Tuning Video Segmentation for Creating Masks of Pedestrians

Quinn McIntyre, Yunqi Richard Gu, Aaryan Singhal
Stanford University
450 Serra Mall, Stanford, CA 94305
qam@stanford.edu, yrichard@stanford.edu, aaryan04@stanford.edu

Abstract

In this paper, we use FAIR's recent Segment Anything Model (SAM) [1] to create masks of pedestrians from driving footage. Fine-tuning SAM for pedestrians can lead to easier applications of machine learning models for pedestrian agent prediction that would otherwise be too computationally expensive to train on video data (ViT [2], eg.). We compare two approaches, one using SAM and the architecture XMem [3] and another using R-CNNs [4] and SAM. The first method encountered difficulties when pedestrians entered the frame or were obscured, but ran faster than the second method. The second method detected and segmented nearly all walking pedestrians.

1. Introduction

This project builds and compares several pipelines that perform image segmentation of pedestrian footage from vehicles. Given the increasing prevalence of self-driving cars, it is useful to process visual data from vehicles for prediction and analysis of pedestrian behavior to improve the safety of the vehicles. Two common ways to handle object detection are bounding boxes and image segmentation. Image segmentation is generally more computationally expensive, but for downstream analysis tasks like behavior prediction, it is crucial that pedestrian segmentation early on in the pipeline is done accurately. We hypothesize that, by performing pedestrian segmentation, we can provide downstream models that perform behavior prediction with more information than they receive traditionally via bounding boxes.

For example, if pedestrian prediction work was to be done with an architecture like the Vision Transformer [2], it would be incredibly computationally expensive to train the model to predict actions based on the raw footage. Therefore, extracting a simple representation from the footage could lead to easier computation in architectures like the ViT, while still capturing relevant features like body language, movement, and positioning.

There has been significant work on generating bounding boxes around pedestrians as seen in [5] and [4], but less recent work has been done on pedestrian image segmentation. Recently, the Segment Anything Model was released [1], which performs zero-shot segmentation of any image. This is demonstrated on pedestrian footage taken from a vehicle in Figure 1.

Figure 1. Segment Anything Model on still image from footage containing pedestrians.

In this paper, we describe two distinct models that both take in videos as input and return a video with image segmentation masks around pedestrians. For pedestrian masks to be useful for downstream prediction applications, it is important to be able to generate the masks in as short a time as possible (ideally as fast as the video plays). Such a consideration is particularly important when considering the implications of deploying such pedestrian detection and behavior prediction models in the real world. With this understanding, we decided to use a fine-tuned Segment Anything Model alongside existing architectures for bounding boxes and video memory as ways to segment pedestrians both accurately and efficiently.

Both architectures use a fine-tuned version of SAM. In our first approach, we use SAM to generate a mask of the first frame in a video clip and then use XMem [3], which uses an attention-like mechanism to propagate the initial mask through the frames in the video, to generate a video sequence of the masks. The second approach first generates bounding boxes around every pedestrian in

the video using an existing pedestrian bounding box architecture, Pedestron [5], and then uses the aforementioned bounding boxes at each frame as input for SAM, which generates a video where pedestrian segmentation has been performed via masks for each frame. We hypothesize that the latter approach will outperform the former, because it is able to access and process each frame of the video individually. However, due to reasons discussed previously, the former approach is more practical for deployment in the real-world, where computational limitations and time constraints disable frame-by-frame analysis.

We finally benchmark the performance of these two approaches against an existing architecture for general video object segmentation [6] that uses click prompting to select objects and track them throughout the video (seen in Figure 2) using an untuned SAM and XMem.

Figure 2. Clip from baseline Track Anything applied to pedestrian footage.

2. Related Work

2.1. Segment Anything

Recently, the Segment Anything Model (SAM) [1] was released. It was trained on the largest image segmentation dataset ever created (over 11M images and 1B masks). The Segment Anything architecture consists of an image encoder which uses a pre-trained ViT [2], a prompt encoder for the possible prompts of SAM (points, boxes, and text), and finally a mask decoder which uses a modification of a transformer decoder block [7].

Segment Anything performs remarkably well. Most interestingly, it is able to maintain quality even when performing zero-shot segmentation tasks on out-of-domain data. As scored by human annotators, the quality of the segment anything performs either comparably to or outperforms all other segmentation architectures.

SAM can be augmented to perform video segmentation tasks by slicing a video into sequential frame, where segmentation is performed on each frame separately and an output video is generated by stitching the segmented frames. It is worth noting that SAM can be given a keyword prompt of ”pedestrians,” but running SAM with keyword prompts on every frame is far too computationally expensive given our application. Furthermore, such an approach is likely to run into performance issues with variability in the masks between individual frames, due to the frames being segmented independently.

2.2. Track Anything

The Track Anything Model (TAM) [6] proposes an architecture that uses SAM alongside a video segmentation model called XMem [3]. TAM functions by prompting the user for the object of interest (which the user then specifies via a click). TAM then uses segment anything with the point-prompt (corresponding to the click location) to generate a segmentation mask of the desired targets. It feeds this initial mask into XMem, which uses an attention-like mechanism to propagate the initial segmentation through the video based on the masks of previous frames.

TAM is a general model that performs reasonably well on general tasks, but fails to recognize when new pedestrians enter the frame. Additionally, it requires a level of human interaction in the form of prompting and has not been fine-tuned for pedestrian detection tasks.

2.3. R-CNNs

A standard architecture used in image segmentation is the Mask R-CNN [8], which leverages convolutional layers to classify objects within an image. More specifically, each pixel in the image is classified and this information is then aggregated to produce the desired output. The architecture is applied to videos in a frame-by-frame manner, but produces lower accuracy masks than SAM and generalizes less well for out-of-domain tasks.

The R-CNN approach performs well at generating bounding boxes around objects and runs significantly faster than the mask R-CNN can segment an image. For video tasks, a frame-by-frame approach works less well for masking. In addition to this, it is important to note that the boundary boxes require less precision to be accurate than masks. With this information, our team decided to proceed with a frame-by-frame approach, where bounding boxes were generated for every frame in the video.

Large scale models for bounding box generation use the mask R-CNN architecture to perform at levels near the current state-of-the art models for pedestrian detection [4]. Specifically, these models use a ResNet [9] for the CNN in the R-CNN architecture.

3. Methods

3.1. Fine-tuning SAM

We fine-tuned the Segment Anything Model in two steps to produce the highest accuracy results for detecting the

pedestrians. The first step involved fine-tuning SAM to detect humans on a general human dataset and the second step involved fine-tuning the resulting model on a dataset of just pedestrian segmentation.

To fine-tune SAM, we start with the pretrained weights of vanilla SAM and proceed to update the weights of the mask decoder block by fine-tuning to the specific desired class for segmentation. We are able to do so by modifying the values of `decoder.parameters()` from the pretrained architecture. To perform fine-tuning, we use as input the relevant image and its corresponding bounding boxes, generated for the specified target of interest. Loss is then calculated against a ground truth segmentation mask.

SAM is first fine-tuned on the OCHuman dataset [10] to improve its segmentation of human characters. Given the new weights, we then further tune on pedestrian specific datasets using the CityScapes dataset [11].

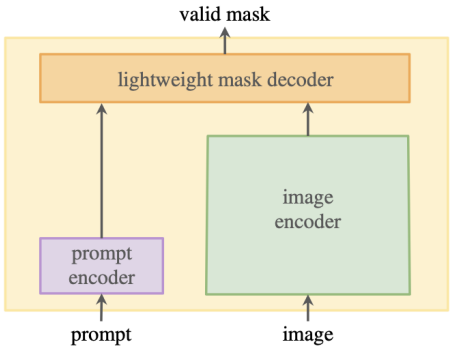

Figure 3. Segment Anything Model Network Architecture taken from [1].

After this, we performed hyper-parameter tuning to produce the most robust model. Beyond different choices for the optimizer and the loss function, there several hyper-parameters that were experimented with.

An area of particular concern for us was ensuring that the run time of the model - specifically when performing segmentation - was minimized. Given that SAM runs faster on smaller input sizes, we decided to lower the resolution of images used for training and testing. Performing this augmentation significantly improved the speed at which the model was able to evaluate pedestrian videos without sacrificing its accuracy.

Furthermore, when training segmentation models, the choice of the loss function being used can lead to wildly varying training performance. There are many ways to quantify how a segmentation mask fails compared to the ground truth and specify how pixels are weighted in the segmentation differently. We experimented with four different loss functions to determine which would lead to the most accurate model.

3.1.1 Binary Cross Entropy

One loss we used was binary cross entropy [12]. This calculates loss by comparing every pixel in the predicted mask to the pixels in the ground-truth segmentation. Using the formula below, it compares the probability of every pixel in the predicted mask against the binary value from the ground-truth. The pixel-wise losses are then averaged to calculate the total loss.

$$BCELoss(y, \widehat{y}) = -y \log(\widehat{y}) - (1 - y) \log(1 - \widehat{y})$$

3.1.2 Dice Loss

Another loss function that our team tested was Dice Loss [13]. The dice loss comes from calculating the dice coefficient, which is a measure of overlap between two masks (1 if the masks are identical and 0 if there is no overlap). It has an additional one in both the numerator and the denominator to prevent errors caused by division by zero and numerical instability.

$$\text{Dice Loss}(y, \bar{p}) = 1 - \frac{(2y\bar{p} + 1)}{(y + \bar{p} + 1)}$$

3.1.3 Boundary Loss

As an extension to this loss function test, we decided to use the boundary loss [14] to try to weight the difference between the boundaries of the segmented regions as the largest factor in the loss. The continuous difference between the regions is given by

$$\begin{aligned}\frac{1}{2}\text{Dist}(\partial G, \partial S) &= \int_S \phi_G(q)dq - \int_G \phi_G(q)dq \\ &= \int_\Omega \phi_G(q)s(q)dq - \int_\Omega \phi_G(q)g(q)dq,\end{aligned}$$

and the loss is an approximation of the boundary difference. It is given by

$$\mathcal{L}_B(\theta) = \int_\Omega \phi_G(q)s_\theta(q)dq.$$

We implement this using a discrete approximation of the difference between the boundaries. The boundaries can be identified with a simple filter - specifically, a 3x3 matrix convolved to detect edges. Although the output is a rough estimation, it provides a discrete approximation for the boundary loss which is given above.

3.1.4 Focal Loss

We tried focal loss [15]. Focal loss adds an additional γ hyper-parameter which incentivizes the function to focus

learning on examples that have a higher rate of classification error. Specifically, this is done by exponentially scaling the difference term such that more accurate predictions outputted by the model contribute relatively less to the loss, compared to predictions that are less accurate.

$$\mathrm{FL}\left(p_{\mathrm{t}}\right)=-\left(1-p_{\mathrm{t}}\right)^{\gamma} \log \left(p_{\mathrm{t}}\right)$$

3.2. XMem with SAM

Once SAM had been fine-tuned, we integrated it with the XMem architecture [3]. The XMem architecture uses an attention-like mechanism as a means of continually segmenting an object in a video. After receiving initialization via the pedestrian masks generated by the fine-tuned SAM architecture, XMem propagates its input mask frame-by-frame as a means of segmenting the desired target across all the frames in the video.

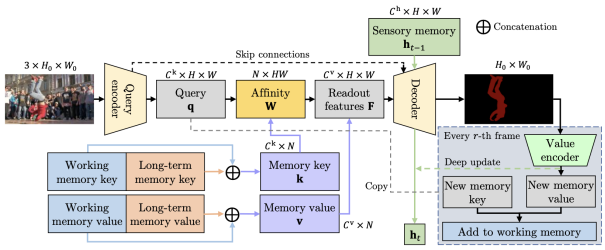

Figure 4. XMem Architecture from [3].

XMem stores three different types of 'memories,' each of which it accesses at every frame to compute new masks. XMem has sensory memory, which updates every frame, working memory, which updates every r frames (where r is a given hyperparameter), and long term memory, which is updated from the working memory. Each memory store learns feature representations to predict the location of the object in new frames. Once generated, these predictions then enter the sensory memory.

Learning to read from these three memory stores is done by learning parameters for query, key, and value matrices, which use an attention process to update the segmentation. Segmentation is then added to working memory using another encoder.

One successful method for pedestrian video segmentation was to seed XMem with the precise segmented image in the first frame and then propagate that frame throughout the rest of the video using the XMem architecture. Such an approach results in near-real-time video segmentation.

3.3. R-CNN Bounding Boxes with SAM

Our second approach is to use an existing pedestrian detection model, Pedestron [4], to generate bounding boxes around the pedestrians in video footage.

We break the video into still frames, which we feed into the fine-tuned SAM with the bounding boxes for pedestrians as the prompts. Using such a methodology proved considerably more effective than prompting SAM with just the still frames. When given still frames, even with the fine-tuned model, there would be higher variability in the segmentation due to the wider possible field of view. Instead, this allows the model to be prompted by the same input as used in fine-tuning and the prompt of a bounding box.

4. Dataset

4.1. SAM Fine-tuning Dataset

4.1.1 OCHuman

To fine-tune the model to recognize humans, we used the OCHuman dataset [10] which contains 13,360 annotated human instances within 5,081 images, with an average MaxIoU (which measures the severity of an object being occluded) of each person being 0.573. The focus on occlusion is particularly valuable, as one of the key instances we are training for is when pedestrians are obscured from view by some object. As part of the training set of SAM, we used its bounding boxes and instance masks. We used OCHuman as the first phase of SAM training since the figures in the images are larger, which may have led to improved performance in our tuning process.

4.1.2 Cityspaces

The Cityspaces dataset [11] was used to further fine-tune SAM to detect pedestrians. This consists of 5000 finely annotated images as well as 20000 annotated images with course annotations. It features footage from 50 cities in Europe and contains annotation for 30 classes. We used the images and bounding boxes of the pedestrians as well as the ground truth segmentation of the pedestrian class.

4.2. Video Segmentation Dataset

To train and test the video segmentation approaches, we have drawn our video data from the publicly available PIE dataset [16]. This dataset is in the form of 10 minute long videos as footage from moving vehicles containing annotations of bounding boxes of pedestrians. It has 53 ten-minute videos of car footage taken from various cities. This footage is higher resolution than is needed for segmentation, and it takes far too long to segment the higher resolution clips. It is necessary to undergo pre-processing to put it into a form that is useful to tune the segmentation models to.

We proceeded by breaking down the footage into many shorter segments (5 seconds) containing pedestrians which we will use to tune the model. These segments may contain some overlapping footage for the purpose of giving the model more data on which to train. Additionally, the raw

footage was reduced in resolution and bit-rate to near the minimum of what SAM could successfully generate segmentation masks (we found this to be a bit-rate of 1000). For training, we used a randomly selected 80/10/10 training, validation, test split of these 10 second clips. In total, there are 6360 5-second clips used.

5. Experiments

5.1. Performance of Fine-tuned SAM

The first step is to test various loss functions. The common loss functions for segmentation work are focal loss, dice loss, boundary loss, and BCE loss. After trials of these functions or their combinations (see Figure 5) on the OCHuman, the general human detection dataset, we discovered that only dice loss and dice + BCE + focal is suitable, while the latter updates the parameters much slower than the former. Thus, we adopted dice loss as the loss function.

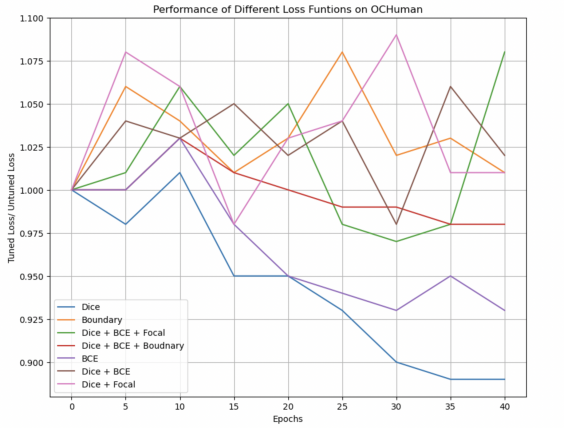

Figure 5. Loss function performance on the OCHuman dataset (the loss is normalized by the zero-shot loss of untuned SAM).

After fixing our loss function, we checked the performance on three optimizers: Adam, AdamW, and RMSprop. AdamW outperforms the other two optimizers.

Next we tuned the hyperparameters of AdamW on the CitySpace dataset for further improvement. We observed similar metrics for the choice of loss function so we proceeded with dice loss for the CitySpace set. We found that the model works the best when fixing learning rate to be 5e-7, beta1 to be 0.9, and beta2 to be 0.99.

The model increased its accuracy on humans, by having greater success in identifying the target, as well as more connected, continuous shapes. This success is demonstrated in Figure 6. It additionally can handle adjacent people well at times as shown in Figure 7

The performance of the tuned model still struggled when segmenting multiple humans overlapping, particularly as it seems to fail to determine which body parts belong to which

Figure 6. Example of the success of fine-tuned SAM from OCHuman dataset.

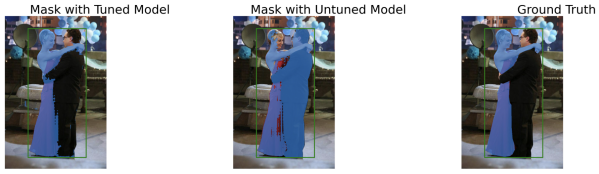

Figure 7. Example of the success of fine-tuned SAM from OCHuman dataset.

as seen in Figure 8 where even the fine-tuned model segments the torso of one player and the legs of another.

Figure 8. Example of a failure case of the fine-tuned segment anything model.

5.2. SAM + XMem Results

The use of SAM + XMem performed identically to Track Anything. It is important to note that, qualitatively, SAM + XMem had slightly finer masks. It is worth noting that fine pedestrian segmentation, while useful for many use-cases, is not the utmost priority in the autonomous vehicle space. However, it is the manner in which SAM + XMem processes inputs (specifically, bounding boxes generated by Pedestron) that make it more appealing than Track Anything, given the end use-case. The result from the video can be seen in Figure 9, with the extracted mask at that step visualized in 10.

5.3. R-CNN Results

The R-CNN took longer to run than XMem, but produced higher accuracy and did not have the difficulties of failing to track new pedestrians that entered the frame. It is, however, worth noting that R-CNN faced difficulties finding people that were sitting (for example in a cafe along the side of the street) and sometimes failed to identify individuals

Figure 9. Example of XMem propagation on pedestrians.

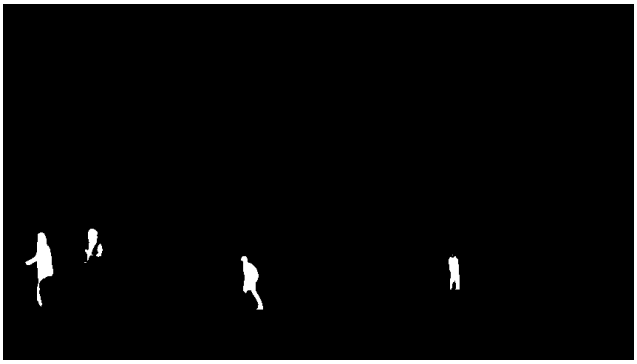

Figure 10. Mask after some propagation.

walking in a group. Generally, the model was able to handle partial-observability of pedestrians really well. The model used in this paper was parameterized such that bounding boxes were only outputted in the video if the model had greater than or equal to 80 percent confidence that the bounding box contains a pedestrian. As a result, there were a negligible number of bounding boxes generated without pedestrians in them. A higher required minimum confidence would increase the number of bounding boxes generated in the video, meaning almost no pedestrians could be missed. Such an approach, however, was specifically avoided, given that the primary purpose of Pedestron was to generate inputs for SAM. The R-CNN can be seen in 11.

5.4. Benchmarking Pedestrian Tracking

The benchmarking process contains two sections: first, a comparison between the performance of SAM and the click-style input of Track-Anything model in generating correct masks for the pedestrians; second, a comparison between the performance of XMem, SAM + R-CNN, and Track-Anything in tracking the pedestrians given correct masks.

The test dataset has 20 5-sec video clips extracted from 5 different 10-min video clips from PIE dataset. Each 5-sec test video are randomly picked from these 10-min clips, varying from weather, light, number of people, and car speed, and thus covers different situations.

Figure 11. R-CNN bounding boxes of pedestrians.

Table 1: This table quantifies the accuracy of pedestrian identification. In the case of Track-Anything, we permit a maximum of two clicks on the pedestrians to create the mask. For SAM+XMem, we count the accurate masks generated by SAM. In the scenario of RCNN+SAM, we count the accuracy of the mask produced by SAM, using the bounding box provided by RCNN.

Table 2: This table is made from a thorough examination of the resulting video, maintaining a count of the total number of pedestrians appearing within a certain time period. It also counts the number of pedestrians that the model was unable to successfully track.

Table 3: This table analyzes reasons of inaccurate tracking. Two potential causes are considered: either the tracking is interrupted (for instance, being obstructed by another vehicle, or due to two pedestrians intersecting each other, leading to a confused tracking of the mask) or the pedestrian does not appear in the initial frame of the graph and thus lacks a corresponding mask.

Table 1. Identification Accuracy Contrast

Category	**Count**
Track-Anything(2-click)	90%
SAM + XMem	90%
R-CNN + SAM	93%

Table 2. Tracking Accuracy Contrast

Category	**Percentage**
Track-anything	97 (74%)
SAM + XMem	97 (74%)
R-CNN + SAM	122 (93%)

On the test set, identification accuracy based on given location is 100%, with 86% successfully located in the first

Table 3. Tracking Loss Analysis Contrast

Category	Percentage
Track Interrupted in Track-anything	11.8%
Appearing during tracking in Track-anything	88.2%
Track interrupted in XMem	11.8 %
Appearing during tracking in XMem	88.2 %
Track interrupted in RCNN-SAM	0 %
Appearing during tracking in RCNN-SAM	0 %

click and 14% in the second click. The estimated accuracy of tracking is 76%, and out of the loss there is about 25% caused by being intermittently blocked by other objects and 75% caused by that pedestrians appear in the middle of the 5-sec clip, therefore not identifiable in the first frame.

6. Conclusion and Future Work

We found that fine-tuning SAM leads to more accurate and precise segmentation around pedestrians than the zero-shot capabilities of Segment Anything Model. We found that the higher quality masks with XMem performed as well the click-prompted Track Anything Model and that the higher quality pedestrian masks led to better tracking of pedestrians in the video in some cases. While more computationally expensive, the process of prompting SAM with bounding boxes of each pedestrian led to higher accuracy for pedestrian tracking.

A potential area of future work would be to modify the memory structure of XMem such that it could be prompted by new pedestrians entering the frame, because currently it fails when new objects enter the frame. Modifying it to learn when to re-segment the image, or have the working memory store temporarily updated by new SAM applications could lead to efficient tracking of new pedestrians who enter the frame.

7. Contributions

- Quinn McIntyre: Reviewed literature, decided the appropriate approach, found relevant papers and architectures and determined how to perform both segmentation methods. Decided the two architectures and planned their implementations. Found the video dataset and wrote the code to process the data and downsample to the lowest resolution SAM could perform well on. Helped write loss functions for fine-tuning SAM. Wrote sections 1, 2, 3, 4, and half of section 5. Set up and ran the XMem architecture on the fine-tuned masks. Built the pipeline to run and evaluate SAM + XMem. Benchmarked XMem on test set.

- Yunqi Richard Gu: Completed the fine-tuning work of SAM. Decided the appropriate approach of automatic detection from original image to successful tracking. Found the OCHuman and CitySpace datasets for SAM tuning. Implemented, visualized, and decided among different optimizers and loss functions in the tuning process. Wrote the code for the visualization of masks and collected the results of Track-anything. Established benchmark standards and completed benchmarking for the performance of SAM and Track-anything.

- Aaryan Singhal: Reviewed literature, decided the appropriate approach, found relevant papers and architectures and determined how to perform both segmentation methods. Wrote the code to break .mov files into frames, set up and ran Pedestron to do bounding box generation, stitch frames with bounding boxes into continuous videos, and extract/store bounding box coordinates in a .csv file to be processed for SAM inputs. Helped edit sections 1, 2, 3, 4, 5.

References

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 1, 2, 3

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1, 2

[3] Ho Kei Cheng and Alexander G. Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 1, 2, 4

[4] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Pedestrian detection: Domain generalization, cnns, transformers and beyond. *arXiv preprint arXiv:2201.03176*, 2022. 1, 2, 4

[5] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Generalizable pedestrian detection: The elephant in the room. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11328–11337, June 2021. 1, 2

[6] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos, 2023. 2

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[10] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul Rosin, Zixi Cai, Xi Han, Dingcheng Yang, and Haozhi Huang. Pose2seg: Detection free human instance segmentation, 2018. OCHuman dataset available at https://github.com/liruilong940607/OCHumanApi. 3, 4

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 4

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 3

[13] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer International Publishing, 2017. 3

[14] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. *Medical Image Analysis*, 67:101851, jan 2021. 3

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 3

[16] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K. Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *International Conference on Computer Vision (ICCV)*, 2019. 4