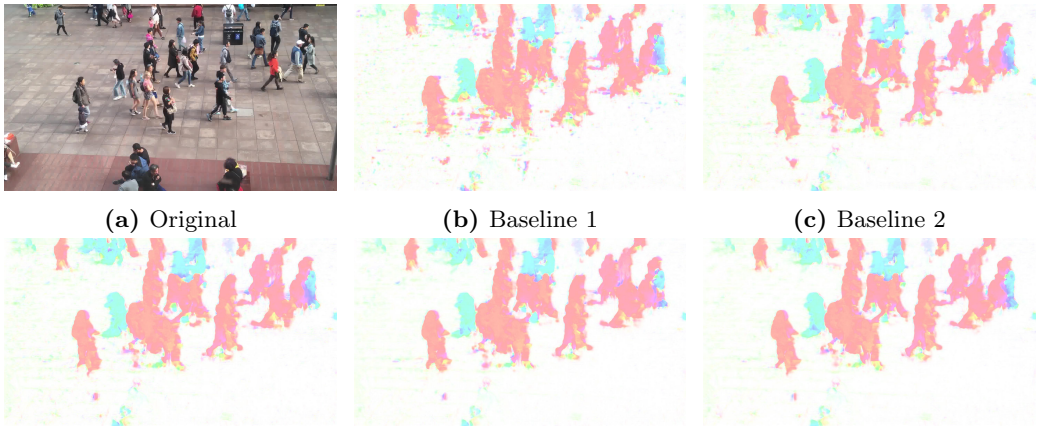
The effect of Flow estimation

Since the flow estimation is done in an unsupervised manner, there is no absolute measure to an accurate flow. However, the photometric loss, as described in formula 3.3, indicates the performance. Therefore, in table 4.1 the photometric loss is compared. The table shows that the loss only increases slightly. Where the standalone PWCNet (Baseline 2) performs the best. The loss increases according to the use of the PWCNet modules for predicting the density map. Baseline 3 and baseline 4 only use the encoder; the proposed method uses the PWCNet output for warping as well, and baseline 1 uses the same modules of PWCNet for both predicting the density map and the flow map.

Method	Photometric loss
Baseline 1	0.6915
Baseline 2	**0.6793**
Baseline 3	0.6860
Baseline 4	0.6859
Proposed	0.6891

Table 4.1: Photometric loss on Fudan dataset

A sample of all flows, in figure 4.8, shows almost identical performance for all results. Baseline 1 is the exception and shows some small noise, which can be caused by the single density decoder; however, this difference is negligible due to the maxing filter.



(a) Original **(b)** Baseline 1 **(c)** Baseline 2

(d) Baseline 3 **(e)** Baseline 4 **(f)** Proposed

Figure 4.8: Flow comparing on CrowdFlow dataset

The impact of Frame rate

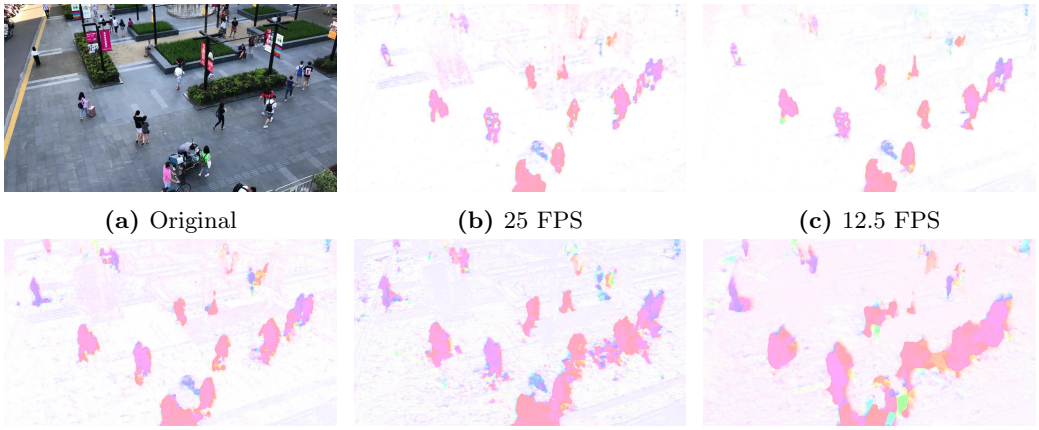
FPS	MAE	RMSE	MAPE
25	1.38	1.89	0.251
12.5	1.36	1.85	0.253
5	1.41	1.94	0.259
2.5	1.84	2.47	0.327
1	3.04	3.37	0.566

Table 4.2: Compare FPS on Fudan dataset with baseline 3

Table 4.2 shows the performance over different frames per second for the Fudan dataset with baseline 3. Comparing the performance between 25FPS and 5FPS, the decrease in all metrics is rather small. A smaller FPS is preferred to reduce computational cost. At an FPS of 2.5 and lower, a significant performance decrease is visible.

In figure 4.9, the difference is shown qualitative. As shown with the velocity maps, an FPS of 5 and higher shows no large incorrect estimations. When applying a lower frame rate, deformations start to appear, and the model starts to create large velocities that are not there in reality. In 10FPS and higher, some smooth areas show as if there is no movement at all due to the low spatial difference, which happens at a high frame rate. As shown in the results, this has no real effect on performance due to the maxing filter, which smooths these areas.

Due to these experiments in the rest of the thesis, an FPS of 5 is used to compare performance.



(a) Original **(b)** 25 FPS **(c)** 12.5 FPS

(d) 5 FPS **(e)** 2.5 FPS **(f)** 1 FPS

Figure 4.9: Flow impact different framerates

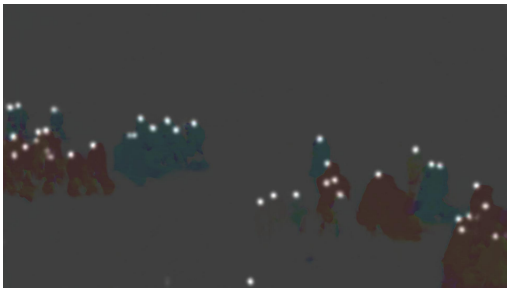
The effect of maxing filter

For the effect of the maxing filter, all datasets are tested on several baselines. Additionally, each dataset contains a small qualitative comparison to explain the differences.

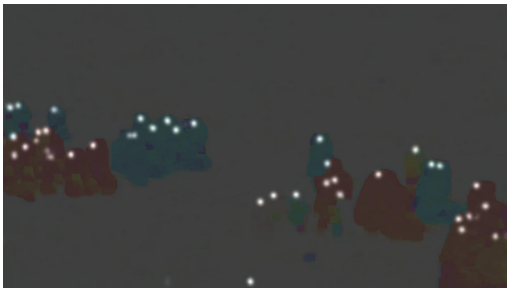
Method (LOI)	MAE	With Maxing	Improved	MAPE	With Maxing	Improved
Baseline 1	3.31	1.40	57.7%	0.576	0.252	56.3%
Baseline 2	3.29	1.42	56.8%	0.580	0.259	55.3%
Baseline 3	3.15	1.42	54.9%	0.555	0.259	53.3%

Table 4.3: LOI results for Fudan

(a) Original



(b) No maxing



(c) With maxing

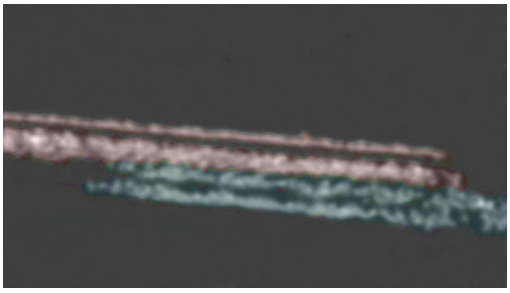
Figure 4.10: Comparing maxing filter on Fudan sample

Table 4.3 shows the difference for the Fudan dataset. The maxing filter improves the performance on both the MAE and the MAPE by more than half. In figure 4.10b, it shows that there is a misalignment between the density map and the velocity map. The predicted heads in the density map are larger than the velocity map area of the respective pedestrians. In figure 4.10c, the same velocity map with the maxing filter applied is shown. Here it is clear that the large velocity area caused by the maxing filter fully encapsulates the blobs in the density map.

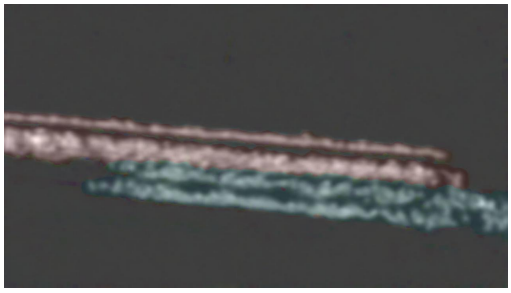
Method (LOI)	MAE	With Maxing	Improved	MAPE	With Maxing	Improved
Baseline 1	23.17	8.98	61.2%	0.184	0.078	57.6%
Baseline 2	19.95	12.24	38.6%	0.185	0.093	49.7%
Baseline 3	21.19	7.69	63.7%	0.168	0.079	52.9%

Table 4.4: LOI results for Crowdflow

(a) Original



(b) No maxing



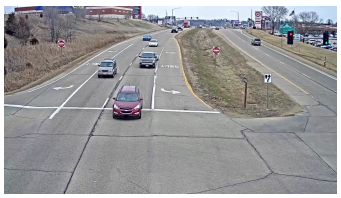
(c) With maxing

Figure 4.11: Comparing maxing filter on CrowdFlow dataset with baseline 3

In table 4.4, the impact of the maxing filter for the CrowdFlow dataset is shown. It has a large impact as well, just as on the Fudan dataset. As shown in figure 4.11b, there is not a large visible misalignment between the density map and the velocity map. However, in the images, the difference in speed is not clearly visible. This could mean that, because of the space between pedestrians, there is a difference in speed, not visible in the figures. By applying the maxing filter, the speed is adjusted accordingly.

Method (LOI)	MAE	With Maxing	Improved	MAPE	With Maxing	Improved
Baseline 2	8.71	5.08	41.6%	0.336	0.22	34.5%
Baseline 3	3.82	3.89	-0.2%	0.258	0.24	7.0%

Table 4.5: Maxing comparison for AI City



(a) Original

(b) No maxing

(c) With maxing

Figure 4.12: Comparing maxing filter on AICity dataset

In table 4.5 a smaller increase is noticed. Due to the larger areas of the cars and the central tagging of the cars, there is a smaller chance of a misalignment between the density map and velocity map (As shown in figure 4.12b and figure 4.12c). This would explain the smaller increase in performance in comparing the Fudan dataset and the CrowdFlow

dataset. Several other inaccuracies could explain the increase in performance. It could be that the labeling of the dataset is inaccurately done, which results in inconsistent locations of the density. Small appearing cars could still have a misalignment problem. Additionally, the maxing filter could remove underestimating velocity map pixels.

All in all, the maxing filter does increase performance significantly. Especially on pedestrian datasets, the increase in performance is large. There is still an increase in performance on the cars dataset, however much smaller, which can be explained by the small underestimating velocities.

4.3.3 Fudan-ShanghaiTech

Method (LOI)	MAE	RMSE	MAPE
Baseline 1	1.40	1.90	0.252
Baseline 2	1.42	1.86	0.259
Baseline 3	1.42	1.94	0.259
Baseline 4	1.40	2.04	0.255
Proposed	**1.28**	**1.78**	**0.240**

Table 4.6: LOI results for Fudan

Method (ROI)	MAE	RMSE
Baseline 1	5.80	7.09
Baseline 2	2.99	3.93
Baseline 3	2.65	3.69
Baseline 4	**2.45**	3.39
Proposed	**2.45**	**3.30**

Table 4.7: ROI results for Fudan

In table 4.6 and table 4.7, different baselines and proposed models are compared on LOI and ROI performance for the Fudan-ShanghaiTech dataset.

Baseline 1 seems to perform very poor in table 4.7, but equally well in table 4.6 as the rest of the baselines. It seems that baseline 1 mainly focuses on moving pedestrians, which works well for the LOI performance, but the ROI performance takes non-moving pedestrians into account as well.

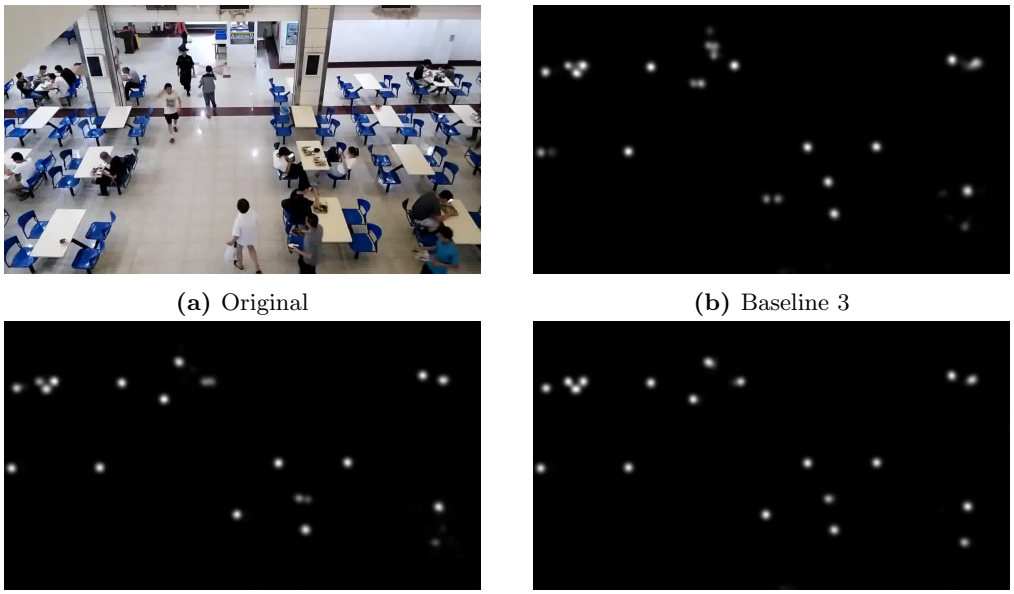
Baseline 3 shows an improvement over baseline 2 in the ROI performance, which shows the strength of our proposed upscaler modules in the network. Applying positional context in baseline 4 and our proposed model shows extra improvements as well. Due to no importance of precise location, both baseline 4 and the proposed model perform on par on the ROI metrics.

In table 4.6, the baselines are all performing similarly. The proposed method improves overall the baselines. Comparing baseline 4 and the proposed model, this can be explained by the applied warping. For the LOI, there is higher importance of precise localization of the pedestrian, which is harder to obtain with the baselines.

For qualitative comparison figure 4.13 and figure 4.14 are two samples from the Fudan dataset. In figure 4.13 a clear difference is visible between aided models and the model relying on a single image. Baseline 3 has a hard time distinguishing the pedestrian on the top side, which is partially overlapped with the pedestrian above. Baseline 4 and the proposed model can clearly identify this pedestrian. In the middle and the right, a difference between baseline 3 and the proposed model is noticeable, where baseline 3 has difficulties determining the pedestrian's location in the density map.

The same pattern is noticeable in figure 4.14. In the left-top, a lot of occluded pedestrians are missed by baseline 3, which can be identified by baseline 4 and the proposed model; however, the proposed model has better separation of the pedestrians. In the middle, both baseline 3 and 4 have difficulties with localizing the exact position of pedestrians.

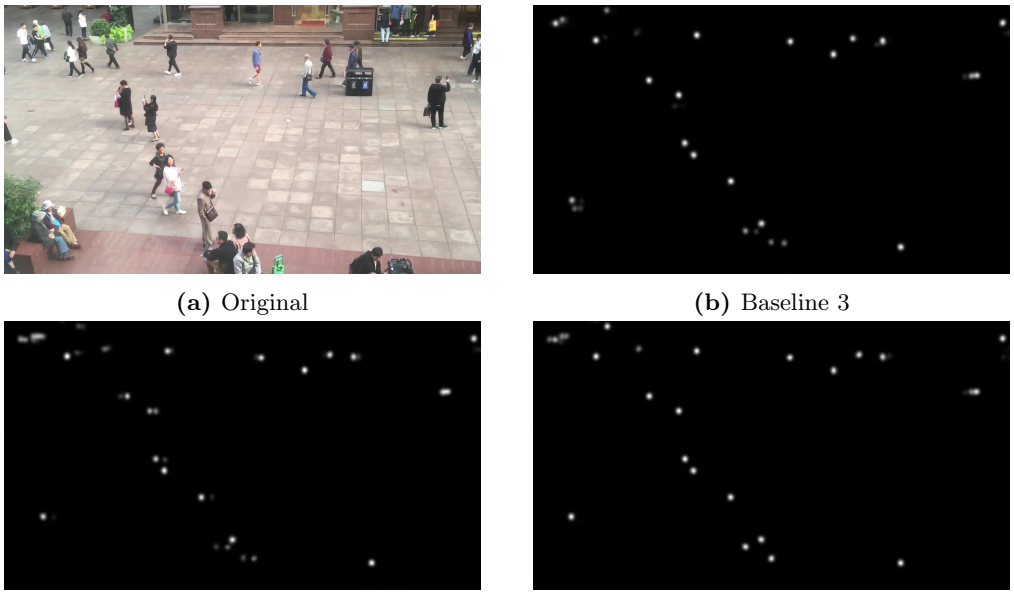
Adding context and warping compared to baseline 3 clearly improves the proposed model in identifying and localizing the pedestrians, improving both ROI and LOI over the baselines.



(a) Original **(b)** Baseline 3

(c) Baseline 4 **(d)** Proposed

Figure 4.13: Density map sample for Fudan dataset



(a) Original **(b)** Baseline 3

(c) Baseline 4 **(d)** Proposed

Figure 4.14: Density map sample for Fudan dataset

4.3.4 CrowdFlow

Method	MAE	RMSE	MAPE
Baseline 1	8.98	10.07	0.078
Baseline 2	12.24	14.56	0.093
Baseline 3	7.69	9.36	0.079
Baseline 4	7.15	9.54	0.054
Proposed	**6.55**	**8.72**	**0.053**

Table 4.8: LOI results for Crowdflow

Method	MAE	RMSE
Baseline 1	17.611	22.99
Baseline 2	37.939	54.53
Baseline 3	13.686	19.82
Baseline 4	13.134	21.16
Proposed	**12.888**	**18.05**

Table 4.9: ROI results for Crowdflow

In table 4.9 and table 4.8 the LOI and ROI performance in CrowdFlow are shown.

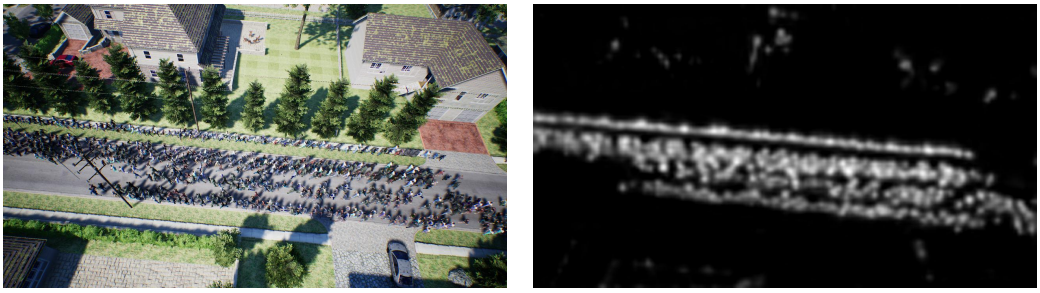
Baseline 2 shows a terrible performance in comparison to the rest of the models. This can be due to the lack of training data, which adds much noise in the density map on places where are no pedestrians, as shown in figure 4.15b. CSRNet in baseline 2 has a much larger encoder with higher complexity; hench requires more training data to finetune correctly.

Baseline 1 is not underperforming as much as in Fudan; this could be because all pedestrians are moving. Still, it is not performing as well as the baselines with separate density map decoder and our proposed model.

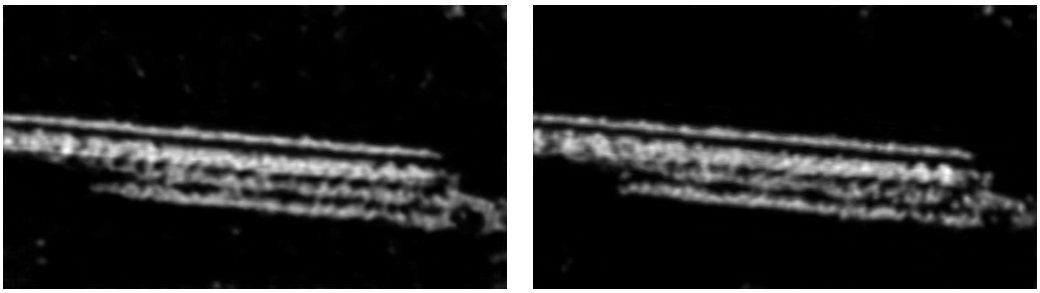
The importance of temporal/spatial context is shown in table 4.9 and 4.8. Both baseline 4 and our proposed model outperform baseline 3.

Due to the added spatial context, the noise is further reduced. In the left bottom corner of figure 4.15, it is visible that baseline 3 has a harder time with noise than the proposed model. It seems that both baseline 2 and baseline 3 have a harder job in distinguishing pedestrian and non-pedestrian areas.

In figure 4.16, the results of baseline 2 show large weird-shaped blobs, whereas baseline 4 and the proposed model show better separation between the blobs. Which indicates a hard time locating the pedestrians as well. In figure 4.17, baseline 4 still has difficulty location pedestrians. Baseline 4 has a more smoothed out density maps, where the proposed model shows rounded blobs, which could be explained by the warping, which solves the spatial difference in the proposed model and improves predicting the location of the pedestrians

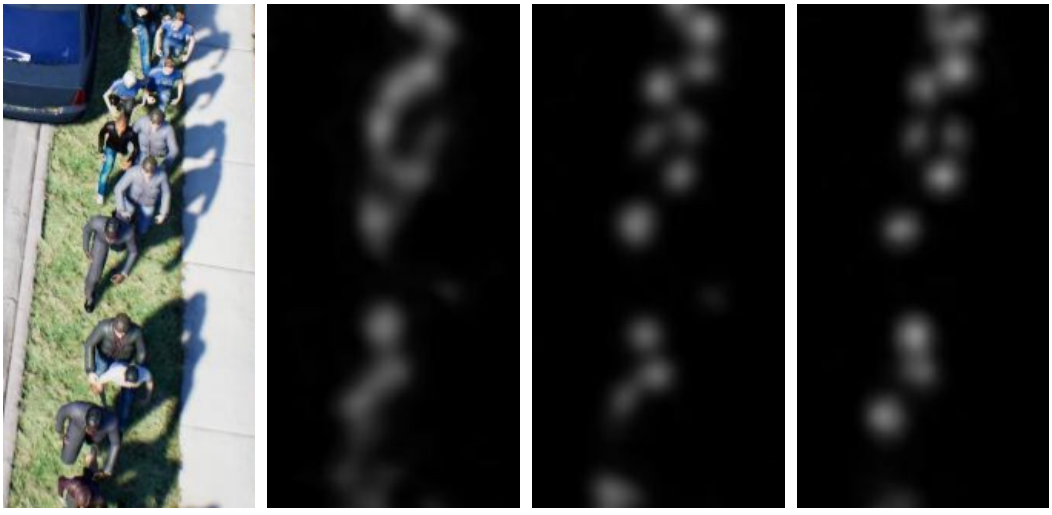


(a) Original **(b)** Baseline 2



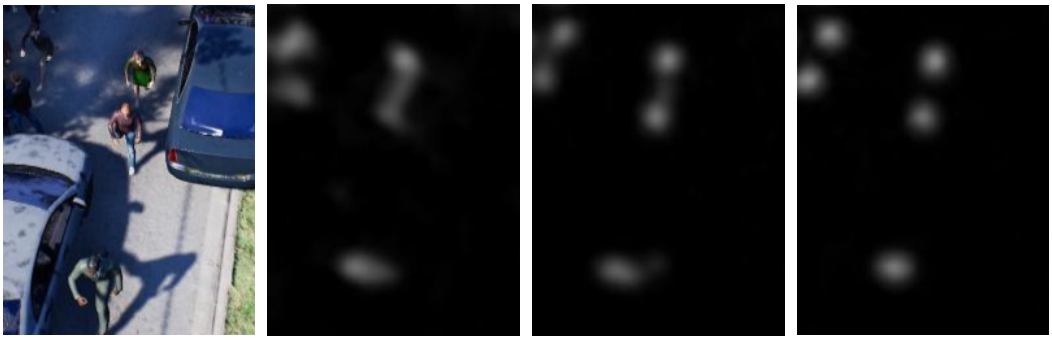
(c) Baseline 3 **(d)** Proposed

Figure 4.15: Density map sample for CrowdFlow dataset



(a) Original **(b)** Baseline 3 **(c)** Baseline 4 **(d)** Proposed

Figure 4.16: Density map sample for CrowdFlow dataset



(a) Original **(b)** Baseline 3 **(c)** Baseline 4 **(d)** Proposed

Figure 4.17: Density map sample for CrowdFlow dataset

4.3.5 AI City Challenge

Method (LOI)	MAE	RMSE	MAPE
Baseline 1	14.09	16.54	0.59
Baseline 2	5.08	6.00	0.22
Baseline 3	3.89	4.24	0.24
Baseline 4	4.16	3.71	0.23
Proposed	5.31	6.49	0.24

Table 4.10: LOI results for AI City

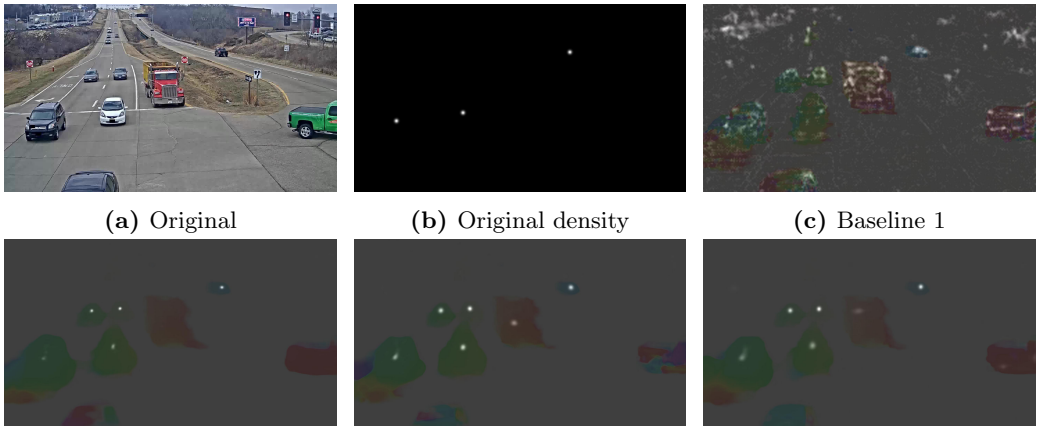
In table 4.10, the LOI performance of AICity is shown.

The performance overall, the models show very inconsistent results. This could be due to inconsistent labeling. Baseline 1 even has difficulties generalizing the model and separating background and foreground (As shown in figure 4.18c)

The rest of the models seems to vary in performance on both MAE and RMSE. Due to only 4 different video segments, a single test sample's weight is larger than in the other datasets. This would why the MAPE is almost equal over all the models, except baseline 1. The MAPE metric normalizes for each sample, suggesting that the whole dataset's performance is almost identical, which suggests that context has no influence on tasks where temporal occlusion is minimal, and the objects are large to minimize miss positioning in the density map.

In figure 4.18b the original density map is visible. It is clear that several distant cars and even some close cars are not labeled. Even though the density maps result in correctly detecting those cars. Due to the spatial information, it seems that especially baseline 4 and the proposed model tend to locate the non-labeled cars as well in figure 4.18.

Due to incorrect labeling and a small number of test samples, it is hard to conclude that our model performs better or worse than the baselines. The MAPE metrics show performance similar to the other models. The qualitative results show that the proposed model performs better in detecting all the cars than baseline 3, even though the metrics show this is not the case.



(a) Original **(b)** Original density **(c)** Baseline 1

(d) Baseline 3 **(e)** Baseline 4 **(f)** Proposed

Figure 4.18: Density map sample for the AI City dataset

4.4 Real world performance

To compare the real-world performance, the models are compared on processing speed. This is done on the Fudan-ShanghaiTech dataset with baseline 3.

Method	FPS	ms	MAE	RMSE	MAPE
Baseline 1	8.9	112	1.40	1.90	0.252
Baseline 2	5.8	172	1.42	1.86	0.259
Baseline 3	7.9	127	1.42	1.94	0.260
Baseline 3+optimized	**35.7**	**28**	1.41	1.93	0.258
Proposed	6.6	152	**1.28**	1.78	0.240
Proposed+optimized	27.0	37	**1.28**	**1.77**	**0.237**

Table 4.11: Processing time

Looking at processing time in table 4.11, even the slowest model is processing more than 5FPS on the used server, which means that a live stream can run on the server using all the compared models. However, the current server uses a powerful GPU, which is in most cases not a feasible option and extra optimizations have to be made to run on slower servers.

Baseline 1, without optimization, shows the best processing time, which is deducible from the architecture, which is the smallest of all the models. Baseline 2 performs the slowest due to the use of two completely separate estimators, whereas the density estimator has a much larger encoder than the encoder used in PWCNet as well. The proposed model shows some extra overhead in comparison to baseline 3 by adding the warped context.

Because predicting the LOI only requires the area around the Line of Interest, an optimized version is tested as well, with only a small area around the LOI used for density map and flow map estimation. Both optimized versions of baseline 3 and the proposed model show a large decrease in processing time compared to using the complete frame. Additionally, the optimized versions show no decrease in LOI performance, which shows that the bigger context of the frame does not influence the prediction of both estimated maps. So cropping the full image can be done to optimize for speed without sacrificing performance.

Chapter 5

Conclusion

This thesis presents a new method for crossing-line counting, which performs unsupervised flow estimation and supervised density estimation in a unified network and shows good initial results. Next, we provide some conclusions with our experimental results.

- The unified model, which both predicts density map and velocity map, shows that the density map decoder on top of the existing PWCNet performs well on the datasets, without decreasing the performance of the standalone PWCNet model.
- The proposed architecture, with PWCNet encoder and the upscaler modules, provides strong standalone density estimations. On both the Fudan and CrowdFlow dataset, both baseline 1 [Zhao et al., 2016] and baseline 2 [Li et al., 2018] are outperformed by all baselines with the proposed architecture.
- Adding temporal information, by adding a consecutive frame as context, shows an increase in identifying pedestrians. The increase is most notable in places with short temporary occlusions. The added temporal information has the most influence on the ROI performance and some impact on LOI performance.
- The added temporal information introduces spatial differences between the consecutive frames. Due to the spatial difference, locating the positions of the pedestrians is still a problem. By introducing the warping module, the spatial difference is tackled. The LOI performance increases and the ROI performance to a lesser extent because LOI greatly benefits from better localization.
- The misalignment problem, where the misalignment between pedestrians in the velocity map and density map due to unsupervised Flow Estimation, is tackled by a maxing filter. The maxing filter enlarges each pedestrian's velocity map area, so each pedestrian's blob in the density map is encapsulated by the velocity map. The maxing filter shows the most significant increase in line-crossing performance compared to all other improvements. It shows an improvement of more than 50% on all pedestrian datasets.
- For real-world usage of Crowd Crossing Counting, the results are promising. Low processing time can be achieved. Additional speed optimizations reduce the processing time significantly, without any significant loss of accuracy.

Further research

Several other realigning methods can be researched in further research to reduce the number of hyperparameters per dataset by proposing trainable realigning methods or more human-shaped density map blobs. Additionally, due to qualitative research, we can conclude that

locating pedestrian improves due to warping; however, no quantitative method is used. A focus on the quantitative research of locating pedestrians in a density map would be a promising continuation. Lastly, new applications regarding both optical flow and crowd counting can be researched. The unified model provides a single model to predict both, where Crowd Crossing Counting is only one application.

All in all, the thesis shows a new approach in predicting Line Crossing Counting by introducing unsupervised flow estimation, as well as improving Crowd Counting results by additional temporal information using a warping module.

Bibliography

[Belagiannis et al., 2015] Belagiannis, V., Rupprecht, C., Carneiro, G., and Navab, N. (2015). Robust optimization for deep regression. In *ICCV*.

[Bruhn et al., 2005] Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):1–21.

[Cao et al., 2015] Cao, L., Zhang, X., Ren, W., and Huang, K. (2015). Large scale crowd analysis based on convolutional neural network. 48(10):3016–3024.

[Chan and Vasconcelos, 2009] Chan, A. and Vasconcelos, N. (2009). Bayesian Poisson regression for crowd counting Cited by me. *Computer Vision, 2009 IEEE 12th International* . . . .

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I:886–893.

[Denman et al., 2018] Denman, S., Fookes, C., Yarlagadda, P. K., and Sridharan, S. (2018). Scene invariant virtual gates using dnns. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2637–2651.

[Dollár et al., 2012] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischery, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P. V. D., Cremers, D., and Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2758–2766.

[Fang et al., 2019a] Fang, Y., Zhan, B., Cai, W., Gao, S., and Hu, B. (2019a). Locality-constrained spatial transformer network for video crowd counting. *Proceedings - IEEE International Conference on Multimedia and Expo*, 2019-July:814–819.

[Fang et al., 2019b] Fang, Y., Zhan, B., Cai, W., Gao, S., and Hu, B. (2019b). Locality-constrained spatial transformer network for video crowd counting. In *ICME*.

[Gadde et al., 2017] Gadde, R., Jampani, V., and Gehler, P. V. (2017). Semantic video cnns through representation warping. In *ICCV*.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[Hui et al., 2018] Hui, T. W., Tang, X., and Loy, C. C. (2018). LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8981–8989.

[Idrees et al., 2013] Idrees, H., Saleemi, I., Seibert, C., and Shah, M. (2013). Multi-source multi-scale counting in extremely dense crowd images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2547–2554.

[Ilg et al., 2016] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2016). FlowNet 2.0: Evolution of optical flow estimation with deep networks.

[Janai et al., 2018] Janai, J., Güney, F., Ranjan, A., Black, M., and Geiger, A. (2018). Unsupervised Learning of Multi-Frame Optical Flow with Occlusions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11220 LNCS:713–731.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

[Lempitsky and Zisserman, 2010] Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. *Advances in neural information processing systems*, 23:1324–1332.

[Li et al., 2018] Li, Y., Zhang, X., and Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1091–1100.

[Lin and Davis, 2010] Lin, Z. and Davis, L. S. (2010). Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618.

[Liu et al., 2008] Liu, C., Freeman, W. T., Adelson, E. H., and Weiss, Y. (2008). Human-assisted motion annotation. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

[Liu et al., 2019a] Liu, P., King, I., Lyu, M. R., and Xu, J. (2019a). DDFlow: Learning optical flow with unlabeled data distillation.

[Liu et al., 2019b] Liu, P., Lyu, M., King, I., and Xu, J. (2019b). SelFlow: Self-supervised learning of optical flow.

[Liu et al., 2019c] Liu, W., Salzmann, M., and Fua, P. (2019c). Context-aware crowd counting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5094–5103.

[Ma and Chan,] Ma, Z. and Chan, A. B. Counting people crossing a line using integer programming and local features. 26(10):1955–1969.

[Ma and Chan, 2013] Ma, Z. and Chan, A. B. (2013). Crossing the line: Crowd counting by integer programming with local features. In *CVPR*.

[Mémin and Pérez, 1998] Mémin, E. and Pérez, P. (1998). Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing*, 7(5):703–719.

[Menze and Geiger, 2015] Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070.

[Pock et al., 2008] Pock, T., Schoenemann, T., Graber, G., and Bischof, H. (2008). Learning optical flow. 5304(May 2014).

[Ranjan and Black, 2017] Ranjan, A. and Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:2720–2729.

[Shi et al., 2019] Shi, Z., Mettes, P., and Snoek, C. G. (2019). Counting with focus for free. In *ICCV*, pages 4200–4209.

[Shi et al., 2020] Shi, Z., van Woerden, J. E., Mettes, P., and Snoek, C. (2020). Counting in motion: Video density estimation with unsupervised flow. In *CVPR*.

[Shi et al., 2018] Shi, Z., Zhang, L., Liu, Y., Cao, X., Ye, Y., Cheng, M.-M., and Zheng, G. (2018). Crowd counting with deep negative correlation learning. In *CVPR*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27:568–576.

[Sreenu and Saleem Durai, 2019] Sreenu, G. and Saleem Durai, M. A. (2019). Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1):1–28.

[Subburaman et al., 2012] Subburaman, V. B., Descamps, A., and Carincotte, C. (2012). Counting people in the crowd using a generic head detector. *Proceedings - 2012 IEEE 9th International Conference on Advanced Video and Signal-Based Surveillance, AVSS 2012*, pages 470–475.

[Sun et al.,] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. PWC-net: CNNs for optical flow using pyramid, warping, and cost volume.

[Sun et al., 2018] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*.

[Wan and Chan, 2019] Wan, J. and Chan, A. (2019). Adaptive density map generation for crowd counting. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:1130–1139.

[Wang et al., 2020] Wang, Q., Gao, J., Lin, W., and Li, X. (2020). Nwpu-crowd: A large-scale benchmark for crowd counting. *arXiv preprint arXiv:2001.03360*.

[Wedel et al., 2009] Wedel, A., Cremers, D., Pock, T., and Bischof, H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1663–1668.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266.

[Xiong et al., 2017] Xiong, F., Shi, X., and Yeung, D.-Y. (2017). Spatiotemporal modeling for crowd counting in videos. In *ICCV*.

[Xu et al., 2015] Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[Yu et al., 2016] Yu, J. J., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9915 LNCS:3–10.

[Zhang et al., 2017] Zhang, S., Wu, G., Costeira, J. P., and Moura, J. M. (2017). Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *ICCV*.

[Zhang et al., 2016] Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:589–597.

[Zhao et al., 2016] Zhao, Z., Li, H., Zhao, R., and Wang, X. (2016). Crossing-line crowd counting with two-phase deep neural networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, volume 9912, pages 712–726. Springer International Publishing.

[Zheng et al., 2019] Zheng, H., Lin, Z., Cen, J., Wu, Z., and Zhao, Y. (2019). Cross-line pedestrian counting based on spatially-consistent two-stage local crowd density estimation and accumulation. 29(3):787–799.