VRU Pose-SSD: Multiperson Pose Estimation For Automated Driving

Chandan Kumar, [1] **Jayanth Ramesh,**[1] **Bodhisattwa Chakraborty,**[1] **Renjith Raman,**[1] **Christoph Weinrich,**[2] **Anurag Mundhada,**[3] **Arjun Jain,**[3,4] **Fabian B. Flohr** [5]

[1] Mercedes-Benz R&D, India, [2]Robert Bosch GmbH, Germany, [3]Axogyan AI, India, [4]Indian Institute of Science, India.
[5]Mercedes-Benz AG, Germany.

Abstract

We present a fast and efficient approach for joint person detection and pose estimation optimized for automated driving (AD) in urban scenarios. We use a multitask weight sharing architecture to jointly train detection and pose estimation. This modular architecture allows us to accommodate different downstream tasks in the future. By systematic large-scale experiments on the Tsinghua-Daimler Urban Pose Dataset (TDUP), we obtain multiple models with varying accuracy-speed trade-offs. We then quantize and optimize our network for deployment and present a detailed analysis of the efficacy of the algorithm. We introduce a two-stage evaluation strategy, which is more suitable for AD and achieves a significant performance improvement in comparison to state-of-the-art approaches. Our optimized model runs at 52 fps on full HD images and still reaches a competitive performance of 32.25 LAMR. We are confident that our work serves as an enabler to tackle higher-level tasks like VRU intention estimation and gesture recognition, which rely on stable pose estimates and will play a crucial role in future AD systems.

Introduction

Significant progress has been made over the last decade on video-based Vulnerable Road Users (VRU) detection. In the area of driver assistance, this has led to the first commercial active VRU systems reaching the market. In the context of automated driving (AD) or future driver assistance systems, it might be desired to infer a more detailed model of the VRU behavior to cope also with more complex scenarios in urban environments. While a human driver makes constant judgments and driving decisions based on perceived VRU context cues, such as intention (e.g. inferred by head and body orientation, gait cycle) or explicit gestures (e.g. cyclist hand gestures), this also motivates the estimation and use of such cues for future driver assistance or AD systems (Aparicio et al. 2017) (Kooij et al. 2019).

Unfortunately, the diversity of VRU appearance, owing to clothing, pose, size, ethnicity, gender etc. presents a challenge at perceiving and modeling VRU context cues accurately using image features. One way to solve this is an appearance invariant representation, which is discriminative for downstream tasks as intention and gesture estimation. As shown in Fig. 1, articulated pose can be utilized to generate an appearance-invariant intermediate representation and is widely supported in literature for modeling VRU behavior like action recognition (Hariyono and Jo 2015), crossing intention estimation (Fang and López 2019), trajectory prediction (Rasouli et al. 2019) and gesture recognition (Tripathi et al. 2019). Even though there have been great strides in human pose estimation, there has been less focus from the perspective of real-time applications such as AD use-cases (Kothari et al. 2017) (Kress et al. 2018). Most renowned approaches are either computation-intensive (He et al. 2017) or memory-intensive (Huang, Zhu, and Huang 2019), making them today not optimal for real-time AD applications which demands inference on high resolution images with high frame rates. Another challenge in tackling this problem is the lack of AD domain-specific pose datasets. While there exist datasets (Lin et al. 2014) (Jhuang et al. 2013), (Andriluka et al. 2014) for human action and pose estimation, they are not of automotive-grade.

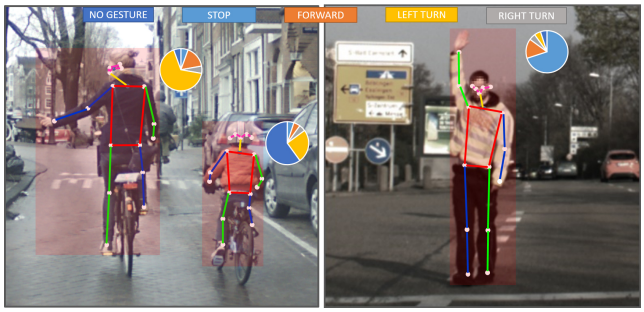


Figure 1: An estimated articulated pose can serve as an effective intermediate representation for inferring the direction of motion, intention or gesture of a VRU. Based on such contextual cues automated vehicles can better understand and interact with their environment.

This paper proposes VRU Pose-SSD, an extensible VRU detection and 2-D pose estimation model, optimized for deployment on an AD platform. We explain the design choices of our architecture along with adopted changes to attain a faster model inference with minimum trade-off in accuracy. We adopt a new evaluation strategy to focus on the AD use-case. Experiments are performed on the TDUP dataset (Wang et al. 2020), an automotive-grade dataset fo-

cused on VRU in challenging urban street scenarios.

Related Work

Detection Object detection is a fundamental computer vision problem, often necessary as the first step in computer vision pipelines to facilitate valuable semantic scene understanding. CNN-based object detectors can be classified into two broad families, namely, *two-stage detectors* and *single-stage detectors*. Two-stage detectors use Region Proposal Network (RPN) (Ren et al. 2015), which generates object proposals, and extracts feature crops from each proposal in the first stage. Then, a classifier in the second stage predicts labels for each object proposal produced by the RPN. Such detectors mainly include the R-CNN family of detectors. On the contrary, single-stage detectors directly predict the location and class of the objects in a single step. The family of single-stage detectors comprises of YOLO (Redmon et al. 2016) and its variants, SSD (Liu et al. 2016), RetinaNet (Lin et al. 2017), etc. Both families incorporate *anchors* or *prior boxes*, first introduced in Faster R-CNN (Ren et al. 2015), which enables incorporating prior information on the aspect ratio of the objects. Two-stage detectors tend to be heavy but more accurate, while single-stage detectors are light and more suitable for real-time object detection. There have been works focusing on pedestrian and cyclist detection (Li et al. 2016) and (Braun et al. 2019) is a valuable benchmark dataset, focusing on VRU in Urban Traffic Scenes. Recent work of (Uhrig et al. 2018) Box2Pix has adapted SSD, a single-stage detector for the AD use-case.

Human Pose Estimation Human pose estimation is defined as the problem of the localization of human joint-points in an image. Approaches to multi-person pose estimation falls into two categories: bottom-up and top-down. The bottom-up approaches directly predict the joint-points and later associate them, to obtain the corresponding pose for each human. Pifpaf (Kreiss, Bertoni, and Alahi 2019) and Higher HRNet (Cheng et al. 2019) are few notable works in this category. In contrast, the top-down approaches constitute a detector, to capture all instances of humans in an image and thereafter, the joint-points are localized for each instance. The family of top-down approaches includes Mask R-CNN (He et al. 2017), Dense Pose (Alp Güler, Neverova, and Kokkinos 2018), etc. For high-resolution images (2MP and above), the association step in bottom-up methods suffers because of increased complexity and memory constraints. Whereas the run-time performance of top-down approaches are directly proportional with the number of person present in the frame (Dang et al. 2019).

Run-time Optimization and Deployment Techniques of model *optimization* and *compression* constitute a critical role for their deployment in real-time systems. Various approaches to model compression, acceleration, and their consequences are summarized in (Cheng et al. 2017). The authors have classified *optimization* approaches into four categories: parameter pruning and quantization, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. Additionally to the four categories, (Liu et al. 2017) proposed a learning scheme for CNNs that reduces the run-time and model size without compromising the accuracy. The proposed method achieves this by removing insignificant channels of a wide and large model automatically during training. Model *compression* can be accomplished either by converting a pre-trained floating-point model into a fixed point model without re-training (Settle et al. 2018) or by training a deep neural net with a fixed point constraint (Chen et al. 2017). We prefer converting a floating-point model approach as it does not need re-training and fits in a multi-task training environment.

Contribution We consider our main contribution to be a principled, modular and extensible architecture for VRU detection and pose estimation (VRU Pose-SSD) optimized for the AD use-case. We describe a general and systematic experimentation methodology and explore various possible optimization techniques and their benefit, targeting a fast but accurate detection and pose estimation model. Furthermore, we introduce a joint evaluation scheme for detection and pose estimation based on the log-average miss rate, which accounts for various AD-specific VRU characteristics such as their proximity and visibility to the ego vehicle. While our optimized models reach a competitive accuracy, they run 1.6 - 4.1 times faster than well-known state-of-the-art models (He et al. 2017) (Kreiss, Bertoni, and Alahi 2019).

Proposed Approach

Since bottom-up architectures often suffer from an increased complexity and memory constraints when used with high-resolution images, we opt for a top-down architecture based on Mask-RCNN for human pose estimation as our starting point and enhance it for our real-time requirements. During inference, Mask R-CNN has five stages that run in sequence: (1) feature extractor, (2) region proposal network, (3) detector head (classification and regression branches), (4) mask head, and (5) non-maximum suppression (NMS). This order of operations in the pipeline bottlenecks the inference run-time making it run at 5 fps, which is considered slow for our target application. Therefore, we modify the Mask R-CNN architecture as follows to achieve a faster model. We simplify the pipeline by replacing parts (1)-(4) of the Mask R-CNN pipeline by adapting a single-stage detector (SSD). Consequently we merge the Region Proposal Network (RPN) and the detector head and finally performing NMS prior to pose estimation, which eliminates the need to perform pose estimation on a large number of detections.

We call this *VRU Pose-SSD* architecture. To summarize, our architecture consists of four functional blocks: core network, detector (SSD), Non-Maximum Suppression (NMS) and Pose head (pose estimator) as shown in Fig. 2.

Architecture

Core Network We choose Inception V1 (Szegedy et al. 2015) as it enables our model to process high resolution images effectively and provide us the right trade-off w.r.t computation and memory compared to other networks in literature (Huang et al. 2017). The features from the core network are used for the detection and pose estimation tasks and we train the network end-to-end in a multitask setting for run-time efficiency. We adapt the Inception V1 architecture to

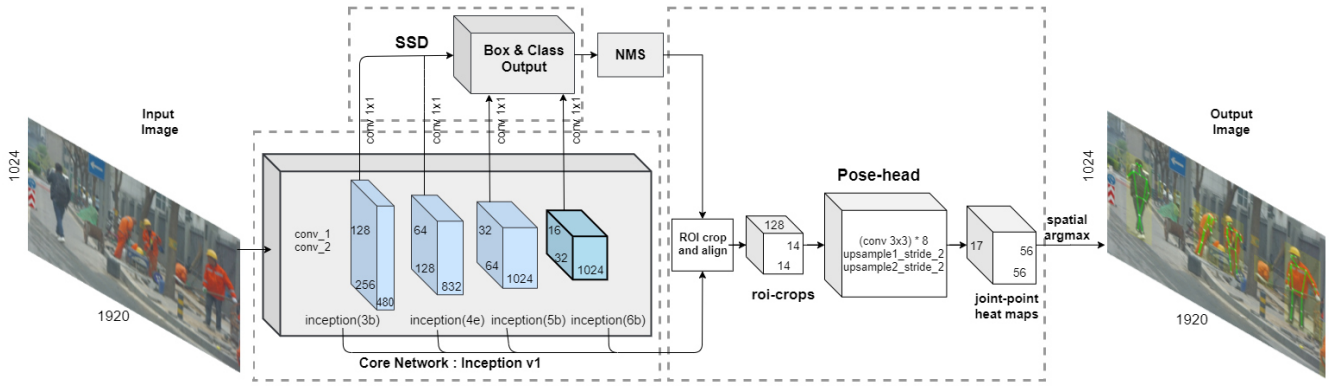


Figure 2: VRU Pose-SSD architecture with four main blocks: core network, SSD, NMS and pose head. Inception V1 core network is modified with addition of two more layers, inception (6a) and inception (6b), to capture larger receptive field.

have a larger receptive field by adding two additional inception modules, i.e. inception (6a) and inception (6b) as suggested also in (Uhrig et al. 2018). This enables us to capture also larger objects, i.e. VRUs close to the ego-vehicle and thus critical for our use case (e.g. pedestrian behind parked cars, traffic control persons, cyclist driving in front). We initialize the core network with ImageNet pre-trained weights and the additional new layers are randomly initialized.

SSD adaptions: We adapt the SSD used in Box2pix approach (Uhrig et al. 2018) and modify it for detection of VRU classes by adding distinct classes for pedestrian and rider instead of human in addition to background. The modified SSD is as follows: (**a**) relative box parameter changes are used instead of IoU for box matching, (**b**) nine different prior boxes with different aspect ratios are designed to cover different scales of VRU's, i.e. one for inception (3b), two for inception (4e) and inception (5b), and four for inception (6b). The prior boxes are assigned to these layers based on the receptive field of the respective layers to cover VRUs of all scales and aspect ratios. To design prior boxes, we use the strategy proposed by Redmon et al. (Redmon and Farhadi 2017), i.e. k-means clustering is applied to all the bounding boxes in the training split to compute optimal cluster centers for the data distribution. As the aspect ratios of both the classes are similar, the assigned priors provides a perfect trade-off between maximum recall and computation complexity (**c**) a 1×1 convolution layer is used after every inception block to regress box offsets and classification scores. The SSD output is a $N \times (4 + 3)$ sized vector, where N is the number of total prior boxes across all the scales, and for each prior box we estimate four bounding box offsets, and three scores for background, pedestrian and rider classes. We perform NMS independently for both foreground classes and use an optimized version that runs on the GPU to eliminate redundant boxes.

Pose head We follow the COCO convention (Lin et al. 2014) and represent VRU pose as a set of 17 joint-points: eyes, ears, nose, shoulders, elbows, wrists, hips, knees, and ankles. This representation, widely used in literature encodes meaningful information about the VRU pose. We use 2-D pose estimation because this representation is sufficient for our targeted use cases of intention and gesture recognition and a better fit, taking into account our hard real-time constraints. The bounding boxes after NMS, are cropped using TensorFlow crop_and_resize which is similar to ROI align (He et al. 2017) but uses a different sampling strategy. Both share the fundamental idea of preserving pixel to pixel alignment while cropping features, crucial for localization tasks. We experiment with features extracted from different layers - inception (3b), inception (4e), inception (5b), and inception (6b). As the actual pose head, we use a modified version of the fully convolutional network from Mask R-CNN. ROI crop output of size 14×14 is passed through a stack of eight 3×3 128-d *conv* layers followed by two *de-conv* layers (bilinear up-sampling), to produce the output features of size $56 \times 56 \times 17$. This is a set of 17 heatmaps for all 17 joint-points of a VRU instance. Spatial argmax is performed across each of these heatmaps to estimate the joint-point location, and the coordinates are then transformed to the image frame.

Multi-task training

The modularity offered by a two-stage architecture of VRU Pose-SSD allows a multitask network architecture, i.e. multiple perception tasks can use the same core network. Thereby, we expect this architecture to better scale for additional tasks. We are also interested in inferring several features of VRU such as segmentation, intention, orientation, gestures, etc. A top-down method allows us to offload specialized feature inference to the second stage, while the first stage can be learned as a generic feature extractor that can be shared across tasks. Thus, we can 'zoom-in' on selective VRU instances (output from NMS) and do additional processing, preserving computation on the entire image. We perform joint training in an end-to-end fashion, with three losses: detection, classification, and pose. We use Focal Loss (Lin et al. 2017) for training the classification branch and L2 regression loss for the bounding box offsets. For the pose head, we use spatial cross entropy loss to learn a 56×56-way classification problem for every joint-point.

Since we have three losses of different magnitudes and semantics, we balance the losses by using Task Uncertainty Weighing (Kendall, Gal, and Cipolla 2018) and loss weighing.

Network Optimization

We aimed to achieve a frame rate of at least 50 fps for the deployed VRU pose estimation model in order to allow the computation of downstream algorithms in time. We optimize our trained network in TensorRT to attain a low-latency and a memory-efficient inference. Running the optimized network on the GPU involves two phases: build and deployment. In the build phase, the network parameters are optimized to generate an inference execution engine. The following optimizations are performed using TensorRT: layer fusion, deletion of redundant layers, FP32 to INT8 calibration, and target hardware-specific auto-tuning. In the deployment phase, we load and de-serialize the plan file (saved during the build phase) and run inference on the deployment hardware. Throughout the training pipeline, the model is trained using FP32 precision. Nevertheless, at the time of inference, computations are executed with INT8 precision with insignificant loss of accuracy. The quantization from FP32 to INT8 needs an intermediate step called calibration. Calibration minimizes the loss accuracy by adjusting the dynamic range and granularity of representable values for INT8 based on the activation. Calibration data is chosen randomly from the validation sub set and is used in the calibration step. The calibration data should be a representative of the data distribution. As a result of the described optimization, our model runs faster, yields higher throughput, and also holds low latency during inference. Additionally, we explore a simple scheme of linearly reducing the number of channels throughout the core network, which can serve as an effective method of obtaining networks of reduced runtime. All the above optimization techniques mentioned here are applicable to any of the state-of-the-art backbone architectures.

Evaluation

COCO mAP (Lin et al. 2014) and PASCAL VOC AP (Everingham et al. 2010) are widely used metrics to benchmark and compare the state-of-the-art object detection results. Our focus being on AD use-case where the effects of False Positives per Image (FPPI) and Miss Rate (MR) are better interpretable than precision and recall, we choose log-average miss rate (LAMR) (Dollar et al. 2011) as evaluation metric. LAMR gives the MR over FPPI curve as a single value. For pose estimation, COCO mAP metric is the widely used metric, which introduces a similarity measure for joint-points called Object Keypoint Similarity (OKS) (Lin et al. 2014). OKS behaves like IoU for measuring how similar two skeletons are based on a score [0,1]. As we are not treating pose as a primitive, which we detect directly, rather as a property of an already detected VRU box, we adapt the metric to reflect this. In the COCO mAP metric for pose estimation, OKS is exclusively used for matching joint-point, making the bounding boxes redundant. Instead, we propose a 2-step matching procedure: first boxes are matched, followed by a second step that compares joint-point of the matched boxes to confirm matching of joint-points. We modify the LAMR metric with the two-stage matching proposed above. As we evaluate the AD use-case, we are more interested in high-risk VRU instances in the vicinity of the ego car. Hence we adopt a specific evaluation methodology based on VRU size and visibility. See Table 1. We use three bins and classify the instances into *reasonable*, *occluded*, and *small* based on their height. The reasonable category is given a higher weight as these instances are of utmost importance. The occluded category is given next importance as it includes VRUs that might be directly relevant for some scenarios, e.g. pedestrian occluded by parked car. Moreover, the small instances are far away from the car, they have been given the least importance. The detection and pose estimation accuracies for each category are multiplied with their corresponding factors and are summed up to a single value for the pedestrian and rider class.

	Pedestrian/Rider		
	Reasonable	**Occluded**	**Small**
Settings	bbox >= 40px skel >= 60px occl. < 40% trunc. < 40%	bbox >= 40px skel >= 60px occl. > 40% trunc. > 40%	30 px < bbox < 60px 60 px < skel < 100px 40% < occl. < 80% 40% < trunc. < 80%
Detection weightage	0.35	0.10	0.05
Skeleton weightage	0.35	0.10	0.05

Table 1: Evaluation settings based on height and visibility of the VRU instances. The metric is reported for pedestrian and rider separately, and averaged over the two classes to get one final value for joint optimization.

Experiments

Dataset

We use the Tsinghua-Daimler Urban Pose Dataset (TDUP) (Wang et al. 2020) recorded in urban environments around Beijing area in China. It consists of full HD image and annotated for VRUs, categorized into pedestrian and rider. Every VRU instance with height greater than 20 pixels is annotated with a bounding box and instances greater than 60 pixels are annotated for both bounding box and additional 17 joint-points. Annotations contain additional bounding box attributes for *occlusion*, *truncation*, etc. and joint-point attributes like *visible*, *occluded* and *self-occluded*. The dataset contains 21.4k images with 40k pedestrian instances and 53k rider instances. Refer to (Wang et al. 2020) for more details on the dataset,

Implementation Details

We train our model for 100 epochs with a batch size of 4 images using Adam optimizer with initial learning rate set to 10^{-4}. Learning rate is manually dropped by a factor of 10 twice at 64 and 80 epochs. Weight-decay is set to 10^{-4}. While training SSD Head, we train on boxes with a minimum height of 40 pixels. We ignore boxes with occlusion and truncation greater than 40%. We also reject boxes with attributes *depiction*, *sitting-lying* and *behind-glass*, since we observe that these type of instances do not help the detec-

(Id) Experiment	Core N/W	ImageNet init	ROI crop from	ROI size	Pose head			Ped. LAMR	Rider LAMR
					# convs	# chnls	Loss weight		
(-) Baseline	GoogLeNet	Yes	4e	14 × 14	8	128	1.0	45.92	22.56
(a) ROI extraction	GoogLeNet	Yes	3b	14 × 14	8	128	1.0	45.46	23.25
	GoogLeNet	Yes	4e	14 × 14	8	128	1.0	46.47	22.70
	GoogLeNet	Yes	5b	14 × 14	8	128	1.0	47.35	23.08
	GoogLeNet	Yes	6b	14 × 14	8	128	1.0	48.20	23.99
	GoogLeNet	Yes	PRA	14 × 14	8	128	1.0	46.64	22.90
(b) Pose head	GoogLeNet	Yes	4e	14 × 14	4	64	1.0	46.38	22.41
	GoogLeNet	Yes	4e	14 × 14	6	64	1.0	46.78	22.91
	GoogLeNet	Yes	4e	14 × 14	8	64	1.0	46.14	22.36
	GoogLeNet	Yes	4e	14 × 14	8	256	1.0	46.20	22.50
	GoogLeNet	Yes	4e	14 × 14	8	512	1.0	46.17	22.82
(c) ROI and output resolution	GoogLeNet	Yes	4e	12 × 12	8	128	1.0	45.34	23.13
	GoogLeNet	Yes	4e	24 × 24	8	128	1.0	46.92	23.25
	GoogLeNet	Yes	4e	28 × 28	8	128	1.0	46.75	23.24
(d) Loss weighting	GoogLeNet	Yes	4e	14 × 14	8	128	0.5	45.29	22.44
	GoogLeNet	Yes	4e	14 × 14	8	128	1.5	47.95	24.23
	GoogLeNet	Yes	4e	14 × 14	8	128	2.0	48.77	24.93
	GoogLeNet	Yes	4e	14 × 14	8	128	2.5	49.52	25.15
(e) Feature pyramid network (FPN)	FPN-64	Yes	4e	14 × 14	8	128	1.0	44.90	22.19
	FPN-128	Yes	4e	14 × 14	8	128	1.0	44.07	21.60
	FPN-256	Yes	4e	14 × 14	8	128	1.0	43.99	21.51
	FPN-512	Yes	4e	14 × 14	8	128	1.0	44.50	21.21
(f) Core Network Capacity	GoogLeNet	No	4e	14 × 14	8	128	1.0	50.77	25.35
	GoogLeNet-90	No	4e	14 × 14	8	128	1.0	51.21	25.08
	GoogLeNet-80	No	4e	14 × 14	8	128	1.0	51.63	26.10
	GoogLeNet-70	No	4e	14 × 14	8	128	1.0	53.82	26.91
	GoogLeNet-60	No	4e	14 × 14	8	128	1.0	54.48	27.45
	GoogLeNet-50	No	4e	14 × 14	8	128	1.0	56.55	28.25

Table 2: Experimental results of our method trained and evaluated on TDUP dataset to determine the best performing model (E_1). GoogLeNet-X refers to reduced capacity of the core network with 'X' being the capacity of the original GoogLeNet.

Exp. ID	Experiment Details	Ped. LAMR	Rider LAMR	Final Timing
A	Baseline	45.92	22.56	22.45
	Performance Experiments			
B	Baseline + ROI crop from 3b	45.46	23.25	21.57
C	+ ROI crop size 12x12 , heat map output res. 48x48	45.86	23.32	20.23
D	+ Pose head conv/channels 8/64, loss weight 0.5	43.56	20.94	19.41
	Prune Experiments			
E	- ImageNet initialization removed	48.99	24.35	19.41
F	+ Core Network channel reduction 90%	49.68	25.32	18.76
G	+ Core Network channel reduction 80%	50.54	25.46	18.05
H	+ Core Network channel reduction 70%	51.42	26.25	17.26
I	+ Core Network channel reduction 60%	52.75	27.46	16.43
J	+ Core Network channel reduction 50%	54.40	28.54	15.14

Table 3: Incremental adaptations of best results from Table 2, along with incremental timing improvements by channel reduction (E_2). The LAMR values are calculated using the weighted sum as explained in the evaluation section.

tion training. Similarly for the pose head, we reject joint-points that are labeled as *self-occluded*. Results are reported as weighted sum of LAMR values as discussed in the evaluation section. We train and fine-tune the experiments on the train-val split and present the final results on the test set.

Results and Inferences

The baseline model configuration is adopted from the state-of-the-art Mask R-CNN architecture. Refer to Table 2 for the detailed configuration. We perform two sets of experiments to achieve the final run-time optimized model. In the first set of experiments E_1 we focus on identifying the best hyper-parameters to improve the accuracy. We achieve this by changing only one parameter across training runs. This allows us to measure the individual contribution of each parameter. For the second set of experiments E_2 we improve run-time and preserve accuracy. For this we use the set of parameters from E_1 and add them sequentially to the baseline model. The results from E_1 are presented in Table 2. We only present results that show significant change in performance.

We target the important blocks in the architecture that potentially can impact the accuracy and fine-tune them as

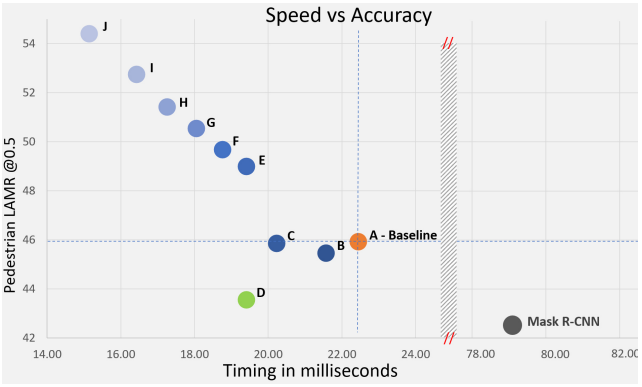


Figure 3: Accuracy vs. Speed plot for E_2. The orange circle depicts our baseline model. D (in green) depicts the best performance model. Refer Table 3 for experiment id and names in the plot. The ideal model would be closest to the origin as LAMR metric lower is better.

follows: **(a) ROI extraction strategy**: We experiment with different ROI extraction methods such as Adaptive Feature Pooling (AFP) and pyramid ROI align (PRA) for ROI extraction and ROI crops from different layers. We notice that crop features from inception (3b) improves the performance over the baseline. **(b) Pose head**: Here we vary the number of *conv.* layers and channels in the pose head. We find that a much lower number of channels (64) as compared to the baseline (128) works equally well. **(c) ROI and Output resolution**: We apply different output sizes of the ROI crops and we observe that a ROI layer size of 12×12 gives the best result. **(d) Loss Weighting for pose head**: We use different loss weights for the pose head (0.5, 1.5, 2.0 and 2.5) and observe that using loss weight of 0.5 results in best performance. **(e) Feature Pyramid Network** (FPN) (Lin et al. 2017): We use different number of channels (64, 128, 256 and 512). The experiment result shows that all the configurations for FPN gives similar improved accuracy but we do not consider these experiments to achieve our final model as FPN consume more memory and run-time. **(f) Core Network capacity**: We prune the core network capacity to 40%, using a decremental factor of 10. We perform this set of experiments to study the impact on the performance with reduction on the number of channels in the core network. We do not use ImageNet initialization for these experiments. Accuracy drops with decrease in number of channels.

Based on the observations from E_1, the second set of experiments E_2 is performed in a sequential manner and we present the result in Table 3. All the timings reported here are after TensorRT INT8 quantization as described in the Network Optimization section. The outcome of E_2 is the final model optimized for both performance and run-time. In the final model, we extract features from inception (3b), use ROI crop size of 12×12, pose head size is 8 *conv.* layers and 64 channels. We use loss weight of 0.5 for the pose head. We chose these parameters such that it facilitates us to reduce the model run-time with minimum loss in accuracy. The final model is 3.1 ms faster than the baseline model with

LAMR	Pedestrian / Rider		
	Reasonable	Occluded	Small
Detection	37.55 / 16.86	68.55 / 40.44	64.27 / 35.67
Skeleton	34.39 / 13.48	62.62 / 36.47	41.10 / 17.10
Combined (Det. and Skel.)		43.56 / 20.94	
Final (Ped. and Rider)		32.25	

Table 4: Detailed numbers showcasing the final metric calculation for Pedestrian and Rider LAMR as explained in Table 1.

equal performance. After we obtain our best model, we continue with a set of experiments by pruning the core network channels to get a speed-vs.-accuracy trade-off curve. This curve helps us to determine various operating point from which we can select a model based on either performance or run-time. The results of these pruning experiments are presented in Table 3 and the speed-vs.-accuracy curve is shown in Fig. 3. We remove ImageNet initialization for this set of experiments. From the Fig. 3 we can observe that channel reduction is a straightforward way to reduce time with minimum accuracy trade-off. In Table 4 we present the detailed metrics for our final model. We showed how our model is performing on reasonable, occluded and small boxes and also presented the combined LAMR values for detection and skeleton and final LAMR value for pedestrian and rider.

We benchmark the results of our final model and compare it with the state-of-the-art Mask R-CNN model both trained on TDUP dataset in Table 5. The inference time in TensorFlow for our final model is 48.5 ms during inference, while after quantization to TensorRT (INT8), we achieve a run-time of 19.41 ms. That is, the model runs 2.5 times faster following TensorRT optimization. We did not optimize the Mask R-CNN model as we already notice that our un-optimized final model is significantly faster than the Mask R-CNN model. Also the Mask R-CNN architecture can not be directly converted to tensorRT model and is out of scope for this paper. Pedestrian LAMR is significantly worse when compared to the rider LAMR. Our initial analysis suggests that this is because riders are found mostly *on* the street, and thus background clutter and occlusion are less pronounced compared to the pedestrian case.

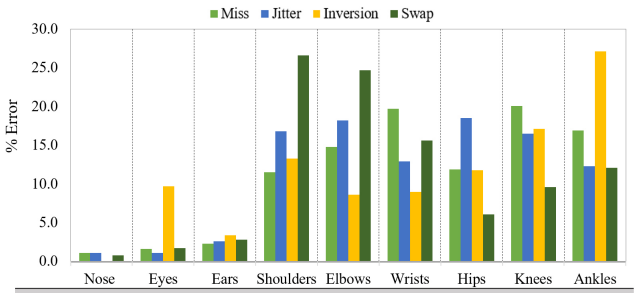
Qualitative results in Fig. 4 showcase the performance of our final model on the test set. We observe a good performance even in challenging and crowded urban scenarios. We present the analysis of join-point errors in Fig. 5. We use COCO-analyze tool (Ronchi and Perona 2017) and mainly concentrate on localization errors for our analysis which

Model	Ped. LAMR	Rider LAMR	Avg. LAMR	Timing (ms.)	
				Tensorflow	INT8
Mask R-CNN	42.47	19.32	30.90	79	-
Our final model	**43.56**	**20.94**	**32.25**	**48.5**	**19.41**

Table 5: Performance comparison to state-of-the-art Mask R-CNN architecture using a ResNet-50 backbone compared to our final model, on the TDUP test set. The LAMR values are calculated using the weighted sum as explained in the evaluation section.

Figure 4: Qualitative results of our algorithm on the test set of TDUP.

are: **(a) Miss:** large localization error, **(b) Jitter:** small error around the joint-point location, **(c) Inversion:** confusion between similar parts belonging to the same instance, e.g. the estimated left and right joint-point locations are switched for an instance, **(d) Swap:** confusion between similar parts belonging to different instances. Please refer (Ronchi and Perona 2017) for more detailed information. From Fig. 5, it is evident that for knee and ankle joint-points, i.e the joint-points at the lower-body are more prone to inversions and miss errors, as there are fewer visual cues to distinguish left from right. Whereas, for joint-points in the face i.e nose, eyes, and ears, the miss error rate is very low, indicating that they are easily detectable compared to other joint-points. We present some of the instances with the best results and a few examples of failure cases for our pose estimation model in Fig. 6.



	Nose	Eyes	Ears	Shoulders	Elbows	Wrists	Hips	Knees	Ankles
Miss	1.1	1.6	2.3	11.5	14.8	19.7	11.9	20.1	16.9
Jitter	1.1	1.1	2.6	16.8	18.2	12.9	18.5	16.5	12.3
Inversion	0.0	9.7	3.4	13.3	8.6	9.0	11.8	17.1	27.1
Swap	0.8	1.7	2.8	26.6	24.7	15.6	6.1	9.6	12.1

Figure 5: Analysis of types of errors in pose estimation using COCO analyse. The stats are generated on test set.

Figure 6: Some close-up results of best cases (first two rows) and failure cases (last row). We observe that the model has generalised for poses like walking and riding, even in scenarios with occlusion and truncation. However, our algorithm tends to fail for rare poses and in heavily crowded scenes. The last row depicts the type of common errors as described in Fig. 5. From left to right: Miss error (1-3), inversion (4-6), swap error (7, 8).

Discussion

Our optimized architecture reaches an inference time of 48.5 ms / 19.41 ms per frame with the TensorFlow / TensorRT model. This is 1.6 / 4.1 times faster compared to the Mask-RCNN TensorFlow implementation and enables the usage in real-time AD applications. At the same time we still reach a competitive combined accuracy of 32.25 LAMR (cf. Mask-RCNN with 30.90 LAMR). As we use a top-down approach, pose estimation is dependant on the accuracy of the detector. One way to improve the detector performance is to use active learning techniques (Haussmann et al. 2020). One of the most prominent errors in the pose estimation are *inversion and swap errors*. We plan to correct those wrong joint-point estimates by using additional temporal information for pose estimation (Girdhar et al. 2018). Pose estimates also suffer when the detector fails to detect multiple overlapped VRUs in a crowded scenario. While the downstream task of intention estimation expects a robust pose estimation also in crowded scenario, we plan to adopt the part association field methodology from (Kreiss, Bertoni, and Alahi 2019) to improve performance in such cases occlusion in particular.

We further plan to derive several contextual cues based on our robust pose estimates to compute more informed predictions (Rudenko et al. 2020). Preliminary experiments suggest that the estimated pose quality gives already a decent performance in estimating such cues for intention estimation and gesture recognition. This can be further improved when combined with time series modeling approaches.

Conclusions

We have presented a fast and efficient approach to VRU detection and pose estimation for real-time AD applications. With extensive experiments, we designed and optimized the network for accuracy and speed trade-offs. While the accuracy of our model (32.25 LAMR) is competitive with Mask-RCNN (30.90 LAMR), our TensorFlow model is 1.6 times faster and respectively 4.1 times faster after TensorRT conversion, resulting in a final inference time of 19.41 ms per frame (52 fps). The introduced evaluation strategy will benefit the systematic evaluation of estimated VRU poses for the AD use-case. We are convinced that the reported performance enables reliable estimation of important contextual cues like intentions or explicit gestures, making future AD systems better react to their environment.

References

Alp Güler, R.; Neverova, N.; and Kokkinos, I. 2018. Densepose: Dense human pose estimation in the wild. In *Proc. CVPR*, 7297–7306.

Andriluka, M.; Pishchulin, L.; Gehler, P.; and Schiele, B. 2014. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *Proc. CVPR*.

Aparicio, A.; Sanz, L.; Burnett, G.; Stoll, H.; Arbitmann, M.; Kunert, M.; Flohr, F. B.; Seiniger, P.; and Gavrila, D. 2017. Advancing active safety towards the protection of vulnerable road users: the PROSPECT project. In *Proc. ESV*.

Braun, M.; Krebs, S.; Flohr, F. B.; and Gavrila, D. M. 2019. EuroCity Persons: A novel benchmark for person detection in traffic scenes. *IEEE PAMI* 1–1. ISSN 0162-8828. doi:10.1109/TPAMI.2019.2897684.

Chen, X.; Hu, X.; Zhou, H.; and Xu, N. 2017. FxpNet: Training a deep convolutional neural network in fixed-point representation. In *Proc. IJCNN*, 2494–2501.

Cheng, B.; Xiao, B.; Wang, J.; Shi, H.; Huang, T. S.; and Zhang, L. 2019. HigherHRNet: Scale-aware representation learning for bottom-up human pose estimation. *arXiv preprint arXiv:1908.10357* .

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* .

Dang, Q.; Yin, J.; Wang, B.; and Zheng, W. 2019. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology* 24(6): 663–676.

Dollar, P.; Wojek, C.; Schiele, B.; and Perona, P. 2011. Pedestrian detection: An evaluation of the state of the art. *IEEE PAMI* 34(4): 743–761.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* 88(2): 303–338.

Fang, Z.; and López, A. M. 2019. Intention recognition of pedestrians and cyclists by 2D pose estimation. *IEEE Trans. on ITS* .

Girdhar, R.; Gkioxari, G.; Torresani, L.; Paluri, M.; and Tran, D. 2018. Detect-and-track: Efficient pose estimation in videos. In *Proc. CVPR*.

Hariyono, J.; and Jo, K.-H. 2015. Pedestrian action recognition using motion type classification. In *Proc. IEEE CYBCONF*, 129–132.

Haussmann, E.; Fenzi, M.; Chitta, K.; Ivanecky, J.; Xu, H.; Roy, D.; Mittel, A.; Koumchatzky, N.; Farabet, C.; and Alvarez, J. 2020. Scalable active learning for object detection. In *Proc. IEEE IV*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. In *Proc. ICCV*, 2961–2969.

Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proc. CVPR*, 7310–7311.

Huang, J.; Zhu, Z.; and Huang, G. 2019. Multi-stage HRNet: Multiple stage high-resolution network for human pose estimation. *arXiv preprint arXiv:1910.05901* .

Jhuang, H.; Gall, J.; Zuffi, S.; Schmid, C.; and Black, M. J. 2013. Towards understanding action recognition. In *Proc. ICCV*, 3192–3199.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. CVPR*, 7482–7491.

Kooij, J. F.; Flohr, F.; Pool, E. A.; and Gavrila, D. M. 2019. Context-based path prediction for targets with switching dynamics. *Int. J. Comput. Vis.* 127(3): 239–262.

Kothari, N.; Gupta, M.; Vachhani, L.; and Arya, H. 2017. Pose estimation for an autonomous vehicle using monocular vision. In *Proc. ICC*, 424–431. IEEE.

Kreiss, S.; Bertoni, L.; and Alahi, A. 2019. Pifpaf: Composite fields for human pose estimation. In *Proc. CVPR*, 11977–11986.

Kress, V.; Jung, J.; Zernetsch, S.; Doll, K.; and Sick, B. 2018. Human pose estimation in real traffic scenes. In *2018 IEEE SSCI*, 518–523. IEEE.

Li, X.; Li, L.; Flohr, F. B.; Wang, J.; Xiong, H.; Bernhard, M.; Pan, S.; Gavrila, D. M.; and Li, K. 2016. A unified framework for concurrent pedestrian and cyclist detection. *IEEE Trans. on ITS* 18(2): 269–281.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proc. ICCV*, 2980–2988.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common objects in context. In *Proc. of the ECCV. Lecture Notes in Computer Science*, 740–755.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. SSD: Single shot multibox detector. In *Proc. of the ECCV. Lecture Notes in Computer Science*, 21–37.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *Proc. ICCV*, 2736–2744.

Rasouli, A.; Kotseruba, I.; Kunic, T.; and Tsotsos, J. K. 2019. PIE: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *Proc. ICCV*, 6262–6271.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 779–788.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In *Proc. CVPR*, 7263–7271.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in NIPS*, 91–99.

Ronchi, M. R.; and Perona, P. 2017. Benchmarking and error diagnosis in multi-instance pose estimation. In *Proc. ICCV*.

Rudenko, A.; Palmieri, L.; Herman, M.; Kitani, K. M.; Gavrila, D. M.; and Arras, K. O. 2020. Human motion trajectory prediction: A survey. *Int. J. Robot. Res* 39(8): 895–935.

Settle, S. O.; Bollavaram, M.; D'Alberto, P.; Delaye, E.; Fernandez, O.; Fraser, N.; Ng, A.; Sirasao, A.; and Wu, M. 2018. Quantizing convolutional neural networks for low-power high-throughput inference engines. *arXiv preprint arXiv:1805.07941* .

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proc. CVPR*, 1–9.

Tripathi, P.; Keshari, R.; Ghosh, S.; Vatsa, M.; and Singh, R. 2019. AUTO-G: Gesture Recognition in the Crowd for Autonomous Vehicl. In *Proc. ICIP*, 3482–3486.

Uhrig, J.; Rehder, E.; Fröhlich, B.; Franke, U.; and Brox, T. 2018. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *Proc. IEEE IV*, 292–299.

Wang, S.; Wang, B.; Ramesh, J.; Mitra, A.; Weinrich, C.; and Flohr, F. B. 2020. Tsinghua-Daimler Urban Pose Dataset (TDUP), http://www.urbanpose-dataset.com/ *(to be published soon)* .