Scene and Task Adaptive Event and Object Detection

A Dissertation Presented

by

Zekun Zhang

to

The Graduate School

in Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2 2024

Stony Brook University

The Graduate School

Zekun Zhang

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation,

Minh Hoai Nguyen, **Advisor**
Associate Professor, Computer Science Department, Stony Brook University

---

Haibin Ling, **Chair**
Professor, Computer Science Department, Stony Brook University

---

Dimitris Samaras
Professor, Computer Science Department, Stony Brook University

---

Lingqiao Liu, **External**
Associate Professor, Australian Institute for Machine Learning, University of Adelaide

---

Chao Chen, **Co-Advisor**
Associate Professor, Department of Biomedical Informatics, Stony Brook University

This dissertation is accepted by the Graduate School

Celia Marshik
Dean of the Graduate School

August 2 2024

Abstract of the Dissertation

Scene and Task Adaptive Event and Object Detection

by

Zekun Zhang

Doctor of Philosophy

in

Computer Science

Stony Brook University

2024

Event and object detection in video streams is an important task in many computer vision applications, from building surveillance and traffic monitoring to autonomous driving and quality assurance. Typically, a detection model is trained and deployed across various scenes and situations. This model remains unchanged after deployment, except for sporadic updates, and it lacks the capability to adapt to individual scenes or specific tasks to improve its performance on ever-changing scenes. This leads to unsatisfactory performance and, in many cases, degrading performance over time. In this dissertation, I seek to address this problem by developing methods that can adapt a trained detector to diverse scenes or novel tasks, with the objective of improving its detection timeliness or precision. I aim for algorithms that go beyond the trivial approach of manual data collection and fine-tuning, which often requires a large amount of human annotation and leads to separate models for different scenes, significantly requiring more processing memory and decreasing the processing throughput.

In this dissertation, I present several methods that can adaptively improve trained detection models in terms of timeliness or precision. These methods do not require a large amount of human annotation nor prohibitively increase the computational costs. First, I will illustrate the adaptation of a model—from recognizing target events post-occurrence to an early event detector—by leveraging a video stream and updating the detector through self-supervision. Second, I will introduce a framework capable of adaptively enhancing the detection performance of an object detector across various new scenes via self-supervised object labeling and exploiting consistency among objects and backgrounds. Third, I will present a method that can adapt an object detector to a large number of scenes while maintaining a similar computational cost, by integrating a mixture-of-experts architecture and through self-distillation training. Lastly, I will demonstrate a framework that improves the detection precision of an open-vocabulary object detector by enhancing the prompt feature without changing the model parameters. The enhancement is actuated by a lightweight module that only requires a small number of sample images with target objects labeled to train.

Contents

List of Figures

List of Tables

Publications

- Zekun Zhang, Farrukh M. Koraishy, and Minh Hoai. "Exemplar-Based Early Event Prediction in Video". In: *British Machine Vision Conference*. 2021.
- Vinh Tran, Yang Wang, Zekun Zhang, and Minh Hoai. "Knowledge Distillation for Human Action Anticipation". In: *IEEE International Conference on Image Processing*. 2021.
- Zekun Zhang and Minh Hoai. "Object Detection With Self-Supervised Scene Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023.
- Shih-Po Lee, Zijia Lu, Zekun Zhang, Minh Hoai, and Ehsan Elhamifar. "Error Detection in Egocentric Procedural Task Videos". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2024.
- Zekun Zhang, Vu Quang Truong, and Minh Hoai. *Efficiency-Preserving Scene-Adaptive Object Detection*. British Machine Vision Conference. 2024.
- Zekun Zhang, Vu Quang Truong, and Minh Hoai. *Prompt Adaptation for Open-Vocabulary Object Detection*. under review. 2024.

Acknowledgements

I am fortunate to have met and collaborated with many wonderful people during my Ph.D. study here at Stony Brook University. I would like to express my gratitude at the beginning of my dissertation.

I thank my advisor Prof. Minh Hoai Nguyen for his guidance and support. He showed me the way of becoming a computer science researcher. Every time I feel blocked in my research, an in-depth discussion with him has always been the elixir. I thank Prof. Dimitris Samaras for being on the committee for my RPE exam, dissertation proposal, and defense. He has been moving things forward in the computer vision lab for years, especially during the hard COVID times. Parties organized by him have always been the highlights of our lives too. And I thank Prof. Haibin Ling, Prof. Lingqiao Liu, and Prof. Chao Chen for being on my defense committee. Their feedbacks and suggestions have been very helpful. I also thank Prof. Michael Ryoo, who cannot be on my defense committee due to schedule conflicts, for giving me valuable advices during for my RPE exam and dissertation proposal. And I thank Prof. Ehsan Elhamifar and Dr. Farrukh M. Koraishy for their collaboration in my research.

I thank Prof. C.R. Ramakrishnan, Ms. Kenna O'Leary, Ms. Cynthia Santa-Cruz-Scalzo, and Ms. Kathy Germana for their tremendous help during the years. They have helped me keep on top of all the requirements and deadlines on the path to my Ph.D. degree. And I thank Prof. Hong Qin for his valuable advice at the beginning of my study.

I thank my lab mates and other coauthors, Yang Wang, Boyu Wang, Vinh Tran, Viresh Ranjan, Supreeth Narasimhaswamy, Huy Anh Nguyen, Yifeng Huang, Sounak Modal, Zijun Wei, Le Hou, Zhixin Shu, Ke Ma, Huidong Liu, Hieu Le, Vu Nguyen, Qiaomu Miao, Jingwei Zhang, Lei Zhou, ShahRukh Athar, Sagnik Das, Eugenia Soroka, Jiaxiang Ren, Ruyi Lian, Jingyi Xu, Ruoyu Xue, Shih-Po Lee, Zijia Lu, Vu Quang Truong, and many many others. I wish them success in their study and career, and above all, to be happy.

Lastly, I thank all my family members and friends. Their unconditional support gave me strength to push through hardships during my study. I look forward to spending more wonderful time with them in the future.

Zekun Zhang August 2024

Chapter 1

Introduction

Event and object detection in video streams have been of high research interests in modern computer vision, as such detection models can be used in a wide range of applications such as surveillance [48, 155, 186, 201], content creation [7, 77, 99], traffic [1, 78, 160, 164, 233], and procedural task assessment [34, 112]. As cameras can produce high quality videos at an affordable price, it is desired to apply detectors to a number of video streams to perform certain tasks. It is common to develop a model architecture according to the specific needs, such as a regional convolutional network for object localization [73, 74, 80, 172], and train the model on a specific dataset. Then this trained model is deployed to various cameras, where it continuously observe the video streams and perform the tasks.

However, this model faces the risk of performing poorly, as each camera is installed in dynamic scene, and new cameras can be installed after the model is trained. What is more, when the need to detecting novel objects or events appears, the model cannot perform on them without retraining. In such cases, it is desired to adapt the trained model to specific scenes or tasks, in order to improve its performance. This improvement is critical in many cases. For instance, a model that detects anomalies from video streams can be much more useful if it can detect earlier, giving more time for the responses. An object detector deployed on a traffic camera will give much more accurate results if it can recognize vehicles under adversarial conditions like snow and fog. Since the video streams contain rich information about the scene and task, this kind of improvement can be actuated by training the model on more data collected on the cameras and annotated by human experts. But this approach is not practical in terms of cost and time, and especially not scalable when hundreds of cameras are used. And if a adapted version of the model is created for each camera, the computational cost of hosting hundreds of models will not be acceptable.

In this dissertation, I present several methods that adaptively improve the timeliness

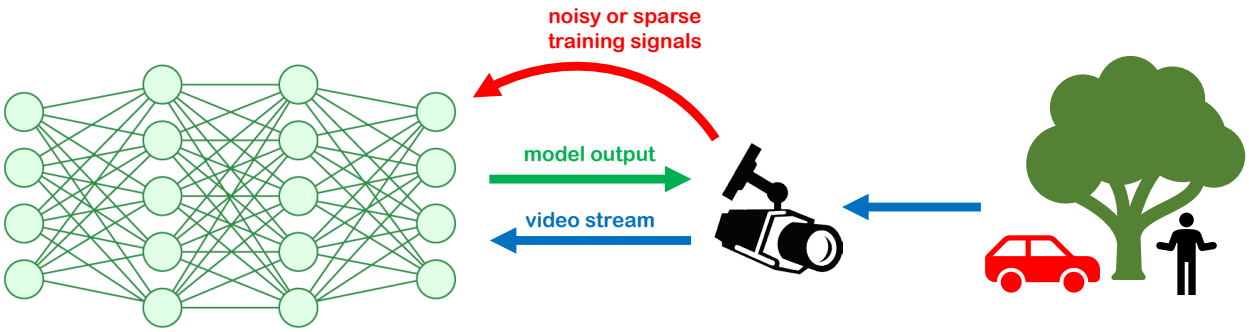


FIGURE 1.1: Illustration of adaptive model improvement from video streams. When a computer vision model observes the video stream and performs tasks, it also get feedback from self-supervision or sparse human labels, which can be used as training signals to update and improve the model.

or precision of trained event and object detection models. Such adaptive improvement can be actuated via self-supervision or from sparse human labels, thus making it practical in cases when annotation is difficult or expensive to obtain. As illustrated in Figure 1.1, when the model performs its designated tasks on the incoming video stream, it sends the outputs of the model, and gets a feedback signal. This signal is either generated via a self-supervised mechanism, or from minimal human labor. The model then use this as the training signal to update and improve itself. This framework enables the model to improve at the same time of operation, and get better overtime when more frames are observed. It also lets the model to improve according to the scene and task it has been deployed on, instead of increasing metrics that are generic to a collection of video streams. I apply this adaptive improvement framework on several event and object detection problems and demonstrate its effectiveness.

In chapter 2, I demonstrate how to improve the timeliness of an event detector, which is beneficial for early and effective responses, by letting a event detector, which can recognize the target event with high precision but only when the event has occurred fully, to continuously observing and labeling video segments from a video stream. The detection results from this full event detector acts as the supervision for training the early detector. I implement the idea of exemplar learning to train an ensemble of exemplar classifiers to separate events from non-events. The performance of each exemplar classifier on a held-out sample set is used to calibrate the output score. This essentially identifies the detectable events undetectable ones, effectively reduces the noise in the training signal.

The learn ensemble model has the ability to detect the target events early with high precision when they are detectable.

I shift gear and focus on object detection in chapter 3, as object information is important and is used by many high-level computer vision tasks. I investigate the situation of applying a trained object detector on a collection of stationary scene cameras, and aim to improve the detection performance. The detector adapts to each scene by training on self-supervised pseudo labels from an ensemble of detection and tracking models, which improves the quality of the pseudo labels. Both object and background consistency are utilized for more precise separation of objects. I also introduce a large-scale and diverse dataset called Scenes100 for the development and evaluation of scene-adaptive object detection models. Experiments on this dataset show significant improvement of detection performance from the original detector using the proposed methods.

However, if detection precision is prioritized, the adapted models often have higher computational cost and more parameters than the original model, which leads to lower efficiency. And since a separate model is created for each scene, this adaptation process is difficult to scale. In chapter 4 I show that it is possible to obtain improvement in detection performance on a collection of scene cameras while still preserving efficiency and scalability. I implement a mixture-of-expert architecture, which only create parallel branches for more scene-specific modules in an object detector, and uses shared scene-generic modules. The model is trained through self-distillation using upscaled images, and outperforms baseline models while using less memory and computation.

Yet in certain real-world applications, the model needs to be fixed after inference-time optimizations. This makes adaptively changing the parameters of the model impractical. So finally in chapter 5, I present a framework that adapts open-vocabulary object detectors to various scenes without the need of changing the weights of the original model. Instead, I propose to optimize the prompt feature input to the model for the adaptive improvement. The prompt feature is optimized by a lightweight enhancer module, which is trained from a limited number of sample images with the target objects labeled. This module can obtain consistent detection performance improvement on different datasets and different types of target objects.

Chapter 2

Exemplar Based Early Event Detection

In this chapter, I aim to improve the timeliness of a trained event detection model through self-supervised training signals. By letting an event detector to continuously observe a video stream, it is expected to make connections between visual features that appear in the early stage of the target event and the event itself, thus learn to make decisions before the event fully occurs. Improvement in timeliness is highly desired in many cases. For example, an anomaly event detection model that makes alert early can enables quick and effective responses. It is also challenging, as the model needs to recognize more subtle signals that will lead to the full occurrence of the event. I tackle this issue by utilizing the output of a full event detector. This is made possible by the fact that detection of the target event when it has fully occurred is relatively easy, as it shows distinctive indications. Thus a trained computer vision model can detection the full event with high confidence and precision. I present a method for early detection of visual events, trained from detection results from a full event detector. The early detector is based on an ensemble of exemplar classifiers. Each exemplar classifier is associated with an instance of the target event, being trained to separate the target event from non-events. The exemplar classifiers can be calibrated and integrated to create a stronger detector. Experiments on several datasets show that the proposed exemplar-based framework outperforms other methods, yielding higher precision given fewer training samples. The content of this chapter is based on the published paper [243], and "we" in this chapter refers to the authors of this paper. The code and datasets can be found at `https://github.com/cvlab-stonybrook/EnEx`.

2.1 Introduction

Being able to monitor a video stream and detect an event of interest before it fully occurs is profound capability that has numerous applications in a wide range of fields, ranging

from entertainment and environment to healthcare and security. For example, the ability to recognize dangerous events in a forest, such as wild fires, will enable quick response, minimizing damages and improving safety. However, compared to detecting the event when it already happens fully, which can be easily actuated by detecting the wild-spread flames, this early detection requires the ability to read the subtle signals such as smoke. We aim for an algorithmic framework that will turn a full event detector running on a video camera into an early detector that continually improves its performance based on delayed self-supervision signals. We focus on scenarios where the camera can observe the same type of scene for an extended period of time and visual events from a target category of interest occur infrequently but repeatedly.

In general, it is difficult to detect visual events early due to the stochastic nature of visual data and the limited field of view of the camera. An event can be detected in advance only if there is some precursory evidence and that evidence can be recognized. However, many visual events, such as picking up a phone or opening a door, might occur without any prior visual indication. Even when prior visual indication does exist, it might not be observable or detectable from the perspective of the camera. The difficulties of early detection of visual events can be seen from the training perspective. Let us refer to a video segment that leads to a target event of interest as *pre-positive*, and a video segment that is not followed by a target event as *pre-negative*. For early detection, we need to distinguish pre-positive video segments from pre-negative ones. To do so, we can collect the previously observed pre-positive and pre-negative video segments by running a full event detector on a video stream and label when the target event occurs. Then we can backtrack to the preceding video segments of the events and non-events to train a classifier to separate these two sets. However, visual data is highly stochastic and visual events can occur without any prior indication. Many pre-positive video segments are similar or even identical to pre-negative segments, and for a naïve approach, much of the training effort would be wasted on the failed attempt to separate the non-separable, which leads to overfitting and low precision.

The premise of our work is that not all target events can be detected early, but many of them can be detected early with a high level of certainty. Instead of aiming to detect everything and ending up with a model that is not confident about anything, we propose to focus on identifying and detecting the precursory patterns that will surely lead to a target event. In other words, we should prioritize high precision over high recall. Having a detector with high precision is very important for many applications, especially

when it is much more likely for the target event not to happen and early detection is only meaningful when the event does happen.

We develop here a non-parametric method for early detection of visual event based on an ensemble of exemplar classifiers. Each exemplar classifier is obtained by training a classifier to separate a single pre-positive video segment from many pre-negatives. Each exemplar classifier is like a local distance function that is discriminatively trained, representing a candidate precursory pattern that should be monitored. Our approach does not suffer from the drawbacks of parametric classifiers, which treat all pre-positive segments as a whole, implicitly assuming that pre-positive segments are visually related and separable from the pre-negatives. Experimental results on multiple visual events show that the proposed method outperforms both parametric baseline methods and other exemplar based methods in terms of precision. The proposed method is faster to train and more robust to the noisy training data.

2.2 Related Work

2.2.1 Detection and prediction in video

We propose a framework that can detect high-level semantic properties of events early, unlike methods for recognizing mid-level features [167, 188, 207, 209] or low-level feature maps such as raw video frames [111, 139, 216, 225], optical flow maps [68, 208]), segmentation maps [135], dynamic images [174], and human poses [56, 106, 199, 210, 211, 212]. There are methods for recognizing partial human actions by training frame-based classifiers [49, 138, 179, 234] or segment-based classifiers [88, 168, 177, 211]. More recently, many end-to-end neural network based methods are proposed for early recognition or anticipation [3, 53, 60, 61, 183], but they rely on large offline datasets with high-quality and dense annotation to learn good video representation and classifiers. None of the aforementioned works are designed for self-supervised continual learning, where target events happen infrequently. They do not consider the difficulties of separating the non-separable and they do not aim for a predictor with high precision.

One might assume the sequence of video frames or events as a Markov process and use methods such as Hidden Markov Models [165] for prediction. This is the framework of many existing prediction methods such as [23, 36, 158, 180], but this approach is not suitable for early detection of high-level visual events for two reasons. First, visual

events are complex and do not usually satisfy the Markov property. Second, to model a video with a Markov process, we need to specify a state space, and this approach is only appropriate if the states are what we aim to predict. Similarly, time series forecasting techniques, such as ARMA [218] and ARIMA [14], can be used for forecasting future observation values. These methods are appropriate if we are interested in detecting raw observation values such as the Dow Jones Utilities Index [15]. However, knowing raw observation values is not enough to predict high-level semantic events.

2.2.2 Exemplar based learning

In machine learning, the term exemplar refers to those training samples that are representative and can serve as prototype of certain given dataset, so model trained on them can achieve similar performance as the model trained on the whole dataset. Exemplar based learning are successfully used for image classification [236], detection and segmentation [32, 136, 137, 190, 198], and action recognition [219, 229]. Since each exemplar only represents a local region of the problem, it is related to local learning and local distances [12, 58, 59]. Our work is most similar to the object detection framework of [137], where instead on learning an universal model to separate object windows with non-object windows, a separate linear SVM is trained on every single object window with many non-object windows. In this way, a per-exemplar distance is learned. In reference, the exemplar that is closest to the query sample is used. If none of the exemplar is close enough, the query samples is regarded as negative. Their approach, however, is inefficient for continual learning, which is an issue that we address in this chapter. Moreover, we propose a more robust approach for calibrating and combining multiple exemplar classifiers.

2.3 Framework Description

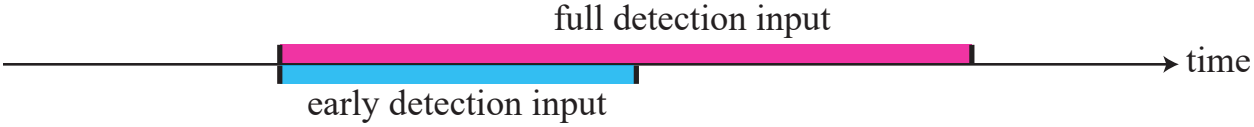


FIGURE 2.1: Early detection of events in a video stream. Compared to a full event detector, an early detector needs to make decisions based on the observation before the event has fully occurred.

We envision a self-supervised learning framework that will turn a full event detection on a video camera into an early event detector that continually improves its performance based on delayed self-supervised signals. At time t, the full event detector analyzes the video sequence up until t and detects whether an event of interest has occurred between times $t - \tau_1$ and t, inclusively. This is represented by a binary variable y_t. The early event detector only observes the video sequence up to time $t - \tau_2$. We assume $\tau_2 > \tau_1$ and refer $\tau_2 - \tau_1$ as the lead time τ_* of early detection. Let the input of the early detector be $\mathbf{v}_t$, which is the feature representation for the video segment in time $[t - \tau_2 - l, t - \tau_2]$, where l is the length of the input window. The goal of the early detector is learning how to predict y_t from $\mathbf{v}_t$, as illustrated in Figure 2.1.

Suppose we can let the full event detector keep observing and labeling training examples for the early detector from our past observations. At time t, the accumulated set of labeled training examples is $\{(\mathbf{v}_i, y_i)|i = 1, \ldots, t\}$. In general, there are not many positive examples where an event is detected, and let $\mathcal{P} = \{\mathbf{z}_1, \ldots, \mathbf{z}_p\}$ denote this subset. On the other hand, there are many negative non-event samples, and we do not need to use all of them for training. We will sample two disjoint subsets of negative examples $\mathcal{N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathcal{N}^h = \{\mathbf{x}_1^h, \ldots, \mathbf{x}_n^h\}$. The former will be used for training the exemplar classifiers, while the latter will be used for calibration. The training and calibration of the exemplar classifiers will be explained in details below.

In this section, we describe our method for early detection based on an ensemble of exemplar predictors. Each exemplar classifier $f_{\mathbf{z}}(\cdot)$ is obtained by learning to separate a *single* pre-positive video segment $\mathbf{z}$ from many pre-negatives. We propose a novel formulation to efficiently train multiple exemplar classifiers together, making it suitable for continual learning. We also propose a novel approach for calibrating and combining the exemplar classifiers using rank pooling to create a stronger predictor.

Notation. We will use bold uppercase letters to denote matrices (e.g., $\mathbf{D}$), bold lowercase letters to denote column vectors (e.g., $\mathbf{d}, \boldsymbol{\alpha}$). $\mathbf{d}_j$ represents the j^{th} column of the matrix $\mathbf{D}$, while d_j is the j^{th} element of the column vector $\mathbf{d}$. d_{ij} denotes the scalar in the row i and column j of the matrix $\mathbf{D}$ and the i^{th} element of the column vector $\mathbf{d}_j$. Non-bold letters represent scalar variables. $1_n \in \mathbb{R}^n$ is a column vector of ones. $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is an identity matrix.

2.3.1 Exemplar classifier from single positive example

Given a particular positive example $\mathbf{z} \in \mathcal{P}$ and the set of $\mathcal{N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of negative examples, we propose to learn the exemplar classifier based on Kernel Ridge Regression [191]. Specifically, we consider the RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||_2^2)$, where γ is set to the inverse of the average squared distance between elements of $\mathcal{P}$ and $\mathcal{N}$. Let $\phi(\mathbf{x})$ be the implicit feature function that maps a data point $\mathbf{x}$ to the feature space. For the exemplar predictor, we will learn a weight vector $\mathbf{w}$ and a bias term so that: $\mathbf{w}^T\phi(\mathbf{z})+b = 1$ and $\mathbf{w}^T\phi(\mathbf{x}_i) + b \approx 0$. We can learn $\mathbf{w}$ and b as an optimization: $\text{minimize}_{\mathbf{w},b}\ \lambda||\mathbf{w}||^2 + \sum_{i=1}^n(\mathbf{w}^T\phi(\mathbf{x}_i) + b)^2$, s.t. $\mathbf{w}^T\phi(\mathbf{z}) + b = 1$, where λ is a regularization parameter. From the constraint, we have $b = 1 - \mathbf{w}^T\phi(\mathbf{z})$ and the objective is equivalent to:

$$\underset{\mathbf{w}}{\text{minimize}}\, \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(1 - \mathbf{w}^T(\phi(\mathbf{z}) - \phi(\mathbf{x}_i)))^2. \tag{2.1}$$

Let $\mathbf{l}_i = \phi(\mathbf{z}) - \phi(\mathbf{x}_i)$ and $\mathbf{L} = [\mathbf{l}_1, \ldots, \mathbf{l}_n]$. The solution of the above optimization problem can be found by setting the derivative to zero: $\mathbf{w} = (\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{L}\mathbf{1}_n$. On the other hand, we have $(\mathbf{L}\mathbf{L}^T+\lambda\mathbf{I}_n)\mathbf{L} = \mathbf{L}(\mathbf{L}^T\mathbf{L}+\lambda\mathbf{I}_n)$, then $\mathbf{L}(\mathbf{L}\mathbf{L}^T+\lambda\mathbf{I}_n)^{-1} = (\mathbf{L}\mathbf{L}^T+\lambda\mathbf{I}_n)^{-1}\mathbf{L}$. So we must have $\mathbf{w} = \mathbf{L}(\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{1}_n$. Thus the weight vector has the form $\mathbf{w} = \mathbf{L}\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{1}_n$. In practice, there is no need to explicitly recover $\mathbf{w}$. It is sufficient to solve for $\boldsymbol{\alpha}$, and the decision value for any data point $\mathbf{q}$ can be calculated:

$$f_{\mathbf{z}}(\mathbf{q}) = \mathbf{w}^T\phi(\mathbf{q}) + b = \sum_{i=1}^{n}\alpha_i(k(\mathbf{z}, \mathbf{q}) - k(\mathbf{x}_i, \mathbf{q}) - k(\mathbf{z}, \mathbf{z}) + k(\mathbf{x}_i, \mathbf{z})) + 1. \tag{2.2}$$

2.3.2 Efficient training of multiple classifiers

For a single exemplar classifier, we need to find $\boldsymbol{\alpha}$ using: $\boldsymbol{\alpha} = \mathbf{C}^{-1}\mathbf{1}_n$, where $\mathbf{C} = \mathbf{L}^T\mathbf{L}+\lambda\mathbf{I}$. Let $\mathbf{K}$ be the $n\times n$ kernel matrix with $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Let $\mathbf{k}_{\mathbf{z}} = [k(\mathbf{x}_1, \mathbf{z}), \ldots, k(\mathbf{x}_n, \mathbf{z})]^T$. We have $\mathbf{C} = \mathbf{L}^T\mathbf{L} + \lambda\mathbf{I}_n = \mathbf{K} - \mathbf{k}_{\mathbf{z}}\mathbf{1}_n^T - \mathbf{1}_n^T\mathbf{k}_{\mathbf{z}} + k(\mathbf{z}, \mathbf{z}) + \lambda\mathbf{I}_n$. Since we use an RBF kernel, $k(\mathbf{z}, \mathbf{z}) = 1$. Let $\mathbf{A} = \mathbf{K} + 1 + \lambda\mathbf{I}_n$ and $\mathbf{B} = \mathbf{A} - \mathbf{k}_{\mathbf{z}}\mathbf{1}_n^T$. We have $\mathbf{C} = \mathbf{B} - \mathbf{1}_n^T\mathbf{k}_{\mathbf{z}}$.

Using Sherman-Morrison [75] we have:

$$\boldsymbol{\alpha} = \mathbf{C}^{-1}\mathbf{1}_n = \left(\mathbf{B}^{-1} + \frac{\mathbf{B}^{-1}\mathbf{1}_n\mathbf{k}_{\mathbf{z}}^T\mathbf{B}^{-1}}{1 - \mathbf{k}_{\mathbf{z}}^T\mathbf{B}^{-1}\mathbf{1}_n}\right)\mathbf{1}_n = \frac{\mathbf{B}^{-1}\mathbf{1}_n}{1 - \mathbf{k}_{\mathbf{z}}^T\mathbf{B}^{-1}\mathbf{1}_n} = \frac{\mathbf{h}_{\mathbf{z}}}{1 - \mathbf{k}_{\mathbf{z}}^T\mathbf{h}_{\mathbf{z}}}, \tag{2.3}$$

where $\mathbf{h_z} = \mathbf{B}^{-1}\mathbf{1}_n$. Again, using Sherman-Morrison:

$$\mathbf{h_z} = \mathbf{B}^{-1}\mathbf{1}_n = \left(\mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\mathbf{k_z}\mathbf{1}_n^T\mathbf{A}^{-1}}{1 - \mathbf{1}_n^T\mathbf{A}^{-1}\mathbf{k_z}}\right)\mathbf{1}_n = \mathbf{u}^* + \frac{\mathbf{A}^{-1}\mathbf{k_z}\mathbf{1}_n^T\mathbf{u}^*}{1 - \mathbf{k_z}^T\mathbf{u}^*} = \mathbf{u}^* + \frac{\mathbf{1}_n^T\mathbf{u}^*}{1 - \mathbf{k_z}^T\mathbf{u}^*}\mathbf{u_z}, \quad (2.4)$$

where $\mathbf{u}^* = \mathbf{A}^{-1}\mathbf{1}_n, \mathbf{u_z} = \mathbf{A}^{-1}\mathbf{k_z}$. Since $\mathbf{A}$ and $\mathbf{u}^*$ do not depend on the exemplar instance $\mathbf{z}$, the above suggests an efficient method to train multiple exemplar classifiers. This method requires computing $\mathbf{A}^{-1}$ only once, so the total complexity for p exemplars is the complexity of computing $\mathbf{A}^{-1}$ and the complexity of computing $\mathbf{u_z}$ and $\mathbf{h_z}$ p times. The total complexity is $O(n^3 + pn^2)$. Here we assume the complexity for computing $\mathbf{A}^{-1}$ is $O(n^3)$ but there are algorithms with lower complexity of $O(n^{2.3727})$. In practice, for numerical stability, there is no need to solve for $\mathbf{A}^{-1}$. We can find $\mathbf{u}^*$ and $\mathbf{u_z}$ for $\mathbf{z} = \mathbf{z}_1, \ldots, \mathbf{z}_p$ using the backslash operation: $[\mathbf{u}^*, \mathbf{u}_{\mathbf{z}_1}, \ldots, \mathbf{u}_{\mathbf{z}_p}] = [\mathbf{1}_n, \mathbf{k}_{\mathbf{z}_1}, \ldots, \mathbf{k}_{\mathbf{z}_p}]\backslash\mathbf{A}$.

2.3.3 Calibrating exemplar classifiers

The exemplar classifiers are efficiently trained together, but they are still independent. To compare and combine them, we will need to calibrate their early detection scores. One approach is to use Platt scaling [163] as in [137], but it is unsuitable for our scenarios due to the imbalance between positive and negative classes and the need to prioritize high precision over recall. Here, we propose to use a non-parametric method based on precision values instead.

Consider a particular exemplar $\mathbf{z}$ with the corresponding decision function $f_{\mathbf{z}}(\cdot)$ given in (2.2), we will first obtain a function g that maps the raw output of f to the corresponding precision value, which is calculated based on the set of other positive examples other than $\mathbf{z}$ (i.e., $\mathcal{P}_{\mathbf{z}} = \mathcal{P} \setminus \{\mathbf{z}\}$) and the set of holdout negatives $\mathcal{N}^h$. In other word, $g(\theta)$ is the precision value of the detector that uses $f_{\mathbf{z}}$ as the decision function and θ the detection threshold, i.e., $g(\theta) = a/(a+b)$, where a and b are the number of positive and negative examples with decision values greater or equal to θ.

We further ensure that the calibration function is monotonic non-decreasing by defining $\hat{g}(\theta) := \sup_{\theta' \leq \theta} g(\theta')$. Hereafter, we will use $f^c_{\mathbf{z}}$ to denote the calibrated decision function for the exemplar $\mathbf{z}$, i.e., $f^c_{\mathbf{z}}(\mathbf{q}) := \hat{g}(f_{\mathbf{z}}(\mathbf{q}))$, for any query point $\mathbf{q}$. $f^c_{\mathbf{z}}(\mathbf{q})$ is essentially the estimated probability for the query point $\mathbf{q}$ to be positive, where the estimation was performed on the holdout data.

2.3.4 Combining classifiers: the EnEx detector

Given a query data $\mathbf{q}$ and the calibrated scores of the exemplar classifiers, our objective here is to calculate a value to indicate the likelihood for $\mathbf{q}$ being positive. Recall that the calibrated score of an exemplar predictor is the precision value evaluated on a holdout dataset, so the calibrated score is the direct estimate for the probability of the query point to be positive. One approach is to use mean pooling, but the estimated probability will be corrupted by unreliable prediction outputs. This is because each exemplar classifier is trained with a single positive exemplar, so its decision is expected to be reliable only for data points that are sufficiently similar to the exemplar. A more intuitive approach is to use max pooling as in [137], assigning the decision value for $\mathbf{q}$ based on the exemplar classifier that has the highest calibrated score for $\mathbf{q}$. Max pooling, however, is not robust [89] given the noisiness of an individual exemplar classifier. We instead use Orderly Weighted Averaging [89, 217, 226, 227]. This consists of two steps: (1) rank the calibrated scores of the exemplar classifiers from highest to lowest, and (2) use a learned weight vector to combine the scores.

For a query data $\mathbf{q}$, let $\mathbf{s_q}$ denote the vector of calibrated scores $\mathbf{s_q} = [f^c_{\mathbf{z}_1}(\mathbf{q}), \ldots, f^c_{\mathbf{z}_p}(\mathbf{q})]^T$. We first calculate $\mathbf{s_q}$ for $\mathbf{q}$ in $\mathcal{P}$ and $\mathcal{N}^h$. If $\mathbf{q}$ is from $\mathcal{P}$, the entry of $\mathbf{s_q}$ that comes from the exemplar classifier $\mathbf{q}$ cannot be used, because $\mathbf{q}$ was used for training the classifier. We therefore treat it as a missing value and replace it with the average value. We then sort the entries of each vector $\mathbf{s_q}$ from the highest to the lowest, and learn a classifier to separate $\{sort(\mathbf{s_q})|\mathbf{q} \in \mathcal{P}\}$ from $\{sort(\mathbf{s_q})|\mathbf{q} \in \mathcal{N}^h\}$. Following[89], we learn a weight vector $\mathbf{v}$ by optimizing the following linear program:

$$\begin{aligned} \underset{\mathbf{v},b}{\text{minimize}} & \frac{1}{|\mathcal{P}|}\sum_{\mathbf{q}\in\mathcal{P}} \max(\mu - (\mathbf{v}^T sort(\mathbf{s_q}) + b), 0) + \frac{1}{|\mathcal{N}^h|}\sum_{\mathbf{q}\in\mathcal{N}^h} \max(\mu + (\mathbf{v}^T sort(\mathbf{s_q}) + b), 0), \\ & \text{s.t. } v_1 \geq v_2 \geq \cdots \geq v_p \geq 0, \text{and } v_1 + v_2 + \cdots + v_p = 1. \end{aligned} \tag{2.5}$$

Here, μ is the margin term of the Hinge loss, and it is set to: $|\mathbb{E}_{\mathbf{q}\in\mathcal{P}}\|\mathbf{s}_q\|_1/2 - \mathbb{E}_{\mathbf{q}\in\mathcal{N}^h}\|\mathbf{s}_q\|n_1|$. The learned weight vector $\mathbf{v}$ will be then used for combining the ranked calibrated scores. We refer to this combined exemplar classifiers as the **EnEx** detector.

2.3.5 Forgetting mechanism based on clustering

The presented training method is efficient, but can still be time-consuming if thousands of exemplars are kept as more samples are observed. Thus we introduce a forgetting mechanism based on clustering. Suppose the maximum number of exemplar classifiers is p_{max}. When the number of observed exemplars exceeds this budget, each exemplar classifier $f_{\mathbf{z}}(\cdot)$ is used on all the training samples ($\mathcal{P}$ and $\mathcal{N}$) to obtain a vector of scores. And this vector is used as the representation of the exemplar $\mathbf{z}$. Then k-medoids is used to cluster the exemplars into p_{max} clusters, and only the medoids are kept as representative exemplars. This method performs clustering on the set of exemplar classifiers (i.e., their classification scores) instead of the representations of the exemplars in the feature space.

2.4 Experiments

	$\tau_1 = 2$s			$\tau_1 = 4$s		
model	@0.1	@0.3	@1.0	@0.1	@0.3	@1.0
MLP	54.5	54.1	31.1	45.8	41.0	21.3
LogReg	65.9	66.7	38.2	46.3	43.1	22.4
LSSVM	61.3	59.7	39.1	36.6	33.2	22.9
SVM	70.8	65.1	40.8	50.6	41.6	25.5
kNN (k=5)	69.6	66.8	38.1	45.0	32.8	20.4
eSVM	**80.3**	**76.3**	43.1	49.5	41.3	23.1
EnEx	79.2	**76.3**	**44.8**	**57.0**	**47.9**	**27.4**

TABLE 2.1: Results of detecting OpenDoor events with lead times $2s$ and $4s$. Each column shows the the average precision values at a particular recall threshold r, $AP@r$. Each number is actually the running average $AP@r$ (averaged over time) for a detector that is continually updated as more samples are seen. Non-parametric methods (kNN, eSVM, and EnEx) work relatively well. EnEx outperforms the other methods by a wide margin in several cases, especially when considering the AP for a lower recall threshold.

We evaluate the proposed framework on its ability to detect different types of target events early in video. Specifically, we detect if phone will shortly be picked up, if a door will shortly be opened, and if certain cooking actions are on the way of being performed. Our experiments are performed on self-recorded videos and the EpicKitchen dataset[34].

Other than specifically mentioned, the hyper-parameters of the models are kept the same across all the experiments.

2.4.1 Experiment setting

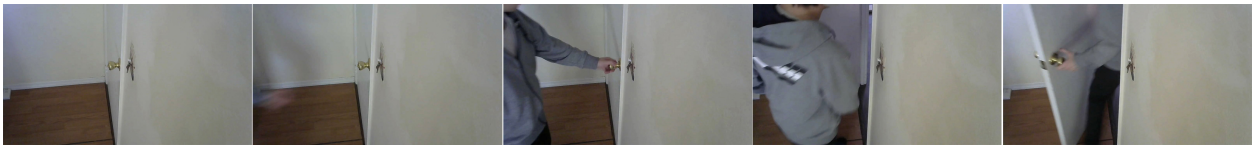
Replayed simulation from realistic data. We aim to develop a framework that keeps observing a video stream, generating delayed supervision, and improving early detection ability. This can be an everlasting process once the framework is deployed. However, for development and evaluation, we speed up the video stream by simulating the sequential arrival of events as follows. Given a particular category of target events, a set of prerecorded videos, and the set of hyper-parameters τ_1, τ_2, and l, the first step is to identify where target events occurred and extract the video segments leading up to the events within the set time range. The second step is to sample video segments which precede non-events. This creates a collection of T early video observations, each with a binary label for whether a target event eventually happens. We randomly shuffle the video segments and simulate their sequential arrival. At a certain step t, a pair of video segment and label $(\mathbf{v}_t, y_t)$ is used to update the early detector. We evaluate the performance of this detector snapshot on the future video segments $(\mathbf{v}_{t+1}, y_{t+1}), \ldots, (\mathbf{v}_T, y_T)$. Note that the detector does not gain any information from the evaluation process. For meaningful evaluation results, we will use all unseen video segments for evaluation and only simulate the process up until $t = 0.75T$, so that at least 25% of the data is used for evaluation.
Evaluation metric. Our goal is early detection of target events, and we are more interested in the moments when the events do occur than the moments when the events do not occur. In general, events of interest do not occur very often, so an unintelligent model that never makes positive detections already achieves very high accuracy. So accuracy is not a good performance measure. A better choice for data with class imbalance would be average precision. However, as noted earlier, events might occur without any prior indication, so it is unrealistic to aim for the detection of every event. Therefore, we propose to use the the average precision of the detector up until certain recall value only. Let $Pr(r)$ be the precision of the detector at recall r, the average precision until a particular recall threshold r_t ($0 < r_t \leq 1$) is calculated as

$$AP@r_t = \int_0^{r_t} Pr(r)dr/r_t. \tag{2.6}$$

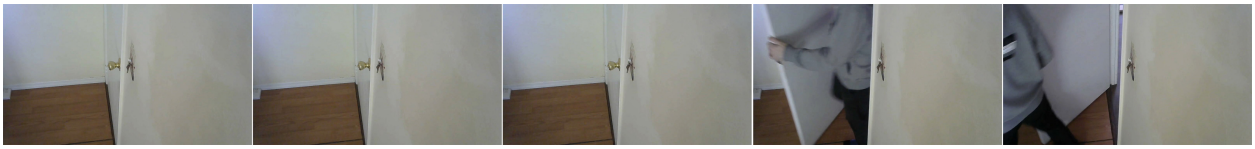
Note that $AP@1$ is the ordinary average precision. Every recall threshold r_t corresponds to a score threshold s_t for the decision function, and this average precision is equivalent to calculating the average precision only with test samples that have scores higher than s_t. For a properly trained model, lower recall threshold should result in higher average precision.

2.4.2 Early detection of OpenDoor events

We consider the task of detecting a door being opened from the visual stream of a camera. The camera is mounted inside of a room facing the door. The door can be opened from the inside or outside. If the door is opened from the outside, there will be no prior visual indication. Our goal is to detect the OpenDoor events before the door is separated from the door frame. We recorded for an extended period of time and observed 27 OpenDoor events. About half of the times the door was opened from the outside. From the remaining part of the video, we sample a negative set of video segments that did not contain an OpenDoor event. Altogether, we have a dataset of 323 video segments, among which 8.36% are positive. Figure 2.2 shows two frame sequences from the video. Figure 2.2a shows the frame sequence of the door being opened from the inside, and Figure 2.2b shows the frame sequence of the door being opened from the outside.



(A) door being opened from inside



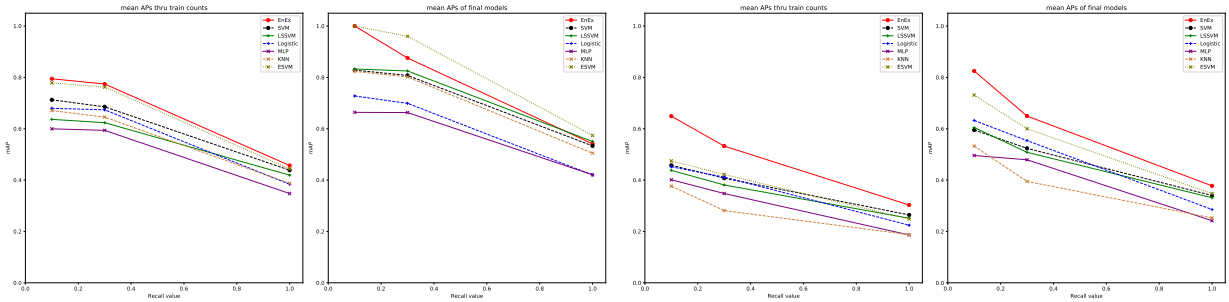
(B) door being opened from outside

FIGURE 2.2: Sample frame sequences taken from OpenDoor video. Both cases of door being opened from inside and outside are shown.

The input feature $\mathbf{v}_t$ is a 512-dim vector, which is obtained by max-pooling the ResNet-34 [81] feature vectors for all the frames in the time window $[t-\tau_2-l, t-\tau_2]$, with $l = 5$s. We experiment with different values for the lead time τ_* of 2s and 4s. The tolerance duration

τ_1 is always 2s. We compare with four parametric and two non parametric models: multiple layer perceptron (MLP), logistic regression (LogReg), least-squared SVM (LSSVM) [191], SVM [203], k nearest neighbors (kNN with $k = 5$), and exemplar SVM (eSVM) [137]. We use an RBF kernel for SVM, LSSVM, and eSVM. The results averaged over 20 experiment runs, are shown in Table 2.1. Non-parametric methods work better than parametric ones in this experiment, and EnEx yields the best or close to the best result for different measurements and settings.

We also report the final scores, which is the AP scores at the end of the training, when the number of training samples is at maximum. Figure 2.3 shows both the final scores and average scores of different models under different recall thresholds as precision-recall curves. For this metric, EnEx remains the best or close to the best model compared to other methods.



(A) $\tau_* = 2$s (B) $\tau_* = 4$s

FIGURE 2.3: Average and final performance of different models under different recall thresholds. Experiments are run 10 times for predicting OpenDoor with lead times $\tau_* = 2$s and $\tau_* = 4$s.

2.4.3 Effect of EnEx hyper-parameters

We did some exploration on the hyper-parameters of the described EnEx model. Those are the coefficient γ used in the RBF kernel, and the regularization strength λ. In experiments we set $\gamma = k_\gamma/\bar{d}$, where $\bar{d}$ is the averaged Euclidean distance between the training samples; and $\lambda = k_\lambda(n_- + 1)$, where n_- is the number of negative training samples. We try different combinations of k_γ and k_λ, and run each combinations to detect OpenDoor actions early with lead time $\tau_* = 2$s for 20 times. The average $AP@0.1$ and $AP@1$ are reported in Table 2.2. Note that in all other experiments, we have set $k_\gamma = 1$ and $k_\lambda = 10^{-3}$.

		k_γ				
		10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
k_λ	10^{-4}	79.9	77.3	78.9	85.3	55.7
	10^{-3}	74.1	76.0	79.2	75.9	57.9
	10^{-2}	63.9	77.4	81.9	81.4	66.3

(A) Average $AP@0.1$

		k_γ				
		10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
k_λ	10^{-4}	45.7	45.6	44.4	48.4	35.8
	10^{-3}	41.3	45.6	44.8	43.6	36.3
	10^{-2}	35.6	42.9	44.4	47.5	39.1

(B) Average $AP@1$

TABLE 2.2: Results of different hyper-parameters combinations in EnEx model on early detection of OpenDoor actions with lead time of 2s. The combination used in all other experiments is highlighted.

2.4.4 Early detection of PickupPhone human actions

We consider the task of early detecting if a cell phone will be picked up. A camera is mounted on top of a shelf pointing at a desk on which the phone is placed. The phone will be picked up when a notification pops up on the phone's screen, but it can also be picked up without any screen notification. The field of view is limited, and the phone owner is not visible most of the time, only his hand appears when the phone is being picked up. We recorded for an extended period of time to capture multiple instances of the phone pickup action. Altogether there are 105 pickup actions, roughly half of them occur without a notification on the phone's screen. From the remaining part of the video, we sample a negative set of video segments that did not contain the pickup action. Altogether, we have a dataset of 446 video segments, among which 22.53% are positive. In Figure 2.4 two frame sequences from the PickupPhone video are shown. Figure 2.4a shows the frame sequence of the phone being picked up after a notification pops up on the screen, and Figure 2.4b shows the frame sequence of the person picking up the phone without any notification.

The 512-dim input feature $\mathbf{v}_t$ is obtained by max-pooling the ResNet-34 feature vectors for all the frames in $l = 2$s window. The tolerance duration τ_1 is always set to 2s. We compare with several types of parametric and non-parametric classifiers, as described in subsection 2.4.2. The results for $\tau_* = 0.5s$, averaged over 10 experiment runs, are shown in Figure 2.5a. EnEx outperforms other methods by a wide margin. The gap is particularly large when when there are only 20 positive examples of the target event. The other exemplar-based classifier eSVM does not perform as well in this case, possibly due to the use of different ways for calibrating and combining the exemplar classifiers.

Additionally, we set the lead time to $\tau_* = 0.25s$ and $\tau_* = 2s$. The results are shown in

(A) notification pops up on screen

(B) no notification on screen

FIGURE 2.4: Sample frame sequences taken from PickupPhone video.

Figure 2.5b and Figure 2.5c. For shorter lead time, the detection task becomes easier, and all models achieve higher AP. But EnEx stills outperforms other methods. When $\tau_* = 2$, the early detection becomes essentially impossible, and all models fail to make correct detections.

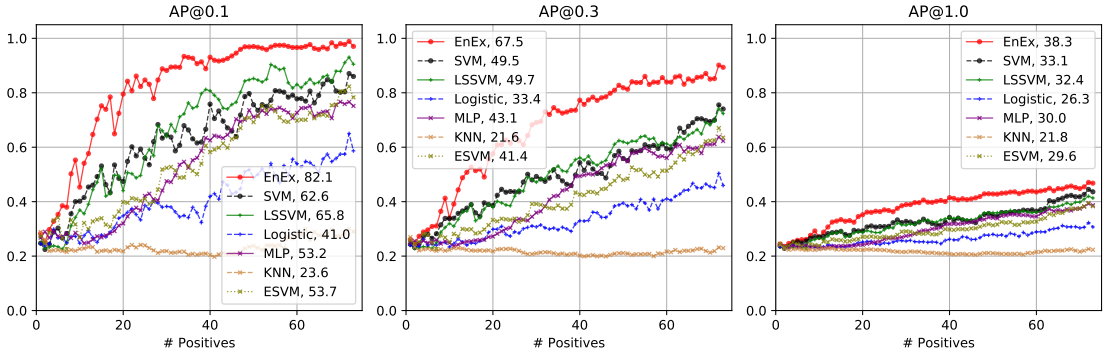
We run the experiments on PickupPhone events with $\tau_* = 0.25$s for $n = 100$ times. We calculate the Standard Errors (SE) of the results across different runs, which is defined as

$$SE = \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} (x_i - \bar{x})^2}, \tag{2.7}$$

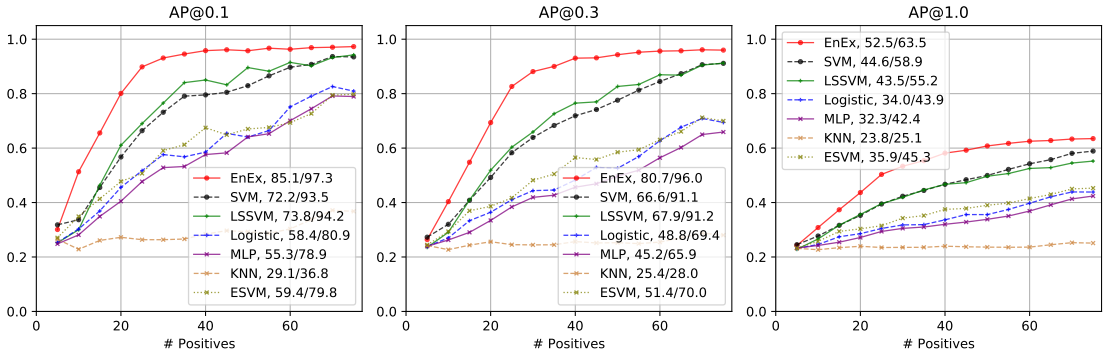
where $\bar{x}$ is the average result of all experiment runs. The results are shown in Figure 2.6. The proposed method is very stable, and the variance across different training runs is insignificant.

2.4.5 Illustration of EnEx output scores

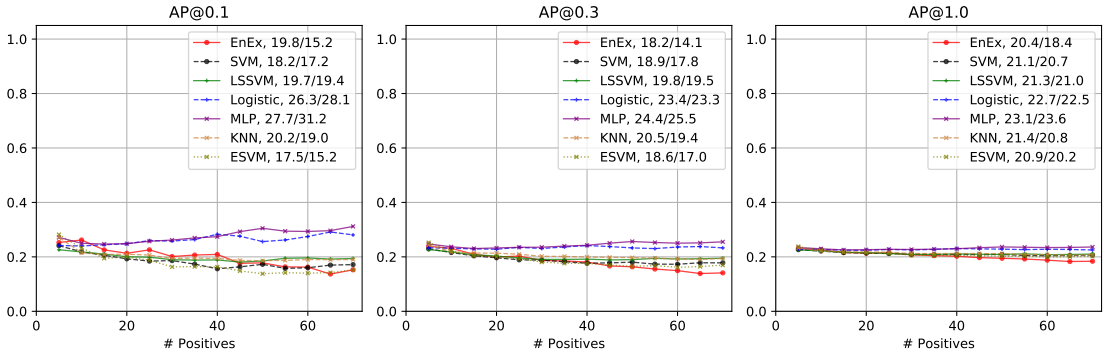
We visualize how the detection scores produced by EnEx model changes over time when new frames in the video is observed on PickupPhone actions with lead time $\tau_* = 0.5$, and OpenDoor events with lead time $\tau_* = 2s$. The trained model is applied to the unseen samples for visualization. Sliding window is applied to obtain how the prediction scores change continuously when every new frame is observed. Figure 2.7a and Figure 2.7b show the scores when the phone gets picked up after the person sees a notification on the screen. It is clear that the detection score increases immediately when the notification pops up, indicating that after training, EnEx can learn to detect with high confidence after



(A) Lead time $\tau_* = 0.5$ second, average of 10 runs



(B) Lead time $\tau_* = 0.25$ second, average of 40 runs



(C) Lead time $\tau_* = 2$ second, average of 40 runs

FIGURE 2.5: Performance of different classifiers for early detection of PickupPhone actions with different lead times. Each subplot shows the different AP@r. Each curve shows how the performance of a classifier changes as it is trained on more seen target events. The number next to each model in the legend box takes the form of average score/final score.

seeing the precursory clue that will surely lead to the PickupPhone action. Figure 2.7c and Figure 2.7d show the phone gets picked up without any notification is seen. This time the

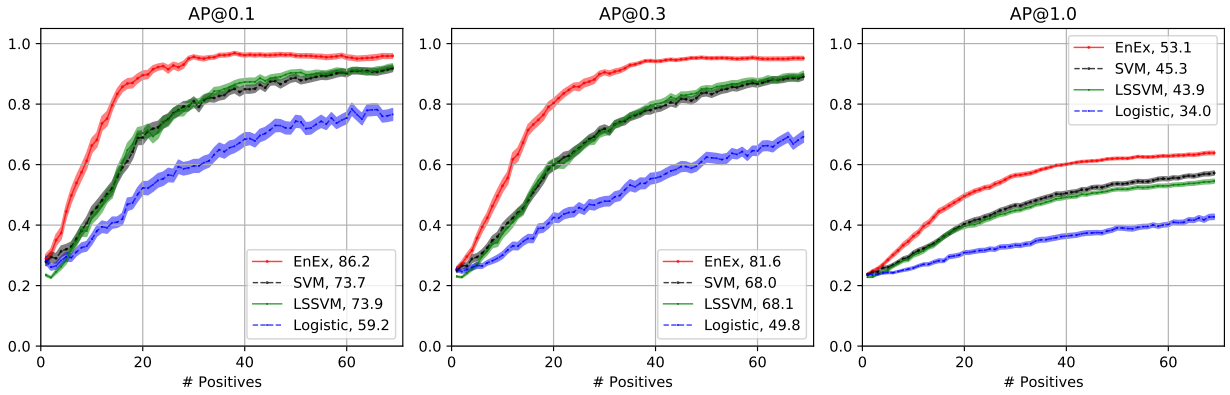
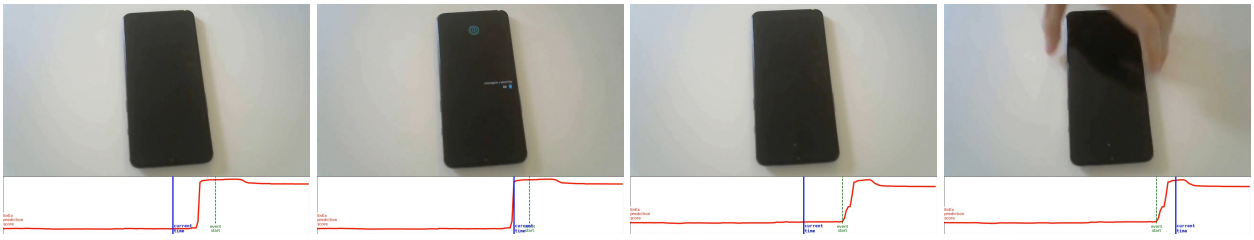


FIGURE 2.6: Standard error of results over different experiment runs at every training iterations on PickupPhone events with $\tau_* = 0.25$s. Standard error is plotted using filled error bars.

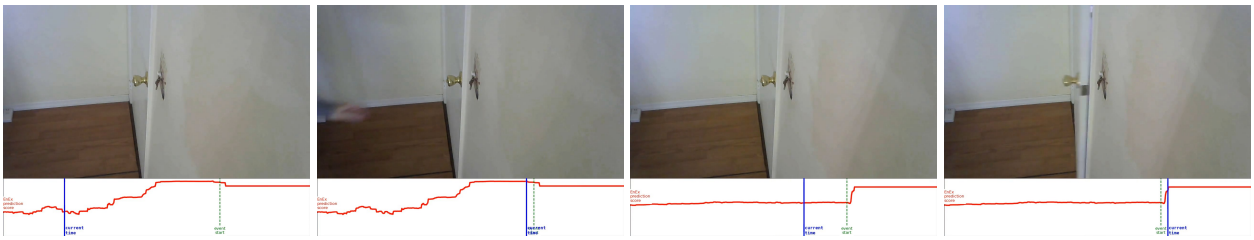
precursor is missing, and the score stays low until the action actually starts. This show that after training, EnEx can distinguish events with and without precursors. By doing this, it can make early detections with high precision.



(A) notification has not popped up (B) notification is now visible (C) the phone is not picked up (D) the phone is being picked up

FIGURE 2.7: Visualization of how EnEx detection score changes with input videoo of PickupPhone. Sliding window is applied to obtain how the prediction scores change continuously when every new frame is observed.

Figure 2.8a and Figure 2.8b show the event of the door being opened from inside. Compared to PickupPhone, the precursory clues are more subtle to observe. When the person approaches the door, the lighting and shadow on the wall changes. The trained EnEx model can pick such clues, and produces higher scores. Figure 2.8c and Figure 2.8d show the event of the door being opened from outside. The score stays low until the event actually starts. Similar to the illustration for PickupPhone action, this shows that EnEx can make early detection with high precision.



(A) person has not approached the door (B) person is walking towards to door (C) the door is staying closed (D) the phone is being opened

FIGURE 2.8: Visualization of how EnEx detection score changes with input videoo of OpenDoor. Sliding window is applied to obtain how the prediction scores change continuously when every new frame is observed.

2.4.6 Early detection of human actions in kitchens

FIGURE 2.9: Sample frame sequences taken from EpicKitchen video P01_01.MP4.

We consider the task of early detection of actions of a person in a kitchen using an egocentric camera mounted on the head. For this experiment, we use the publicly available EpicKitchen dataset [34]. Videos from this dataset are densely annotated with *verb-noun* action labels (e.g. *open-drawer*). Figure 2.9 shows a frame sequences taken from the EpicKitchen video `P01_01.MP4` around time `00:05:01`, when an *open-tap* action is being performed. We do experiments on the most 20 popular action categories, listed in Table 2.3. The data is sampled similarly as in previous experiments. We use the I3D [21] network trained on the Kinetics-400 [104] to extract action feature at each second of the video. $\mathbf{v}_t$ is the temporally max-pooled I3D feature vectors of the $l = 5$s window. We set $\tau_* = 1$s, and $\tau_1 = 2$s. Note that different action instances have different length, so after sampling, the most popular action might not be the one that has the highest positive rate in the training sample set.

Table 2.3 reports the running average $AP@0.1$ and $AP@1.0$ scores of different methods, averaged over 20 experiment runs. The results show that the proposed method EnEx can make early detections with higher precision, especially at low recall threshold. For the AP at the low recall threshold $AP@0.1$, EnEx outperforms other methods on 16 out

	positive		AP@0.1					AP@1.0				
action	%	#	**EnEx**	eSVM	SVM	LSSVM	LogReg	**EnEx**	eSVM	SVM	LSSVM	LogReg
open-cupboard	11.6	580	**30.2**	20.9	28.3	27.7	25.1	20.0	13.2	18.3	**20.4**	18.8
open-drawer	10.6	441	**32.6**	24.3	31.7	27.5	25.0	**20.4**	13.0	18.4	20.2	18.5
put-plate	8.8	377	**83.2**	32.9	79.3	76.4	51.2	31.5	15.5	33.3	**33.4**	28.6
take-plate	8.3	372	**27.9**	18.1	27.8	24.3	21.6	**16.6**	10.2	14.8	16.4	15.0
close-cupboard	9.3	362	53.0	32.5	**53.4**	46.6	50.1	27.8	14.4	27.3	27.7	**28.6**
take-spoon	7.6	345	**20.4**	14.1	14.3	15.9	13.7	**11.7**	8.9	10.2	11.4	10.8
open-fridge	7.8	344	**28.3**	13.9	19.6	21.5	20.7	**14.1**	9.2	12.6	13.8	13.4
open-tap	14.1	333	**64.4**	22.2	55.6	50.6	37.8	**30.8**	16.8	28.5	28.5	25.2
wash-hand	7.1	330	36.1	7.8	35.7	**37.0**	30.4	17.1	7.2	15.5	**17.2**	15.2
put-spoon	8.4	317	52.3	25.3	**52.8**	48.0	32.8	**21.5**	12.4	20.6	21.1	17.9
take-knife	8.1	306	**22.9**	17.5	19.2	19.3	15.3	**12.3**	9.9	11.1	12.0	11.1
put-knife	7.4	287	**49.2**	15.9	45.2	45.9	31.0	19.7	9.5	17.8	**20.3**	16.7
close-fridge	7.1	280	**68.5**	27.2	67.5	62.9	53.7	31.5	11.8	**33.1**	32.6	31.1
close-tap	10.3	267	60.1	22.3	**67.6**	59.9	48.4	33.0	14.0	**34.1**	33.6	29.8
put-pan	7.5	256	42.9	28.4	**43.0**	33.1	40.0	18.6	11.9	18.0	19.0	**20.0**
close-drawer	8.0	249	**59.9**	23.6	51.6	53.2	33.0	**25.8**	11.7	23.4	**25.8**	21.5
turnoff-tap	7.4	227	**80.5**	28.7	69.6	62.4	44.5	**31.1**	12.7	29.4	28.8	24.8
wash-plate	11.0	218	**74.6**	54.1	70.9	66.6	56.9	**38.6**	23.6	**38.6**	37.9	33.9
put-bowl	8.5	217	**47.1**	37.9	42.9	41.3	36.4	**21.2**	15.0	19.7	20.3	19.2
turnon-tap	7.4	214	**45.5**	15.5	26.3	31.9	25.0	**15.4**	9.4	12.9	14.5	13.2
average	8.8	316	**49.0**	24.2	45.1	42.6	34.6	**22.9**	12.5	21.9	22.7	20.7

TABLE 2.3: Results on top-20 EpicKitchen actions. % and # are the percentage and number of positive events from the data stream, respectively. Action classes are sorted by #. This table reports the running average $AP@r$ for several types of models.

of 20 action classes, by a wide margin in many cases. For the remaining 4 action classes, the performance gap between the best method and EnEx is smaller than 1%, except for *close-tap*, where SVM obtains much higher $AP@0.1$ than EnEx. Results are more varied for $AP@1.0$. Out of the 20 actions, EnEx is the best in 12 cases (with 2 ties). For actions that EnEx is not the best, it is also not far behind from the best. On the other hand, the other methods do not perform as consistently and robustly. SVM outperforms EnEx for some actions, but on other actions (e.g. *open-fridge*, *turnoff-tap*, *turnon-tap*), its precision is much lower than EnEx. LSSVM and Logistic Regression also suffer from this problem. eSVM does not performance well on most the actions, probably due to its lack of dedicated calibration and combination methods.

2.4.7 Comparison with end-to-end methods

We compare the performance of EnEx against deep network models for early detection of PickupPhone actions with $\tau_* = 0.5s$, OpenDoor events with $\tau_* = 2s$, and averaged top-5 EpicKitchen actions with $\tau_* = 1s$. The compared models are: i) a pretrained ResNet-34 [81] followed by temporal max-pooling and fully-connected classifier (similar to the feature extractor used in subsection 2.4.2 and subsection 2.4.4) and ii) ruLSTM [61], the state-of-the-art model for early action recognition. The whole networks of the deep models are trained end-to-end using cross-entropy loss. The same data shuffling and evaluation methods in subsection 2.4.1 are used. Similarly, each model is trained and evaluated for 10 runs with different random data shuffling, and the average performance is reported in Table 2.4. Despite being much heavier in computation, deep models still fall behind EnEx in most cases, especially for lower recall thresholds and when very limited positive samples are observed. We also observe that the performance of neural models has higher variance across different runs, implying unstable optimization.

	OpenDoor 2s			PickupPhone 0.5s			Epic 1s	
model	@0.1	@0.3	@1.0	@0.1	@0.3	@1.0	@0.1	@1.0
ResNet	57.7	52.9	31.5	68.1	63.1	**38.4**	41.9	22.5
ruLSTM	61.9	51.3	26.9	76.9	52.4	33.2	40.5	**29.2**
EnEx	**79.2**	**76.3**	**44.8**	**82.1**	**67.5**	38.3	**45.4**	23.3

TABLE 2.4: Comparison of EnEx with end-to-end models. Each column in the table shows $AP@r$. Each number is the average of 10 experiment runs. EnEx outperforms ResNet and ruLSTM in most cases, especially for OpenDoor events where the positive samples are very sparse.

2.4.8 Effect of calibration and combination methods

We perform some ablation experiments to show the effectiveness of the proposed calibration and combination methods. We use the same setting of early detection of PickupPhone actions with $\tau_* = 0.5s$. For calibration, we can use the proposed non-parametric method or Platt scaling as in eSVM [137]. For combination, we can either use the the proposed rank pooling or max pooling of eSVM. The results are shown in Table 2.5. The numbers are the running average of $AP@0.1$. EnEx works well with both calibration

methods, while eSVM must be used with logistic calibration. Max pooling works much better if logistic calibration is used, but it is not as robust as the proposed combination method.

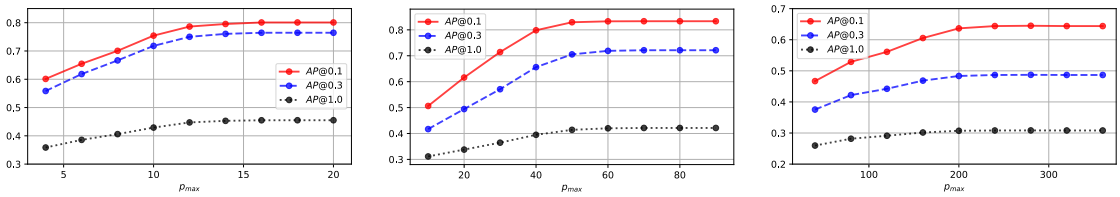
options		options used					
calibration	logistic		✓	✓		✓	
	ours	✓			✓		✓
combination	max	✓	✓	✓	✓		
	ours					✓	✓
classification	eSVM	✓	✓				
	EnEx			✓	✓	✓	✓
$AP@0.1$		36.3	53.2	66.3	36.5	76.9	**82.1**

TABLE 2.5: Ablation study with different methods of calibration and combination. The numbers are the running average of $AP@0.1$.

The proposed non-parametric calibration runs slightly faster than logistic calibration. EnEx training is also faster than eSVM. On a small dataset of 70 positive and 150 negative examples, the training (including calibration) time of EnEx and eSVM are 240ms and 540ms, respectively. On a bigger dataset with 200 positive and 860 negative examples, the training duration are 7.2s and 27.4s respectively. The inference time for one query are 0.4ms and 12ms respectively. All times are measured on 2.6 GHz Quad-Core Intel Core i7 CPU.

2.4.9 Effect of clustering-based forgetting

We use the same setting of early detection of PickupPhone actions with $\tau_* = 0.5s$, OpenDoor events with $\tau_* = 2s$, and *open-tap* actions with $\tau_* = 1s$. In each run, all the EnEx models share the same split of observed samples, but have different exemplar budgets p_{max}. The largest budget is equivalent to unlimited. We run the experiments for 20 times, and report $AP@r$ in Figure 2.10. The results show that using about half of the observed exemplars can reduce the calculation cost, but still achieve similar detection performance. When the number of exemplars increases to the point that computation becomes time-consuming, forgetting mechanism can be applied to trade-off between speed and precision.



(A) OpenDoor (B) PickupPhone (C) *open-tap*

FIGURE 2.10: Performance of different exemplar budgets. Data points are averaged over 20 runs.

2.5 Summary

We have described a non-parametric method for early event detection. This method has shown the possibility of improving the timeliness of event detection using noisy self-supervised signal from a video stream. A full event detector continuously observes a video stream and provides the delayed supervision for training the early event detector. The early event detector is based on an ensemble of exemplar classifiers, each of which is obtained by training a classifier to separate a single pre-positive video segment from many pre-negatives. We have developed an efficient formulation for training multiple exemplar classifiers together, and we have also proposed a novel method for calibrating and combining multiple weak exemplar classifiers to create a stronger ensemble classifier. The ensemble classifier accounts for the non-separable events, and it does not suffer from the drawbacks of many parametric classifiers. We have evaluated our method on its ability to detect visual evens (PickupPhone, OpenDoor, and cooking actions) early, and showed that our method outperformed the others in many cases.

Chapter 3

Object Detection with Self-Supervised Scene Adaptation

High-level computer vision tasks, such as action and event detection (as described in chapter 2), visual reasoning [97], and video captioning [252], rely on lower-level features to operate. For instance, [61] and [252] use a out-of-shelf object detection model to produce object level features that act as the input of the proposed models for high-level tasks. And such object features are proven to be beneficial. However, such trained object detectors are not guaranteed to perform well on unseen videos, as the data distribution can differ greatly from the dataset where those detectors have been trained on. This calls for the ability of an object detector to improve its detection performance by observing a video stream for a period of time.

This chapter presents a novel method to improve the performance of a trained object detector on scenes with fixed cameras based on self-supervised adaptation. Given a specific scene, the trained detector is adapted using pseudo-ground truth labels generated by the detector itself and an object tracker in a cross-teaching manner. When the camera perspective is fixed, the presented method can utilize the background equivariance by proposing artifact-free object mixup as a means of data augmentation, and utilize accurate background extraction as an additional input modality. I also introduce a large-scale and diverse dataset for the development and evaluation of scene-adaptive object detection. Experiments on this dataset show that the presented method can improve the average precision of the original detector, outperforming the previous state-of-the-art self-supervised domain adaptive object detection methods by a large margin. The content of this chapter is based on the published paper [242], and "we" in this chapter refers to the authors of this paper. The dataset and code are available at `https://github.com/cvlab-stonybrook/scenes100`.

3.1 Introduction

Object detection in video streams from stationary cameras is needed for many computer vision applications, including video surveillance and autonomous retail. In general, different applications require the detection of different object categories, and each computer vision based product will have its own detector. However, for a specific product, there is typically a *single* detector that will be used for *many* cameras/scenes. For example, a typical video surveillance product would use the same detector to detect pedestrians and vehicles for network cameras installed at different locations. Unfortunately, a single detector might not work well for all scenes, leading to trivial and unforgiving mistakes.

This fundamental problem of many computer vision tasks stems from the limited generalization power of a single model, due to insufficient training data, lack of model capacity, or both. One can attempt to address this problem by using more training data, but it will incur additional cost for data collection and annotation. Furthermore, in many cases, due to the low latency requirement or the limited computing resources for inference, a product is forced to use a very lightweight network, and this network will have limited representation capacity to generalize across many scenes.

In this chapter, instead of having a single scene-generic detector, we propose using scene-specific detectors. This yields higher detection performance as each detector is customized for a specific scene, and allows us to use a lightweight model without sacrificing accuracy as each detector is only responsible for one scene.

However, obtaining scene-specific detectors is very challenging. A trivial approach is to train a detector for each scene separately, but this requires an enormous amount of annotated training data. Instead, we propose a self-supervised method to adapt a pre-trained detector to each scene. Our method records the unlabeled video frames in the past, uses the trained detector to detect objects in those frames, and generates augmented training data based on those detections. Although the detections made by the pre-trained model can be noisy, they can still be useful for generating pseudo annotated data. We further extend those pseudo bounding boxes by applying object tracking [8, 176] along the video timeline, aiming to propagate the detections to adjacent frames to recover some of the false negatives not returned by the detector. We also use multiple detectors to obtain the pseudo labels and train the detector in a cross-teaching manner, taking the advantage of the ensemble of models [57, 87].

Exploiting the stationary nature of the camera, we use two additional techniques to

boost the detection performance: location-aware mixup and background-augmented input. The former is to generate more samples during training through object mixup [239] that contains less artifacts, based on the aforementioned pseudo boxes generated from detection and tracking. The latter involves estimating the background image and fusing it with the detector's input.

In short, we present in this chapter is a novel framework that utilizes self-supervision, location-aware object mixup, and background modeling to improve the detection performance of a pre-trained object detector on scenes with stationary cameras. We also contribute a large scale and diverse dataset for the development of scene adaptive object detection, which contains sufficient quantity and quality annotations for evaluation.

3.2 Related work

Despite much recent improvement in object detection research [11, 17, 18, 31, 44, 73, 74, 83, 90, 102, 113, 122, 127, 131, 133, 152, 153, 154, 156, 169, 170, 171, 172, 197, 204, 213, 214, 215, 241, 253], trained detectors still encounter problems caused by the change in data distribution referred as domain shift or domain gap. Scene adaptive object detection can be viewed as a special case of domain adaption, in which an object detector trained on a fully-supervised source domain is adapted to a target domain. Most research in this direction focuses on semi-supervised, weakly-supervised, or self-supervised adaptation. In addition, source-free adaptation [95, 119, 121, 223] has been proposed to address the situation where the source domain data is unavailable during adaptation.

Self-labeling methods [38, 98, 105, 119, 123, 132, 149, 176, 185, 228] uses the teacher-student setup from semi-supervised image classification [110, 193]. A teacher detector trained on the source domain generates pseudo bounding boxes on the target domain images, and a student model is trained with those boxes to improve its performance on the target domain. Techniques such as weak/strong augmentation [123, 132, 185], knowledge distillation [38], and weight averaging [119, 123] have been used to deal with noisy pseudo labels from the teacher detector due to the domain gap. Our proposed pseudo-labeling method uses two teacher models trained on the source domain to train a student model, and their pseudo boxes are aggregated and refined. We further apply tracking as in [176], but with both forward and backward directions, to extend the pseudo labels. The pseudo boxes also form the basis for later location-aware mixup and dynamic background extraction steps.

Domain alignment methods [28, 38, 84, 119, 120, 178, 224, 246] aim to reduce the domain gap by enforcing the models to output similarly on the source and target domain at image, proposal, or instance levels. They use domain adversarial learning through gradient reversal [64, 66] or graph matching [120] for domain alignment. This approach involves adding domain alignment losses to existing object detection models and is complementary to our method.

Another seemingly related research area is continual or incremental learning. Scene adaptive and domain adaptive object detection are in between task-continual learning [124] and data-continual learning [161], for that the object categories of the source and target domains are the same, but the data distribution differs. It is possible to formulate scene adaptive object detection as an online learning problem, but this approach would require additional annotations.

3.3 Methods

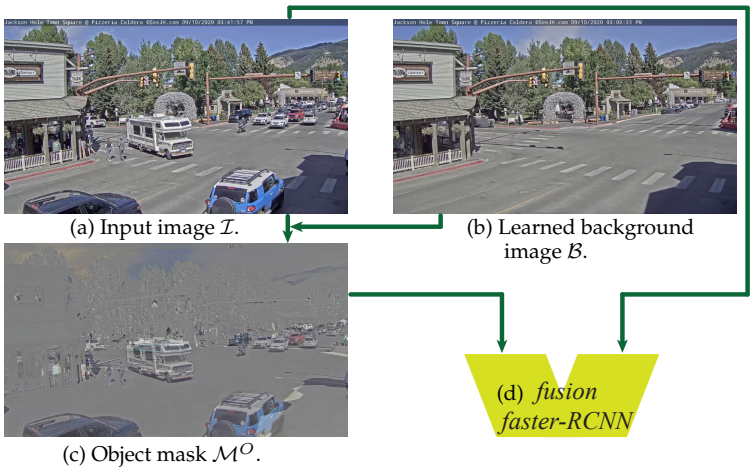


FIGURE 3.1: Inference flow of the proposed detector that can adapt to a specific scene. The background image and fusion faster-RCNN model are learned during training, discussed in section 3.3.

For an object detector pre-trained on a source dataset referred as the base model, we aim to improve its performance on video stream from a stationary camera in a given scene, with unlabeled videos captured by the same camera in the past as adaptation training samples. The categories of object of interest in the scene are also present in the source

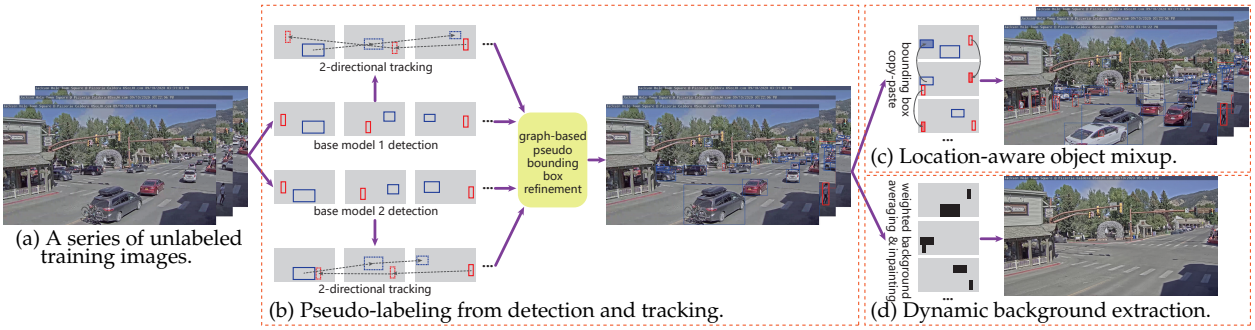


FIGURE 3.2: Training flow of the proposed self-supervised scene adaptation object detection framework with sample images. The mixup training images with pseudo bounding boxes and background image are used in the fusion training. The details are discussed in section 3.3.

dataset, but the data distribution is shifted. The base model is adapted to each scene independently.

Assuming the distribution shift is moderate, the base detector can generate partially decent bounding boxes on the unlabeled frames. We use them as pseudo-labels to improve the base model itself. Many objects are ignored by the base detector, as their appearance can be very different from the source dataset. However, videos contain temporal correlation information, making it possible to apply object tracking in both time directions to recover some of those false negatives. The pseudo-labels are then refined to remove low-confidence boxes and duplicates. We refer to this process of obtaining pseudo bounding boxes as pseudo-labeling.

As data augmentation is beneficial for training on less reliable labels [157], we take advantages of object mixup to generate new sample images. Since the background is stationary, those images contain fewer artifacts, which is beneficial for detectors. We further adopt the idea of ensemble models [57, 87] by using two base detectors to obtain the pseudo labels, for they can be complementary to each other. The pseudo boxes are used to train one base model, which we refer as cross-teaching.

Since the job of a detector is to separate foreground objects from the background, having information about the background is advantageous. In a stationary camera video with moving objects, different parts of the background are covered by objects in different frames, making it possible to model the complete background. We use the aforementioned pseudo bounding boxes to mark the uncovered parts of the background in each frame. Combining the partial background from a sequence of frames can give an accurate and complete background model. We then modify the Faster-RCNN [172] architecture to

fuse the background with the input image to improve detection performance.

The overall structure and data flows of the proposed methods are shown in Figure 3.1 and Figure 3.2. Our proposed methods are highly flexible and can be configured to trade-off between performance and speed. Object tracking can be bypassed to reduce the time to get the pseudo-bounding boxes. Mixup and object mask fusion can also be disabled for faster training and inference. In subsection 3.5.7 we will show ablation study results on how those components affect the performance of the adapted models. We now describe each components in more details. For consistency, the images are all from the first video in our collected dataset, which will be described in section 3.4.

3.3.1 Base detection models on source domain

We use Faster-RCNN [172] in our experiments for its high precision and flexible modular architecture. Among the techniques proposed for self-supervised scene adaptation, both pseudo-labeling and mixup can be directly applied to any object detection model. The object mask fusion technique requires modification to the network architecture, but it can be easily extended to most types of detectors by adding a parallel branch to the existing network.

The object labels in MSCOCO training set are remapped to *person* and *vehicle* as described in subsection 3.4.2. All objects other than *person*, *car*, *bus*, and *truck* are discarded. We take Faster-RCNNs pre-trained on the original MSCOCO training set, and finetune them on the remapped training set. We choose two models with ResNet-50 and ResNet-101 backbones as base models, referred as **M1** and **M2**.

3.3.2 Pseudo labels from detection and tracking

We take the frames from the first training portion of each video, and adopt the idea of detection and tracking similar to [176], then apply our proposed bounding box refinement and cross-teaching schemes. Base detectors described in subsection 3.3.1 are applied on the frames, and any bounding box with confidence score higher than λ_{det} is kept as a valid pseudo box.

Single-object trackers are initialized with detected objects having confidence scores higher than λ_{sot}. We use DiMP-50 [8] to track the objects in both forward and backward directions in the video timeline, and the tracked object bounding boxes are used as pseudo boxes. The reason for tracking in both directions is that if an object is moving towards

the camera and appears bigger by time, backward tracking is more accurate because the initial bounding box contains more detailed texture. To prevent drifting, the maximum length of each track is capped at two seconds.

Bounding box refinement is applied to eliminate duplicated boxes of the same object from combining pseudo bounding boxes from detector and tracker. Each candidate box in the same frame is regarded as a graph node. If two boxes have the same object label, and their IoU is above a threshold λ_{iou}, then an edge is added to connect them. We assume that each connected component of the graph represents one single object, and the nodes in the component are duplicated. We only keep the node with the highest degree (*i.e.*, the node with the highest number of connected edges). An example of refinement results is shown in Figure 3.2b. Refinement can effectively remove duplicated bounding boxes. However, it can neither remove false positives nor recover false negatives.

Both M1 and M2 are used in detection and tracking. The pseudo bounding boxes from both base models are then refined jointly, and the refined boxes are used to train M2 in adaptation. We refer this setting as cross-teaching.

3.3.3 Location-aware object mixup

Mixup [239] is originally proposed as a simple but powerful data augmentation method in for image classification. It is quickly improved by many modifications [6, 85, 116, 205, 232]. Mixup is also used in object detection tasks [13, 232], where it can be applied at image or instance levels. During adaptation training, a frame with its pseudo bounding boxes has the probability p_{mixup} of being paired with another randomly select source frame for mixup. A portion r_{mixup} of the pseudo bounding boxes in the source frame are cropped out and pasted onto the original frame, along with their locations and labels. If a pasted box covers more than α_{cover} of a pseudo box in the original frame, the pseudo box will be excluded from the pseudo labels. An example of this location-aware object mixup is shown in Figure 3.2c and Figure 3.3. Since the background is stationary, we propose to use location-aware mixup instead of the original random mixup to reduce the amount of artifacts in the generated mixup images and to keep the location distribution of the objects in the scene. This will be shown to be beneficial in the experiments.

(A) Random mixup. (B) Location-aware mixup.

FIGURE 3.3: An example of location-aware object mixup. Compared to random mixup, less artifacts are introduced.

3.3.4 Dynamic background extraction

For a video frame $\mathcal{I}$ of dimension $W \times H$ associated with a set of K pseudo-annotated object bounding boxes $\{(x_k^1, y_k^1, x_k^2, y_k^2) | k = 1 \dots K\}$, a background mask $\mathcal{M}$ of the same dimension as $\mathcal{I}$ can be constructed as follows. For each location (x, y) in $\mathcal{M}$, we set $M[x, y] = 0$ if (x, y) is inside of any pseudo-annotated bounding box and 1 otherwise. Then for a sequence of frame-mask pairs $\{(\mathcal{I}_l, \mathcal{M}_l) | l = 1 \dots L\}$, the background image is determined as

$$\mathcal{B} = \frac{\sum_{l=1}^{L} \mathcal{I}_l \otimes \mathcal{M}_l}{\sum_{l=1}^{L} \mathcal{M}_l}, \tag{3.1}$$

where $\otimes$ is the pixel-wise multiplication operator.

There might exist a location (x', y') that lies inside an object bounding box in every image, *i.e.*, $\mathcal{M}_l[x', y'] = 0$. In this case, the background at this location is never observed and its pixel value cannot be determined. In this case, we "guess" the background value by inpainting inside such areas using the inpainting algorithm of [194].

Figure 3.2d shows an example background, which is reasonably accurate. Although in each video, the camera perspective is fixed, the background can change overtime due to some factors such as illumination change. To incorporate this, the background extraction is operated every T_{bg} seconds, giving a dynamic background modeling.

We construct an object mask as an additional input modality for the detector, to utilize the extracted background to improve detection performance during adaptation. For a frame $\mathcal{I}$ and its corresponding background model image $\mathcal{B}$, we define its object mask

image as

$$\mathcal{M}^O = (\mathcal{I} - \mathcal{B} + 1) \times 0.5, \tag{3.2}$$

where we assume the pixel values in both $\mathcal{I}$ and $\mathcal{B}$ have been normalized to the range $[0, 1]$. An object mask example is shown in Figure 3.1c, in which objects are clearly separated from the background. Note that since $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{M}^O$ are linearly dependent, no information is lost.

Before the adaptation training process, the base detector needs to be modified to use object mask input modality and trained on the MSCOCO training set with background images. However, dynamic background extraction on MSCOCO is impossible since it only consists of static images. Therefore, we rely on the object mask annotations in the dataset and apply the same inpainting algorithm [194] to generate the background image for each training image.

3.3.5 Object mask fusion

The architecture of Faster-RCNN [172] is depicted in Figure 3.4a. First, a CNN backbone with a feature pyramid network (FPN) [126] is utilized to extract the Feature Pyramid (FP). The first layer in the CNN that takes the input image is also shown here. Then, an RPN [126] is employed to produce object bounding box proposals. Lastly, an ROI head [74, 80] is used to assign object labels and refine the bounding boxes on the pooled feature maps of the proposals. The entire network is trained utilizing two sets of losses: localization loss and objectness loss from RPN, and localization loss and classification loss from the ROI head. The fusion models explained below start with the base model that is trained on the remapped MSCOCO training set.

Early-fusion. The input to the Faster-RCNN network is the stacked image $[\mathcal{I}; \mathcal{M}^O]$, with the two modalities stacked along the color channels, as shown in Figure 3.4b. To enable the network to take a 6-channel image as input instead of the vanilla 3-channel image, the first layer of the backbone CNN is duplicated and stacked. The convolution kernel weights are first copied, then halved. This ensures that the feature map after the first layer conv1|conv2 is the same as in the vanilla model when $[\mathcal{I}; \mathcal{I}]$ is fed into it, providing a smooth start for adaptation training. This modification introduces only a small number of additional model parameters, and the increase in computational cost is negligible.

Mid-fusion. Both $\mathcal{I}$ and $\mathcal{M}^O$ are fed into the FPN in parallel, which yields two feature pyramid FP1 and FP2, as shown in Figure 3.4c. Two sets of RPNs and ROI heads are used

in parallel, and their initial weights are copied from the base model. FP1 is used in the left branch with RPN1 and ROI1, which is the same as the vanilla model. In the right branch with RPN2 and ROI2, the input feature pyramid is the fusion of FP1 and FP2:

$$\text{FP}_{fusion} = \text{FP1} \oplus \text{FP2}. \tag{3.3}$$

The fused feature pyramid has the same dimension as both FP1 and FP2. The fusion operator can be either non-parametric such as average pooling, or parametric such as convolution layers. The whole network is trained with losses from both branches

$$\mathcal{L}^{mid} = (1 - \alpha_{mid})(\mathcal{L}^{\mathcal{I}}_{rpn} + \mathcal{L}^{\mathcal{I}}_{roi}) + \alpha_{mid}(\mathcal{L}^{fusion}_{rpn} + \mathcal{L}^{fusion}_{roi}), \tag{3.4}$$

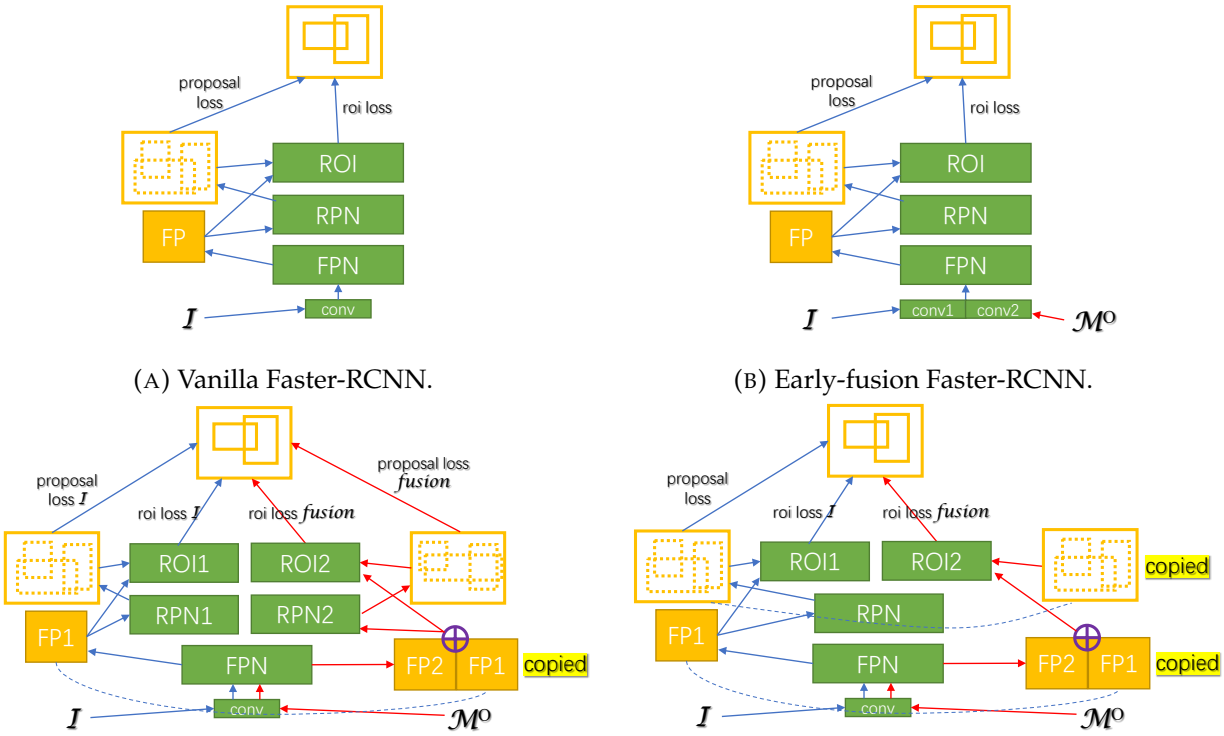
where α_{mid} is the weights of different losses.

Late-fusion is similar to mid-fusion as shown in Figure 3.4d. The key difference is that only the ROI head is branched. The region proposals are only generated from FP1, and the fused feature pyramid is only used in ROI head in the right branch. As a result, the total loss is

$$\mathcal{L}^{late} = \mathcal{L}^{\mathcal{I}}_{rpn} + (1 - \alpha_{late})\mathcal{L}^{\mathcal{I}}_{roi} + \alpha_{late}\mathcal{L}^{fusion}_{roi}. \tag{3.5}$$

Both mid-fusion and late-fusion introduce significant additional parameters and computation. $\mathcal{I}$ and $\mathcal{M}^O$ need to pass the backbone, requiring more computation even at inference time. We show the inference throughput of vanilla faster-RCNN, early-fusion faster-RCNN, mid-fusion faster-RCNN, and late-fusion faster-RCNN in Table 3.1. All models use the same R-101 backbone, and are ran on the same NVIDIA RTX 4090 GPU using the same set of images. The dataloader is carefully optimized to eliminate any possible CPU bottleneck. Please note that although mid-fusion and late-fusion introduces significantly more parameters by duplicating the RPN and ROI networks, only the branch for the fused feature pyramid will be deployed for inference. Thus they do not enlarge the model compared to vanilla faster-RCNN at inference time.

To train the fusion models in adaptation, we first finetune the base models on MSCOCO training set with the inpainted background models described in subsection 3.3.4 until convergence. During evaluation, we use the last background image to obtain the object masks, as it is the closest to the evaluation images on the timeline.



(A) Vanilla Faster-RCNN.

(B) Early-fusion Faster-RCNN.

(C) Mid-fusion Faster-RCNN.

(D) Late-fusion Faster-RCNN.

FIGURE 3.4: Illustration of different fusion models for Faster-RCNN. Network modules are indicated by green, and feature maps and outputs are indicated by yellow.

fusion model	inference throughput (images/s)
vanilla model	17.78
early-fusion	17.33
mid-fusion	11.93
late-fusion	11.96

TABLE 3.1: Inference throughput of different fusion models in term of images per second. As stated in the main paper, early-fusion adds very limited computational cost, while mid-fusion and late-fusion impact the speed more significantly, but they give higher precision.

3.4 Scenes100 dataset

For our proposed scene-adaptive object detection problem, there is no existing dataset with long enough videos of stationary backgrounds for the development and evaluation of self-supervised adaptation techniques. We therefore collected a new dataset called

Scenes100, which will be described in this section. This dataset is also used for experiments in chapter 4 and chapter 5.

3.4.1 Data collection

We used keywords such as "live view", "street camera", and "live webcam" to search for live streams on YouTube. We look for streams that are of decent quality and have fixed camera perspective. Around 200 candidate videos were recorded between September 2020 and February 2022. We picked 100 videos to compile our Scenes100 dataset, ensuring that each video is longer than two hours and has minimum width and height of 720 pixels. We also aimed for a dataset with high diversities. The locations of the cameras of the videos in Scenes100 is shown on a world map in Figure 3.5. Unlike most other object detection or scene understanding datasets, which are captured in a smaller range of locations, the 100 videos of Scenes100 were recorded in a variety of 16 countries and territories, giving great diversity.

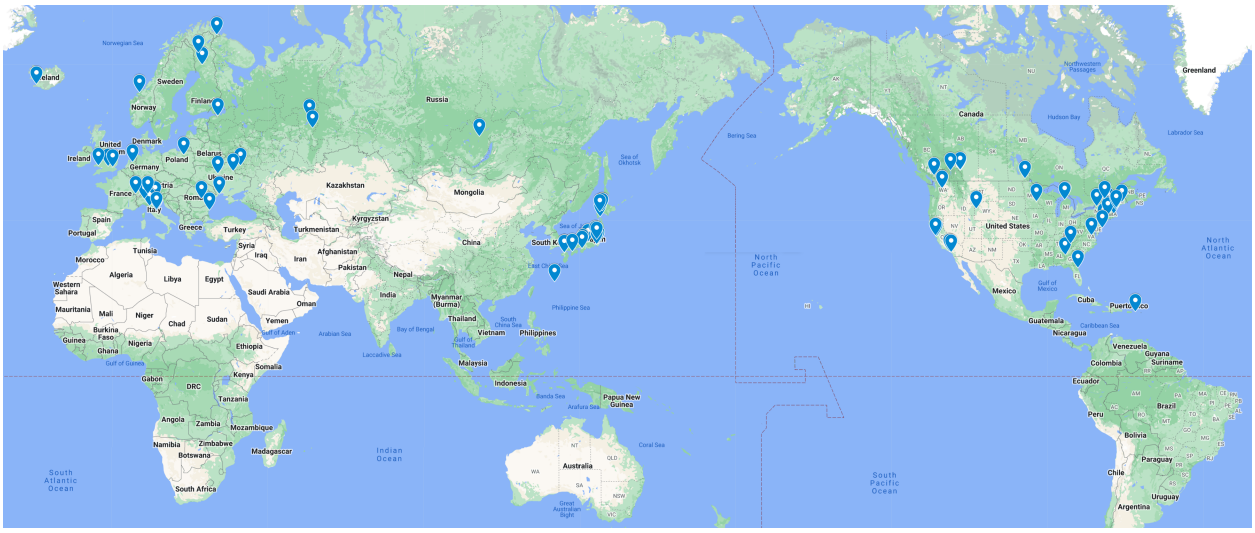


FIGURE 3.5: The locations of the cameras in Scenes100 on the world map. Antarctica and Arctic regions are not included. Each location is represented by a blue pin. The number of pins is smaller than 100, as some of the videos are captured in the same city. Map is created using tools provided by Google Maps. Please see https://support.google.com/maps/answer/3145721 for its conventions on region names and borders.

The dataset contain videos from both indoor and outdoor scenes, with different camera perspectives. The density and scale of visual objects also vary significantly from one

video to another. Some sample frames is presented in Figure 3.6 to show the diversity and therefore usefulness of our dataset. Please refer to the sub-captions for our comments for each of the scenes. As will be shown in the experiments, since all videos share the same set of hyper-parameters in the adaptation experiments, it also shows the effectiveness and robustness of the proposed self-supervised scene adaptive object detection method.

3.4.2 Manual annotation

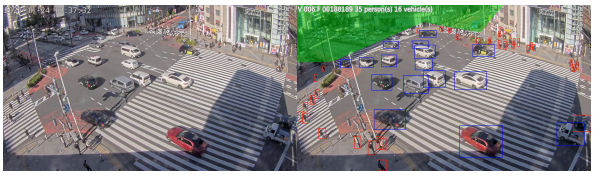
We are interested in detecting objects from two categories: *person* and *vehicle*. The *person* category includes any visible and recognizable people in the frames; this corresponds to the *person* category in MSCOCO [128] dataset. The *vehicle* category includes all vehicles with four or more wheels, so motorbikes, bicycles, tricycles are not included. This category corresponds to the *car*, *bus*, and *truck* categories in MSCOCO.

For each video, we used the first 1.5 hours for self-supervised and weakly-supervised training, and the remaining part for evaluation. Since many videos were taken with wide-angle lenses, some frames included faraway objects that were too small to be considered in evaluation. To address this, we manually drew polygon masks to exclude these faraway parts from annotation and evaluation. We then estimated the number of visual objects in the unmasked parts by first using a pre-trained detector, and then refining the estimate through human annotation. Frames for annotation were uniformly sampled from the second portion of each video, with the number of frames being inversely proportional to the estimated number of objects in the frames. This ensured that each video had roughly the same number of annotated objects. We contracted a data annotation company to perform bounding box annotation on those sampled frames. Some annotated frame samples for evaluation is shown in Figure 3.6.

3.4.3 Dataset statistics

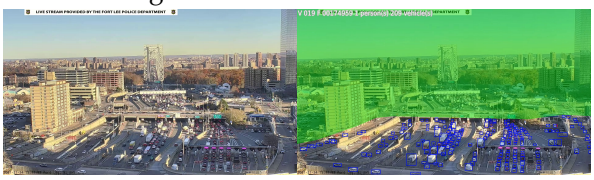
In Table 3.2, we compare Scenes100 with some popular object detection datasets. MSCOCO [128] is a fully-supervised general-purpose object detection dataset, which acts as the supervised source domain. KITTI [71] only contains object detection annotation associated to individual images. In BDD100K [231], object detection annotation is given for one key frame in each video. For CityScapes [33], we consider the finely labeled instance-level segmentation masks, which are given for the 20^{th} image from a 30 frame video snippets lasting 1.8s.

(A) Video 001 recorded in Jackson, Wyoming, USA at September 2020. The field of view and occlusion are moderate. Image quality is very clear.



(B) Video 006 recorded in Tokyo, Japan at November 2021. The field of view is very wide causing significant corner distortion.

(C) Video 016 recorded in Varna, Bulgaria at November 2021. The field of view is extremely wide. The lighting is low.



(D) Video 019 recorded in New York City, USA at November 2021. The field of view is extremely wide. Object are densely occluded.

(E) Video 040 recorded in Osaka, Japan at November 2021. It is a shadowed walkway in business district with heavy object occlusion.

(F) Video 049 recorded in Tokyo, Japan at November 2021. The field of view is extremely wide and the objects appear very small.

FIGURE 3.6: Sample frames from Scenes100 with their non-annotation masks and annotation bounding boxes. As described in the sub-captions, the videos cover a variety of locations, weather, lighting conditions, image qualities, camera perspectives, and indoor/outdoor environments. The diversity of our dataset makes it representative for various scenes and thus useful for the understudied scene adaptive object detection task.

dataset	contain videos	average length	frames	countries	bounding boxes	boxes per video
MSCOCO [128]	No	-	-	-	897K	-
KITTI [71]	No	-	-	1	80K	-
BDD100K [231]	Yes	40s	120M	1	1.8M	18
CityScapes [33]	Yes	1.8s	150K	2	65K	13
Scenes100	Yes	2h	21.6M	16	84K	840

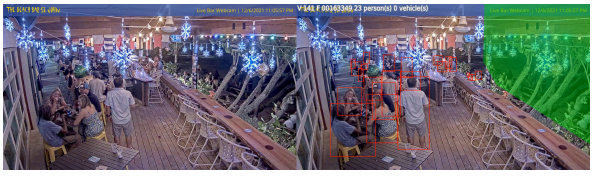
TABLE 3.2: Comparison of Scenes100 with other object detection datasets (K= 10^3, M= 10^6).

(G) Video 074 recorded in San Francisco, California, USA at November 2021. There is very strong contrast between light and shadow.

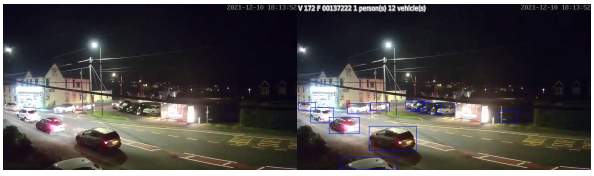
(H) Video 090 recorded in Ust-Kut, Russia at December 2021. The weather is snowy, leaving mostly white background.

(I) Video 114 recorded in Kennebunkport, Maine, USA at December 2021. Most of the objects are in a strong back-light condition.

(J) Video 128 recorded in Katashina, Japan at December 2021. This is an indoor scene with people occluded by desks and chairs.



(K) Video 141 recorded in St John, U.S. Virgin Islands at December 2021. It is an indoor scene with heavy object occlusion.



(L) Video 172 recorded in Ammanford, Wales at December 2021. The lighting is very low, and motion blur of the objects is strong.

FIGURE 3.6: (Continued) Sample frames from Scenes100 with their non-annotation masks and annotation bounding boxes. As described in the sub-captions, the videos cover a variety of locations, weather, lighting conditions, image qualities, camera perspectives, and indoor/outdoor environments. The diversity of our dataset makes it representative for various scenes and thus useful for the understudied scene adaptive object detection task.

The scale of Scenes100 is comparable to that of KITTI and CityScapes. However, Scenes100 has three unique features. First, the videos in Scenes100 are much longer than those in any other datasets with scenes, providing sufficient data for a model to adapt to each of them. Second, Scenes100 has greater diversity. Unlike some other datasets that are recorded in only a few cities with similar camera angles, background environments, and object types, Scenes100 covers several different countries, with varying weather conditions, road conditions, view of field, camera perspectives, and vehicle types. Third, each video in Scenes100 has a fixed camera perspective and stationary background scene, making background modeling possible.

Note that many currently available datasets are not primarily intended for self-supervised

adaptive object detection tasks. They may possess other features, such as segmentation masks, multi-modal sensor data, depth maps, or annotations for object tracking. However, despite the absence of these attributes, Scenes100 holds value as the first extensive and varied dataset that features fixed camera perspectives and lengthy videos. As such, it can complement existing dataset collections and serve as a valuable resource for researchers exploring scene adaptive object detection.

3.5 Experiments

3.5.1 Evaluation protocols

We evaluate the performance of object detectors following the COCO evaluation protocol [128] to calculate the average precision at $IoU = 50\%$ (AP^{50}) and mean average precision of different IoU thresholds from 0.5 to 0.95 (AP^m), with some modifications as follows.

Non-evaluation region: before feeding the ground-truth and detected bounding boxes to the COCO evaluator, the bounding boxes that have at least one corner inside the non-evaluation mask will be removed. So the faraway parts in the frames where objects are deemed too small and blurry will not affect the evaluation results. Please see subsection 3.4.2 and Figure 3.6 for more details.

Category weighting: in the standard COCO evaluation protocol, the overall multi-class AP is the simple average of the APs of the different categories. If a class is sparse in the training set and validation set, the model tends to perform poorly due to the scarcity of the training samples. In this case the average AP does not truly reflect the performance of a detector. Hence we propose an additional multi-class AP metric with weighted average where the weight is determined by the prevalence of ground truth object instances in the evaluation set. This weighted AP can better portray the performance of a detector in the case of long-tailed distribution. We refer the standard classes-mean COCO AP as AP_{co}, and the proposed weighted AP as AP_w. We refer the AP averaged over IoU thresholds and $IoU = 50\%$ as AP^m and AP^{50}.

Per-video evaluation: we aim to evaluate the performance of a scene adaption method, instead of a specific detection model. For each of the video, a model is adapted to it from the same base model trained on the source dataset. The performance of the base model and adapted model on each video is calculated individually as $\{(AP_{v,base}, AP_{v,adapt})|v = 1 \dots 100\}$. Then we evaluate the overall effectiveness of the adaptation method using the

averaged AP gain as:

$$APG = \frac{1}{100}\sum_{v=1}^{100}(AP_{v,adapt} - AP_{v,base}). \tag{3.6}$$

3.5.2 Implementation details

Our implementation is based on Detectron2 [222]. We start the finetuning from the models provided by the Detectron2 [222] model zoo. M1 and M2 are based on the configurations `"COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"` and `"COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"`, respectively. We keep the weights of the backbone and RPN, but re-initialize the weights of the new ROI heads, as the number of classes changes during the object categories remapping described in the main paper. Then the whole network is trained end-to-end on remapped MSCOCO training set. We use learning rate scheduling with base of 5×10^{-4}, image batch size of 4, and ROI batch size of 128. The models are trained for 15,000 iterations. We examine the models' performance on validation set and losses periodically during training to ensure convergence.

For self-supervised adaptation training, the training video portions are down-sampled uniformly to 5 frames per second, which gives 27,000 training frames per video. All spatial resolutions of the frames are kept. For pseudo-labeling, we set $\lambda_{det} = 0.5$, $\lambda_{sot} = 0.9$, and $\lambda_{iou} = 0.85$. The DiMP tracker is directly taken from the official PyTracking [35] implementation without any change. For location-aware mixup, we set $p_{mixup} = 0.3$, $r_{mixup} = 0.5$, and $\alpha_{cover} = 0.65$. For dynamic background extraction we set $T_{bg} = 90s$. For fusion models training, we use average pooling for feature pyramid fusion, and set $\alpha_{mid} = \alpha_{late} = 0.5$. In adaptation training, we use learning rate scheduling with base of 10^{-4}, image batch size of 4, and ROI batch size of 128. The models are trained for 20,000 iterations. We examine the models' performance on validation set and losses periodically during training to ensure convergence.

During training, we include the same number of images from MSCOCO training set as the number of unlabeled frames from the videos. The hyper-parameters are the same for all videos. We check the training loss for all experiments, including the baseline methods, to ensure convergence. We also train "compound" models where all videos are used together for adaptation.

3.5.3 Domain adaptation baselines

We compare our methods with several domain adaptive object detection baselines. In their original papers, the methods are evaluated on other domain adaptive object detection datasets. However, those datasets do not contains videos with fixed camera perspectives, so the proposed location-aware mixup and dynamic background extraction cannot be applied. We instead apply the methods in other papers on Scenes100. For a fair comparison, all methods start with the same base model and weights, and then are adapted to and evaluated on each scene individually. We keep the values of the hyper-parameters that are specific to a baseline method used in the original papers. We use the same learning rate, batch size, and number of iterations as our experiments in those baseline methods. More details can be found in the supplementary material.

Self-Train (ST) [176] is a method that uses detection and tracking to obtain pseudo bounding boxes, similar to ours. But ST applies tracking only in one direction. It assigns different weights to the pseudo bounding boxes during training, but no refinement nor cross-teaching. The official implementation[1] does not include the code for detection, tracking, and hard negative mining, so we re-implemented it in our framework.

STAC [185] uses the base model to get pseudo bounding boxes, and trains the model on a strongly augmented version of the target domain image with those boxes. We take the code of the core algorithm from the official implementation[2], which is the image augmentation function set, and integrate it into our framework.

Adaptive Teacher (AT) [123] also uses self-training, but the pseudo bounding boxes are detected on the fly. It utilizes exponential moving average to update the model gradually. It also uses weak/strong data augmentation, and a domain classifier to align the features of the two domains. We directly use the official implementation[3], and simply replace the initial base model and data loader with ours.

H^{2}FA R-CNN [224] applies feature alignment of source and target domains at various levels of Faster-RCNN. It assumes that image-level weak annotation is available for the target domain. So we use the labels from our pseudo-labeling process as image-level class labels. We directly use the official implementation[4], and simply replace the initial base model and data loader with ours.

[1] https://github.com/AruniRC/detectron-self-train
[2] https://github.com/google-research/ssl_detection
[3] https://github.com/facebookresearch/adaptive_teacher
[4] https://github.com/XuYunqiu/H2FA_R-CNN

TIA [246] is a method that instantiates feature alignment by using both domain classifier and auxiliary classification and localization heads. The model is trained in a mini-max manner using gradient reverse layers [64]. The official implementation[5] is based on another framework not compatible with our dataset and model architecture. So we implement it following the core logic in the official code base. We have to reduce the coefficient of auxiliary consistency losses by a factor of 10 to avoid training divergence. Number of training iterations is also reduced to 250, for longer training schedule leads to strong performance degradation.

LODS [119] uses style enhancement as data augmentation. The pseudo bounding boxes are generated from the original images, and the features from both version of the same image are aligned. It also utilizes exponential moving average for the weights. It does not use any supervised training data from the source domain. The official implementation[6] is based on another framework not compatible with our dataset and model architecture. So we implement it following the core logic in the official code base. Number of training iterations is also reduced to 250 due to performance degradation.

3.5.4 Comparison with baselines

Model	# params	MSCOCO		Scenes100			
		AP^m_{co}	AP^{50}_{co}	AP^m_{co}	AP^{50}_{co}	AP^m_w	AP^{50}_w
M1 (R-50)	41.4M	50.05	76.48	40.52	62.12	41.28	64.65
M2 (R-101)	60.5M	51.29	77.46	41.11	63.10	41.96	65.74

TABLE 3.3: Performance of base models on MSCOCO validation set with remapped categories and Scenes100 before any adaptation. For Scenes100, APs are evaluated on each video individually and then averaged. R- stands for ResNet. # indicates the number of parameters (M= 10^6). ^{m} and ^{50} stand for mean over IoU thresholds and $IoU = 50\%$, respectively. $_{co}$ and $_w$ stand for standard classes mean and proposed classes-weighted mean.

The performance of the base models is shown in Table 3.3, along with their number of trainable parameters. M2 is a bigger model with more parameters and representation capacity, so it outperforms M1 by a noticeable margin in all the metrics before adaptation. All the AP gains shown in other tables are based on the performance of base model M2.

[5] https://github.com/MCG-NJU/TIA

[6] https://github.com/Flashkong/Source-Free-Object-Detection-by-Learning-to-Overlook-Domain-Style

Note that the AP numbers on Scenes100 is already relatively high at only about 10 points below the performance on MSCOCO, so we do not observe very high (> 5 points) AP gains in the results shown later.

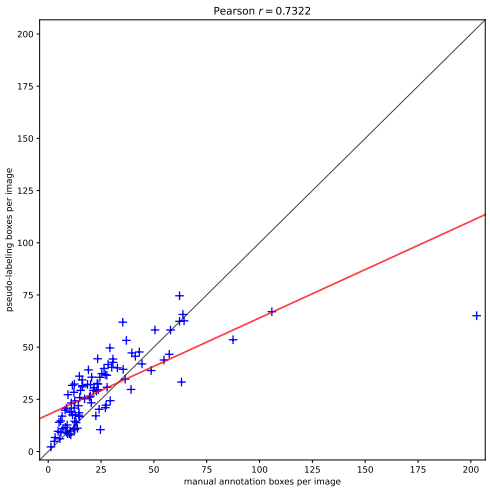


FIGURE 3.7: Compare the number of object bounding boxes per image of manual annotation and pseudo-labeling. Each scatter point + represents one video. The red line — indicates the least-square linear model fit to the scatter. The Pearson correlation coefficient is displayed.

In Figure 3.7, we compare the number of manually labeled object bounding boxes per image with the number of pseudo bounding boxes after the proposed pseudo-labeling procedure. The same set of hyper-parameters are used as in the experiments. All the pseudo boxes with at least 1 corner inside the non-evaluation mask are removed for consistency. Please note that the mask is not used in the adaptation training experiments. It is clear the number of bounding boxes from pseudo-labeling is correlated with actual number of bounding boxes from human annotation. Please note that the comparison is not precise, since the frames used for pseudo-labeling and human annotation come from different parts of the videos, which means the density of objects can change. When the object density is not very high (less than 50 object per image), the correlation is strong, showing the proposed pseudo-labeling can identify most of the objects. However, when the object density is very high, meaning that there is heavy occlusion or the field of view is extremely wide, the scene becomes more difficult for object detectors and the number of pseudo bounding boxes is smaller compared to the actual number.

We compare the performance of different adaptation methods in terms of average AP gain in Table 3.4a. Here we show our best combination of methods, which incorporates

pseudo-labeling, location-aware mixup, and object mask mid-fusion. Among the baseline methods for domain adaptive object detection, only ST and LODS improves the performance significantly. AT only yields marginal improvement in classes-weighted AP. STAC, H^{2}FA, and TIA actually degrade the performance on Scenes100. LODS performs surprisingly well considering it does not use source domain images. However, we found that any training schedule longer than 1000 iterations would lead to severe performance degradation. The proposed method yields much higher, consistent, and stable improvement.

In general, self-training-based methods (ST, AT, STAC, and the proposed method) perform better than domain alignment-based methods (H^{2}FA and TIA). Domain alignment is usually achieved through adversarial learning, which is known to have unstable objectives and is more difficult to optimize [62, 129, 184, 187]. Therefore, these methods are more sensitive to hyperparameters and training schedules, making it challenging to determine the optimal settings for all datasets. Applying domain alignment methods directly to our scene adaptive dataset without careful tuning can cause problems as our dataset has less in-domain variance than domain adaptation datasets. Furthermore, since the scenes in our dataset are very different, they might require different settings. This implies that Scenes100 has its uniqueness compared to more generally purposed domain adaptive object detection datasets. Our proposed method is more robust to this setting and does not rely heavily on hyperparameter tuning, as we use the same set of hyperparameters across all 100 videos.

We also try to treat Scenes100 in a manner closer to the standard domain adaptive object detection problem. All 100 videos are regarded as a whole target domain, and the models are trained on all the training frames from them, resulting a generic (compound) model for all videos. For consistency reasons, we keep all the hyper-parameters and settings during training unchanged from the individual adaptation settings used in the main paper, only to increase the number of training iterations by $20\times$ to incorporate larger training set. After training, we still use the same independent evaluation protocol as in the main paper, and report the average AP gains in Table 3.4b.

Interestingly, different methods performs very differently under this compound adaptation setting compared to individual adaptation setting. ST is the only method the performs noticeably better, implying that it benefits from higher variance in the training data. AT and TIA perform significantly worse. Our proposed method sees about 1-point drop in the APs, but is still the best one by a considerable margin. This shows that the proposed

method is more suitable for a fine-grained scene adaptive learning setting compared to more generic domain adaptive one.

Method	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
ST	+0.80	+0.24	+1.39	+1.03
STAC	-1.26	-5.12	-1.97	-6.64
AT	-0.75	-1.11	+0.06	+0.04
H^{2}FA	-3.10	-4.97	-3.77	-6.01
TIA	-0.32	-0.37	-0.32	-0.33
LODS	+0.45	+1.28	+1.02	+2.28
Proposed	**+3.76**	**+4.45**	**+3.78**	**+4.65**

(A) Adaptation performance of models that are adapted to each video independently.

Method	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
ST	+1.69	+1.47	+1.69	+1.46
STAC	-1.03	-4.81	-1.56	-6.03
AT	-4.64	-6.92	-3.57	-5.50
H^{2}FA	-3.78	-5.98	-3.99	-6.66
TIA	-1.82	-2.76	-1.58	-2.34
LODS	+0.59	+1.01	+0.69	+1.33
Proposed	**+2.77**	**+3.68**	**+2.56**	**+3.48**

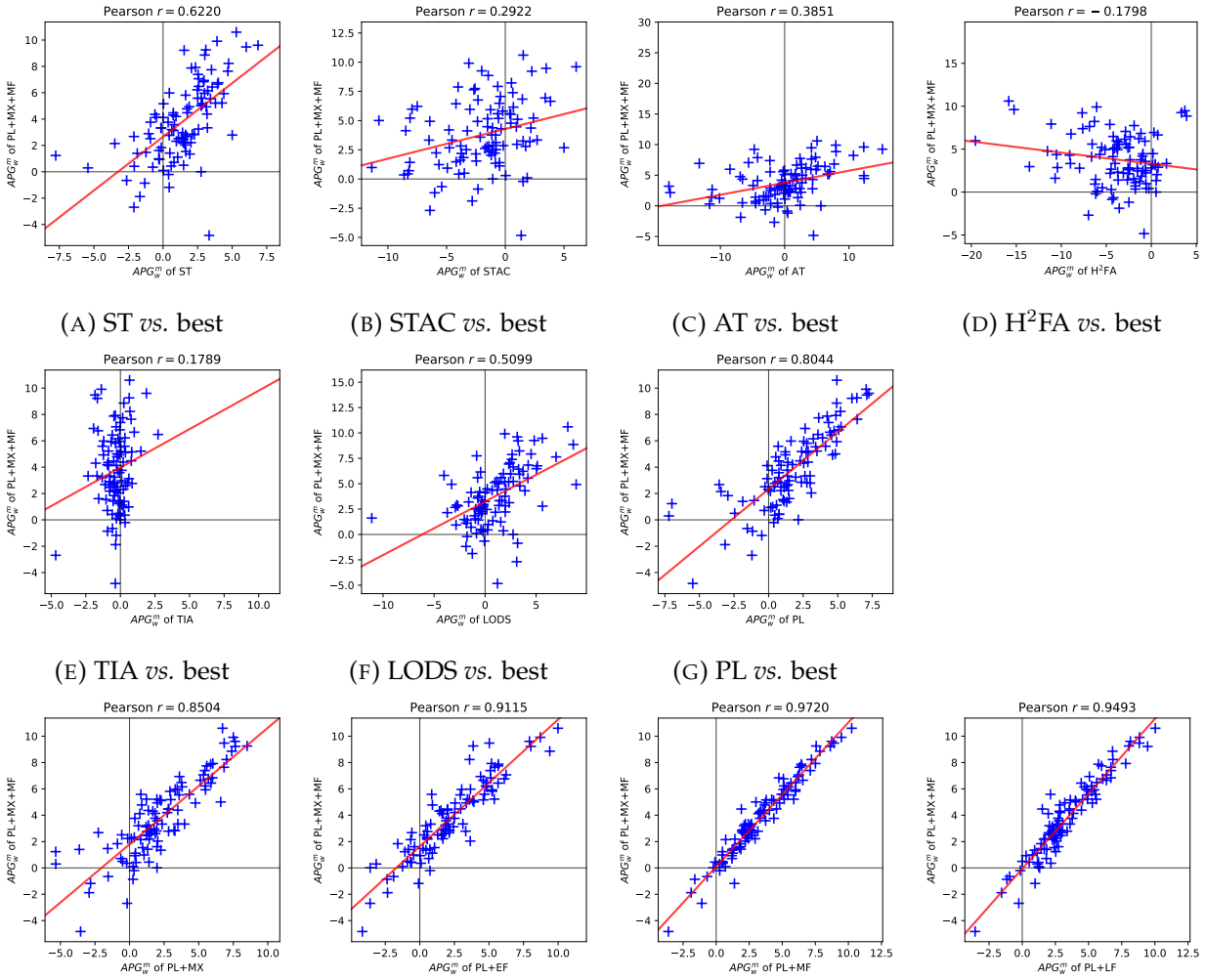
(B) Adaptation performance of models that are adapted to the whole Scenes100 dataset.

TABLE 3.4: Averaged AP gain of different adaptation methods. ^{m} and ^{50} stand for mean over IoU thresholds and $IoU = 50\%$, respectively. $_{co}$ and $_{w}$ stand for standard classes mean and proposed classes-weighted mean.

3.5.5 Correlation of AP gains of methods

We take a framework either from the baseline methods or from our ablation study, pair it with our best proposed method (pseudo-labeling + location-aware mixup + object mask mid-fusion, indicated by PL+MX+MF). we plot the individual AP gain after adaptation on each video of the 2 models in the pair as a scatter and calculate the Pearson correlation coefficient. The results are shown in Figure 3.8.

It is clear that only ST and LODS, which give moderate AP gain during adaptation, is weakly correlated with the proposed method. STAC, AT, H^{2}FA, and TIA all are not correlated with the proposed method. However, the combinations PL (pseudo-labeling), PL+MX (pseudo-labeling + location-aware mixup), PL+EF (pseudo-labeling + object mask early-fusion), PL+MF (pseudo-labeling + object mask mid-fusion), and PL+LF (pseudo-labeling + object mask late-fusion) are all strongly correlated with the best method. This indicates that our proposed methods are consistent in scene adaptation performance. The scenes that all methods perform poorly can be regarded as hard samples.



(A) ST *vs.* best (B) STAC *vs.* best (C) AT *vs.* best (D) H^{2}FA *vs.* best

(E) TIA *vs.* best (F) LODS *vs.* best (G) PL *vs.* best

(H) PL+MX *vs.* best (I) PL+EF *vs.* best (J) PL+MF *vs.* best (K) PL+LF *vs.* best

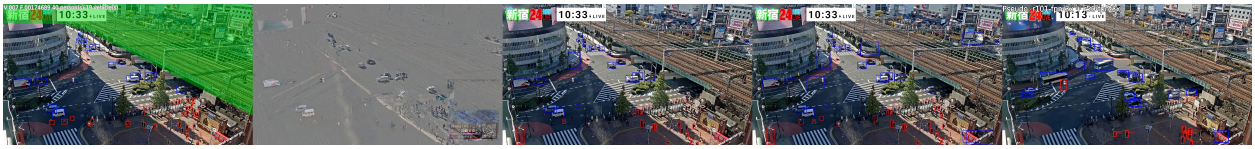
FIGURE 3.8: Compare the AP gains of different frameworks against the proposed best model. Each scatter point + represents one video. The red lines — indicate the least-square linear models fit to the scatters. The Pearson correlation coefficients are displayed. See subsection 3.5.5 for notation of different models.

3.5.6 Study of success and failure cases

In Figure 3.9 we present the videos that the best proposed method performs extraordinary well or bad in term of APG_w^m, and discuss the possible causes. Since it is the AP gains being examined, good performance means not only that the adapted model achieves high precision, but also that the base model cannot perform very well so there is room for improvement. Bad performance means the adaptation process actually degrade

the detection ability of the base model. Please note that the case study is qualitative and empirical.

(A) Success case of video `154`, adaptation increases APG_w^m from 17.59 to 27.19. The base model misses some of the cars, probably due to the complexity of the scene and varied light and shadow conditions. The pseudo-labeling can identify most of the target objects, and the adapted model is able to pick up those cars missed by the base model.



(B) Success case of video `007`, adaptation increases APG_w^m from 25.13 to 33.36. Similar to video `154`, the base model misses many of the smaller objects. Although the pseudo-labeling cannot find all of objects, the adapted model still performs significantly better.

(C) Success case of video `014`, adaptation increases APG_w^m from 42.53 to 51.75. The base model cannot properly detect the vehicles in the shadow. However, in the object mask they are more clearly outlined and can be identified by the adapted model.

FIGURE 3.9: Some success and failure cases of the best proposed method. For each case, from left to right, 5 images are presented: i) 1 sample frame from the evaluation split with its non-annotation masks and annotation bounding boxes, ii) its corresponding object mask image, iii) the detection output of the base model on the evaluation frame, iv) the detection output of the adapted model on the evaluation frame, and v) 1 sample frame from the training split with its pseudo-bounding boxes from pseudo-labeling. For the detection output, we only show object bounding boxes with confidence score higher than 0.5.

3.5.7 Ablation study

We try to exclude tracking pseudo bounding boxes from training. We also test only using M2 to generate the detection results. The results are shown in Table 3.5a. Including tracking bounding boxes in pseudo-labeling provides consistent performance gain. The results of only using M2 in detection also show that ensemble of models is beneficial.

(D) Failure case of video `051`, adaptation decreases APG^m_w from 32.84 to 30.96. Adaptation seems to make the model produce less bounding boxes at the heavily-occluded regions (*e.g.* the parked cars at middle-right of the frame). This is probably due to the fact that pseudo-labeling cannot identify each object accurately at those regions.

(E) Failure case of video `093`, adaptation decreases APG^m_w from 54.91 to 50.08. The base model already performs very well, likely because the background is covered by snow and the objects are well-separated. Pseudo-labeling introduces some false positive bounding boxes, which is learned by the adapted model. Tuning the pseudo-labeling hyper-parameters can probably fix this issue.

FIGURE 3.9: (Continued) Some success and failure cases of the best proposed method. For each case, from left to right, 5 images are presented: i) 1 sample frame from the evaluation split with its non-annotation masks and annotation bounding boxes, ii) its corresponding object mask image, iii) the detection output of the base model on the evaluation frame, iv) the detection output of the adapted model on the evaluation frame, and v) 1 sample frame from the training split with its pseudo-bounding boxes from pseudo-labeling. For the detection output, we only show object bounding boxes with confidence score higher than 0.5.

We explore the effect of the different components in our methods by experimenting various combinations of them. For all experiments, we keep the pseudo-labeling unchanged, as it is the basis of other components. We compare different types of object mask fusion (Fusion): early, mid, and late. We also compare the proposed Location-aware object mixup with Random mixup. Table 3.5b shows the results. The experiments on hyper-parameters can be found in the supplementary material.

The proposed pseudo-labeling, mixup, and object mask fusion all benefit the adaptation performance. Applying only the pseudo-labeling already leads to AP gain higher than ST, which is more complicated involving pseudo box weighting. Location-aware mixup is also more advantageous than Random mixup, validating the assumption in subsection 3.3.3 that having fewer artifacts is better for adaptation. Among three fusion options, mid-fusion yields the best result. That is perhaps because in a mid-fusion model, the RPN utilizes the fused feature pyramid, which can contain critical information of object boundaries. The combination of mixup and object mask fusion can further improve

detectors	tracking	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
M1+M2	✓	**+0.95**	**+0.54**	**+1.67**	**+1.55**
M1+M2	✗	+0.73	+0.22	+1.49	+1.31
M2	✓	+0.58	-0.30	+1.21	+0.67

(A) Effects of tracking and ensemble of models.

Mixup	Fusion	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
✗	✗	+0.95	+0.54	+1.67	+1.55
Location-aware	✗	**+1.72**	**+1.67**	**+2.25**	**+2.53**
Random	✗	+1.22	+1.13	+1.76	+2.03
✗	early	+1.85	+2.12	+2.22	+2.73
✗	mid	**+3.40**	**+3.81**	**+3.67**	**+3.98**
✗	late	+3.34	+3.60	+3.38	+3.72
Location-aware	early	+2.25	+2.82	+2.59	+3.46
Location-aware	mid	**+3.76**	**+4.45**	**+3.78**	**+4.65**
Location-aware	late	+3.66	+4.10	+3.73	+4.31

(B) Effects of different mixup strategies and fusion stages.

Mixup	Fusion	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
✗	AVG	**+3.40**	**+3.81**	**+3.67**	**+3.98**
✗	CNN	+3.25	+3.52	+3.26	+3.79
✗	ATTN	+3.22	+3.73	+3.23	+3.95
Location-aware	AVG	**+3.76**	**+4.45**	**+3.78**	**+4.65**
Location-aware	CNN	+3.60	+4.20	+3.71	+4.50
Location-aware	ATTN	+3.30	+3.89	+3.45	+4.26

(C) Effects of different feature pyramid fusion methods. AVG, CNN, and ATTN indicate mid-fusion based on average pooling, CNN, or attention module, respectively.

TABLE 3.5: Ablation study for the proposed components of our scene-adaptive object detection method, on adaptation performance. The numbers shown are the averaged AP gain. ✓ means being applied and ✗ means not being applied. ^{m} and ^{50} stand for mean over IoU thresholds and $IoU = 50\%$, respectively. $_{co}$ and $_{w}$ stand for standard classes mean and proposed classes-weighted mean.

the performance.

We also compare different feature pyramid fusion methods described in (3.3). In our main experiments, we simply average the feature pyramids from the original image and the object mask. Now at each level of the pyramid, we use a separate CNN or an attention

module to fuse the feature maps. The comparison is shown in Table 3.5c. The parametric method performs slightly worse than average-pooling, probably because the newly introduced modules have randomly initialized weights and thus require additional supervised training data.

3.5.8 Effect of hyper-parameters

We change the hyper-parameters λ_{det} and λ_{iou} for pseudo-labeling, and see how they effect the performance of the adapted models. To avoid interfering of other factors, location-aware mixup and object mask fusion are not used. The results are shown in Table 3.6a. The results show that increasing λ_{det} to certain level can lead to increase of APG^m, but decrease of APG^{50}. The performance will be degraded if λ_{det} is too high. Lower λ_{iou} leads to slightly higher performance.

We change the hyper-parameters p_{mixup}, r_{mixup}, and α_{cover} for our proposed location-aware object mixup, and see how they effect the performance of the adapted models. To avoid interfering of other factors, object mask fusion is not used. The results are shown in Table 3.6b. It can be seen that increasing p_{mixup} or r_{mixup}, meaning stronger mixup, can improve the APs slightly.

Please note that the results shown in Table 3.6a and Table 3.6b should not be viewed as a full-scale hyper-parameter tuning. The hyper-parameters in pseudo-labeling and location-aware mixup can interfere with each other. The situation is more complicated when object mask fusion is used. A proper tuning will require a search in the full hyper-parameter space, which is far beyond the computational capacity we have. It can also be reasonably expected that each video in Scenes100 has it own set of optimal hyper-parameters. Nevertheless, we argue that our proposed methods are mostly insensitive to the hyper-parameters, and can still perform well even no video-specific tuning is applied.

3.6 Summary

We have presented a self-supervised framework for scene-adaptive object detection. Base detectors and generic trackers are used to generate pseudo object labels as the adaptive training targets in a cross-teaching manner. To fully utilize the background equivariance when camera perspective is fixed, we propose artifacts-free location-aware object mixup to augment the input images, and dynamic background extraction for additional input

λ_{det}	λ_{iou}	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
0.5	0.85	+0.95	+0.54	+1.67	+1.55
0.7	0.85	**+1.47**	+0.43	+1.79	+0.78
0.9	0.85	+0.33	-3.02	-0.12	-4.09
0.5	0.75	+1.28	**+0.60**	**+2.03**	**+1.75**
0.5	0.95	+0.36	+0.50	+1.00	+1.51

(A) Effects of pseudo-labeling hyper-parameters. λ_{det} is the minimum score for a detected object to be included in the pseudo bounding boxes. λ_{iou} is the IoU for 2 bounding boxes to be merged during refinement.

p_{mixup}	r_{mixup}	α_{cover}	APG^{m}_{co}	APG^{50}_{co}	APG^{m}_{w}	APG^{50}_{w}
0.3	0.5	0.65	+1.72	+1.67	+2.25	+2.53
0.5	0.5	0.65	+1.77	+1.72	+2.41	+2.75
0.7	0.5	0.65	**+2.06**	**+2.16**	**+2.47**	**+2.90**
0.3	0.3	0.65	+1.67	+1.67	+2.15	+2.47
0.3	0.7	0.65	+1.60	+1.59	+2.20	+2.57
0.3	0.5	0.45	+1.81	+1.70	+2.29	+2.52
0.3	0.5	0.85	+1.73	+1.74	+2.23	+2.60

(B) Effects of locate-aware mixup hyper-parameters. p_{mixup} is the probability that a frame is selected for mixup. r_{mixup} is the probability a pseudo bounding box in the mixup source frame is copied and pasted. α_{cover} is the threshold that a covered pseudo bounding box to be removed if exceeded.

TABLE 3.6: Effects of pseudo-labeling and locate-aware mixup hyper-parameters on adaptation performance, in term of averaged AP gain. ^{m} and ^{50} stand for mean over IoU thresholds and $IoU = 50\%$, respectively. $_{co}$ and $_{w}$ stand for standard classes mean and proposed classes-weighted mean.

modality to the detector. We also introduce the first large-scale scene adaptive object detection dataset, **Scenes100**, with several unique features compared to other domain adaptive object detection datasets. Our method outperforms domain adaptive object detection baselines by a large margin on Scenes100. We also conduct extensive experiments to illustrate the effectiveness of the components in our framework.

Chapter 4

Efficiency-Preserving Scene-Adaptive Object Detection

I have described a scene-adaptive object detection framework in chapter 3. Experimental results in subsection 3.5.4 indicate that for scene-adaptation to perform well, dedicated and independently adapted models are required. This poses challenges to the computational resources needed to host all the models as the number of scene cameras increases. In real-world applications, not only detection precision but also efficiency are desired. To address the demand for improved detection precision as well as efficiency, in this chapter I introduce a framework that enhances a base object detector for specific scene adaptability, without compromising inference speed or system throughput. This method is compatible with established object detectors such as Faster-RCNN, YOLOv8, and DINO, and involves augmenting a base detector's network architecture with a mixture-of-experts structure. This structure incorporates additional trainable network branches, minimally increasing the overall parameter count, thus avoiding the considerable expense of replicating the entire model. The resultant enhanced detector operates as a cohesive single unit, facilitating an efficient client-server architecture with a shared inference engine for multiple video streams. The presented framework supports self-supervised learning, which eliminates reliance on manually annotated data. Experimental results on the Scenes100 dataset demonstrate that our approach significantly outperforms existing models in detection accuracy while maintaining operational efficiency. The content of this chapter is based on the published paper [244], and "we" in this chapter refers to its authors. The code is available at `https://github.com/cvlab-stonybrook/scenes100/tree/main/moe`.

4.1 Introduction

Real-time object detection in video feeds is critical for various computer vision applications, including anomaly detection in security systems, obstruction spotting on railway lines, and vehicle detection for traffic monitoring. While detection accuracy is critical for these applications to function correctly, one also needs to be mindful of the required computational resources so that the costs do not outweigh the benefits. Optimizing for detection accuracy within computational resource constraints is challenging. Smaller and quantized networks require less computation and can reduce cost. However, they also have limited representation capacity and generalization ability. This issue becomes even more pronounced when analyzing video streams from multiple scenes with a single detector, as these scenes can vary greatly in perspective, lighting, and appearance.

As described in chapter 3, scene-specific detectors outperform a single generic detector, since each detector is tailored to a particular scene. However, using separate models for individual scenes would prevent the use of an efficient model serving architecture with a shared inference engine (e.g., [39, 40, 41, 42, 140, 144]), as shown in Figure 4.1. Each model would be tied to a separate inference engine and consume a portion of the limited memory. Consequently, the number of video streams that can be processed simultaneously by the same inference hardware is limited by the available memory for these models, thereby reducing the system's throughput and increasing costs.

We propose to address the challenge of adapting a base object detector to diverse scenes, with the aim of improving detection precision while preserving inference speed and system throughput. Our method begins with the network architecture of a base detector, which we enhance with a sparse mixture-of-experts structure. In contrast to the significant overhead of replicating the entire model, only a small number of parameters are added. This enhanced detector operates as a single entity, enabling an efficient client-server architecture with a shared inference engine for multiple video streams. It is capable of adapting to diverse scenes via self-supervised learning, eliminating the need for manual annotation. This framework is compatible with various object detector architectures, including Faster-RCNN [172], YOLOv8s [103], and DINO-5scale [238]. Our experiments on Scenes100 demonstrate that our proposed method substantially surpasses existing models, delivering improved detection accuracy while maintaining runtime efficiency.

Note that our focus is to enhance the detector deployed on a centralized processing

server. Centralized AI processing has many benefits compared to distributed setups in scenarios with numerous video streams, even with the increasing prevalence of AI cameras [69, 150, 159]. AI-enabled cameras require expensive processing chips, and the model is hard to update. In contrast, a centralized processing server can support many cheaper, regular cameras, and updating a centralized detection model is relatively easier.

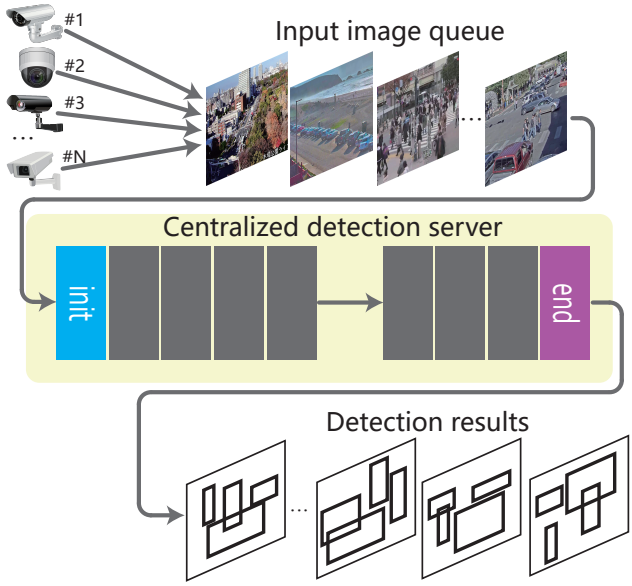


FIGURE 4.1: Illustration of the model-serving architecture. The centralized model serving architecture involves hosting a detector on a centralized server that serves multiple video streams. This setup is preferred for cost efficiency in scenarios with many video streams. However, this design also presents challenges in adapting the model to diverse video streams, which our proposed framework aims to address.

4.2 Related work

Network efficiency is a key consideration of this chapter. Various approaches can improve efficiency, including designing efficient architectures [91, 94, 96, 192], neural architecture searching [50], quantizing the parameters [72], and pruning network parameters [10]. Our proposed method complements these approaches and can be applied to various network architectures, including those that are optimized, quantized, or pruned. Recently LoRA [2, 93, 250, 254] has been utilized to fine-tune trained models for new tasks with

reduced memory usage. Our proposed method has the advantage of not increasing memory costs as the number of adapted scenes grows. Additionally, LoRA can be integrated as a plugin if memory consumption remains a concern after implementing our method.

Scene-adaptive object detection is discussed in chapter 3. However, if a separate model is trained on each scene, the total cost of both training and inference is proportional to the number of scenes, making scaling difficult. And the proposed method achieve adaptation using modality fusion (described in subsection 3.3.5), which needs more computation compared to the base detector. In this chapter we propose a self-supervised scene-adaptive object detector that is designed with computational resource limitations and scalability in mind while still outperforming methods that solely concentrate on improving detection accuracy.

Mixture-of-experts (MoE) [4, 24, 30, 47, 54, 182, 220] is the network architecture utilized by the proposed method to achieve both efficiency and scene-adaptability. An MoE model contains a set of experts, and a gating module activates a subset of the experts according to each input sample. We show that only a small portion of a detector network is scene-specific and needs to be converted to MoE, greatly reducing the parameter count. Our gating module is sparse and only activates one expert for each input image, so the computational cost for both training and inference is the same as the non-MoE base model. We also apply a two-stage training schedule to mitigate the overfitting issue caused by MoE and gating.

Knowledge distillation is the primary technique used by the proposed framework to actuate scene-adaptation training. It was originally proposed [86] to transfer the learned information from a teacher network to another student network. Self-distillation refers to the case when the teacher and student have the same architecture, and it achieves remarkable results in self-supervised vision representation learning [19, 25, 26, 27, 79, 146]. Many of such frameworks apply the idea of contrastive learning, which involves different data augmentations on the input of the teacher and the student. In our proposed self-supervised adaptation method, the teacher and the student also have identical architectures and weights at the beginning of training. We apply upscaling as the data augmentation for the teacher. Comparable methods are used in self-supervised domain-adaptive object detectors [22, 38, 119, 123, 247]. Our proposed method does not require complicated data augmentation methods, nor careful tuning of the hyper-parameters. Yet, it still achieves a significantly higher detection precision boost. Self-distillation methods can suffer from model collapse, which requires special treatments such as negative

pair sampling, regularization, or diversity enforcement. We avoid this issue by not updating the teacher during training.

4.3 Efficiency-preserving scene adaptation

This section describes our proposed framework which enhances an object detector with the capability of self-adapting to many different scenes while also preserving its suitability as the core component of a shared inference engine for all scenes. It utilizes a sparse mixture-of-experts (MoE) strategy, which assigns each video stream to its corresponding route within the model to improve adaptability. Only one expert is activated for each input image. It ensures that the model has enough capacity for scene-specific adaptation to improve its precision, while maintaining both the processing latency and the throughput of the inference engine.

4.3.1 Problem definition

Given a base detector $\mathcal{M}$, which is the core component of an inference engine that processes multiple camera streams from diverse scenes, instead of creating a separate model for each scene which linearly increases the overall parameter count, we aim to obtain an enhanced model $\mathcal{M}^*$ to replace $\mathcal{M}$, while ensuring the following characteristics:

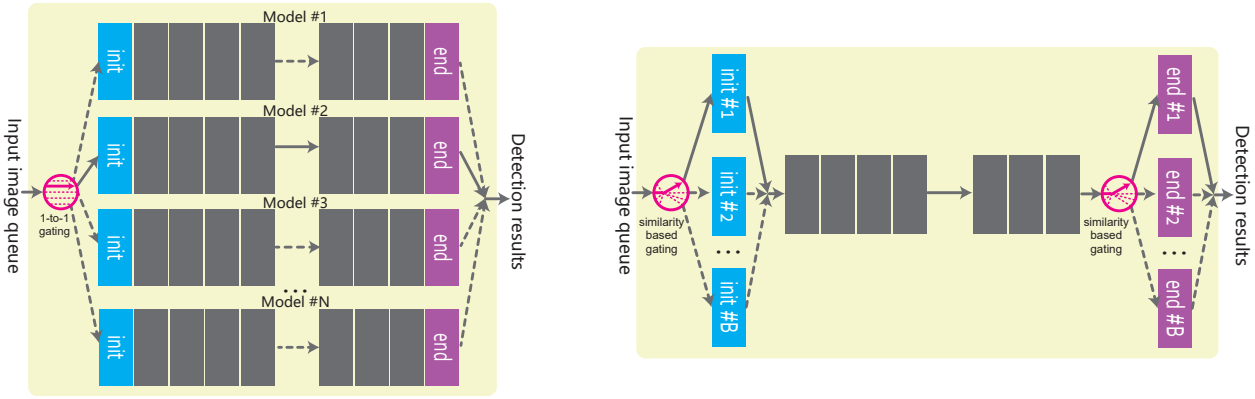
1. **Improved precision**: The model $\mathcal{M}^*$ should have the capacity to adapt to different scenes, thereby providing improved detection precision over $\mathcal{M}$.

2. **Consistent latency and throughput**: On the same inference hardware, the time $\mathcal{M}^*$ takes to process each image should be equivalent to that of $\mathcal{M}$. $\mathcal{M}^*$ should also match $\mathcal{M}$ in its ability to process the same number of video streams at an identical frame rate.

3. **Memory efficiency**: The parameter count for $\mathcal{M}^*$ must not be excessively large, even with a large number of scenes, ensuring that it can operate on the existing inference hardware without the need for additional memory.

4. **Self-supervised learning**: $\mathcal{M}^*$ should adapt to the scenes and through self-supervised learning, eliminating the necessity for manually-labeled data or human oversight.

4.3.2 Architecture of enhanced model



(A) Dedicate a scene-specific detector to adapt to each scene

(B) Proposed mixture-of-experts detector adapts to all scenes simultaneously

FIGURE 4.2: Different scene adaptation approaches. (a) The conventional adaptation approach, generating a separate model copy for each scene, significantly increases memory usage and decreases processing throughput. (b) The proposed sparse mixture-of-experts (MoE) approach with a budget of B. At each branching point, a similarity-based gating module routes the image to an expert. This method preserves the computational cost per image of the base model. The increase in the number of parameters is only sublinear relative to the number of scenes. The maximum number of video streams that can be supported depends solely on the computational speed of the inference hardware.

To design a detector with the aforementioned characteristics, we divide the detector's modules into scene-generic and scene-specific ones. Scene-generic modules are shared by all scenes, while each scene-specific module only adapts to a subset of scenes. This enables the model to adapt to diverse scenes while scaling sublinearly with the number of scenes. It allows for serving as many concurrent video streams as possible, limited only by the computational speed capacity of the inference hardware and not by additional memory constraints.

Most of the existing object detection architectures [11, 18, 20, 73, 74, 102, 103, 113, 127, 131, 169, 170, 171, 172, 213] are composed of two primary components: (1) a feature extractor network that generates feature maps from images, and (2) a detection head for localization and category classification of object instances. For a collection of diverse scenes, it is apparent that the most significant differences between them are object sizes,

camera perspectives, and lighting conditions. Consequently, the most appropriate layers for scene-specific adaptation are those closest to the input or output. In particular, the initial layers of the feature extractor play a pivotal role in adjusting to geometric and lighting variations, and the terminal layers of the head are crucial for finalizing the detector's judgments. The intermediary layers, which deal more with abstract visual representation and semantics, tend to differ less across different scenes. This division is shown in Figure 4.2a. These scene-specific modules usually comprise only a few percent of the network's total parameters. In subsection 4.4.3 we show that using scene-specific modules benefits scene-adaptation.

We enhance the base detector by selectively duplicating these scene-specific modules, as illustrated in Figure 4.2b. Let $|\mathcal{M}|$ be the parameter count of the original model, α the parameter proportion of scene-specific modules, and each of the scene-specific modules is duplicated for B times. B can be smaller than the number of scenes N, as a single module can still adapt to multiple similar video streams adequately. The parameter count of the enhanced MoE model is: $|\mathcal{M}^*| = (1 + \alpha(B - 1))|\mathcal{M}|$, which is significantly smaller than $B|\mathcal{M}|$. For instance, α = 1.37% for a Faster-RCNN with ResNet-101 backbone. If we use B=10 for 100 video streams, $\mathcal{M}^*$ is only 12% larger than $\mathcal{M}$, while being able to adapt to 100 different scenes.

The processing path for each input image in the MoE model $\mathcal{M}^*$ depends on the unique *scene ID* indicating the camera from which the image originates. At each branching junction within $\mathcal{M}^*$, a gating module determines the image's route based on its scene ID. This sparse mixture-of-experts approach ensures that, despite the branching, the computational cost for processing one image remains unchanged from the base detector $\mathcal{M}$. In this paper, we apply consistent gating rules across all gating modules in $\mathcal{M}^*$. This means the number of parallel modules at each branching point equals B, resulting in a total of B distinct paths in $\mathcal{M}^*$. It is feasible to vary the number of branches at different branching points and combine a branch from one point with an arbitrary one from another, yielding a combinatorial number of pathways. We reserve such considerations for future work.

4.3.3 Similarity-based gating

The MoE detector $\mathcal{M}^*$ is designed to handle B distinct processing paths. B is a hyperparameter adjusted for system memory constraints. When memory is very limited, we can

set B=1, effectively adapting a single model to all scenes. Conversely, with more memory, B could match the total number of scenes N, though it is commonly unnecessary. When $1 < B < N$, effective scene-to-branch assignment is crucial. As random allocation can be suboptimal, we propose a data-driven clustering strategy to assign similar scenes to the same branch. M images are sampled from each of the N scenes to compile a representative set on which the grouping is based on. A feature extractor $\mathcal{F}$ then calculates an image-level feature vector for every image. k-means clustering is performed on the feature vector set. Then, for every scene, we evaluate the cluster IDs assigned to its sample images and select the one for the entire scene using majority voting. This algorithm for scene-to-branch assignment, named *B-Means*, is outlined in algorithm 1. It is desirable to use a feature extractor $\mathcal{F}$ that already demonstrates effective object detection performance. In our experiments, we take the backbone of the warmed up model described in subsection 4.3.4 as $\mathcal{F}$. In subsection 4.4.3 we verify that B-Means improves performance over random allocation.

Algorithm 1: Similarity-based gating (B-Means)

Input: N video streams and a feature extractor $\mathcal{F}$
Parameters: Branching budget B and the number of samples per scene M
Output: Branch assignments for each scene $a_1, \ldots, a_N$, where $1 \leq a_i \leq B$

1 Sample M frames from each of N scenes: $\{X_{ij} | 1 \leq i \leq N, 1 \leq j \leq M\}$.
2 Extract feature vectors: $f_{ij} \leftarrow \mathcal{F}(X_{ij})$.
3 Run k-means on $\{f_{ij}\}$ with $k = B$, get the cluster assignments $\{k_{ij}\}$.
4 Assign branch IDs to scenes using voting: $a_i \leftarrow \text{argmax}_{1 \leq k \leq B} \sum_{j=1}^{M} \delta(k_{ij} = k)$.

4.3.4 Training for adaptation

Adapting the MoE detector to different scenes requires images from those scenes with corresponding training signals to update the parameters of the base detector. The images are already produced by the cameras, and various strategies can be used to generate self-supervised pseudo annotation. For instance, tracking [5, 9, 52, 141, 142, 143, 181] aids in identifying missed detections, as enforcing temporal persistence reduces false positives. Model ensembles [63, 147] enhance accuracy but demand more computational power. Alternatively, input data augmentation [148, 162] transforms the original image into variants, each analyzed by the base detection model. The aggregated results from all variants can be used as the pseudo annotation. Employing tracking, model ensembles, and input

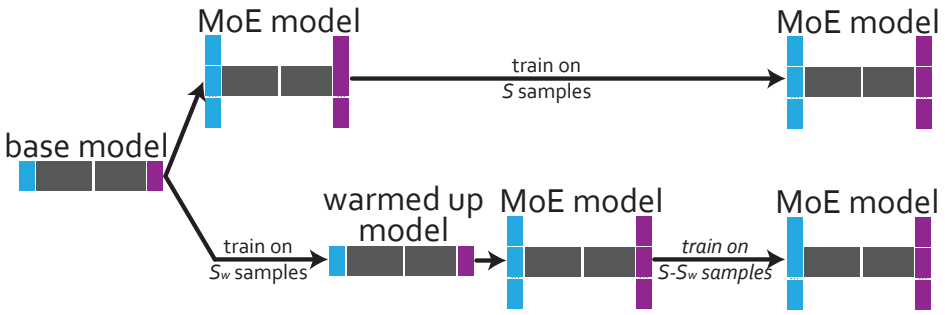


FIGURE 4.3: Two-stage (bottom) compared with one-stage (top) training schedule. With the same total of S training samples, the model is first trained on S_w samples without branching to obtain the warmed up single model. Then, it is enhanced with MoE branching and further training on $S - S_w$ samples. Two-stage training mitigates the overfitting issue brought by MoE and gating.

augmentation individually or in combination could likely yield more reliable labels than the base detector's outputs.

However, determining the optimal pseudo label generation method is not the primary focus of this work. We show that for several state-of-the-art detectors, self-distillation can significantly increase the precision of the MoE model. The base model serves as the teacher network, and the adapted MoE model is the student. Teacher and student networks have the same parameters at the beginning of adaptation training, while the input of the teacher is augmented by bilinear upscaling. Upscaling greatly improves the detection precision of the base model, likely because in the tested dataset, the object is considerably smaller than the dataset on which the base models are trained, and upscaling narrows this data distribution gap. Pseudo labels for training the student network can be generated by keeping detected objects from the teacher with confidence scores above a threshold θ. Please note that the input images of the student network keep their original size to preserve runtime efficiency. We avoid the model collapse issue of self-distillation by not updating the teacher during training, which is shown to be beneficial by experiments in section 4.7. In section 4.6, we also examine the effectiveness of our method with pseudo labels generated through tracking and model ensembles described in subsection 3.3.2.

Branching certain modules of a model can enhance its ability to adapt to diverse scenes, but it also potentially causes overfitting. In the vanilla model, every parameter

is exposed to all the training samples. In contrast, with the same number of training samples, each training sample follows only one path during training in an enhanced MoE model, resulting in each of the parallel modules being exposed to far fewer samples. We propose a two-stage training schedule for the MoE model. The first stage involves training a vanilla model without branching on samples from all scenes, producing the so-called "warmed up" model. In the second stage, this warmed up model is enhanced with MoE and further trained with gating on the remaining samples. This *two-stage* training schedule is depicted in Figure 4.3. In subsection 4.4.3, we show that two-stage training results in higher detection performance over one-stage training while using the same number of total training samples.

4.4 Experiments

This section describes our extensive experiments, starting with the dataset, implementation details, and baselines. Afterwards, it compares the performance of the proposed method with the baselines in terms of both precision and efficiency.

4.4.1 Dataset, Evaluation Metrics, and Baselines

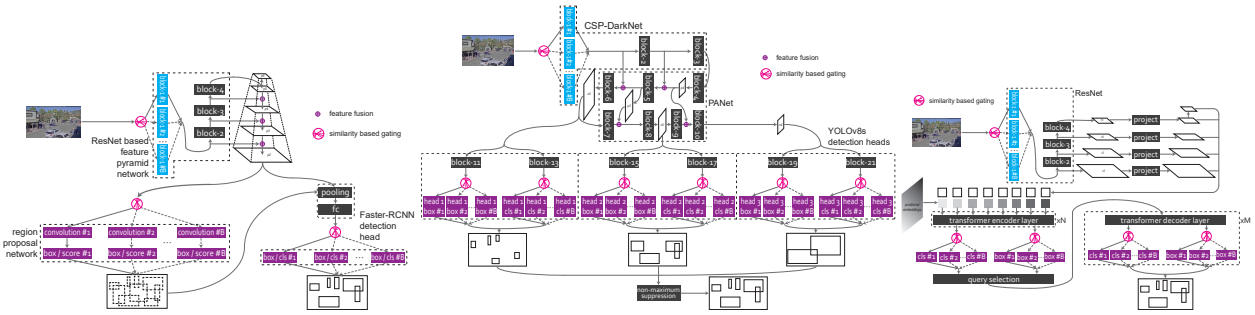
Model	Backbone	COCO	Scenes100
Faster-RCNN [172]	R-101	52.77	41.96
Faster-RCNN [172]	R-18	46.96	35.68
YOLOv8s [103]	D-53	48.61	44.10
DINO-5scale [238]	R-50	54.65	40.54

TABLE 4.1: Mean Average Precision (AP^m) of base models on COCO2017-val [128] and Scenes100. Models are trained on COCO2017 training set with remapped object classes. R-101, R-50, R-18, and D-53 stand for ResNet-101, ResNet-50, ResNet-18, and CSPDarkNet-53 backbones, respectively.

We use Scenes100 propsed in section 3.4 for the experiments. It is the only publicly available dataset with a sufficient number of lengthy videos and ample bounding boxes for scene-adaptive detection research. Following subsection 3.5.1, we evaluate a detector's precision by calculating the per-class mean Average Precision score across different IoU thresholds from 0.5 to 0.95 (AP^m), and then averaged weighted by the prevalence

of the instances of each class. Adaptation starts from the two-class base detector in section 3.5, which is trained on COCO2017 [128] training set with remapped object classes *person* and *vehicle*. Most of the baseline methods aim to achieve high detection accuracy without considering efficiency, so they are based on bigger models such as Faster-RCNN [172] with a large backbone. In the experiments, for comparison with baseline methods, we use base Faster-RCNN models on ResNet-101 [82] backbone and use the weights trained in section 3.5. We test the proposed method on several other object detector architectures, including Faster-RCNN with a smaller ResNet-18 backbone, the lightweighted YOLOv8s [103], and a transformer-baed DINO-5scale [238]. Our implementation is primarily based on Detectron2 [222]. The code of YOLOv8s and DINO-5scale is ported from their corresponding official repositories [103, 237]. For Faster-RCNN with R-101 backbone base model, we use the same model and weights trained in section 3.5. For Faster-RCNN with R-18 backbone, the backbone ResNet-18 is initialized with ImageNet [37] pretrained weights. It is trained on COCO2017 training set for 270,000 iterations with batch size 4 and learning rate 0.0005. It is then finetuned on COCO2017 training set with remapped two object categories for 15,000 iterations with batch size 4 and learning rate 0.0003. For YOLOv8s, we take the COCO2017 pretrained weights from [103] and finetune it on the remapped COCO2017 training set for 428,262 iterations with batch size 16 and learning rate 0.01. And for DINO-5scale, we take the COCO2017 pretrained weights of DINO-5scale with R-50 backbone from [237] and finetune it on the remapped COCO2017 training set for 415,284 iterations with batch size 1 and learning rate 0.0001. We apply learning rate decay and examine the training curves to ensure convergence. Please note that for each detector architecture, the same base model is always used.

Table 4.1 shows the AP^m scores of the base detectors on both the COCO2017 validation set and Scenes100. For Faster-RCNN models, the larger R-101 backbone enables significantly better performance on COCO2017. YOLOv8s is considerately smaller, thus achieving AP^m more comparable to the smaller R-18 backbone. DINO-5scale, being the most computationally heavy, also achieves the highest AP^m. When comparing the performance on COCO2017 and Scenes100, Faster-RCNN and DINO-5scale models see a AP^m drop of more than 10 points. But AP^m of YOLOv8s only drops about 4 points. This is likely because the objects in Scenes100 are considerably smaller in scale compared to COCO2017. YOLOv8s is more accurate at detecting smaller objects, making it less prone to the data distribution shift. This also implies that YOLOv8s might see less improvement in detection precision from upscaling-based self-distillation discussed in subsection 4.3.4.



(A) Enhanced Faster-RCNN (B) Enhanced YOLOv8s (C) Enhanced DINO-5scale

FIGURE 4.4: Detailed network architecture of MoE enhanced YOLOv8s [103], Faster-RCNN [172], and DINO-5scale [238]. Not all modules are explicitly shown for clarity. (a) In Enhanced Faster-RCNN, the first residual block of the feature extractor, the region proposal network, and the last layer of the detection head are converted. (b) In enhanced YOLOv8s, the first Conv-C2f-Conv-Conv-C2f block and the final convolutional layers of each head are converted. (c) In enhanced DINO-5scale, the first residual block of the feature extractor, the class and bounding box embedding layers are converted. Though different object detectors require different modules to be enhanced, they are all the scene-specific initial and terminal layers described in subsection 4.3.2. In practice, the layers are empirically selected so the performance is satisfactory and the size of the MoE model is acceptable. For all models, the parameters in the selected layers all consist of $< 5\%$ of the total parameter count.

The detailed network architectures of the proposed MoE-enhanced models are shown in Figure 4.4. For B-Means, we use the backbone of the warmed up model to extract feature from images. Faster-RCNN based MoE models downsample the `p4` and `p5` feature maps in the feature pyramid to $256 \times 9 \times 16$ and $256 \times 5 \times 9$, then concatenate them together. This forms a 48,384-dimensional feature vector for each image. YOLOv8s based MoE models downsample the `p3` and `p4` feature maps from PANet to $256 \times 9 \times 16$ and $512 \times 5 \times 9$, then concatenate them together. This forms a 59,904-dimensional feature vector for each image. DINO-5scale based MoE models downsample the `r5` feature maps from R-50 backbone to $2048 \times 5 \times 9$. This forms a 92,160-dimensional feature vector for each image.

We use the default input resolution transformation of Detectron2, where an input image is resized to the largest possible resolution while maintaining its aspect ratio, as long as the shorter edge does not exceed L_1 and the longer edge does not exceed L_2. For the

default input scale (denoted as $\times 1$), $L_1 = 800$ and $L_2 = 1333$. "Upscaling" and "downscaling" refer to applying a multiplier on both L_1 and L_2. During adaptation training, after images are read with the default resolution (*i.e.*, $\times 1$), we apply $\times 2$ upscaling on the input of the teacher network to generate the pseudo labels to train the student network. For Faster-RCNN and YOLOv8s models, this is implemented by resizing the input images of the teacher network without other changes. For DINO-5scale, we instead split the image into 4 equally-sized patches (upper left, upper right, bottom left, and bottom right) and upscale each patch to the resolution of the original whole image. This is equivalent to $\times 2$ upscaling in terms of total image resolution. Each upscaled patch is fed into the teacher network, and the detected object bounding boxes are aggregated. We use this strategy because that for convolution-based detection heads in Faster-RCNN and YOLOv8s, the number of anchors scales with the input resolution. While in transformer based detection head as in DINO-5scale, the number of object queries is fixed for any input resolution. Our patching-based approach allows more queries to be used for an effectively higher-resolution image. In our experiments, we found that simple input upscaling on DINO-5scale does not bring much performance improvement.

We apply confidence score thresholding to keep more reliable detected objects as training labels for the student network. The score thresholds are set at $\theta = 0.5$ for Faster-RCNN, $\theta = 0.4$ for YOLOv8s, and $\theta = 0.3$ for DINO-5scale. All Faster-RCNN based MoE models are trained for 180,000 iterations with batch size 6 and learning rate 0.0001. That is 1,080,000 training samples in total. Training stage 1 and stage 2 are divided at iteration 100,000. YOLOv8s based MoE models are trained for 34,000 iterations with batch size 28 and learning rate 0.0001, which is 952,000 training samples in total. Training stage 1 and stage 2 are divided at iteration 24,000. DINO-5scale based MoE models are trained for 10,000 iterations with batch size 2 and learning rate 0.0001, which is 20,000 training samples in total. Training stage 1 and stage 2 are divided at iteration 8,000.

For Faster-RCNN based 1-for-1 models, the copies of the warmed up model are each trained for 800 iterations with batch size 6 and learning rate 0.0001. In total the same number of 1,080,000 training samples are used. For 1-for-1 YOLOv8s models, the copies of the warmed up model are each trained for 100 iterations with batch size 28 and learning rate 0.0001, which means the same number of 952,000 training samples are used in total. For 1-for-1 DINO-5scale models, the copies of the warmed up model are each trained for 20 iterations with batch size 2 and learning rate 0.0001, which means the same number of 20,000 training samples are used in total.

We compare our proposed framework with several domain-adaptive and scene-adaptive object detection methods as follows. Self-Train (**ST**) [176] uses detection and tracking to obtain pseudo bounding boxes, referred to as *DtTr*, for self-supervised adaptation. Cross-Teach (**CT**, proposed in section 3.3) further utilizes an ensemble of base detectors, and we refer to these pseudo labels as *EnDtTr*. Mid-Fusion with location-aware Mixup (**MFM**, proposed in section 3.3) exploits scene consistency from fixed scene cameras by modeling the background as an additional input modality and applies artifact-free object mixup for data augmentation. We directly use the AP^m numbers reported in the original paper for comparison. Geometric Shift (**GS**) [206] aims to correct the distortion from the camera perspective by learning a set of homography transforms. Learning to Zoom and Unzoom (**LZU**) [195, 196] is a differentiable plugin designed to zoom in on specific parts of the input image. **LODS** [119] is a source-free domain-adaptive object detection method in which a teacher network generates pseudo labels to train the student network. Other baselines [123, 185, 224, 246] are shown to perform poorly on Scenes100 in subsection 3.5.4, so we do not include results from them.

All domain adaptation baseline methods are trained as 1-for-1 models on individual scenes separately. This is because most of them rely on geometrical transforms or background modeling, which are scene-specific. The AP^m numbers of ST, CT, MFM, and LODS are directly taken from subsection 3.5.4. For GS, we reuse the code for warping and unwarping from the official repository and use the EnDtTr pseudo labels for adaptation. We adapt LZU to our self-supervised setting using the EnDtTr pseudo labels. The kernel density estimation-based method from [195] is utilized to generate the saliency map for each scene. GS and LZU are both trained on each scene for 20,000 iterations with batch size 4 and learning rate 0.0001. That is 80,000 training samples per scene, 8,000,000 in total. We monitor the training curves to ensure convergence for the baseline methods.

4.4.2 Comparison of Precision and Efficiency

We first compare the proposed adaptation method with the baseline methods in Table 4.2a. All models are adapted from the same Faster-RCNN R-101 base model for fair comparison. ST is trained on DtTr labels. CT, MFM, GS, and LZU are all trained on the same EnDtTr pseudo labels. ST, CT, and LODS cannot effectively improve the detection precision over the base model. GS and LZU both utilize geometric transforms, so the performance is higher. MFM introduces background extraction and mixup, which significantly

Method	Pseudo label	Training samples ↓	Deployment model size ↓	GFLOPs ↓	Relative latency ↓	AP^m ↑
Base model (no adaptation)	Not applicable		230MB	558	1.00	41.96
ST [176]	DtTr	8.00M	230MB×100	558	1.00	43.35
CT[chapter 3]	EnDtTr	8.00M	230MB×100	558	1.00	43.63
MFM[chapter 3]	EnDtTr	8.00M	230MB×100	987	1.75	45.74
GS [206]	EnDtTr	8.00M	231MB×100	1440	2.71	44.06
LZU [196]	EnDtTr	8.00M	230MB×100	558	1.20	44.06
LODS [119]	teacher	0.10M	230MB×100	558	1.00	42.98
Proposed (B=10)	×2	1.08M	259MB	558	1.01	50.27
Proposed (B=100)	×2	1.08M	547MB	558	1.00	**50.39**

(A) Comparison of the proposed method with other adaptation methods. All models are trained using the same Faster-RCNN base model with an R-101 backbone. The proposed models with B=10 and B=100 perform similarly, outperforming other methods by a wide margin. It does not increase computational costs or consume significantly more memory, as some baselines do.

TABLE 4.2: Comparison of different adaptation methods in terms of detection performance on Scenes100 and computational cost in terms of giga floating-point operations (GFLOPs) and inference latency. Pseudo label column shows the labels used in adaptation, being either *DtTr* from [176], *EnDtTr* from section 3.3, *teacher* labels from a teacher network, or ×2 meaning using upscaled image for the base model to generate pseudo labels for self-adaptation. All models take images without upscaling at inference time. Training samples column shows the total number of images seen by the models during adaptation. It is the sum of all individual models in the case of individual adaptation. Please refer to subsection 4.4.1 and subsection 4.4.2 for more details on evaluation metrics, latency measurement, and input scale.

improve the performance. The proposed B=10 model uses two-stage training with B-Means, and the B=100 model uses two-stage training with 1-to-1 gating. Both models achieve similar detection precision after adaptation, which is significantly higher than all baselines. All the baseline methods adapt to individual scenes by creating a separate model for each scene (1-for-1 approach). This strategy results in significant memory consumption and requires a large number of training samples (except for LODS) for each model to achieve convergence. In contrast, the proposed method, being both data and computation efficient, has a significantly smaller overall model size, requiring much fewer training samples. At inference time, MFM uses two backbone passes. GS and LZU introduce additional geometrical transforms or backbone passes. The increased computational cost is reflected in increased GFLOPs and latency. The proposed method can maintain the computational cost of the base model.

Base model	Method	Pseudo label	Training samples ↓	Deployment model size ↓	GFLOPs ↓	Relative latency ↓	APD^m ↑
Faster-RCNN with R-101 backbone	1-for-100	×2	1.08M	230MB	558	1.00	0
	1-for-100 (no adapt)	Not applicable		230MB	558	1.00	−7.70
	1-for-1×100	×2	1.08M	230MB×100	558	1.00	0.42
	Proposed (B=10)	×2	1.08M	259MB	558	1.01	0.61
	Proposed (B=100)	×2	1.08M	547MB	558	1.00	**0.73**
Faster-RCNN with R-18 backbone	1-for-100	×2	1.08M	107MB	310	0.54	0
	1-for-100 (no adapt)	Not applicable		107MB	310	0.54	−8.96
	1-for-1×100	×2	1.08M	107MB×100	310	0.54	0.37
	Proposed (B=10)	×2	1.08M	134MB	310	0.54	**0.90**
	Proposed (B=100)	×2	1.08M	397MB	310	0.54	0.69
YOLOv8s with D-53 backbone	1-for-100	×2	0.95M	43MB	73	0.22	0
	1-for-100 (no adapt)	Not applicable		43MB	73	0.22	−1.63
	1-for-1×100	×2	0.95M	43MB×100	73	0.22	**0.98**
	Proposed (B=10)	×2	0.95M	55MB	73	0.23	0.73
	Proposed (B=100)	×2	0.95M	174MB	73	0.24	0.75
DINO-5scale with R-50 backbone	1-for-100	×2	0.02M	181MB	1620	5.58	0
	1-for-100 (no adapt)	Not applicable		181MB	1620	5.58	−8.96
	1-for-1×100	×2	0.02M	181MB×100	1620	5.58	1.34
	Proposed (B=10)	×2	0.02M	198MB	1620	5.71	**2.45**
	Proposed (B=100)	×2	0.02M	373MB	1620	5.85	1.95

(B) Comparison of the proposed method with the baselines of adapting a single generic *1-for-100* model to all scenes and creating a scene-specific *1-for-1* model for each scene for adaptation. State-of-the-art detector architectures (Faster-RCNN [172], YOLOv8s [103], and DINO-5scale [238]) are tested. APD^m represents the difference in mean average precision compared to adapted 1-for-100 model with the same detector architecture. The proposed method achieves similar or better performance compared to 1-for-1×100 models, while consuming significantly less memory and maintaining the same computational cost as the base model.

TABLE 4.2: (Continued) Comparison of different adaptation methods in terms of detection performance on Scenes100 and computational cost in terms of giga floating-point operations (GFLOPs) and inference latency. Pseudo label column shows the labels used in adaptation, being either *DtTr* from [176], *EnDtTr* from section 3.3, *teacher* labels from a teacher network, or ×2 meaning using upscaled image for the base model to generate pseudo labels for self-adaptation. All models take images without upscaling at inference time. Training samples column shows the total number of images seen by the models during adaptation. It is the sum of all individual models in the case of individual adaptation. Please refer to subsection 4.4.1 and subsection 4.4.2 for more details on evaluation metrics, latency measurement, and input scale.

The adaptation performance and computational cost of the proposed method on different detector architectures are compared in Table 4.2b. The details of how the computational cost is measured are described and discussed in section 4.8. We compare with

the case when we use a single generic model to adapt to all 100 scenes (*1-for-100*). We calculate the difference between detection AP^m values of an adapted model with this model, which we refer to as APD^m. We also compare the adaptation performance of using 100 scene-specific *1-for-1* models to adapt to each scene, which can be executed when computational resources are unlimited. For a fair comparison, we also apply the two-stage training schedule for 1-for-1 models. Since different architectures have very different base model performances (Table 4.1), the comparison is only meaningful among the models adapted from the same base model.

We again verify that the proposed MoE models do not incur additional neural computation. They have the same GFLOPs and nearly identical inference latency as the corresponding base model, regardless of B. The gating module introduces minimal overhead. Since only a small portion of the network parameters are enhanced, the MoE models are only moderately larger than the base models, resulting in a modest increase in memory consumption at inference time. Though scene-specific 1-for-1 models do not introduce additional computational cost either, for the inference engine to serve all the scenes simultaneously, it needs to host all 100 copies of the network in memory, which is impractical.

When comparing the effectiveness of adaptation, our proposed framework can produce an MoE model with precision similar to or even higher than the scene-specific 1-for-1 models. For Faster-RCNN and DINO-5scale, the proposed model outperforms 1-for-1 models. This suggests that larger models are more prone to overfitting, so sharing certain network parameters among different scenes can be beneficial. In Figure 4.5, we visually demonstrate that the proposed method can obtain models that have increased detection precision over the base model while maintaining the computation cost. We can further reduce the latency by using downscaled input images for the adapted MoE models. Consequently, the models can still obtain improved or similar detection precision over the base models with reduced latency.

4.4.3 Ablation Study on MoE Layers, Training Schedules, and Gating Strategies

The proposed method converts initial and terminal layers to MoE as described in subsection 4.3.2, which we claim to be beneficial for scene-specific adaptation and is shown to outperform the single 1-for-100 model. Here, we instead use intermediate layers to

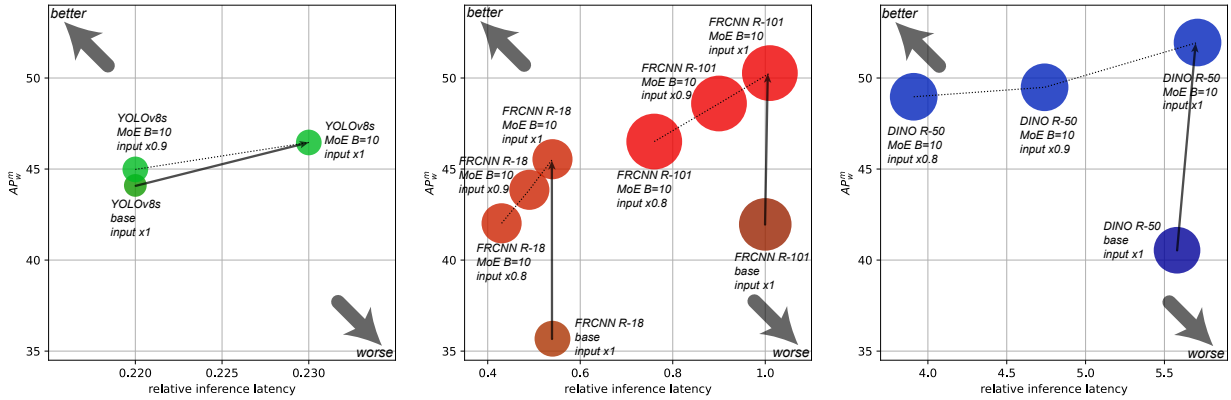
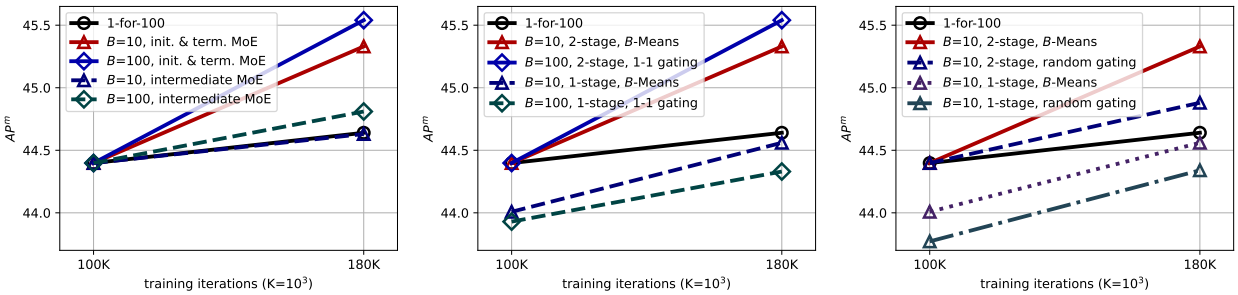


FIGURE 4.5: AP^m and latency of B=10 MoE models for different detector architectures at different input scales. The size of each model is indicated by the area of the corresponding circle. Sub-figures show the same limits for the vertical axis (AP^m) but different limits for the horizontal axis (relative inference latency). For all architectures, our framework can improve the base models in detection precision and efficiency at the same time, while keeping similar memory consumption.



(A) MoE layers (B) Schedules (C) Gating strategies

FIGURE 4.6: Ablation study on different MoE layers, training schedules, and gating strategies. All models are based on Faster-RCNN with R-18 backbone, trained with the same batch size and learning rate. AP^m at training iterations 100,000 and 180,000 are shown. All two-stage trained models start from the warmed up 1-for-100 model at iteration 100,000. In (a) B=10 models use B-Means gating and B=100 models use 1-to-1 gating. In (b) and (c), all B=10 and B=100 models use proposed MoE layers enhancement. The proposed MoE layers, two-stage training schedule, and the B-Means gating strategy are all beneficial over other alternatives.

enhance with MoE. For a fair comparison, we keep the number of parameters of the converted intermediate layer similar to the proposed method, all models start from the same warmed up model, and all use the same gating strategy. The adaptation performance is

compared in Figure 4.6a. It is clear that unlike the proposed method, using intermediate layers for MoE does not provide much performance gain over the non-MoE 1-for-100 model.

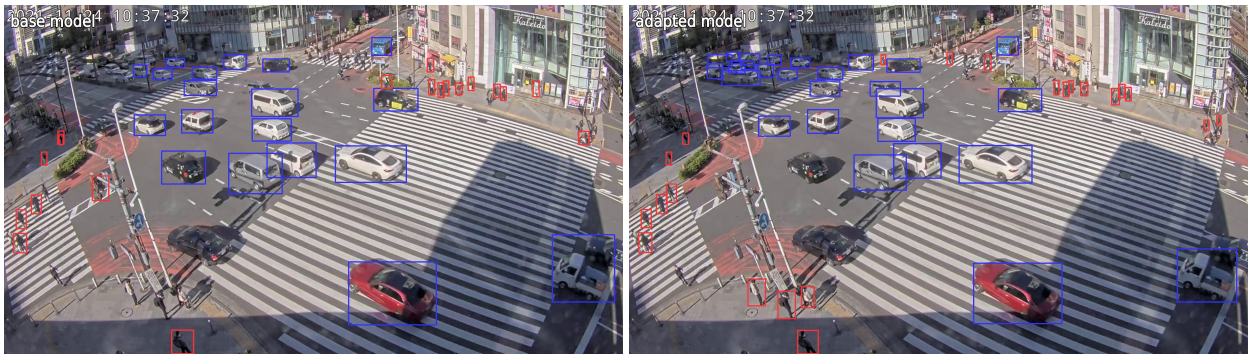
We also conduct experiments using the one-stage training schedule, where the MoE model is constructed, and the gating rules are determined from the base model without warm-up. These results are presented in Figure 4.6b. It is evident that if the model is branched directly from the base model without a warm-up phase, B=10 and B=100 models actually underperform compared to the generic 1-for-100 adaptation model, suggesting overfitting. Two-stage training results in improved performance with the same number of training samples.

We further compare different gating strategies of B=10 MoE models. One of which utilizes the proposed B-Means gating, and the other adopts a random gating strategy where an MoE branch is randomly assigned to each scene. The results are presented in Figure 4.6c. Only the B-Means approach achieves an AP^m comparable to that of the B=100 models (as reported in Table 4.2b), demonstrating the effectiveness of grouping similar scenes into the same branch. Even for one-stage trained models, B-Means is beneficial over random gating.

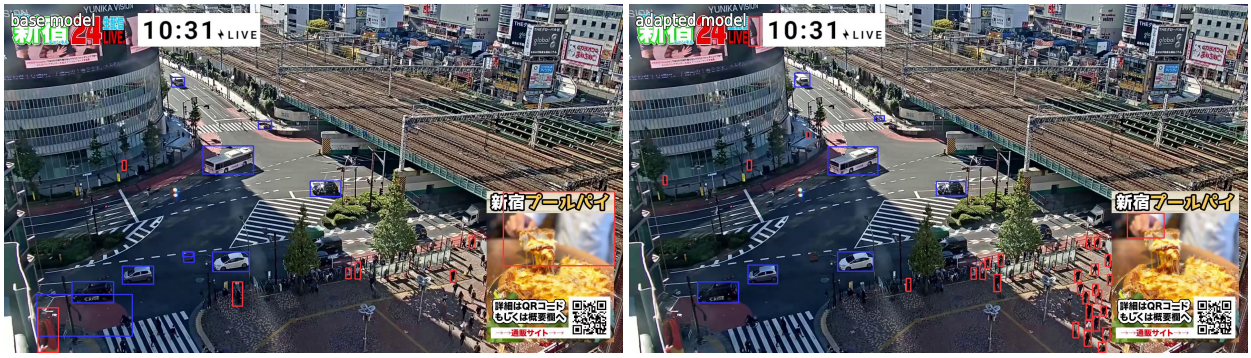
4.5 Case Study and Qualitative Results

In Figure 4.7a−4.7p, we visualize how the detection performance of the adapted MoE model differs from the base model by showing the detection bounding boxes on some images from Scenes100 evaluation set. We show the detected object bounding boxes with a score higher than θ^*, where θ^* is the score threshold that maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. It is clear that after adaptation, the detector can identify the objects that are blurry or small, which are missed by the base model.

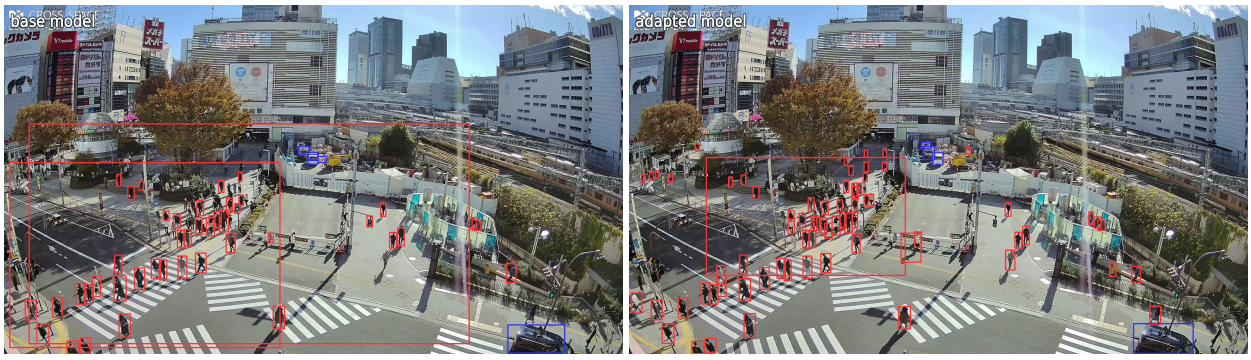
In Figure 4.7q, 4.7r, and 4.7s, we also look into scenes `172`, `159`, and `114`, where the adaptation brings no benefit. These scenes are from cameras in very dark or strongly backlit conditions. Such challenging conditions can be amplified by upscaling, potentially leading to extremely noisy pseudo labels. Hence, the self-distillation training does not improve detection precision.



(A) 006

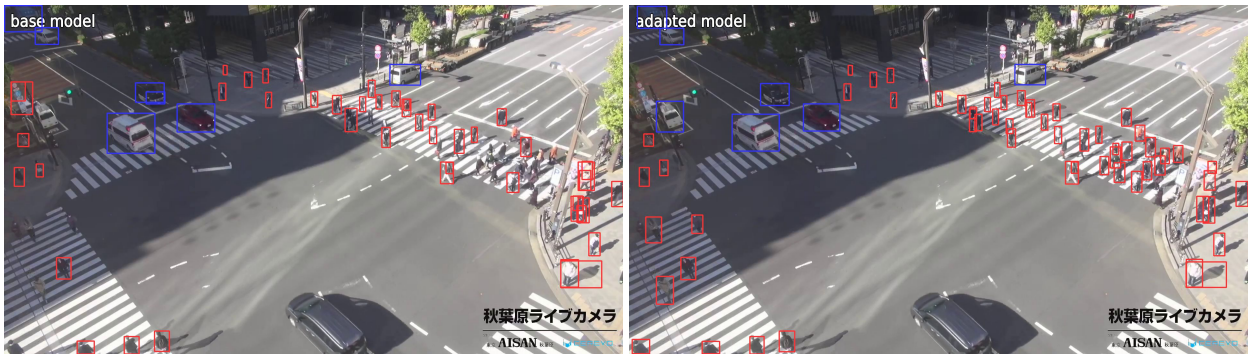


(B) 007

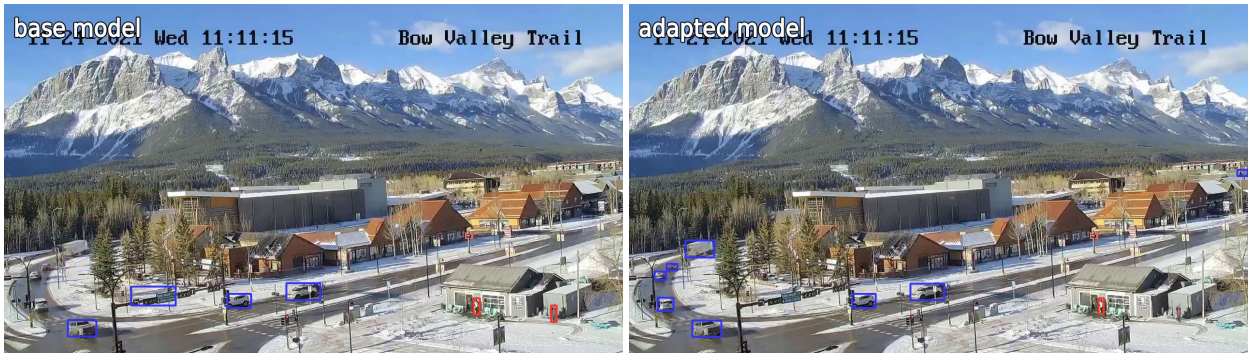


(C) 008

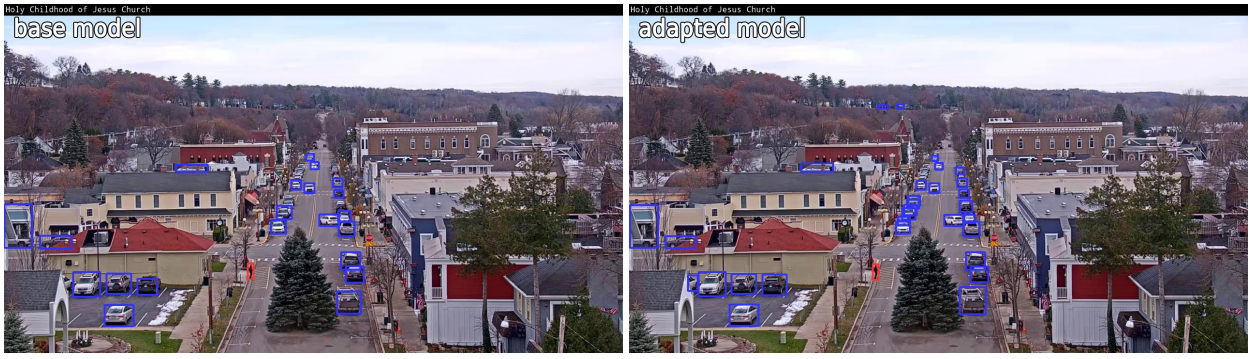
FIGURE 4.7: Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.



(D) 009



(E) 011

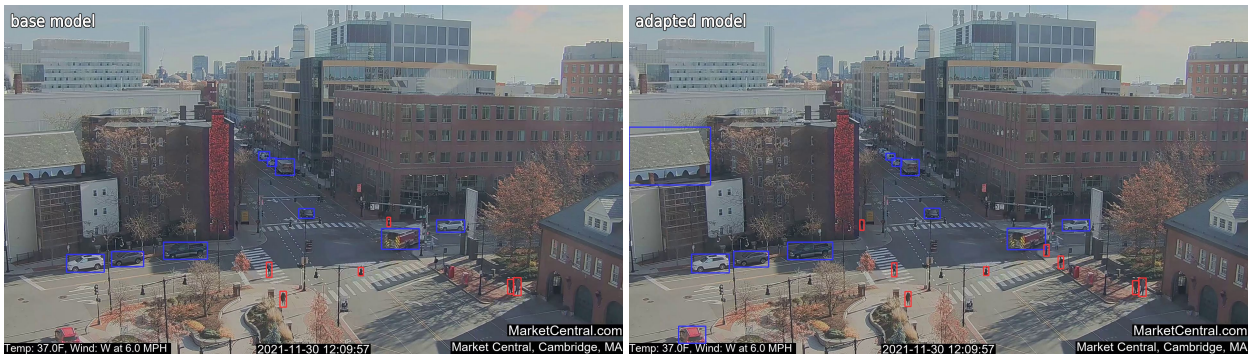


(F) 015

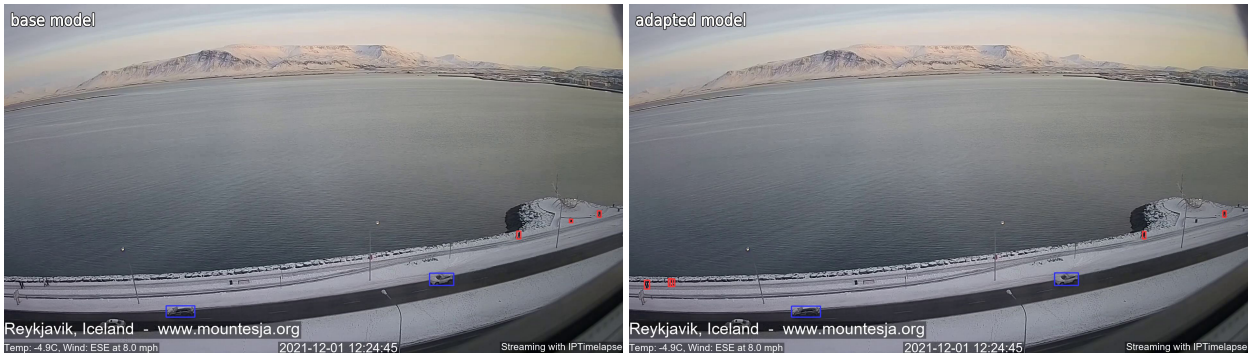
FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

4.6 Adaptation Using Alternative Labels

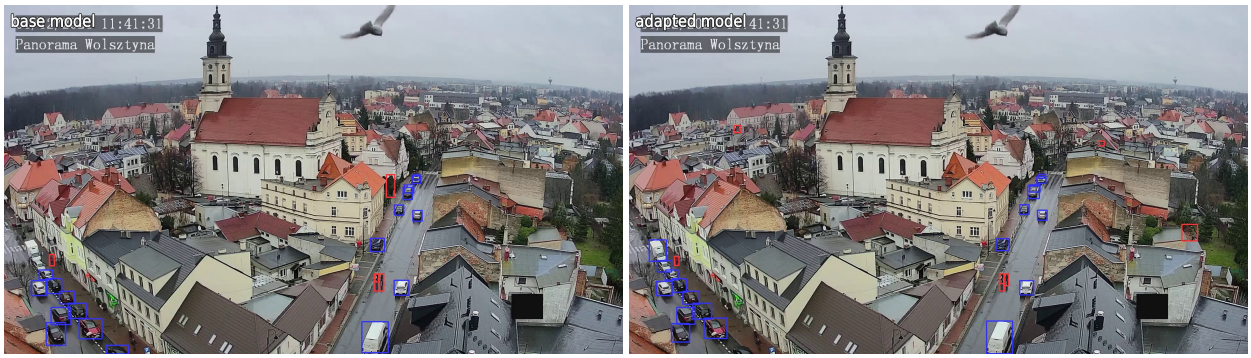
We already show that different detection architectures have very different levels of base model performance and also different magnitudes of improvement from adaptation. We



(G) 058



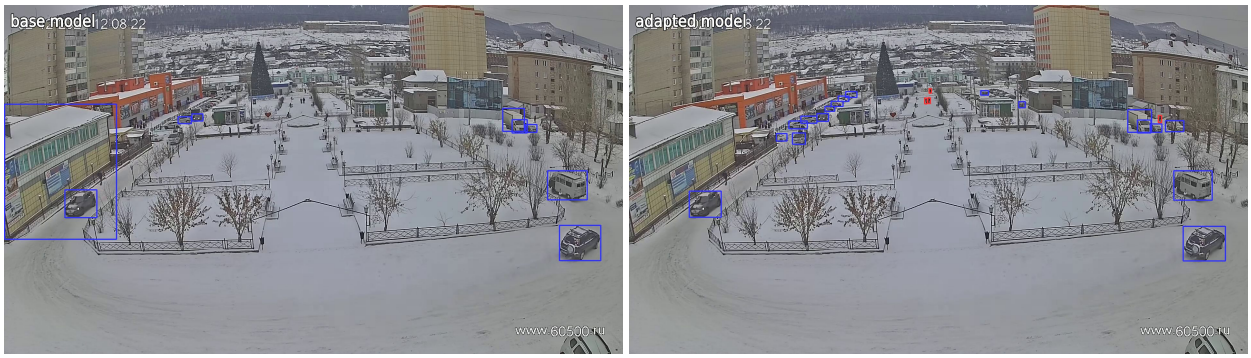
(H) 077



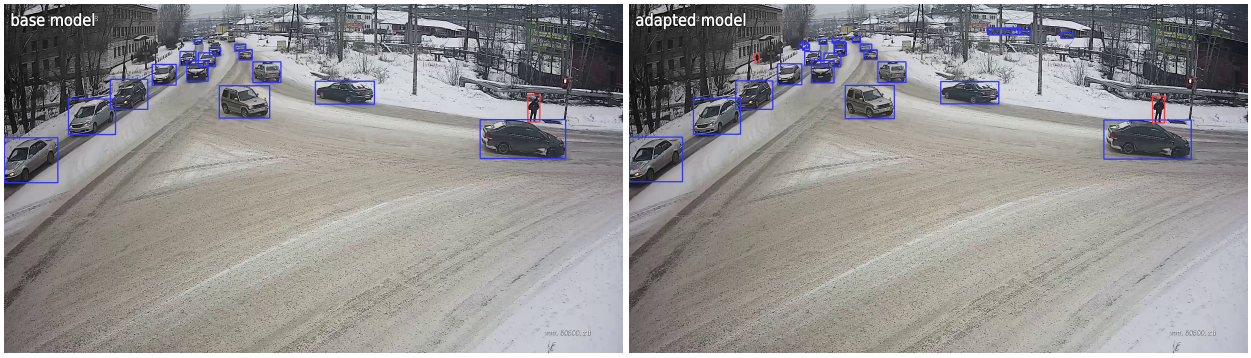
(I) 080

FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

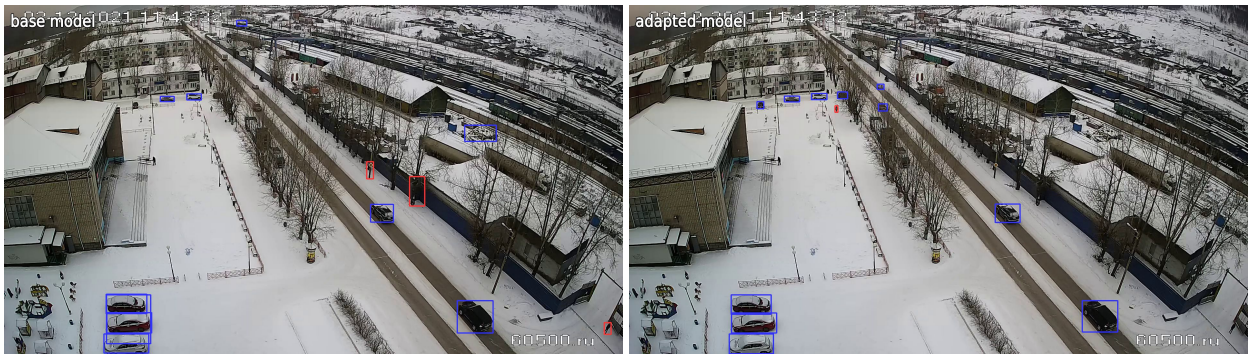
further investigate the upper limit of the detection performance of different architectures on Scenes100 using *supervised* adaptation. Specifically, we sort the evaluation images of



(J) 090



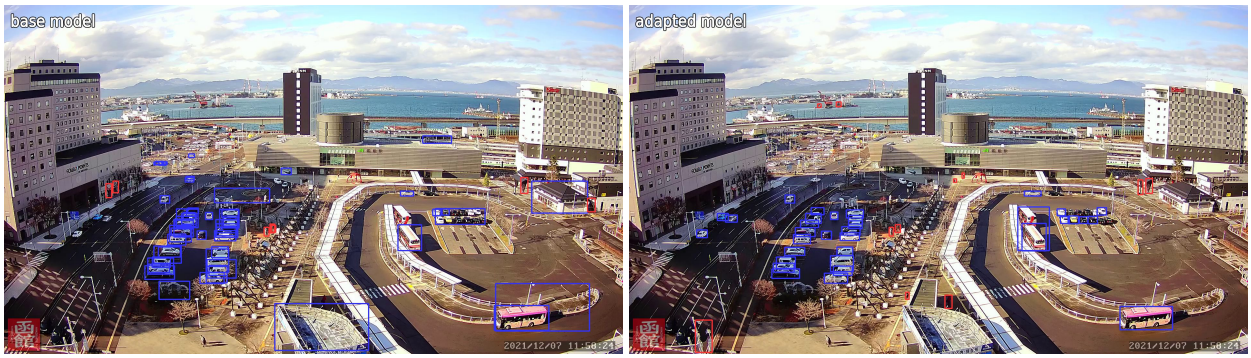
(K) 091



(L) 092

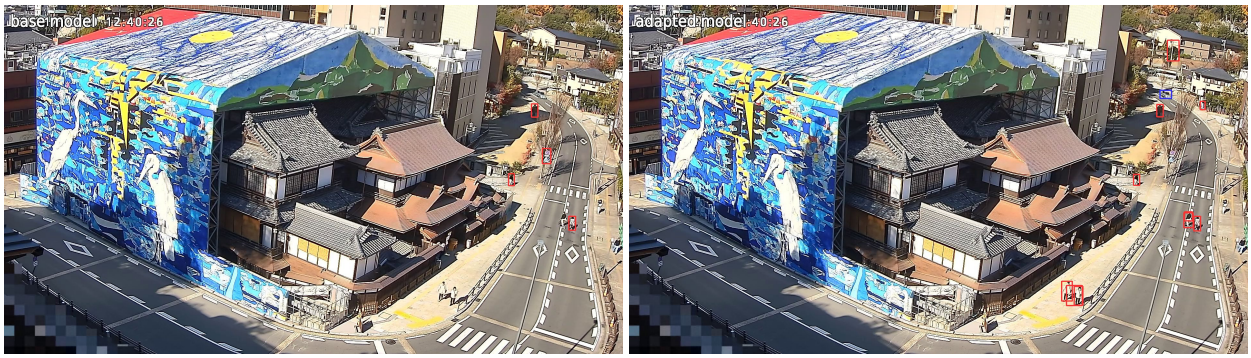
FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

each video in Scenes100 in temporal order and use the first $1/3$ as the training split and the remaining $2/3$ as the evaluation split. We refer to them as *Scenes100-oracle-train* and



(M) 131

(N) 154



(O) 156

FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

Scenes100-oracle-val, respectively. The base detectors are finetuned on Scenes100-oracle-train using the human-annotated ground truth bounding boxes and are evaluated on

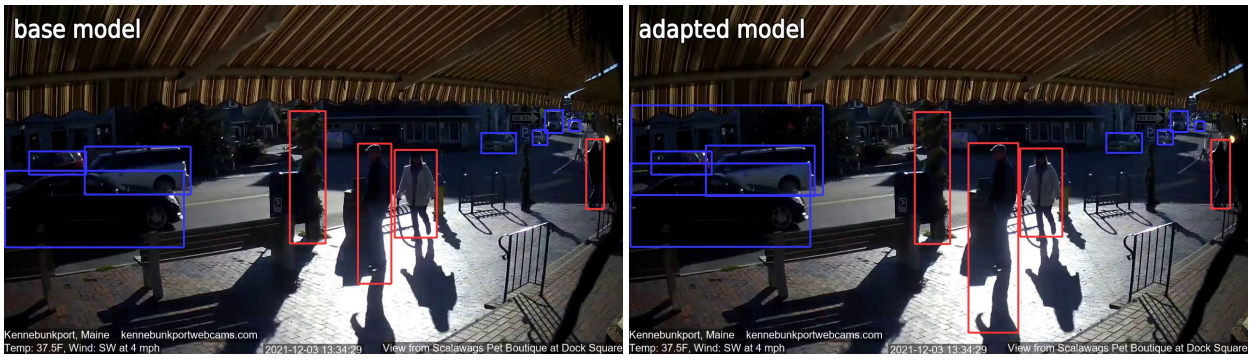
(P) 170

FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

Scenes100-oracle-val. Please note that one model adapts to all scenes (1-for-100). This gives an estimate of the upper limit of detection precision a detector architecture can achieve on Scenes100. This limit is determined by both the capacity of the model and the data distribution of Scenes100. For comparison, we also evaluate the AP^m of the proposed MoE models on Scenes100-oracle-val. Those models all use self-supervised adaptation only. The results are compared in Figure 4.8.

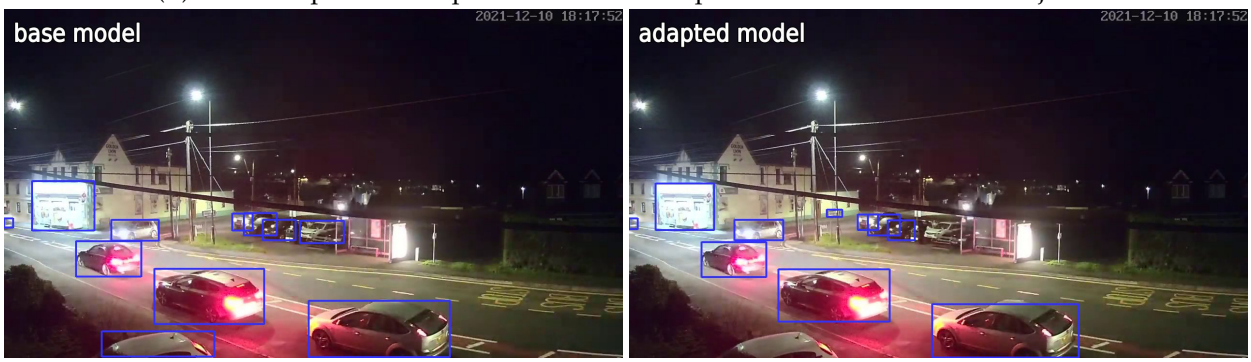
For the same Faster-RCNN architecture, a larger backbone leads to higher precision, likely due to its higher model capacity. Across different architectures, the precision of the oracle model differs greatly, even when they are trained using the same data (COCO2017 training set and Scenes100-oracle-train). This indicates that for a specific data distribution as in Scenes100, certain detector architectures are more suitable than others. YOLOv8s obtains the highest AP^m, which also corresponds to the observation we make that it is more "suitable" for Scenes100. This means YOLOv8s base model already obtains high AP^m using $\times 1$ input, so upscaling is not very beneficial and thus cannot provide high-quality pseudo labels. It also implies that for different detection architectures different types of pseudo labels might be necessary. Again, please note that the focus of the proposed method is efficiency, and it can work with different types of pseudo labels.

In addition to pseudo labels generated by the base model with upscaled input images, we also conduct experiments with EnDtTr pseudo labels as in CT to train our MoE



(Q) 114. Adapted model produces more false positives.

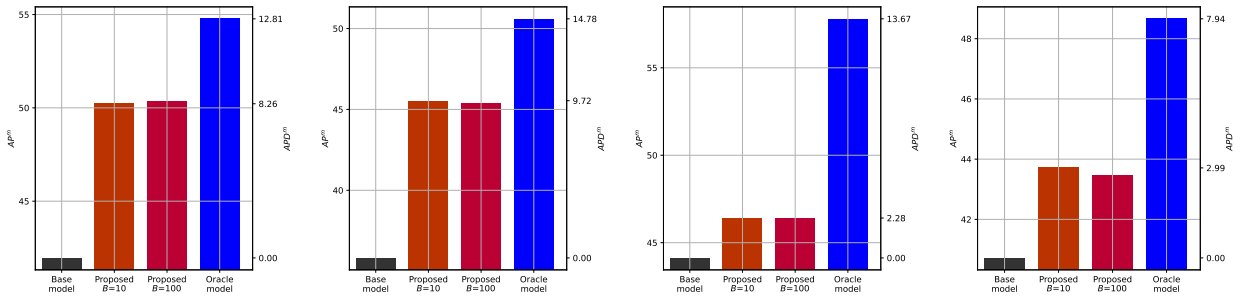
(R) 159. Adapted model produces more false positives and misses more objects.



(S) 172. Adapted model misses more objects.

FIGURE 4.7: (Continued) Detection performance on individual scenes in Scenes100. The results are from the base model and adapted B=10 MoE model using Faster-RCNN with R-101 backbone. Detected *person* and *vehicle* are labeled by red and blue bounding boxes, respectively. We show the detected objects with a score higher than θ^*, where θ^* maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol. These figures are best viewed on screen and when zoomed in.

detectors. For these experiments, we use Faster-RCNN with an R-101 backbone to maintain consistency. The results are shown in Table 4.3. The MoE models with B=10 and



(A) Faster-RCNN R-101 (B) Faster-RCNN R-18 (C) YOLOv8s D-53 (D) DINO-5scale R-50

FIGURE 4.8: Detection AP^m on Scenes100-oracle-val of different detector architectures, which is the last $2/3$ of the validation set of Scenes100. Additionally, the APD^m over the base model is shown on the secondary vertical axis. The oracle models are trained on the first $1/3$ of the validation set, which we refer to as Scenes100-oracle-train.

Method	Model size ↓	APD^m ↑
1-for-100	230MB	0
1-for-100 (no adapt)	230MB	-1.12
1-for-1$\times$100 CT	230MB$\times$100	**0.55**
Proposed (B=10)	259MB	0.54
Proposed (B=100)	547MB	0.36

TABLE 4.3: Adaptation performance of the proposed MoE models trained on pseudo labels *EnDtTr* compared with CT. All models are adaptively trained from the same base Faster-RCNN with R-101 backbone for consistency with CT. Similarly, the proposed method achieves similar performance as CT while consuming much less memory.

B=100 achieve comparable performance to the individually trained CT models. This suggests that the proposed framework is versatile and can work with different types of noisy pseudo labels, while higher quality ones lead to better adaptation performance.

4.7 Ablation Study on Updating Base Detector

In our proposed method, the base teacher model for generating pseudo labels is fixed throughout the self-distillation process. Here, we additionally explore the idea of updating the base model during training, as used by many domain-adaptive object detectors

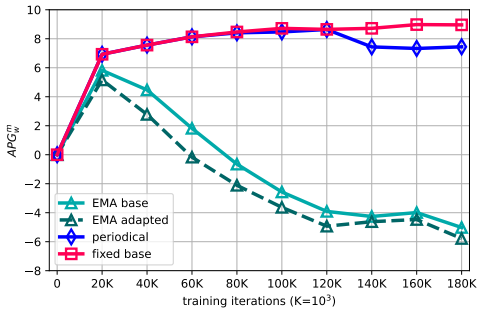


FIGURE 4.9: The detection performance of applying periodical replacement and EMA, compared with not updating the base model. For periodical replacement, we show the performance of the adapted model. For EMA, both the teacher model and the student model are evaluated. Experiments are conducted on 1-for-100 Faster-RCNN with R-18 backbone. Updating the teacher causes the model's performance to degrade.

[22, 119, 123]. For the proposed MoE model, the base model and adapted model have different architectures. Hence, all the experiments here are conducted with 1-for-100 scene-generic models. The base teacher model is updated during training using 2 strategies. **(1) Periodical replacement**: At $1/3$ and $2/3$ of the total training iterations, replace the base teacher model with the current student. And **(2) exponential moving average (EMA)**: gradually updating the weights of the teacher model using the weights of the adapted student model:

$$\boldsymbol{w}_{teacher} \leftarrow \eta \boldsymbol{w}_{teacher} + (1-\eta)\boldsymbol{w}_{student}.$$

We take the common practice of setting $\eta = 0.9999$.

All methods use the same total number of iterations, image batch size, and learning rate. AP^m of the models are evaluated periodically and shown in Figure 4.9. We can observe that at the beginning of the training process, all methods can improve the detection performance effectively. However, updating the teacher will eventually cause the performance to degrade. This implies overfitting and even model collapse. Besides, both periodical replacement and EMA introduce additional hyper-parameters, namely the replacement period and η, which need careful tuning (*i.e.* we find that using $\eta = 0.999$ instead of 0.9999 for EMA causes total model collapse, which leads AP^m to near 0). We conclude that updating the teacher during training in our proposed method is not beneficial. Based on the same reason, we do not use the adapted MoE models to generate pseudo labels during training.

4.8 Detailed Model Efficiency Analysis

We measure the giga floating-point operations (GFLOPs) of neural network computation at inference time of different methods using the `calflops` tool [230]. For all the models, we use the same input image so the numbers are comparable. Unlike GFLOPs which depends only on network architecture and input data, the latency of a model is also related to the hardware and software it runs on. It is impossible to cover all the hardware and software configurations used in real-world applications. Therefore, we test the inference latency of different models on three representative machines with the configurations listed in Table 4.4a. #1 represents up-to-date high-end servers. #2 represents mid-range hardware that was purchased a few years ago. And #3 configuration represents old servers that are out-of-date but still widely used. #3 also has comparable computational power to some of today's mobile devices, such as laptops and smartphones.

The inference latency is measured in milliseconds (ms). The batch size is set to 1 to simulate the on-demand model serving scenario. All models use 32-bit floating point weights. The input images are loaded into RAM and pre-processed before the measurement, so the results reflect mostly the inference speed of the neural network. We measure the average latency from 300 inference calls. For the same model, we use the same checkpoint and images on different machines. All experiments are run 3 times, and the results are averaged.

The latency measurements are shown in Table 4.4b, 4.4c, and 4.4d. The impact of different methods on latency is mostly consistent in different configurations. For all models, the proposed MoE method has very similar latency as the base model. Nvidia GTX 970 only has 4 GB of memory, but the B=100 MoE model based on Faster-RCNN with R-101 backbone can still run in full 32-bit precision, indicating the memory efficiency of the proposed MoE model.

In practice, batching can be used by inference servers to increase throughput, as a single image may saturate the full computational power of the hardware. In the proposed MoE detector, the model can encounter images from different scenes in a batch, which means each of the images needs to take a different route. This can induce additional computational overhead. Here we compare how the inference latency changes with different batch sizes, for both the base detector and the MoE detector. We measure the latency with different batch sizes. For each batch size, we make sure each image is from a *different*

	Config. #1	Config. #2	Config. #3
CPU	Intel Core i7 13700K (2022)	Intel Xeon Gold 6126 (2017)	Intel Xeon E5 1650v3 (2014)
GPU	Nvidia RTX 4090 (2022)	Nvidia RTX 2080 Ti (2018)	Nvidia GTX 970 (2014)
OS	Ubuntu 22.04	Ubuntu 18.04	Ubuntu 14.04

(A) Different configurations used for measuring inference latency. The release years of the hardware are shown for reference. All machines run Detectron2 v0.6 with PyTorch 1.10 and CUDA 11.3.

Method	Backbone	Config. #1		Config. #2		Config. #3		Average
		ms	rel.	ms	rel.	ms	rel.	rel. latency
Faster-RCNN	R-101	27.00	1.00	63.82	1.00	235.29	1.00	1.00
	R-18	16.24	0.60	32.92	0.52	115.87	0.49	0.54
MFM[chapter 3]	R-101	44.60	1.65	117.78	1.85	411.52	1.75	1.75
GS [206]	R-101	71.38	2.64	178.57	2.80	632.91	2.69	2.71
LZU [196]	R-101	36.62	1.36	73.15	1.15	255.00	1.08	1.20
Proposed (B=10)	R-101	27.38	1.01	64.10	1.00	236.99	1.01	1.01
Proposed (B=100)	R-101	27.04	1.00	63.69	1.00	236.41	1.00	1.00
Proposed (B=10)	R-18	16.20	0.60	32.96	0.52	117.51	0.50	0.54
Proposed (B=100)	R-18	16.37	0.61	33.03	0.52	116.96	0.50	0.54

(B) Latency measurements of Faster-RCNN [172] based models.

Method	Backbone	Config. #1		Config. #2		Config. #3		Average
		ms	rel.	ms	rel.	ms	rel.	rel. latency
YOLOv8s	D-53	7.58	0.28	13.52	0.21	36.99	0.16	0.22
Proposed (B=10)	D-53	8.42	0.31	13.30	0.21	38.22	0.16	0.23
Proposed (B=100)	D-53	9.38	0.35	13.92	0.22	38.23	0.16	0.24

(C) Latency measurements of YOLOv8s [103] based models.

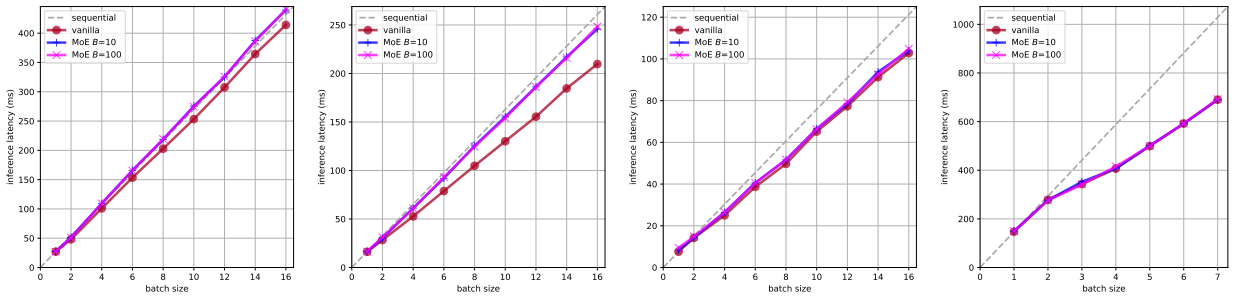
Method	Backbone	Config. #1		Config. #2		Config. #3		Average
		ms	rel.	ms	rel.	ms	rel.	rel. latency
DINO-5scale	R-50	147.01	5.44	365.52	5.73	–	–	5.58
Proposed (B=10)	R-50	149.83	5.55	374.87	5.87	–	–	5.71
Proposed (B=100)	R-50	149.80	5.55	392.49	6.15	–	–	5.85

(D) Latency measurements of DINO-5scale [238] based models. DINO-5scale models cannot run correctly on Config. #3 due to its outdated hardware.

TABLE 4.4: Latency measurements on different configurations. We measure the inference latency by milliseconds using a single image. *rel.* stands for latency relative to Faster-RCNN with R-101 backbone and $\times 1$ input on a configuration. The overall average relative latency is the average over different configurations.

scene to simulate the worst-case scenario and to incur maximum overhead. The experiments are run on the config. #1 machine since it is the most powerful one and the least likely to be saturated. This means that the difference in the latency with different batch sizes is also the most obvious. The results are shown in Figure 4.10. In each plot, we add

the dashed diagonal line which represents the latency when all images in a batch are fed to the base model sequentially instead of in one batch. So if the latency *vs.* batch size curve is under the line, it means batching can bring efficiency benefits.



(A) Faster-RCNN R-101 (B) Faster-RCNN R-18 (C) YOLOv8s D-53 (D) DINO-5scale R-50

FIGURE 4.10: Inference latency of different models at different batch sizes. We measure the latency on the machine config. #1 in terms of milliseconds. In each plot, the dashed diagonal line represents the latency when all images in a batch are fed to the base model sequentially instead of in one batch.

For models based on Faster-RCNN with R-101 backbone, the latency *vs.* batch size curve is very close to the sequential inference line, meaning that batching is not more efficient than processing each image sequentially. Inference with a single image can already utilize almost all the computational power of the GPU. In other models, the benefit of batching is more significant. For Faster-RCNN with R-18 backbone, it can be observed that with a larger batch size, the overhead from gating becomes more obvious. Since we put images from different scenes in each batch, each of the images needs to take a different path, which makes the inference less efficient. Still, the value of B does not have any effect on the latency.

Please note that our implementation of the gating mechanism is preliminary. In real-world vision systems, a model is usually optimized before deployment. Those optimizations are specific to many factors, such as the software library used, the network IO condition, and hardware architecture. Our experiment here is just a simplified scenario that aims to give an insight into how an MoE model performs in different settings.

4.9 Summary

In this chapter, we have presented a novel framework that adapts a base object detector to video streams from various scene cameras without affecting inference speed or system

throughput. By incorporating a mixture-of-experts structure into the base network's architecture, we have achieved an enhanced network that preserves inference latency and memory usage. This architecture can be trained with pseudo labels generated by the base detector itself, enabling self-supervised learning and eliminating the need for human supervision during the adaptation process. Additionally, using downscaled input images with the mixture-of-experts model can improve precision and reduce latency, thus optimizing both critical objectives simultaneously. Our approach has been tested across various state-of-the-art detection network architectures, outperforming the baselines in detection precision, memory consumption, inference latency, and data efficiency.

Chapter 5

Prompt Adaptation for Open-Vocabulary Object Detection

In previous chapters I present how to adapt trained computer vision models to new tasks or datasets. It has been assumed that the whole model is available to the end users, and the parameters of the model can be updated through gradient descent. However, this assumption does not always hold, especially for large multi-modality models that is optimized or run as a service. Open-vocabulary object detector (OVD) models can have such constraints. However, in cases that the target objects are diverse and novel, OVDs are still preferred, as they have the ability of detecting objects based on any input textual prompt. The detection performance of OVDs is sensitive to the prompt input, so it is possible to improve their performance on specific videos by using the optimal prompt without changing the model parameters. However, optimizing these textual prompts can be challenging, often involving a laborious and suboptimal trial-and-error process. In this chapter, I present a data-driven approach featuring an annotation-and-parameter-efficient framework for prompt optimization. Specifically, I propose to use a lightweight prompt enhancement module for adaptation. This module learns to enrich the features of an initial prompt, guiding the detector to better identify the objects of interest using a limited number of images. This method is efficient and achieves significant precision improvements. Experiments with GroundingDINO on several video object detection datasets demonstrate the effectiveness of our proposed method. The content of this chapter is based on the submitted manuscript [245], and "we" in this chapter refers to its authors. The code is available at `https://github.com/cvlab-stonybrook/PromptAdaptOVD`. The experiments in this chapter also use the EgoPER dataset proposed by the published paper [112].

5.1 Introduction

The need to detect objects in images and videos arises in many applications. A typical approach involves using an object detector that has been trained for a predefined set of categories of objects of interest. However, this approach often falls short in various situations. Sometimes, the objects we need to detect may not belong to the existing categories with well-trained detectors. In other cases, even if a detector exists for the desired categories, it may be too general and not specific enough for the objects in the particular videos or scenes being analyzed. This can lead to suboptimal performance. For instance, a person detector trained mostly on indoor images might not be suitable for a street scene in a developing Asian country, where most people are riding bicycles or motorbikes. Therefore, relying on object detectors for a predefined set of categories can be insufficient, as they may lack the ability to detect certain objects or adapt to specific aspects of a category.

Fortunately, OVDs have made significant advancements recently, thanks to multi-modal transformer models that can perform cross-attention between text and visual elements and the availability of large-scale multi-modal datasets to train these models. OVDs are not restricted to a predefined list of object categories. Instead, a text prompt is used to specify the type of objects to detect in an image. This text prompt can range from very general words like `"vehicle"`, to more specific phrases such as `"red double-decker bus"`. This flexibility allows OVDs to detect many different and even novel types of objects.

However, flexibility has its own downside, as determining the optimal prompt for a given application scenario can be challenging. For example, consider a computer vision application that involves assessing a trainee doctor's skills in surgery, which requires detecting surgical tools. Should we use the generic term `"surgical tool"`, or the list of specific tools such as `"retractors . scalpel . forceps"`? We could go even more specific with terms like `"Adson forceps"` or `"Silver Adson forceps"`. We can use category names, alternative names, super-category names, or a list of sub-categories, and there are exponentially many ways to combine them using negation and concatenation. Different people might also describe the same object differently due to variations in language proficiency and word preference. Additionally, different detection models can have inherent biases, further complicating the task of finding the optimal prompt. Given the flexibility and ambiguous nature of human language, iterating through all possible prompts to find the best one is typically not feasible.

Given the time-consuming and suboptimal nature of the trial-and-error process in manual prompt engineering, we propose an alternative approach for finding effective prompts for open-vocabulary object detection. Our premise is that it is possible to learn to provide the OVD model with the appropriate prompt in a data-driven manner. Specifically, we introduce a lightweight prompt enhancer framework. It takes an initial prompt, which can be vague and suboptimal, and learns to improve it from a limited number of images that have target objects labeled. It enables an open-vocabulary detector to adapt to unseen data without changing the parameters of the detector. Our method is more parameter-efficient and less prone to overfitting than fine-tuning the parameters of the whole detector. Experiments using the state-of-the-art open-vocabulary detector GroundingDINO on several video object detection datasets demonstrate the effectiveness and efficiency of the proposed framework.

5.2 Related Work

5.2.1 Open-Vocabulary Object Detection

The open-vocabulary object detection task is proposed [235] as training a model that detects objects beyond the pre-defined vocabulary (*i.e.* closed-vocabulary detectors [11, 18, 73, 74, 102, 103, 113, 127, 131, 169, 170, 171, 172, 213, 238, 253]). ViLD [76] uses an open-vocabulary classifier to train a detector with pre-defined base classes to detect objects in novel classes. With the rise of vision-language models [100, 166], recent works [29, 45, 108, 118, 130, 145, 221, 249] formulate open-vocabulary object detection as an image-text matching problem. Most of the OVD models of this variant take a text prompt and an image as input, and output object bounding boxes corresponding to the objects described by the prompt. However, the detection precision is sensitive to the prompt, which requires careful prompt optimization. This optimization is usually not trivial and requires much experience and understanding of both the detector and data distribution. Our work, on the other hand, aims to directly optimize the prompt text feature according to a limited number of sample images. It does not require the initial prompts to be well-written, and can work on different types of datasets and target objects. We show its effectiveness on the state-of-the-art GroundingDINO [130] model, and it can be implemented on other OVD models as well.

5.2.2 Adaptive Object Detection

Adaptive object detection has been discussed extensively in chapter 3 and chapter 4. Many other methods also exist [28, 38, 65, 67, 84, 92, 95, 98, 105, 114, 117, 119, 120, 121, 123, 125, 132, 134, 149, 175, 176, 178, 185, 189, 202, 223, 224, 228, 242, 246, 248]. Most of them, including ours, however, actuate the adaptation through finetuning the parameters of the detector. This can be computationally heavy and is not easily scalable. Instead, in this chapter we treat the adaptation of OVD models to new datasets as a prompt optimization problem. We claim that for an OVD model with sufficient capacity, its detection performance on a new and unseen dataset can be improved by using data and object-specific prompts without changing the model parameters.

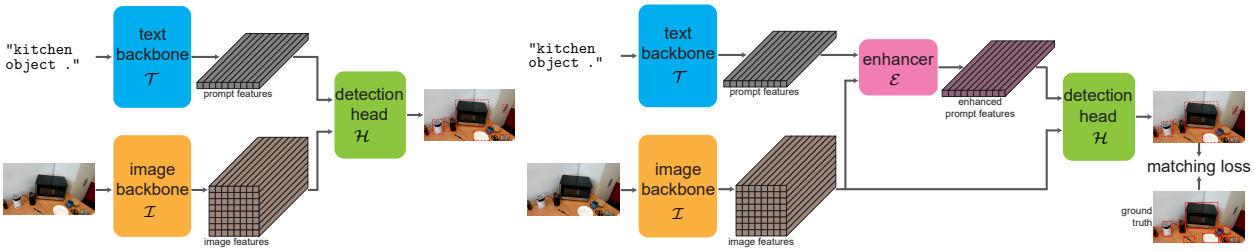
5.2.3 Prompt Engineering

In the context of vision-language models [240], we treat the term "prompt engineering" as the general process of getting the prompt text or prompt feature for models to get the optimal performance on a specific task. Since those models have large capacities and are trained on large-scale datasets, they have the potential to work on different types of data and solve many different problems [101, 115, 166]. Aside from manual prompt tuning, which requires experience, domain expertise, and a deep understanding of the model and the task, many have explored the possibility of optimizing the prompt through machine learning. Several methods are proposed for prompt learning on OVD models [46, 55], which aims to improve the detection performance of OVD models on novel object categories. The prompt representation is learned along with the training process of the OVD models. In this paper, we use prompt learning as a means for adapting a trained OVD model to new tasks and datasets. Prompt learning has been used for domain adaptive image classification [70]. Similarly, we only learn a lightweight module to enhance the prompt feature without changing the parameters of the trained model. This is more efficient, easier to scale, and less prone to overfitting.

5.3 Adaptive Prompt Enhancement

In this section, we describe the proposed framework, which adapts a pre-trained OVD model to new data using a lightweight prompt enhancement module. This module learns

to enrich the text feature of an initial prompt based on a small number of images with labeled target objects.



(A) GroundingDINO (B) GroundingDINO with proposed enhancer

FIGURE 5.1: Model architecture of vanilla GroundingDINO and GroundingDINO with the proposed adaptive prompt feature enhancer. The enhancer is a lightweight transformer decoder that learns to enrich the prompt features according to a limited number of sample images with target objects labeled. The text backbone, image backbone, and detection head are not trained during adaptation.

5.3.1 Preliminaries: GroundingDINO Detector

Our prompt enhancement framework can be applied to any open-vocabulary detector. However, in this paper, we demonstrate our framework for GroundingDINO [130], a state-of-the-art OVD. We first briefly describe the architecture of GroundingDINO to ensure a comprehensive understanding of our proposed method. Similar to most OVDs, GroundingDINO model primarily consists of three components: 1) a text backbone $\mathcal{T}(\cdot)$ that extracts features from a given text prompt. In GroundingDINO, the text backbone is trained from a pre-trained BERT [43] model; 2) an image backbone $\mathcal{I}(\cdot)$ that extracts features from the input image. In GroundingDINO, the image backbone is also trained from a pre-trained SwinTransformer [133]; and 3) a detection head $\mathcal{H}(\cdot,\cdot)$ that calculates the correlation among the text features and image features and ultimately produces bounding boxes of detected objects. GroundingDINO adopts the detection head of DINO [238], which is also based on the transformer architecture [204]. Formally, for a given prompt p and image x, GroundingDINO produces a set of object bounding boxes $\{b_i\}$ along with the corresponding confidence scores $\{s_i\}$:

$$\{b_i, s_i\}_{1 \le i \le Q} = \mathcal{H}(\mathcal{T}(p), \mathcal{I}(x)). \tag{5.1}$$

Here, Q is the number of learnable object queries and is a hyperparameter of the model. The model architecture of GroundingDINO is illustrated in Figure 5.1a. In this paper, we use the Swin-T variant of GroundingDINO. The code and weights are downloaded from the official repository[1].

In the original GroundingDINO model, at the end of the pipeline, a module operates on each object bounding box and text phrase to match the most similar phrase to each detected object. This is useful when fine-grained attributes of the detected objects are needed. For our setting, we only use one prompt for each object category, so this matching step is omitted. We calculate the score of matching each detected object with all possible phrases and use the highest score as the confidence score of this detected object.

5.3.2 Prompt Feature Enhancer

We propose to use a separate and lightweight module for adapting an OVD to new data, instead of directly finetuning its parameters. Specifically, we use a small transformer decoder model $\mathcal{E}(\cdot, \cdot)$ to enhance the text features of the prompt, conditioned on the image features, and use the enhanced prompt feature for detection:

$$\{b_i, s_i\}_{1 \leq i \leq Q} = \mathcal{H}(\mathcal{E}(\mathcal{T}(p), \mathcal{I}(x)), \mathcal{I}(x)), \tag{5.2}$$

where the text feature $\mathcal{T}(p)$ is the query, and image feature $\mathcal{I}(x)$ is the key/value for the enhancer. This modification is illustrated in Figure 5.1b.

We claim that using a separate lightweight module for adaptation between different modules of an OVD is beneficial for several reasons. 1) OVDs are mostly based on large language and vision foundation models, which require a large amount of computation to finetune. The proposed enhancer is small and easy to train. 2) If a detector is adapted to a new dataset, a new copy of the adapted model needs to be stored and deployed. This poses scalability issues when the number of new datasets increases. Please note that this is still an issue even when efficient finetuning techniques such as LoRA [2, 93] are used. On the other hand, creating a lightweight module for each dataset is acceptable. 3) High-quality annotation is usually expensive and thus sparse. In such cases, a smaller model is less prone to overfitting. And 4) in the case when the text or image backbone runs as a service through an API, a direct update on its parameters is not possible. Yet,

[1] `https://github.com/IDEA-Research/GroundingDINO`

the proposed enhancer model can still be trained, as long as the detection head model is available.

For OVDs, using separate modules for adaptation is less likely to cause efficiency issues at inference time. For closed-vocabulary detectors, one-stage detectors [11, 102, 103, 113, 127, 131, 169, 170, 171, 213, 214] usually has efficiency advantages over two-stage ones [18, 73, 74, 172, 238, 253]. Since all instances of all object categories are detected by the model at once, it is desirable to obtain them in one pass. So, it is beneficial to optimize the entire network as a whole, which makes inserting new modules in the network difficult. However, for OVDs, the disentanglement of different modules (text backbone, image backbone, and detection head) and the stage-wise computation are useful. It is common to have the image backbone to exact image features beforehand, and then use different text prompts each time to detect different types of objects. Or vice versa, extract the text features from a fixed prompt first, then detect the same type of objects in different images. Each module can be optimized separately or even run on different devices. Under such setting, inserting a separate module into the existing model pipeline is compatible, and it does not hinder efficiency.

5.3.3 Adaptive Training

We aim to adapt the enhancer module to a dataset with objects that differ greatly from the objects in the training set of the OVD model. The model is given a limited number of images from the new dataset, in each image the target objects are labeled. The enhancer needs to learn from those objects to provide the correct prompt feature to the detector. Similar to many set prediction-based detectors [18, 130, 238, 253], we use a detection loss $\mathcal{L}(\cdot, \cdot)$ based on Hungarian matching. It is a weighted sum of cross-entropy loss, bounding box regression loss, generalized bounding box IoU loss [173], and focal loss [127]. Formally, from the output of the detector in (5.2) and a set of ground-truth object bounding boxes $\{b_j^*\}_{1 \leq j \leq K}$ with K being the number of objects, the enhancer $\mathcal{E}(\cdot, \cdot)$ is optimized using

$$\min_{\mathcal{E}} \mathcal{L}(\mathcal{H}(\mathcal{E}(\mathcal{T}(p), \mathcal{I}(x)), \mathcal{I}(x)), \{b_j^*\}_{1 \leq j \leq K}). \tag{5.3}$$

5.4 Experiments

We conduct experiments on three video object detection datasets. To measure the detection performance, we calculate the mean average precision of intersection-over-union (IoU) thresholds from 0.5 to 0.95 (AP^m) using the COCO protocol [128]. We compare the proposed enhancer-based adaptation with finetuning the parameters of GroundingDINO model, and with manual prompt engineering. We also qualitatively analyze how the adaptation improves detection performance and test the generalization ability of the proposed method.

5.4.1 Datasets

We use three video object detection datasets to test the effectiveness of the proposed method. GroundingDINO is trained on a combination of several general-purpose object detection datasets. However, for specific application scenarios, it needs to adapt to shifted data distribution or more abstract object categories. Each of the three datasets represents such an application scenario.

Scenes100 is proposed in chapter 3 as a scene-adaptive object detection benchmarking dataset. It contains 100 long videos recorded in different scenes, and some frames in the ending portion of each video are labeled with *person* and *vehicle* objects. We treat both categories as a common type. This dataset simulates the scenario where the category of target objects is known. It is easy to write a prompt such as `"person . vehicle"` for the objects. However, the objects still differ from each other, and potentially differ greatly from the training data of the OVD model, so adaptation can still improve detection precision.

EgoPER is proposed [112] as an egocentric video dataset of procedural cooking tasks. In each video, several frames are labeled with objects that are used in cooking, such as ingredients, containers, utensils, kitchen appliances, and hands. We also treat all object categories as one. This dataset simulates the scenario when the general type of the target objects is known, but they belong to different categories, and some of the categories can be novel and not present in the training set of the OVD model. It is difficult to include all the exact object categories in the prompt. One may write a more general prompt like `"kitchen object"`. The OVD model learns to identify the specific object categories from adaptation.

HOIST [151] is proposed as a video object segmentation dataset. It contains a few thousand short videos, each of which has some frames labeled with objects that are held by hand. And the objects in every video differ greatly in type, size, and appearance. It is impossible to specify the object category in the prompt. One can only use an abstract concept such as `"object in hand"`. This simulates the more difficult scenario where the detector needs to learn to identify objects according to their relation with other visual elements through adaptation, instead of just learning the visual appearance.

Please note that Scenes100, EgoPER, and HOIST are all recently proposed for very specific tasks. They are not general-purpose object detection datasets. This means that the objects of interest in them have less in common with those in the training set of GroudingDINO, making the adaptation more challenging.

5.4.2 Dataset Split and Training Hyper-Parameters

For each video in each dataset, several frames are labeled by human experts with the target objects. We adaptively learn the enhancer with both **seen** and **unseen** settings. For each dataset, the videos are divided into training and testing splits. In the **seen** adaptation setting, the model is trained using the first annotated image of videos in both the training and testing splits and is evaluated on the remaining annotated images in the testing split. In the **unseen** setting, the model is trained using the first annotated image of videos in training split only, while still evaluated on the same images in the testing videos. Since the object appearance and type differ from video to video, the unseen setting is more challenging to the model's generalization ability. The dataset splitting and different training settings are illustrated in Figure 5.2.

For Scenes100, we use a pseudo-random algorithm to split the dataset into 50 training videos and 50 evaluation videos, and a total of 2,675 images for evaluation. We follow the same protocol in the original paper to mask out annotation and detection bounding boxes that overlap with the pre-defined non-annotation regions before calculating AP^m. For both seen and unseen adaptation, the model is trained for 2,000 iterations with a batch size of 2 and a learning rate of 1×10^{-4}. For EgoPER, we follow the original dataset split and remove videos that have less than 5 frames labeled. This leaves us with 65 training videos, 27 testing videos, and 1,025 evaluation images. For both seen and unseen adaptation, the model is trained for 2,000 iterations with a batch size of 2 and a learning rate of 1×10^{-4}. Since the original split of HOIST gives a very small testing set, we instead

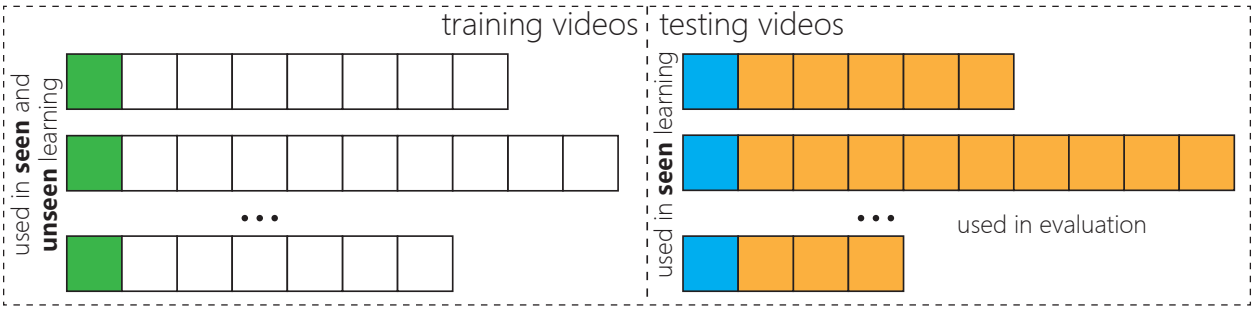


FIGURE 5.2: Illustration of dataset splitting and different adaptation settings. The videos in a dataset are divided into training and testing splits. Each video has a different number of frames that are labeled by human experts with the target objects. In the **seen** adaptation setting, the model learns from the first labeled frame of each video in both training (marked in green) and testing (marked in cyan) splits, and evaluated on all but the first labeled frames of each testing video (marked in orange). In the **unseen** adaptation setting, the model only sees the first labeled frame of each video in the training split while being evaluated on the same set of testing frames.

use a pseudo-random algorithm to split the original training set into 1,862 training videos, 1,863 testing videos, and 35,405 evaluation images. For both seen and unseen adaptation, the model is trained for 20,000 iterations with a batch size of 2 and a learning rate of 1×10^{-4}. More iterations are used as HOIST is considerably larger.

5.4.3 Adaptation from Different Initial Prompts

For each dataset, we choose three different initial prompts to simulate variants in word preference, different levels of language proficiency, and different levels of familiarity with the datasets. The prompts are listed in Table 5.1. This format of descriptive phrases separated by periods is recommended for GroundingDINO. Please note that this list is not exhaustive, and it is not possible to determine which of them works the best without actually evaluating it on the dataset.

The detection AP^m of the pre-trained GroundingDINO using those initial prompts on different datasets are shown in Table 5.2. It is clear that the detection performance from different prompts varies greatly, which means that using the correct prompt is critical to unleash the potential of an OVD. However, the difference between good and bad prompts is subtle and sometimes counter-intuitive. For instance, on Scenes100, a more detailed and verbose prompt `"person . pedestrian . vehicle . automobile . car ."`

Dataset	Prompt		
	I	II	III
Scenes100	`"traffic object ."`	`"person . vehicle ."`	`"person . pedestrian . vehicle . automobile . car ."`
EgoPER	`"kitchen object ."`	`"kitchen . cooking . human body ."`	`"appliance . utensil . cutlery . seasoning . ingredient . food . hand ."`
HOIST	`"common object ."`	`"object in hand ."`	`"hand-held object ."`

TABLE 5.1: Different initial prompts for each dataset. They represent the prompts from end users with different word preferences, different levels of language proficiency, and different levels of familiarity with the datasets.

obtains significantly higher AP^m than the more coarse prompt `"person . vehicle ."`. On the contrary, on EgoPER, the more ambiguous `"kitchen object ."` works better than detailed `"appliance . utensil . cutlery . seasoning . ingredient . food . hand ."`. And on HOIST, the prompts `"object in hand ."` and `"hand-held object ."` get very different detection performances, yet they have the same literal meaning. This sensitivity to wording and inconsistency among different datasets make manual search of the optimal prompt challenging. It is virtually impossible without many rounds of trial-and-error.

We use the proposed method to learn the text feature enhancer under both seen and unseen adaptation settings. The detection performance of the model with the trained enhancer is also shown in Table 5.2. It is clear that for all initial prompts on all datasets, the proposed method can significantly improve the detection performance. The performance improvement from unseen adaptation is lower than seen adaptation, meaning the difference in data distribution among different videos causes slight generalization difficulties. However, the adapted performance is still consistent for different initial prompts on the same dataset. This means that the proposed method eliminates the need for experienced prompt engineers. Even a low-performing initial prompt can reach a high level of detection performance after adaptation.

Dataset	Scenes100			EgoPER			HOIST		
Prompt	I	II	III	I	II	III	I	II	III
no adaptation	9.44	36.60	46.49	45.99	21.59	38.01	14.88	12.25	22.73
seen adaptation	**56.69**	**57.57**	**57.30**	**72.47**	**73.88**	**73.75**	**40.40**	**40.84**	**39.95**
unseen adaptation	54.67	55.16	55.80	69.56	72.00	71.63	35.29	37.25	36.51

TABLE 5.2: Detection AP^m of GroundingDINO before and after adaptation training on different datasets with different initial prompts. Please refer to Table 5.1 for the details on the prompts. We compare the AP^m of models with no adaptation, models with seen adaptation training, and models with unseen adaptation training. The model with the highest AP^m for each prompt-dataset combination is marked in **bold**. Different initial prompts get different detection performances. The proposed adaptation method can improve the performance significantly and consistently.

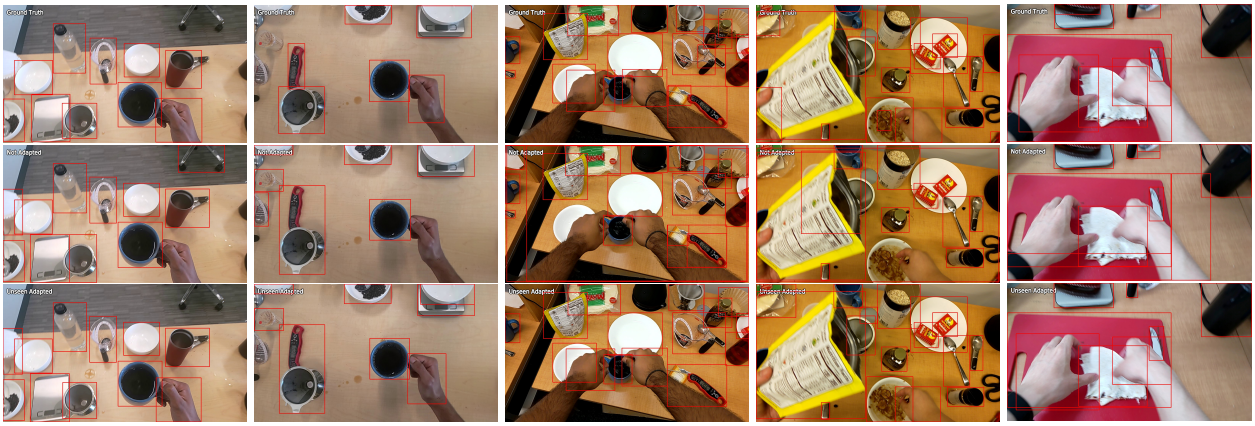
5.4.4 Qualitative Analysis

We visually compare how the behavior of the vanilla GroundingDINO model changes after adding the adaptive enhancer. For each dataset, we choose the initial prompt that is simple but still obtains relatively high AP^m before adaptation. specifically, we use `"person . vehicle ."` on Scenes100, `"kitchen object ."` on EgoPER, and `"hand-held object ."` on HOIST. This simulates the situation when the detector user has some experience and some level of familiarity with the dataset. We show the detected object bounding boxes with a score higher than θ^*, where θ^* is the score threshold that maximizes the F_1 score at IoU=0.75 in the precision-recall evaluation per the COCO protocol [128].

The comparison on several videos from Scenes100 is shown in Figure 5.3a. It is clear that GroundingDINO misses many instances of *vehicle* objects from the prompt phrase `vehicle`. This is possibly caused by the fact that in the training set of GroundingDINO, most of the vehicles are labeled by other words such as `car`, `truck`, `SUV`, etc. This type of bias is very common in datasets [51, 251]. And the bias transfers to models that are trained on those datasets. Our proposed method can adaptively learn to compensate for such bias and identify different types of vehicles.

The detection performance on EgoPER videos is compared in Figure 5.3b. The model can already locate most of the target objects correctly before adaptation. However, the proposed method can still help the model to make more fine-grained decisions, such as not labeling the chair in video `COFF_PINW006-01-0` or identifying the filter dripper in

(A) Results on Scenes100 dataset using initial prompt "`person . vehicle .`". Bounding boxes that overlap with the pre-defined non-annotation regions are removed.



(B) Results on EgoPER dataset using initial prompt "`kitchen object .`".

FIGURE 5.3: Qualitative results on several videos from different datasets. Each column of three images is from one video. The images show the ground-truth bounding boxes, the detected bounding boxes from the pre-trained GroundingDINO model, and the detected bounding boxes from the model with an adaptively trained prompt enhancer. Please refer to subsection 5.4.4 for more details. This figure is best viewed on color screens.

video `V_Tea_3`.

As shown in Figure 5.3c, detecting the objects in hand in HOIST dataset is challenging. To get correct results, a certain level of understanding of the underlying physical world is needed to determine if an object is held by hand. Before adaptation, even with a prompt specifying hand-held objects, the GroundingDINO model still tends to label most objects in the image. After adaptation, the enhancer enables the model to reason about the relation between objects and hands, and find the correct hand-held objects. However, given

(C) Results on HOIST dataset using initial prompt "`hand-held object .`".

FIGURE 5.3: (Continued) Qualitative results on several videos from different datasets. Each column of three images is from one video. The images show the ground-truth bounding boxes, the detected bounding boxes from the pre-trained GroundingDINO model, and the detected bounding boxes from the model with an adaptively trained prompt enhancer. Please refer to subsection 5.4.4 for more details. This figure is best viewed on color screens.

the difficulty of the task, it can still treat objects that are close to hands as objects that are held by hands, as in video `ODxjTJJOkRU` and `PtOS5evrqrQ`. Or still misses the object when the hand is blurry as in video `ZmzY3RILf6c`.

We visually verify that the proposed prompt enhancer module can learn to guide GroundingDINO to the correct direction to improve detection precision, from a limited number of sample images.

5.4.5 Comparison with Finetuning

Instead of using a lightweight module to enhance the text features of the prompts, we try to directly train the modules in GroundingDINO. We keep the same dataset splitting, loss functions, and training hyper-parameters for fair comparison. We test under the more challenging unseen adaptation setting. Formally, we train the BERT text backbone using

$$\min_{\mathcal{T}} \mathcal{L}(\mathcal{H}(\mathcal{T}(p), \mathcal{I}(x)), \{b_j^*\}_{1 \le j \le K}), \tag{5.4}$$

or training the detection head by

$$\min_{\mathcal{H}} \mathcal{L}(\mathcal{H}(\mathcal{T}(p), \mathcal{I}(x)), \{b_j^*\}_{1 \le j \le K}). \tag{5.5}$$

The enhancer only has 1.5 million parameters. The text backbone has 104 million parameters, and the head has 32 million parameters. Both are significantly larger. The detection performance of the adapted models is compared in Table 5.3. Training the text backbone or the detection head results in a similar or lower AP^m than training the enhancer. Since the enhancer operates between the text backbone and the head and is considerably smaller than both, it is theoretically possible for the head or the text backbone to obtain similar performance as the enhancer through training. Their inferior performance is likely caused by the overfitting to a limited number of samples. The text backbone is pre-trained on a large number of tokens, so it has better generalization ability [16] and obtains similar detection performance after adaptation as the enhancer on Scenes100 and EgoPER datasets. The detection head is trained from scratch by GroundingDINO. So, it overfits more and results in significantly lower improvement from adaptation. Please note that aside from higher detection precision, training a separate enhancer has multiple other benefits, as discussed in subsection 5.3.2.

Dataset	Scenes100			EgoPER			HOIST		
Prompt	I	II	III	I	II	III	I	II	III
enhancer	54.67	**55.16**	**55.80**	**69.56**	**72.00**	**71.63**	**35.39**	**37.25**	**36.51**
BERT	**56.09**	54.38	53.93	68.76	69.90	68.05	30.71	30.40	30.37
head	38.06	36.41	40.19	58.49	51.07	38.06	23.84	20.63	19.97

TABLE 5.3: Detection AP^m of GroundingDINO with different modules trained from adaptation on different datasets with different initial prompts. Please refer to Table 5.1 for the details on the prompts. We compare the proposed method of training a separate prompt enhancer with training the text backbone BERT or detection head directly. All models are trained under the unseen adaptation setting. The model with the highest AP^m for each prompt-dataset combination is marked in **bold**. The text backbone and head, despite having significantly more parameters than the enhancer, gain similar or less detection performance from adaptation.

5.4.6 Comparison with Manual Prompt Engineering

Writing the correct prompt for OVD tasks can be difficult, especially when the user has limited knowledge of the categories of the target objects. Here, we assume that the user can obtain detailed object category information and test whether such information is helpful for manual prompt engineering.

In the previous experiments, we treat the EgoPER dataset as having objects that are commonly used in cooking activities. Here, we assume that the user is given some sample images from the dataset with target objects labeled and has done some research on their corresponding categories. Then, a detailed prompt is constructed using the names of the objects as: `"kettle . measuring cup . kitchen scale . coffee grinder . filter cone dripper . paper basket filter . mug . thermometer . spoon . bowl . bottle . hand . paper filter . tortilla . peanut butter . jam . toothpick . knife . chopping board . dental floss . paper towel . plate . paper sheet . tea bag . trash can . honey . nutella . banana slice . cinnamon . oat . microwave . raisin ."`. This prompt gets an evaluation AP^m of 51.97, which is only moderately higher than the prompts used in the adaptation experiments and considerably lower than the AP^m after adaptation.

For HOIST, the object categories are different for each video. We also try to name the object in each video instead of using a prompt as general as `"hand-held object ."`. Specifically, we crop the labeled objects from the first annotated frame of each testing video, and use a pre-trained CLIP [166] model to match the cropped images with an extensive list of 19,995 possible object names from the OpenImages-V6 dataset [107, 109] The most possible match is used to construct a prompt. For generalization, we still include `"hand-held object ."` in all the prompts. We use a separate prompt for each video. For instance, if *ball pen* and *book* are matched for the cropped objects in a video, the prompt is `"ball pen . book . hand-held object ."`. The overall AP^m is still evaluated on all evaluation images. This method gets AP^m of 18.36. We also try to add more in-hand constraints to construct prompts such as `"ball pen in hand . book in hand . hand-held object ."`. The AP^m is 16.08. Both methods actually cause the detection performance to degrade.

This again shows that prompt engineering can be unreliable and inconsistent. Even though all the object categories can be described for EgoPER, the detection precision is still much lower than that of the proposed adaptation method. For HOIST, the detector needs to understand the physical object-hand relation to localize target objects correctly. This is difficult even with the help of object categories and constrained phrases. However, the proposed method can learn an enhancer that can encode needed information into the prompt feature to guide the detection head accurately.

5.4.7 Cross-Prompt Generalization

To test the generalization ability of the learned prompt feature enhancer, we measure the detection performance of the enhancer model trained on a certain prompt and use another prompt during evaluation. We experiment under the more challenging unseen adaptation setting. The evaluated detection performance is shown in Table 5.4. Different prompts on different datasets show different cross-prompt generalization abilities. On HOIST, all three initial prompts are coarse and have similar meanings, so the learned enhancers can all generalize across different prompts without a significant performance drop. On Scenes100 and EgoPER, enhancer learned from shorter and more coarse prompts (`"traffic object ."` on Scenes100 and `"kitchen object ."` on EgoPER) can generalize well to more complex and detailed prompts. In contrast, an enhancer learned from longer and more detailed initial prompts sees a significant detection performance drop when evaluated using shorter and more general prompts. This implies that if cross-prompt generalization ability is desired, the enhancer should learn from more general and coarse initial prompts to avoid overfitting.

tr. \ *ev.*	I	II	III
I	**54.67**	51.69	53.54
II	32.54	**55.16**	48.90
III	23.98	34.45	**55.80**

(A) Scenes100

tr. \ *ev.*	I	II	III
I	**69.56**	63.29	67.62
II	60.43	**72.00**	65.67
III	56.77	57.54	**71.63**

(B) EgoPER

tr. \ *ev.*	I	II	III
I	**35.39**	34.05	35.06
II	35.19	**37.25**	36.61
III	36.02	36.50	**36.51**

(C) HOIST

TABLE 5.4: Cross-prompt evaluation AP^m of the trained enhancer. The model is trained (indicated by *tr.*) and evaluated (indicated by *ev.*) using different prompts. Please refer to Table 5.1 for the details on the prompts. All models are trained under the unseen adaptation setting. The highest AP^m for each training prompt and dataset combination is marked in **bold**. The trained enhancer model shows different generalization abilities on different prompts and datasets.

5.4.8 Inference Efficiency of Enhancer Module

The proposed prompt feature enhancement module only adds about 0.9% of the total number of parameters of GroundingDINO. It is also efficient to train under the adaptation setting. However, it can still potentially increase the latency of model inference. Here, we

practically measure the inference latency of the vanilla GroundingDINO model and the model with the enhancer. For transformer models, the computation complexity depends on the length of the input sequences. For GroundingDINO, that depends on the number of tokens in the prompt and the resolution of the input image. We use the same image that is resized to 750×1333 from EgoPER dataset and use different prompts on it. Specifically, we test the 3 prompts (I, II, and III) in Table 5.1, and the long detailed prompt in subsection 5.4.6, which we refer as prompt IV. All experiments run on the same NVIDIA GTX 2080 Ti GPU. A slower GPU is used, so the difference in latency is more significant. The inference latency is compared in Table 5.5. It is clear that the lightweight enhancer module runs fast at inference time and only adds minimal overhead to the model latency.

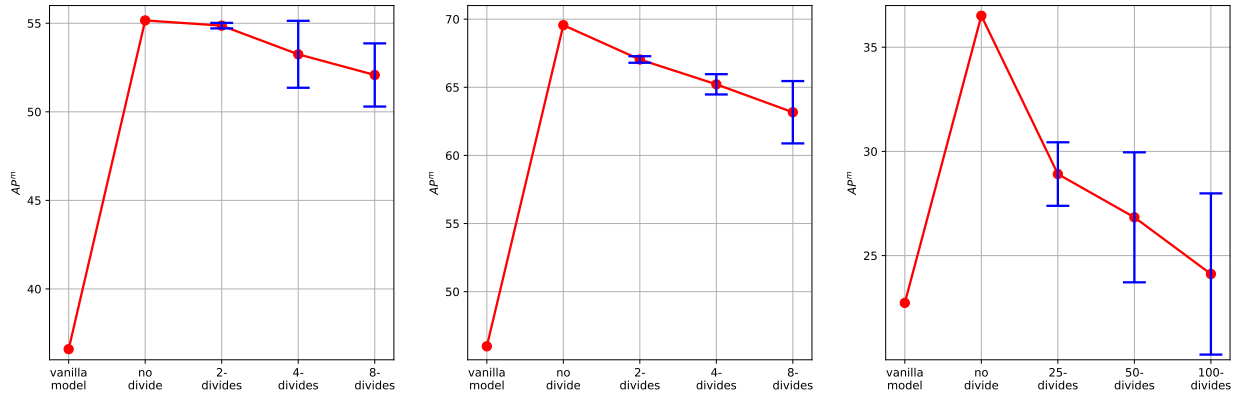
Prompt	# tokens	Inference latency (ms)		Increase
		Vanilla	Enhancer	
I	5	182.0	183.6	0.9%
II	9	186.9	187.6	0.4%
III	22	197.5	199.0	0.8%
IV	95	217.8	218.1	0.1%

TABLE 5.5: Inference latency in terms of milliseconds of vanilla GroundingDINO and GroundingDINO with the proposed enhancer, using different prompts. Please refer to subsection 5.4.8 for more details on the prompts. The column # tokens shows the number of tokens in each prompt. The column Increase shows the relative increase in latency after adding the enhancer module. All inference runs use the same input image for consistency. The enhancer only adds minimal latency overhead.

5.5 Low-Data Adaptation Performance

Here we further test the low-data performance of the proposed method by training it on a smaller number of images. We test under the "N-divides" setting. For instance, for Scenes100 dataset, there are 50 videos in the training split. All the unseen adapted models are trained on 50 sample images for 2,000 iterations. Under a 2-divides setting, we further use a pseudo-random algorithm to divide the 50 images into 2 non-overlapping divides, and each divide has 25 images. Then a separate enhancer is trained on each divide for 1,000 iterations. Each of the 2 resultant enhancers is still evaluated on all the 2,675 images. We calculate the mean and standard deviation of the AP^m from different enhancers and

compare them with the AP^m of the adapted enhancer. Similar approaches are used for EgoPER and HOIST datasets. For this experiment, we use `"person . vehicle ."` on Scenes100, `"kitchen object ."` on EgoPER, and `"hand-held object ."` for HOIST as the initial prompts.



(A) Scenes100 (B) EgoPER (C) HOIST

FIGURE 5.4: Detection performance of the proposed method adapted from smaller number of sample images (N-divides) on different datasets. Please refer to section 5.5 for details of N-divides. The performance of GroundingDINO before any adaptation (vanilla model) and using all the sample images under the unseen adaptation setting (no divide) is shown for comparison. The standard deviation of the AP^m of models adapted on different divides is shown as error bars. Smaller number of sample images lead to lower detection AP^m and more unstable performance.

The detection AP^m is shown in Figure 5.4. It is clear that using fewer sample images leads to lower detection performance and higher variance in the performance of each of the trained enhancers. Still, even with larger N (8-divides on Scenes100 and EgoPER), where there are less than 10 sample images in each divide, the proposed method still achieves significant improvement in AP^m. However, generalization on HOIST dataset is more challenging as the target objects in each video can have different categories. Thus, the detection performance of the enhancer drops more when trained on fewer sample images.

Dataset	Scenes100	EgoPER	HOIST
Faster-RCNN ResNet-50	40.38	59.38	18.00
Faster-RCNN ResNet-101	39.73	59.23	18.43
YOLOv8s	16.43	21.23	3.43
YOLOv8l	26.82	24.43	9.58
Prompt (proposed)	55.80	72.00	37.25

TABLE 5.6: Detection AP^m of closed-vocabulary detectors Faster-RCNN [172] and YOLOv8 [103] on different datasets. All models are trained under the unseen adaptation setting. For each dataset, the highest AP^m from the proposed GroundingDINO based Prompt adaptation method is shown for comparison. Closed-vocabulary detectors achieve significantly lower AP^m compared to the proposed method.

5.6 Adaptation of Closed-Vocabulary Detectors

We try to treat the target objects in each dataset as an object category, and adapt state-of-the-art closed-vocabulary detection models to detect them. We use the same unseen adaptation setting, and the same number of training iterations, learning rate, and batch size for fair comparison. Specifically, we use Faster-RCNN [172] model with ResNet-50 or ResNet-101 [82] backbone and feature pyramid network [126], and YOLOv8 [103] of small (YOLOv8s) or large (YOLOv8l) variants. Both models are trained on MSCOCO dataset [128]. For each model, we use a new classification head, which only has one object category, and finetune the whole network on the training images under the unseen adaptation setting. Faster-RCNN and YOLOv8 use their corresponding sets of loss functions that are different from GroundingDINO [130].

The detection performance is compared in Table 5.6. It is clear that closed-vocabulary detectors fall behind adapted GroundingDINO by a significant margin. Faster-RCNN achieves acceptable performance on Scenes100 and EgoPER, where the target objects still belong to certain categories. But on HOIST dataset, where the target object is a more abstract concept of hand-held object, all closed-vocabulary models get low AP^m. Please note that those two detection models are based on convolution operation, and both have considerably fewer parameters than GroundingDINO. Closed-vocabulary detectors have different overall architectures and training procedures than OVDs. Yet, it still shows the benefits of adapting an OVD model for such difficult detection tasks.

5.7 Summary

In this chapter, we tackle the adaptive object detection problem of open-vocabulary detectors as a prompt learning problem. Instead of finetuning the model parameters or manually searching for the optimal prompt, a lightweight enhancer is used to adaptively enrich the prompt feature and thus improve the detection performance. The enhancer module is efficiently trained using a limited number of sample images, and only adds negligible overhead at inference time. Experiments using GroundingDINO detector and several adaptive object detection datasets show that the proposed method can significantly improve detection precision on different types of objects.

Chapter 6

Conclusion and Future Directions

In this dissertation, I have discussed the idea of adaptive post-deployment improvement of a trained event or object detection model. I have demonstrated that this adaptive improvement can be implemented by letting the model to observe a video stream while obtaining signals through self-supervision or sparse human supervision to update itself adaptively. An event detector that can only detect the target events after they occurred fully can be adapted into a detector that can detect events early, and a generic object detector can adapt to diverse scenes with levitated detection power with minimal efficiency side effects. I also illustrate the idea of adapting an open-vocabulary object detector without changing its parameters to new tasks, which is useful in cases when the parameters of a model cannot be updated after deployment.

In the era of foundation models such as large vision-language models, adaptation is specially useful. Since training large models requires enormous amount of data, hardware, energy, and human labor, most developers and researchers only have the capability to take pre-trained models and adapt them to their specific needs. The issues discussed in this dissertation—the lack of high quality annotated data and computational cost constrains—also poses more importance. It is critical to know what data is useful for adaptation, and how to reduce the cost of adaptation. For instance, for a large model to be customizable to end users, it needs to have the ability to adapt to each user using a limited amount of information. This can be actuated by either finetuning the model parameters in a efficient way such as low-rank approximation or only training a subset of the parameters, or using methods that keep the parameters unchanged such as prompting or in-context learning.

On the other hand, for those who has the capability of training large foundation models from scratch, it is also important to design and train these models with adaptation-friendliness in mind. For instance, a model that has the ability of processing more input

modalities has the potential to be adapted to a wider range of tasks. If a model can take very long input sequences, the end users can fit more information as the prompt, thus enabling in-context adaptation. If the model cannot be published due to commercial reasons, it should has the API for effective and efficient adaptation. In summary, adaptation of models will be of more importance with the rapid development of large foundation models. This will keep being a research area that receives much attention and produces useful work.

Bibliography

[1] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, and Daniel Toal. "A survey of state-of-the-art on visual SLAM". In: *Expert Systems with Applications* 205 (2022), p. 117734. ISSN: 0957-4174.

[2] Sidra Aleem, Julia Dietlmeier, Eric Arazo, and Suzanne Little. "ConvLoRA and AdaBN based Domain Adaptation via Self-Training". In: *IEEE International Symposium on Biomedical Imaging*. 2024.

[3] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. "Encouraging LSTMs to Anticipate Actions Very Early". In: *International Conference on Computer Vision*. 2017.

[4] Tara Baldacchino, Elizabeth J. Cross, Keith Worden, and Jennifer Rowson. "Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems". In: *Mechanical Systems and Signal Processing* 66-67 (2016), pp. 178–200. ISSN: 0888-3270.

[5] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. "Real time robust L1 tracker using accelerated proximal gradient approach". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[6] David Berthelot, Nicholas Carlini, Ian Goodfellow, Avital Oliver, Nicolas Papernot, and Colin Raffel. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems*. 2019.

[7] Rishika Bhagwatkar, Saketh Bachu, Khurshed Fitter, Akshay Kulkarni, and Shital Chiddarwar. "A Review of Video Generation Approaches". In: *2020 International Conference on Power, Instrumentation, Control and Computing*. 2020.

[8] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Learning Discriminative Model Prediction for Tracking". In: *International Conference on Computer Vision*. 2019.

[9] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Learning Discriminative Model Prediction for Tracking". In: *International Conference on Computer Vision*. 2019.

[10] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. "What is the State of Neural Network Pruning?" In: *ArXiv* (2020).

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *ArXiv* (2020).

[12] Leon Bottou and Vladimir Vapnik. "Local Learning Algorithms". In: *Neural Computation* 4.6 (1992), pp. 888–900.

[13] Shahine Bouabid and Vincent Delaitre. "Mixup Regularization for Region Proposal based Object Detectors". In: *ArXiv* (2020).

[14] George Box and G Jenkins. "Some recent advances in forecasting and control". In: *Applied Statistics* 17.2 (1968), pp. 91–109.

[15] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. 2nd ed. Springer, 2002.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. 2020.

[17] Zhaowei Cai and Nuno Vasconcelos. "Cascade R-CNN: Delving into High Quality Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[18] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision*. 2020.

[19] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: *Advances in Neural Information Processing Systems*. 2020.

[20] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging Properties in Self-Supervised Vision Transformers". In: *International Conference on Computer Vision*. 2021.

[21] João Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[22] Meilin Chen, Weijie Chen, Shicai Yang, Jie Song, Xinchao Wang, Lei Zhang, Yunfeng Yan, Donglian Qi, Yueting Zhuang, Di Xie, et al. "Learning Domain Adaptive Object Detection with Probabilistic Teacher". In: *International Conference on Machine Learning*. 2022.

[23] Shengyong Chen. "Kalman Filter for Robot Vision: A Survey". In: *IEEE Transactions On Industrial Electronics* 59.11 (2012), pp. 4409–4420.

[24] Tianlong Chen, Xuxi Chen, Xianzhi Du, Abdullah Rashwan, Fan Yang, Huizhong Chen, Zhangyang Wang, and Yeqing Li. "AdaMV-MoE: Adaptive Multi-Task Vision Mixture-of-Experts". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023.

[25] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *ArXiv* (2020).

[26] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021. DOI: 10.1109/CVPR46437.2021.01549.

[27] Xinlei Chen, Saining Xie, and Kaiming He. "An Empirical Study of Training Self-Supervised Vision Transformers". In: *International Conference on Computer Vision* (2021). URL: https://api.semanticscholar.org/CorpusID:233024948.

[28] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "Domain Adaptive Faster R-CNN for Object Detection in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[29] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. "YOLO-World: Real-Time Open-Vocabulary Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2024.

[30] Kyunghyun Cho and Yoshua Bengio. "Exponentially Increasing the Capacity-to-Computation Ratio for Conditional Computation in Deep Learning". In: *ArXiv* (2014).

[31] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. "Twins: Revisiting the Design of Spatial Attention in Vision Transformers". In: *Advances in Neural Information Processing Systems*. 2021.

[32] Ondrej Chum and Andrew Zisserman. "An Exemplar Model for Learning Object Classes". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[33] 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: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[34] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. "Scaling Egocentric Vision: The EPIC-KITCHENS Dataset". In: *European Conference on Computer Vision*. 2018.

[35] Martin Danelljan, Goutam Bhat, Christoph Mayer, and Matthieu Paul. *PyTracking*. `https://github.com/visionml/pytracking`. 2019.

[36] James W. Davis and Ambrish Tyagi. "Minimal-Latency Human Action Recognition Using Reliable-Inference". In: *Image and Vision Computing* 24.5 (2006), pp. 455–472.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2009.

[38] Jinhong Deng, Wen Li, Yuhua Chen, and Lixin Duan. "Unbiased Mean Teacher for Cross-Domain Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[39] KServe Developers. *KServe*. `https://kserve.github.io/website`. 2023.

[40] PaddlePaddle Developers. *Paddle Serving*. `https://github.com/PaddlePaddle/Serving`. 2022.

[41] TorchServe Developers. *TorchServe*. `https://pytorch.org/serve`. 2023.

[42] Triton Developers. *Triton Inference Server*. `https://github.com/triton-inference-server/server`. 2022.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019.

[44] 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". In: *International Conference on Learning Representations*. 2021.

[45] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. "Learning to Prompt for Open-Vocabulary Object Detection with Vision-Language Model". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[46] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. "Learning to Prompt for Open-Vocabulary Object Detection with Vision-Language Model". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2022), pp. 14064–14073. URL: https://api.semanticscholar.org/CorpusID:247778949.

[47] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. "Learning Factored Representations in a Deep Mixture of Experts". In: *International Conference on Learning Representations*. 2013.

[48] Omar Elharrouss, Noor Almaadeed, and Somaya Al-Maadeed. "A review of video surveillance systems". In: *Journal of Visual Communication and Image Representation* 77 (2021), p. 103116.

[49] Chris Ellis, Syed Zain Masood, Marshall F. Tappen, Joseph J. LaViola Jr., and Rahul Sukthankar. "Exploring the Trade-off Between Accuracy and Observational Latency in Action Recognition". In: *International Journal of Computer Vision* (2013).

[50] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: a survey". In: *Journal of Machine Learning Research* 20.1 (Jan. 2019), pp. 19972017.

[51] Simone Fabbrizzi, Symeon Papadopoulos, Eirini Ntoutsi, and Yiannis Kompatsiaris. "A Survey on Bias in Visual Datasets". In: *ArXiv* (2021).

[52] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. "LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[53] Yazan Abu Farha, Alexander Richard, and Juergen Gall. "When Will You Do What? - Anticipating Temporal Occurrences of Activities". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[54] William Fedus, Barret Zoph, and Noam Shazeer. "Switch transformers: scaling to trillion parameter models with simple and efficient sparsity". In: *Journal of Machine Learning Research* 23.1 (Jan. 2022).

[55] Chengjian Feng, Yujie Zhong, Zequn Jie, Xiangxiang Chu, Haibing Ren, Xiaolin Wei, Weidi Xie, and Lin Ma. "PromptDet: Towards Open-vocabulary Detection using Uncurated Images". In: *European Conference on Computer Vision*. 2022.

[56] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. "Recurrent Network Models for Human Dynamics". In: *International Conference on Computer Vision*. 2015.

[57] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 0022-0000.

[58] Andrea Frome, Yoram Singer, and Jitendra Malik. "Image Retrieval and Classification Using Local Distance Functions". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Cambridge, MA: MIT Press, 2007.

[59] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. "Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2007.

[60] Antonino Furnari and Giovanni Maria Farinella. "Rolling-Unrolling LSTMs for Action Anticipation from First-Person Video". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[61] Antonino Furnari and Giovanni Maria Farinella. "What Would You Expect? Anticipating Egocentric Actions with Rolling-Unrolling LSTMs and Modality Attention." In: *International Conference on Computer Vision*. 2019.

[62] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. "Batch Normalization is a Cause of Adversarial Vulnerability". In: *ArXiv* (2020).

[63] M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. "Ensemble deep learning: A review". In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.

[64] Yaroslav Ganin and Victor Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *International Conference on Machine Learning*. Lille, France, 2015.

[65] Yaroslav Ganin and Victor S. Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *International Conference on Machine Learning*. 2014.

[66] Yaroslav Ganin, E. Ustinova, Hana Ajakan, Pascal Germain, H. Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research*. 2016.

[67] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-Adversarial Training of Neural Networks". In: *Domain Adaptation in Computer Vision Applications*. Springer International Publishing, 2017, pp. 189–209.

[68] Ruohan Gao, Bo Xiong, and Kristen Grauman. "Im2Flow: Motion Hallucination From Static Images for Action Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[69] Andrea Camille Garcia, Jealine Eleanor Gorre, Joshua Angelo Karl Perez, and Mary Jane Samonte. "Deep Learning in Smart Video Surveillance for Crowd Management: A Systematic Literature Review". In: *International Conference on Frontiers of Educational Technologies*. 2021.

[70] Chunjiang Ge, Rui Huang, Mixue Xie, Zihang Lai, Shiji Song, Shuang Li, and Gao Huang. "Domain Adaptation via Prompt Learning". In: *ArXiv* (2022).

[71] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[72] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *ArXiv* (2021).

[73] Ross Girshick. "Fast R-CNN". In: *International Conference on Computer Vision*. 2015.

[74] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[75] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Third. The Johns Hopkins University Press, 1996.

[76] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. "Open-vocabulary Object Detection via Vision and Language Knowledge Distillation". In: *International Conference on Learning Representations*. 2021.

[77] Wilko Guilluy, Laurent Oudre, and Azeddine Beghdadi. "Video stabilization: Overview, challenges and perspectives". In: *Signal Processing Image communication* 90 (2021), p. 116015.

[78] Raad Ahmed Hadi, Ghazali Sulong, and Loay Edwar George. "Vehicle Detection and Tracking Techniques: A Concise Review". In: *Signal and Image Processing: An International Journal* 5.1 (Feb. 2014), pp. 112.

[79] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[80] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: *International Conference on Computer Vision*. 2017.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[83] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *European Conference on Computer Vision*. 2014.

[84] Zhenwei He and Lei Zhang. "Multi-Adversarial Faster-RCNN for Unrestricted Object Detection". In: *International Conference on Computer Vision*. 2019.

[85] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty". In: *International Conference on Learning Representations*. 2020.

[86] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *Advances in Neural Information Processing Systems Workshop*. 2015.

[87] Tin Kam Ho. "Random decision forests". In: *Proceedings of the International Conference on Document Analysis and Recognition*. 1995.

[88] Minh Hoai and Fernando De la Torre. "Max-Margin Early Event Detectors". In: *International Journal of Computer Vision* 107.2 (2014), pp. 191–202.

[89] Minh Hoai and Andrew Zisserman. "Improving Human Action Recognition using Score Distribution and Ranking". In: *Asian Conference on Computer Vision*. 2014.

[90] Minh Hoai and Andrew Zisserman. "Talking Heads: Detecting Humans and Recognizing Their Interactions". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[91] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *ArXiv* (2017).

[92] Cheng-Chun Hsu, Yi-Hsuan Tsai, Yen-Yu Lin, and Ming-Hsuan Yang. "Every Pixel Matters: Center-aware Feature Alignment for Domain Adaptive Object Detector". In: *European Conference on Computer Vision*. 2020.

[93] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022.

[94] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[95] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. "Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data". In: *Advances in Neural Information Processing Systems*. 2021.

[96] Zehao Huang and Naiyan Wang. "Data-Driven Sparse Structure Selection for Deep Neural Networks". In: *European Conference on Computer Vision*. 2018.

[97] Drew A Hudson and Christopher D Manning. "Compositional Attention Networks for Machine Reasoning". In: *International Conference on Learning Representations*. 2018.

[98] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Cross-domain weakly-supervised object detection through progressive domain adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[99] Vanita Jain, Fadi M. Al-turjman, Gopal Chaudhary, Devang Nayar, Varun Gupta, and Aayush Kumar. "Video captioning: a review of theory, techniques and practices". In: *Multimedia Tools and Applications* 81 (2022), pp. 35619 –35653.

[100] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision". In: *International Conference on Machine Learning*. 2021.

[101] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision". In: *ArXiv* (2021).

[102] Glenn Jocher. *YoLov5*. `https://github.com/ultralytics/yolov5`. 2020.

[103] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLO*. Version 8.0.0. Jan. 2023. URL: `https://github.com/ultralytics/ultralytics`.

[104] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. "The Kinetics Human Action Video Dataset". In: *ArXiv* (2017).

[105] Seunghyeon Kim, Jaehoon Choi, Taekyung Kim, and Changick Kim. "Self-Training and Adversarial Background Regularization for Unsupervised Domain Adaptive One-Stage Object Detection". In: *International Conference on Computer Vision*. 2019.

[106] Hema S Koppula and Ashutosh Saxena. "Learning Spatio-Temporal Structure from RGB-D Videos for Human Activity Detection and Anticipation". In: *International Conference on Machine Learning*. 2013.

[107] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." In: *Dataset available from https://storage.googleapis.com/openimages* (2017).

[108] Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. "F-vlm: Open-vocabulary object detection upon frozen vision and language models". In: *ArXiv* (2022).

[109] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale". In: *International Journal of Computer Vision* (2020).

[110] Samuli Laine and Timo Aila. "Temporal Ensembling for Semi-Supervised Learning". In: *International Conference on Learning Representations*. 2017.

[111] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. "Stochastic Adversarial Video Prediction". In: *International Conference on Learning Representations*. 2018.

[112] Shih-Po Lee, Zijia Lu, Zekun Zhang, Minh Hoai, and Ehsan Elhamifar. "Error Detection in Egocentric Procedural Task Videos". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2024.

[113] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei.

"YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications". In: *ArXiv* (2022).

[114] Jichang Li, Guanbin Li, Yemin Shi, and Yizhou Yu. "Cross-Domain Adaptive Clustering for Semi-Supervised Domain Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[115] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation". In: *International Conference on Machine Learning*. 2022.

[116] Junnan Li, Richard Socher, and Steven C.H. Hoi. "DivideMix: Learning with Noisy Labels as Semi-supervised Learning". In: *International Conference on Learning Representations*. 2020.

[117] Kai Li, Chang Liu, Handong Zhao, Yulun Zhang, and Yun Fu. "ECACL: A Holistic Framework for Semi-Supervised Domain Adaptation". In: *International Conference on Computer Vision*. 2021.

[118] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. "Grounded Language-Image Pre-training". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[119] Shuaifeng Li, Mao Ye, Xiatian Zhu, Lihua Zhou, and Lin Xiong. "Source-Free Object Detection by Learning To Overlook Domain Style". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[120] Wuyang Li, Xinyu Liu, and Yixuan Yuan. "SIGMA: Semantic-complete Graph Matching for Domain Adaptive Object". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[121] Xianfeng Li, Weijie Chen, Di Xie, Shicai Yang, Peng Yuan, Shiliang Pu, and Yueting Zhuang. "A Free Lunch for Unsupervised Domain Adaptive Object Detection without Source Data". In: *AAAI Conference on Artificial Intelligence*. 2021.

[122] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. "MViTv2: Improved multiscale vision transformers for classification and detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[123] Yu-Jhe Li, Xiaoliang Dai, Chih-Yao Ma, Yen-Cheng Liu, Kan Chen, Bichen Wu, Zijian He, Kris Kitani, and Peter Vajda. "Cross-Domain Adaptive Teacher for Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[124] Zhizhong Li and Derek Hoiem. "Learning Without Forgetting". In: *European Conference on Computer Vision*. 2016.

[125] Jian Liang, Dapeng Hu, and Jiashi Feng. "Domain Adaptation with Auxiliary Target Domain-Oriented Classifier". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[126] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[127] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". In: *International Conference on Computer Vision*. 2017.

[128] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. "Microsoft COCO: Common Objects in Context". In: *ArXiv* (2014).

[129] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. "On the Loss Landscape of Adversarial Training: Identifying Challenges and How to Overcome Them". In: *Advances in Neural Information Processing Systems*. 2020.

[130] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *ArXiv* (2023).

[131] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision*. 2016.

[132] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. "Unbiased Teacher for Semi-Supervised Object Detection". In: *International Conference on Learning Representations*. 2021.

[133] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *International Conference on Computer Vision*. 2021.

[134] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. "Learning transferable features with deep adaptation networks". In: *International Conference on Machine Learning*. 2015.

[135] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. "Predicting Deeper into the Future of Semantic Segmentation". In: *International Conference on Learning Representations*. 2017.

[136] Tomasz Malisiewicz and Alexei A. Efros. "Recognition by Association via Learning Per-exemplar Distances". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008.

[137] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. "Ensemble of Exemplar-SVMs for Object Detection and Beyond". In: *International Conference on Computer Vision*. 2011.

[138] Syed Zain Masood, Christopher Ellis, Adarsh Nagaraja, Marshall F. Tappen, Joseph J. LaViola, and Rahul Sukthankar. "Measuring and Reducing Observational Latency when Recognizing Actions". In: *International Conference on Computer Vision*. 2011.

[139] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *International Conference on Learning Representations*. 2016.

[140] Chris Maunder and David Cunningham. *CodeProject*. `https://www.codeproject.com`. 2023.

[141] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. "Transforming Model Prediction for Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[142] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. "Learning Target Candidate Association to Keep Track of What Not to Track". In: *International Conference on Computer Vision*. 2021.

[143] Xue Mei and Haibin Ling. "Robust visual tracking using L1 minimization". In: *International Conference on Computer Vision*. 2009.

[144] Microsoft. *Azure Machine Learning*. `https://azure.microsoft.com/en-us/products/machine-learning`. 2023.

[145] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. "Simple open-vocabulary object detection". In: *European Conference on Computer Vision*. 2022.

[146] Ishan Misra and Laurens van der Maaten. "Self-Supervised Learning of Pretext-Invariant Representations". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2019).

[147] Ammar Mohammed and Rania Kora. "A comprehensive review on ensemble deep learning: Opportunities and challenges". In: *Journal of King Saud University - Computer and Information Sciences* 35.2 (2023), pp. 757–774.

[148] Alhassan Mumuni and Fuseini Mumuni. "Data augmentation: A comprehensive survey of modern approaches". In: *Array* 16 (2022), p. 100258.

[149] Muhammad Akhtar Munir, Muhammad Haris Khan, M. Saquib Sarfraz, and Mohsen Ali. "SSAL: Synergizing between Self-Training and Adversarial Learning for Domain Adaptive Object Detection". In: *Advances in Neural Information Processing Systems*. 2021.

[150] Yanjinlkham Myagmar-Ochir and Wooseong Kim. "A Survey of Video Surveillance Systems in Smart City". In: *Electronics* 12.17 (2023).

[151] Supreeth Narasimhaswamy, Huy Anh Nguyen, Lihan Huang, and Minh Hoai. "HOIST-Former: Hand-held Objects Identification, Segmentation, and Tracking in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2024.

[152] Supreeth Narasimhaswamy, Thanh Nguyen, Mingzhen Huang, and Minh Hoai. "Whose Hands are These? Hand Detection and Hand-Body Association in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[153] Supreeth Narasimhaswamy, Trung Nguyen, and Minh Hoai. "Detecting Hands and Recognizing Physical Contact in the Wild". In: *Advances in Neural Information Processing Systems*. 2020.

[154] Supreeth Narasimhaswamy, Zhengwei Wei, Yang Wang, Justin Zhang, and Minh Hoai. "Contextual Attention for Hand Detection in the Wild". In: *International Conference on Computer Vision*. 2019.

[155] Rashmiranjan Nayak, Umesh Chandra Pati, and Santos Kumar Das. "A comprehensive review on deep learning-based methods for video anomaly detection". In: *Image and Vision Computing* 106 (2021), p. 104078.

[156] Thanh Nguyen, Chau Pham, Khoi Nguyen, and Minh Hoai. "Few-Shot Object Counting and Detection". In: *European Conference on Computer Vision*. 2022.

[157] Kento Nishi, Yi Ding, Alex Rich, and Tobias Höllerer. "Augmentation Strategies for Learning With Noisy Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[158] Sebastian Nowozin and Jamie Shotton. *Action Points: A Representation for Low-latency Online Human Action Recognition*. Tech. rep. Microsoft Research Cambridge, 2012.

[159] Adeshina Sirajdin Olagoke, Haidi Ibrahim, and Soo Siang Teoh. "Literature Survey on Multi-Camera System and Its Application". In: *IEEE Access* 8 (2020), pp. 172892–172922.

[160] Darshit J. Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi, and Woong Cho. "A Review on Autonomous Vehicles: Progress, Methods and Challenges". In: *Electronics* (2022).

[161] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* 113 (2019), pp. 54–71. ISSN: 0893-6080.

[162] Luis Perez and Jason Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *ArXiv* (2017).

[163] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 1999.

[164] Prashan Premaratne, Inas Jawad Kadhim, Rhys Blacklidge, and Mark Lee. "Comprehensive review on vehicle Detection, classification and counting on highways". In: *Neurocomputing* 556 (2023), p. 126627.

[165] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[166] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. 2021.

[167] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. "Video (language) modeling: a baseline for generative models of natural videos". In: *ArXiv*. 2015.

[168] Michalis Raptis and Leonid Sigal. "Poselet Key-framing: A Model for Human Activity Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013.

[169] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *ArXiv* (2015).

[170] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *ArXiv* (2016).

[171] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *ArXiv* (2018).

[172] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. 2015.

[173] Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[174] Fahimeh Rezazadegan, Sareh Shirazi, and Larry S. Davis. "A Real-time Action Prediction Framework by Encoding Temporal Evolution". In: *ArXiv*. 2017.

[175] Adrián López Rodríguez and Krystian Mikolajczyk. "Domain Adaptation for Object Detection via Style Consistency". In: *ArXiv* (2019).

[176] Aruni RoyChowdhury, Prithvijit Chakrabarty, Ashish Singh, SouYoung Jin, Huaizu Jiang, Liangliang Cao, and Erik Learned-Miller. "Automatic adaptation of object detectors to new domains using self-training". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[177] Michael S. Ryoo. "Human Activity Prediction: Early Recognition of Ongoing Activities from Streaming Videos". In: *International Conference on Computer Vision*. 2011.

[178] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. "Strong-Weak Distribution Alignment for Adaptive Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[179] Konrad Schindler and Luc Van Gool. "Action Snippets: How Many Frames Does Human Action Recognition Require?" In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008.

[180] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. "People Tracking with a Mobile Robot Using Sample-based Joint Probabilistic Data Association Filters". In: *The International Journal of Robotics Research* 22.2 (2003), pp. 99–116.

[181] Laura Sevilla-Lara and Erik G. Learned-Miller. "Distribution fields for tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[182] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". In: *International Conference on Learning Representations*. 2017.

[183] Yuge Shi, Basura Fernando, and Richard Hartley. "Action Anticipation with RBF Kernelized Feature Mapping RNN". In: *European Conference on Computer Vision*. 2018.

[184] Chawin Sitawarin, Supriyo Chakraborty, and David A. Wagner. "Improving Adversarial Robustness Through Progressive Hardening". In: *ArXiv* (2020).

[185] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. "A Simple Semi-Supervised Learning Framework for Object Detection". In: *ArXiv* (2020).

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

[187] Arvind P. Sridhar, Chawin Sitawarin, and David Wagner. "Mitigating Adversarial Training Instability with Batch Normalization". In: *Proceedings of the International Conference on Learning Representations Workshop*. 2021.

[188] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised Learning of Video Representations using LSTMs". In: *International Conference on Machine Learning*. 2015.

[189] Baochen Sun and Kate Saenko. "Deep coral: Correlation alignment for deep domain adaptation". In: *European Conference on Computer Vision Workshop*. 2016.

[190] Kah-Kay Sung and Tomaso Poggio. "Example-Based Learning for View-Based Human Face Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1998.

[191] Johan A. K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart DeMoor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

[192] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning*. 2019.

[193] Antti Tarvainen and Harri Valpola. "Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in Neural Information Processing Systems*. Mar. 2017.

[194] Alexandru Cristian Telea. "An Image Inpainting Technique Based on the Fast Marching Method". In: *Journal of Graphics Tools* 9 (2004), pp. 23 –34.

[195] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. "FOVEA: Foveated Image Magnification for Autonomous Navigation". In: *International Conference on Computer Vision*. 2021.

[196] Chittesh Thavamani, Mengtian Li, Francesco Ferroni, and Deva Ramanan. "Learning to Zoom and Unzoom". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023.

[197] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. "FCOS: Fully Convolutional One-Stage Object Detection". In: *International Conference on Computer Vision*. 2019.

[198] Joseph Tighe and Svetlana Lazebnik. "Finding Things: Image Parsing with Regions and Per-Exemplar Detectors". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013.

[199] Sam Toyer, Anoop Cherian, Tengda Han, and Stephen Gould. "Human Pose Forecasting via Deep Markov Models". In: *International Conference on Digital Image Computing: Techniques and Applications*. 2017.

[200] Vinh Tran, Yang Wang, Zekun Zhang, and Minh Hoai. "Knowledge Distillation for Human Action Anticipation". In: *IEEE International Conference on Image Processing*. 2021.

[201] Vassilios Tsakanikas and Tasos Dagiuklas. "Video surveillance systems-current status and future trends". In: *Computers and Electrical Engineering* 70 (2018), pp. 736–753.

[202] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. "Adversarial Discriminative Domain Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[203] Vladimir Vapnik. *Statistical Learning Theory*. New York, NY: Wiley, 1998.

[204] 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*. 2017.

[205] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. "Manifold Mixup: Better Representations by Interpolating Hidden States". In: *International Conference on Machine Learning*. 2019.

[206] Vidit Vidit, Martin Engilberge, and Mathieu Salzmann. "Learning Transformations To Reduce the Geometric Shift in Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023.

[207] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Anticipating Visual Representations From Unlabeled Video". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[208] Jacob Walker, Abhinav Gupta, and Martial Hebert. "Dense optical flow prediction from a static image". In: *International Conference on Computer Vision*. 2015.

[209] Jacob Walker, Abhinav Gupta, and Martial Hebert. "Patch to the Future: Unsupervised Visual Prediction". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[210] Boyu Wang and Minh Hoai. "Back to the Beginning: Starting Point Detection for Early Recognition of Ongoing Human Actions". In: *Computer Vision and Image Understanding*. Vol. 175. 2018, pp. 24–31.

[211] Boyu Wang and Minh Hoai. "Predicting Body Movement and Recognizing Actions: an Integrated Framework for Mutual Benefits". In: *Proceedings of the International Conference on Automatic Face and Gesture Recognition*. 2018.

[212] Boyu Wang, Lihan Huang, and Minh Hoai. "Active Vision for Early Recognition of Human Actions". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[213] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *ArXiv* (2022).

[214] Chien-Yao Wang, I-Hau Yeh, and Hongpeng Liao. "You Only Learn One Representation: Unified Network for Multiple Tasks". In: *ArXiv* (2021).

[215] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions". In: *International Conference on Computer Vision*. 2021.

[216] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. "PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs". In: *Advances in Neural Information Processing Systems*. 2017.

[217] Zijun Wei and Minh Hoai. "Region Ranking SVMs for Image Classification". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[218] Peter Whittle. *Hypothesis Testing in Time Series Analysis*. Almquist and Wicksell, 1951.

[219] Geert Willems, Jan Becker, Tinne Tuytelaars, and Luc Van Gool. "Exemplar-based Action Recognition in Video". In: *British Machine Vision Conference*. 2009.

[220] Lemeng Wu, Mengchen Liu, Yinpeng Chen, Dongdong Chen, Xiyang Dai, and Lu Yuan. "Residual Mixture of Experts". In: *ArXiv* (2022).

[221] Size Wu, Wenwei Zhang, Sheng Jin, Wentao Liu, and Chen Change Loy. "Aligning Bag of Regions for Open-Vocabulary Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023.

[222] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[223] Lin Xiong, Mao Ye, Dan Zhang, Yan Gan, Xue Li, and Yingying Zhu. "Source datafree domain adaptation of object detector through domainspecific perturbation". In: *International Journal of Intelligent Systems* 36 (2021), pp. 3746 –3766.

[224] Yunqiu Xu, Yifan Sun, Zongxin Yang, Jiaxu Miao, and Yi Yang. "H^2FA R-CNN: Holistic and Hierarchical Feature Alignment for Cross-domain Weakly Supervised Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[225] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. "Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks". In: *Advances in Neural Information Processing Systems*. 2016.

[226] Ronald R. Yager. "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking". In: *IEEE Transactions on Systems, Man and Cybernetics* 18.1 (1988), pp. 183–190.

[227] Ronald R. Yager and Dimitar P. Filev. "Induced Ordered Weighted Averaging Operators". In: *IEEE Transactions on Systems, Man and Cybernetics* 29.2 (1999), pp. 141–150.

[228] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. "ST3D: Self-training for Unsupervised Domain Adaptation on 3D Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[229] Bangpeng Yao and Li Fei-Fei. "Action Recognition with Exemplar Based 2.5D Graph Matching". In: *European Conference on Computer Vision*. 2012.

[230] Xiaoju Ye. *calflops: a FLOPs and Params calculate tool for neural networks in pytorch framework*. 2023. URL: `https://github.com/MrYxJ/calculate-flops.pytorch`.

[231] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[232] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *International Conference on Computer Vision*. 2019.

[233] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and K. Takeda. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies". In: *IEEE Access* 8 (2019), pp. 58443–58469.

[234] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu. "The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection". In: *International Conference on Computer Vision*. 2013.

[235] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. "Open-vocabulary object detection using captions". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[236] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2006.

[237] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. *DINO*. 2022. URL: `https://github.com/IDEA-Research/DINO`.

[238] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection". In: *International Conference on Learning Representations*. 2023.

[239] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *ArXiv* (2017).

[240] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. "Vision-Language Models for Vision Tasks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[241] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. "Multi-Scale Vision Longformer: A New Vision Transformer for High-Resolution Image Encoding". In: *International Conference on Computer Vision*. 2021.

[242] Zekun Zhang and Minh Hoai. "Object Detection With Self-Supervised Scene Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023.

[243] Zekun Zhang, Farrukh M. Koraishy, and Minh Hoai. "Exemplar-Based Early Event Prediction in Video". In: *British Machine Vision Conference*. 2021.

[244] Zekun Zhang, Vu Quang Truong, and Minh Hoai. *Efficiency-Preserving Scene-Adaptive Object Detection*. British Machine Vision Conference. 2024.

[245] Zekun Zhang, Vu Quang Truong, and Minh Hoai. *Prompt Adaptation for Open-Vocabulary Object Detection*. under review. 2024.

[246] Liang Zhao and Limin Wang. "Task-specific Inconsistency Alignment for Domain Adaptive Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[247] Zijing Zhao, Sitong Wei, Qingchao Chen, Dehui Li, Yifan Yang, Yuxin Peng, and Yang Liu. "Masked Retraining Teacher-Student Framework for Domain Adaptive Object Detection". In: *International Conference on Computer Vision*. 2023.

[248] Yangtao Zheng, Di Huang, Songtao Liu, and Yunhong Wang. "Cross-domain Object Detection through Coarse-to-Fine Feature Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[249] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, et al. "Regionclip: Region-based language-image pretraining". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2022.

[250] Zihan Zhong, Zhiqiang Tang, Tong He, Haoyang Fang, and Chun Yuan. "Convolution Meets LoRA: Parameter Efficient Finetuning for Segment Anything Model". In: *International Conference on Learning Representations*. 2024.

[251] Kankan Zhou, Eason Lai, and Jing Jiang. "VLStereoSet: A Study of Stereotypical Bias in Pre-trained Vision-Language Models". In: *Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*. 2022.

[252] Linchao Zhu and Yi Yang. "ActBERT: Learning Global-Local Video-Text Representations". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[253] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. "Deformable DETR: Deformable Transformers for End-to-End Object Detection". In: *ArXiv* (2020).

[254] Yitao Zhu, Zhenrong Shen, Zihao Zhao, Sheng Wang, Xin Wang, Xiangyu Zhao, Dinggang Shen, and Qian Wang. "MeLo: Low-rank Adaptation is Better than Finetuning for Medical Image Diagnosis". In: *ArXiv* (2023).