Training Domain-invariant Object Detector Faster with Feature Replay and Slow Learner

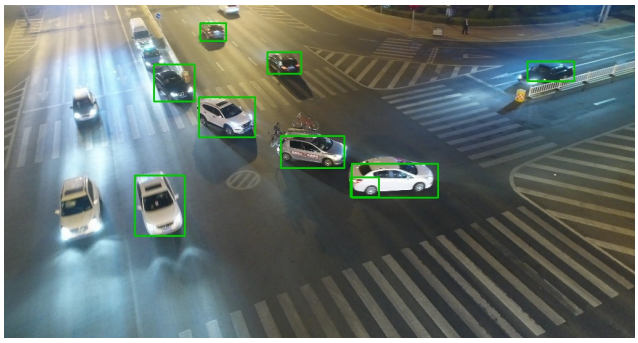
Chaehyeon Lee[1*] Junghoon Seo[2] Heechul Jung[1†]
[1]Department of Artificial Intelligence, Kyungpook National University, Daegu, Korea
[2] SI Analytics, Co., Ltd., Daejeon, Korea
[1]{123456ccdd, heechul}@knu.ac.kr [2]jhseo@si-analytics.ai

Abstract

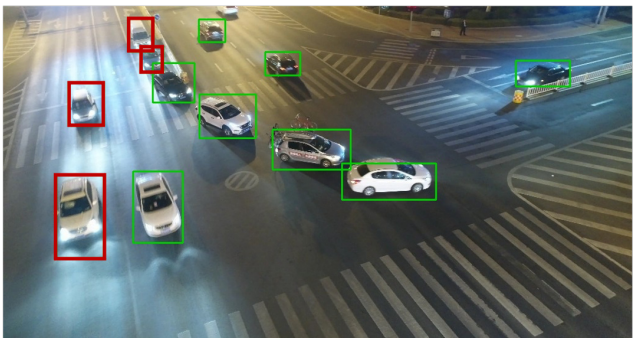
In deep learning-based object detection on remote sensing domain, nuisance factors, which affect observed variables while not affecting predictor variables, often matters because they cause domain changes. Previously, nuisance disentangled feature transformation (NDFT) was proposed to build domain-invariant feature extractor with with knowledge of nuisance factors. However, NDFT requires enormous time in a training phase, so it has been impractical. In this paper, we introduce our proposed method, A-NDFT, which is an improvement to NDFT. A-NDFT utilizes two acceleration techniques, feature replay and slow learner. Consequently, on a large-scale UAVDT benchmark, it is shown that our framework can reduce the training time of NDFT from 31 hours to 3 hours while still maintaining the performance. The code will be made publicly available online[1].

1. Introduction

Nuisance factors matter everywhere in deep learning-based object detection in remote sensing field. In imagery obtained from an unmanned aerial vehicle (UAV), shooting height, shooting angle, and shooting time significantly influence the appearance of scene and objects [4, 44]. In imagery obtained from satellites, the type of satellite, ground sample distance, azimuth/altitude angle of the satellite, azimuth/altitude angle of the sun, and various environmental factors work in the same way [1, 34]. These nuisance variables are marginally independent from target variables, but affect observation variables [37, 30, 17]. Therefore, the learned representation in the deep object detector can be dependent on nuisance variables. It induces over-fitting biased towards frequently appearing nuisance variables.



(a) Baseline



(b) A-NDFT

Figure 1: Detection examples and comparison with Faster R-CNN baseline and the corresponding A-NDFT. Red boxes denote new correct detections where the baseline failed to detect, but A-NDFT did.

In the perspective of domain shift, the issue of nuisance factors in object detection has been addressed from various domain adaptation approaches. [13] addressed the domain adaptation problem from the perspective of noise-resistant robust learning. [6, 5, 31] leveraged image composition between seen object instances and unseen scenes. [39] proposed a novel graphical framework on category-level domain alignment. [16] adopted multiple experts to capture variations of the object appearance better. Each of these

*Work done as intern at SI Analytics.
†Corresponding author.
[1]https://github.com/2-Chae/A-NDFT

methodologies has revealed its strengths, but most of them are not suitable to deal with the nuisance factors appearing in the remote sensing field described above. This is because these methods cannot handle massive numbers of fine-grained domains.

Instead, in this paper, we focus on the target method, nuisance disentangled feature transformation (NDFT) [35]. Motivated from privacy-preserving visual recognition [36] and disentangled feature learning [19], NDFT makes use of an adversarial learning framework to build a domain-robust object detector. Unlike many of the previously mentioned methodologies, it is suitable for dealing with the massive number of nuisance factors. Moreover, to the best of our knowledge, NDFT's original literature is the pioneering work to show the effectiveness of fine-grained domain-invariant learning in UAV images. [35] reported that in the UAVDT benchmark accompanying the Faster R-CNN model, the introduction of NDFT had increased mAP by over 2%. It results from using only meta-data that can be obtained for free without any additional labeling work. For this reason, NDFT is expressed as 'free lunch' in their paper.

Contrary to this attraction, in this paper, we point out that the domain robustness obtained with NDFT is not actually a free lunch. In other words, the use of NDFT requires a lot of training time. We analyze the causes of these phenomena and suggest alternative strategies to train NDFT models. Specifically, our contributions in this paper are as follows:

- We pointed out the problem of slow training of NDFT and analyzed the causes of it.
- Based on the problem analysis, we present an alternative learning methodology for NDFT. One is the feature replay, and another is the slow learner.
- The experiment results in the UAVDT benchmark show that our proposed A-NDFT can significantly reduce the training time while maintaining the performance of the existing NDFT.

2. Related Works

2.1. Domain Shift of Remote Sensing

Domain shift problem is ubiquitous in remote sensing applications. Several works have reported that machine learning models on remote sensing suffered domain shift caused by various nuisance factors, e.g., viewpoint geometry, atmospheric effect, temporal variability, and sensor properties [2]. In [34], it was discussed that most of the existing satellite datasets were collected with only the "at nadir" data and thus showed inferior performance in off-nadir observation data. [14, 35] observed the performance drop of object detector according to the view angle, which is trained on VisDrone dataset [44] and UAVDT dataset [4]. [29] observed the phenomenon that the model performance deteriorated due to the change in color distribution even when sensing different regions with the same satellite.

Under awareness of the problem, a great deal of research has dealt with domain adaptation problem in remote sensing [27, 21, 3, 15, 28, 29]. Most of the existing work has dealt with domain adaptation for classification or semantic segmentation. Moreover, these studies have assumed a multi-source domain adaptation situation in which the source domain and the target domain are entirely separated. On the contrary, our study is based on the NDFT model, so we aim to construct a robust object detector in fine-grained domains.

2.2. Adversarial Learning for Domain Invariance

There are quite a number of studies using adversarial frameworks to impart domain invariance to deep learning models [8, 38, 42, 11, 12, 41, 32]. [8] introduced domain-adversarial neural network (DANN), an adversarial framework between domain predictor and feature extractor. In DANN, the feature extractor is trained to decrease the task loss in the source domain and increase the loss of the domain predictor. In [38], adversarial learning for training the feature extractor was performed in the form of maximizing the entropy of the domain predictor's output. In addition, studies on domain invariance through adversarial learning have been dealt with in various aspects such as neural network architecture [12], extension to multiple domains [42], stabilization techniques of training [11, 41], and methods of dealing with continuous domains [32].

Similarly, our study explores the NDFT model that acquires domain invariance through adversarial learning. Therefore, it can be seen that this study also deals with this category of topics. Nevertheless, at the same time, our work only focuses on convergence speed and acceleration techniques of the NDFT model. Therefore, our research has a point that is differentiated from the viewpoint of the existing researches.

3. Problem Formulation and Motivations

3.1. Domain-invariant Feature for Object Detection

Let $\mathcal{X}$ be the domain of the input image and $\tilde{\mathcal{X}}$ be the space of the intermediate features. The domain to represent localization and classification results of each possible region obtained by the detection task is denoted as $\mathcal{P}$. The backbone network $f_T(\cdot;\theta_T) : \mathcal{X} \mapsto \tilde{\mathcal{X}}$ takes the input image $X \in \mathcal{X}$ and produces an intermediate feature $f_T(X;\theta_T)$. The detection task network $f_O(\cdot;\theta_O) : \tilde{\mathcal{X}} \mapsto \mathcal{P}$ receives this feature and calculates classification and localization for all possible region candidates. When the two networks are combined (i.e. $f_O(f_T(\cdot;\theta_T);\theta_O) : \mathcal{X} \mapsto \mathcal{P}$), it can be interpreted as a generic object detector. By abuse

Algorithm 1: NDFT: Nuisance Disentangled Feature Transform

Input : $f_T(\cdot;\theta_T) : \mathcal{X} \mapsto \tilde{\mathcal{X}}$, a backbone network
$f_O(\cdot;\theta_O) : \tilde{\mathcal{X}} \mapsto \mathcal{P}$, a module for object detection task branch
$\{f_{N,i}(\cdot;\theta_{N,i})\}_{i=1}^{k} : \tilde{\mathcal{X}} \mapsto \mathcal{Y}_{\mathcal{N}}$, an ordered set of k modules for the nuisance prediction branches
D, M-size training dataset consisting of each triplet (X, Y_O, Y_N)

Required: $\{\gamma_i\}_{i=1}^{k}$, an ordered set of weight coefficients
T, a number of training iterations
α, a threshold value for nuisance prediction performance
ψ, a hyper-parameter indicating restart cycle of nuisance predictor
η_U and η_N, two scalar values for learning rate

Defined : $\theta_U \leftarrow \theta_T \cup \theta_O$ and $\theta_{N.U} \leftarrow \theta_{N,1} \cup \cdots \cup \theta_{N,k}$

Output : $f_O(f_T(\cdot;\theta_T);\theta_O) : \mathcal{X} \mapsto \mathcal{P}$, domain-invariant object detector

for t ***in*** range(T) **do**
  Sample a mini-batch of n data $\{(X^j, Y_O^j, Y_N^j)\}_{j=1}^{n}$ from D;
  $\theta_U \leftarrow \theta_U - \eta_U \nabla_{\theta_U} \frac{1}{n} \sum_{j=1}^{n} \left[L_O(f_O(f_T(X^j)), Y_O^j) + \sum_{i=1}^{k} \gamma_i L_{ne}(f_{N,i}(f_T(X^j))) \right]$;
  while $\min_{1 \le i \le k} \left[\frac{1}{n} \sum_{j=1}^{n} [\mathbb{1}(\arg\max(f_{N,i}(f_T(X^j))) = Y_{N,i}^j)] \right] \le \alpha$ **do**
    $\theta_{N,U} \leftarrow \theta_{N,U} - \eta_N \nabla_{\theta_{N,U}} \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{k} L_N(f_{N,i}(f_T(X^j)), Y_{N,i}^j)$;
  end
  if $t \% \psi = 0$ **then**
    Re-initialize $\theta_{N,U}$;
  end
end

of notation, we do not consider the difference between one-stage [18, 43] and two-stage detectors [23] for simplicity. In addition, the model parameter is omitted when it is obvious enough that it is not necessary to specify it, e.g., $f_T(\cdot;\theta_T) = f_T(\cdot)$. The loss function for training this object detector is denoted as $L_O(\cdot,\cdot) : \mathcal{P} \times \mathcal{Y}_{\mathcal{O}} \mapsto \mathbb{R}$, and L_O is often defined as the weighted sum of the localization loss and the classification loss.

In our setup, we have D, M-size dataset as the training data. Every single instance of D is composed of triplet (X, Y_O, Y_N). $X \in \mathcal{X}$, $Y_O \in \mathcal{Y}_{\mathcal{O}}$, and $Y_N \in \mathcal{Y}_{\mathcal{N}}$ are information about image data, object detection labeling, and nuisance factors, respectively. For Y_N to be a nuisance factor, Y_O and Y_N must be marginally independent. This means that the generation process for the observation of X is dependent on Y_O and Y_N, but Y_O and Y_N are independently generated. For example, in object detection on an outdoor UAV image, the location and scale of the detection target are usually independent of whether it is day or night at the time of shooting. However, the appearance of the image may depend on the object in the image and the photographed time.

Typically, the object detector $f_O(f_T(\cdot))$ is trained to map from the image X to the object label Y_O. It can over-fit because of X's dependence on Y_N. The variation of X due to the change in Y_N is expressed as X', and $f_O(f_T(X))$ may be different from $f_O(f_T(X'))$. If the vast majority of image data included in the training dataset were taken during the day, this model might perform poorly on images taken at night. This intuition motivates the feature extractor's domain-invariant properties. In other words, we would like to make the backbone network $f_T(\cdot)$ to have the domain-invariant property $f_T(X) = f_T(X')$.

3.2. Nuisance Disentangled Feature Transform

For building a domain-invariant object detector, [35] formulated fine-grained cross-domain object detection as an adversarial training framework [9]. Motivated by privacy-preserving visual recognition [36], they proposed three-party game among where three players, a backbone network $f_T(X;\theta_T)$, a detection task network $f_O(\cdot;\theta_O)$, and a nuisance predictor $f_N(\cdot:\theta_N) : \tilde{\mathcal{X}} \mapsto \mathcal{Y}_{\mathcal{N}}$.

The nuisance predictor f_N is usually a softmax classifier that predicts Y_N by receiving features from the backbone network. Therefore, the nuisance predictor f_N is trained to minimize the loss L_N, which is the surrogate of the nuisance prediction performance, e.g., multi-class cross-entropy loss. Conversely, the backbone network f_T and the detection task network f_O are trained to maximize the nuisance prediction loss L_N while minimizing the object detection task loss L_O. When considering multiple nuisance

prediction tasks, the training procedure of the three modules is formulated with an alternative optimization of the following two objectives:

$$\begin{aligned} &\arg\min_{\theta_O,\theta_T} L_O(f_O(f_T(X)),Y_O) - \sum_{i=1}^{k} \gamma_i L_N(f_{N,i}(f_T(X)),Y_{N,i}), \\ &\arg\min_{\theta_{N,1},\cdots,\theta_{N,k}} L_N(f_{N,i}(f_T(X)),Y_{N,i}) \end{aligned} \tag{1}$$

where k denotes the number of nuisance factors, and the i-th elements related to them are denoted as $(\cdot)_{N,i}$. $\gamma_i \in \mathbb{R}_{\geq 0}$ is weight coefficient for balancing L_O and L_N.

To avoid convergence problems in GAN training [7], the authors of [35] designed a sophisticated training strategy. This novel training strategy includes the use of negative entropy loss [20], performance monitoring of nuisance predictors, and re-starting tricks [36]. They named the $f_T(\cdot;\theta_T)$ learned by the training loop, including this modification as the Nuisance Disentangled Feature Transform (NDFT). Algorithm 1 depicts the main training loop of NDFT, and the three essential details described in each line are:

1. Line 3: Instead of using gradient reversal trick [8] (i.e., using $-L_N$), the negative entropy of the softmax vector, L_{ne}, are adopted as the adversarial loss. This option was also introduced in other works [20, 41].

2. Line 4-6: To prevent each $f_{N,i}$ from becoming too weak, the training performances of all k nuisance prediction tasks are monitored. In a single alternative optimization, $\theta_{N,U}$ is updated until the prediction accuracy of all nuisance tasks is greater than α.

3. Line 7-9: To help prevent falling into bad local minima, we reinitialize $\theta_{N,U}$ every ψ iterations. It is motivated from [36].

The original work of NDFT reported that by using (almost free) meta-data, the performance of the standard Faster R-CNN [23] in the UAVDT benchmark dataset [4] could increase mAP above 2%. For more details on NDFT, refer to its original work [35].

3.3. Slow Convergence of NDFT

Although several UAV object detection benchmarks revealed NDFT's effectiveness, we point out its serious issue for practice usage. That is, NDFT requires enormous training time. In our early experiments, we tried to run the author's official code of NDFT[2], and we found that the training time of NDFT on UAVDT dataset was above 30 hours. In contrast, we also found that training their baseline, a standard Faster R-CNN model, required only about three hours. This training time difference indicates that NDFT is not truly 'free lunch' in the aspect of computation and time cost.

We point to several causes of NDFT's slow training speed. First, the nuisance prediction parameters $\theta_{N,U}$ is updated with the backbone network parameters θ_U fixed. Thus, repeated feed-forward operations through the backbone network (i.e., $f_T(X^j)$) in Line 5 result in computational inefficiency. Second, in the object detection network, it usually needs a lot of training iteration to fine-tune specific predictors with the parameters of the backbone network fixed [33]. Considering the periodic parameter initialization heuristics in Line 7-9 together, the strict performance monitoring policy (Line 4) can significantly reduce the overall training speed. As mentioned in Section 3.2, they control the over-fitting and under-fitting of the nuisance predictor. Therefore, it is not easy to achieve both this adjustment and fast training by simply controlling hyper-parameters α and ψ. We may need another strategy to achieve rapid training while adjusting them appropriately.

4. Our Approach for Acceleration of NDFT

4.1. Feature Replay

NDFT [35] extracts features two times during each training iteration when training (1) NDFT module and object detection module and (2) nuisance prediction modules. It uses different mini-batches of examples, respectively. However, we point out that the NDFT module's feature extraction process is computationally expensive and time-consuming. So here we introduce feature replay, which avoids this redundancy and guarantees faster learning.

We implement feature replay as a pooling queue containing features from the past steps. During each training iteration, features are extracted from the feature extractor, and they are used as an input of the object detection branch and the nuisance prediction branches, respectively. Before moving on to the next step, we store these features in the pooling queue (Figure 2). Note that we store features as many as a batch size at once; the computational gain is also dependent on the batch size. When training the nuisance prediction branches, we do not extract features but use the top element of the queue, thereby cutting down on training time. However, in this way, the nuisance prediction branches might lose information about past inputs as they only learn mini-batches at that point. We train the nuisance prediction branches with all the elements in the queue for every k iterations to learn without losing (Figure 3, Right).

Our feature replay can be seen as similar to 'image history' in SimGAN [26] or 'experience replay' in reinforcement learning [22] and continual learning [25]. These are intended to avoid catastrophic forgetting [24] of neural networks and stabilize the training process. However, unlike other studies, our feature replay focuses on the redundancy of computation and stores data points in feature space $\tilde{\mathcal{X}}$,

[2] https://github.com/VITA-Group/UAV-NDFT

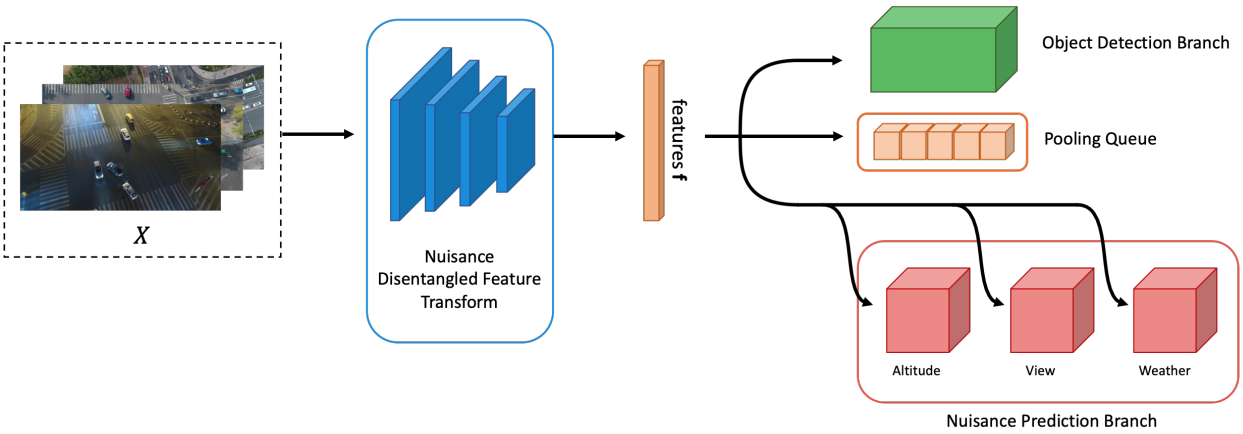


Figure 2: Our proposed framework (A-NDFT). We have added a pooling queue used for feature replay to the original NDFT framework. The encoded feature will be used for object detection and nuisance prediction, just as NDFT. Not only that, we store the feature in the pooling queue to avoid an additional feed-forward process of the backbone network.

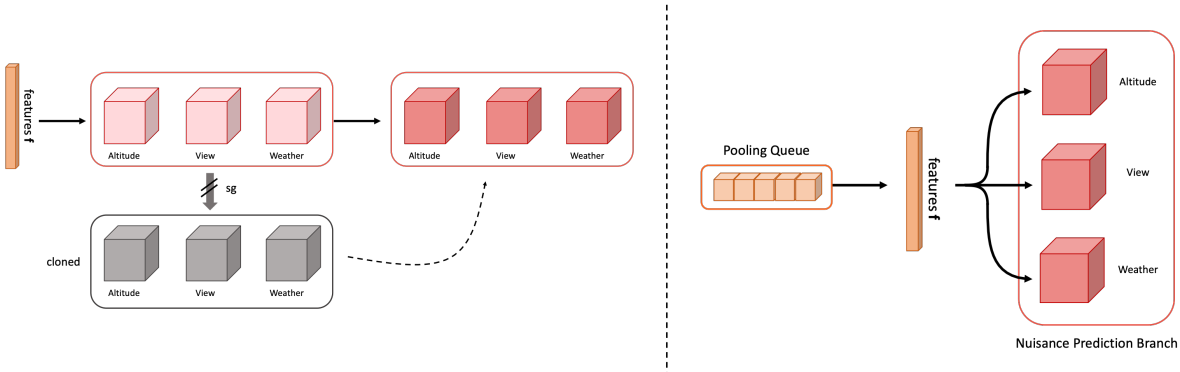


Figure 3: Training procedure of nuisance prediction branches. We adopt exponential moving average on parameters to avoid forgetting past information while accepting current information. Here, *sg* means stop gradient (Left: slow runner). When training nuisance branches for each iteration, we use features extracted and stored in the pooling queue when we learned object detection branches from the previous step (Right: feature replay).

not in data space $\mathcal{X}$.

4.2. Slow Learners

We notice that the nuisance prediction branches' performances keep fluctuating when training because, for every iteration, we train the branches with the top element of the queue of which the nuisance attributes may not be evenly distributed. As a result, they continue to forget past information while accepting the current ones. We want them to learn slowly to make sure they are sufficiently trained, so we adopt exponential moving average (EMA) of parameters. We use EMA operation on nuisance predictors' parameters to keep the average representation and accept new data simultaneously, which slowly trains the model.

As in the left of Figure 3, we have model parameters before an update. We clone and annotate them as θ. These parameters θ are not updated during training; for that reason, we denote stop-gradient (sg) operation [10] in that figure. Let the current model parameters be ξ which have equivalent to θ at this moment. Now we compute the parameters of the next step as:

$$\xi \leftarrow \beta\xi + (1-\beta)\theta \tag{2}$$

where β is a decay rate between 0 and 1.

Although this update method has almost the same context as decreasing the learning rate in finding the center point between the current and historical parameters, we can still reduce the effort of finding the appropriate hyperparameters. Moreover, our slow learner is closely related to the EMA-GAN [40]. However, unlike EMA-GAN, which

Algorithm 2: A-NDFT: Our Modified Version of Nuisance Disentangled Feature Transform

Input : Same with Algorithm 1

Required: $\{\gamma_i\}_{i=1}^{k}$, an ordered set of weight coefficients
T, a number of training iterations
Q_s, a s-size FIFO queue for restoring intermediate features
β, the degree of weighting decrease for slow learning
ϕ, a hyper-parameter to indicate training cycle using Q_s
η_U and η_N, two scalar values for learning rate

Defined : $\theta_U \leftarrow \theta_T \cup \theta_O$ and $\theta_{N,U} \leftarrow \theta_{N,1} \cup \cdots \cup \theta_{N,k}$

Output : $f_O(f_T(\cdot;\theta_T);\theta_O) : \mathcal{X} \mapsto \mathcal{P}$, domain-invariant object detector

1 **for** t_o ***in*** $\mathrm{range}(T)$ **do**

2 Sample a mini-batch of n data $\{(X^j, Y_O^j, Y_N^j)\}_{j=1}^{n}$ from D;

3 $\mathbf{F} \leftarrow \{f_T(X^j)\}_{j=1}^{n}$ and Q_s.enqueue($\mathbf{F}$);

4 $\theta_U \leftarrow \theta_U - \eta_U \nabla_{\theta_U} \frac{1}{n} \sum_{j=1}^{n} \left[L_O(f_O(F_j), Y_O^j) + \sum_{i=1}^{k} \gamma_i L_{ne}(f_{N,i}(F_j)) \right]$ where $F_j = f_T(X^j)$;

5 $\theta_{N,U} \leftarrow \beta\theta_{N,U} + (1-\beta)\left\{ \theta_{N,U} - \eta_N \nabla_{\theta_{N,U}} \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{k} L_N(f_{N,i}(F_j), Y_{N,i}^j) \right\}$;

6 **if** $t\%\phi = 0$ **then**

7 **for** t_i ***in*** $\mathrm{range}(\mathrm{floor}(Q_s.\mathrm{num\_of\_items}/n))$ **do**

8 Sample a mini-batch of n features $\{F^j\}_{j=1}^{n}$ from Q_s where each $F^j \in \tilde{\mathcal{X}}$;

9 $\theta_{N.U} \leftarrow \theta_{N,U} - \eta_N \nabla_{\theta_{N,U}} \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{k} L_N(f_{N,i}(F^j), Y_{N,i}^j)$;

10 **end**

11 **end**

12 **end**

aims to increase the stability and performance of the generative model, we show that slow learning of the adversarial predictors is also effective in fine-grained domain adaptation.

4.3. Overall Algorithm

Here, we present an overall training procedure of our proposed algorithm, A-NDFT. We handle the model training process delicately so that there is no performance difference from the existing technique NDFT but shorten training times. This procedure is summarized in Algorithm 2 and explained below.

As our A-NDFT is a faster version of the original NDFT, we follow almost all the steps from them. Just like NDFT [35], we jointly optimize the backbone model and object detection branch by minimizing the same objective equation as NDFT does. At this point, we store extracted features in the pooling queue to avoid the redundant feed-forward process of the backbone network. After that, NDFT tried to keep monitoring nuisance prediction branches so that their performance does not go down. With its performance monitoring strategy, we observed the fluctuation of these branches' performance while training. We assumed that this comes from an imbalance of mini-batch examples and oblivion of past information. To overcome this, we decided to train nuisance prediction branches with the top element of the queue and apply the slow learner technique, the exponential moving average update on module weights. By updating weights to average points of current and past weights, modules slowly forget less and learn new information (Algorithm 2 Line 5). Besides, to guarantee the adequate performances of branches, we retrain the modules with whole elements of the pooling queue every ϕ iterations (Algorithm 2 Line 6-11).

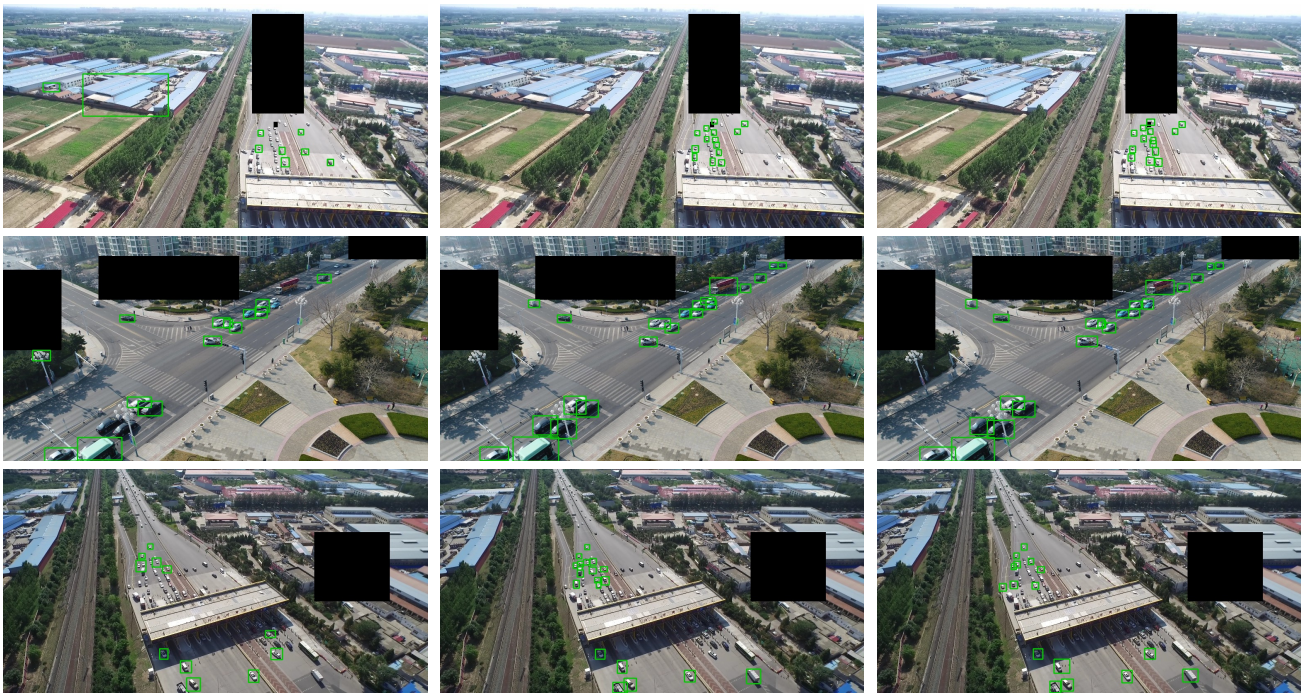
5. Experiments

5.1. Experiment Settings

Dataset: UAVDT [4] The Unmanned Aerial Vehicle Detection and Tracking (UAVDT) is large-scale object detection and tracking benchmark dataset obtained by UAVs. The objects of interest in this benchmark are vehicles annotated over 2,700 with 0.84 million bounding boxes. This benchmark dataset has about 80k frames consisted of 100 video sequences captured from UAVs in various scenarios. Moreover, it provides fully-annotated 14 kinds of attributes for each frame. We use only three categories of them for evaluating our model on the object detection track on UAVDT; flying altitudes, weather conditions, and camera views.

	Baseline [23]	A	V	W	A+V	A+W	V+W	A+V+W	A+V+W (NDFT)	A+V+W [35]
	Flying Altitude									
Low	54.19	55.06	52.48	55.17	46.68	55.27	55.64	**56.91**	57.46	74.84
Med	39.48	43.04	39.13	44.74	34.76	45.16	38.62	**46.20**	46.60	56.24
High	7.54	10.10	5.82	11.06	5.65	11.15	**11.42**	8.42	10.10	20.55
	Camera View									
Front	47.10	48.03	46.58	48.07	39.24	48.40	47.51	**52.86**	53.26	64.88
Side	37.87	39.33	37.29	39.16	30.44	38.90	36.79	**39.56**	40.11	67.50
Bird	73.35	66.40	61.55	68.44	58.58	67.57	**76.30**	72.89	86.71	28.79
	Weather Condition									
Day	45.50	47.39	41.30	47.45	38.14	48.43	45.39	**49.24**	49.40	45.91
Night	49.18	49.19	43.79	50.51	41.53	48.14	49.40	**51.13**	53.07	64.16
Overall	45.59	46.51	43.90	46.82	37.90	47.00	45.43	**48.12**	48.37	47.91

Table 1: Performance of A-NDFT-Faster-RCNN with multiple attribute disentanglement. Note, A+V+W (NDFT) is the result of reproduced NDFT with our settings and A+V+W [35] is the original result from [35].



(a) Baseline (b) NDFT (c) A-NDFT

Figure 4: Examples of the proposed A-NDFT that performs object detection on the UAVDT benchmark. Our A-NDFT framework performs better than the baseline and shows comparable performance to NDFT. Best viewed in color at high-resolution (black rectangles in figures are ignored regions provided by UAVDT benchmark itself).

Flying Altitudes have three levels; low, medium, and high. Weather conditions include daylight, night, and fog. Nevertheless, we did not use fog examples just as NDFT because of its small size. Camera views have three different views, *i.e.*, front-view, side-view, and bird-view. In addition to these three views, there are examples with a front-side view in the dataset. These are frames taken at the border between front and side. The processing of them is not specified in [35], so we decide to merge them into side-view category. By merging frames with front-side view attribute into side-view frames, the number of frames belonging to each attribute can differ from NDFT as a whole, which can raise a discrepancy with NDFT's performance. We will denote these three nuisances as **A**, **W** and **V**, respectively.

Implementation Details We used the same codebase of the NDFT's official repository[2], a Faster-RCNN model using ResNet-101 as a backbone model. γ_1, γ_2, and γ_3 denote the coefficients for altitude, view, and weather nuisances, respectively. To maintain their official setting as unchanged

as possible, we initially tried to train the model for five epochs and size-up the batch to 32 and the learning rates η_U and η_O to 0.08. However, we only report the NDFT's performance up to three epochs because it takes more than two days to train NDFT for five epochs and NDFT showed a tendency to converge at three epochs. In the case of A-NDFT, we trained the model for five epochs and set the size of the pooling queue as $s = 256$. Nevertheless, even in this setting, A-NDFT ends up 10x faster compared to NDFT. We set hyper-parameters indicating the degree of weighting β decrease for slow learning and a training cycle ϕ using the entire pooling queue as 0.99 and 325, respectively. Also, we trained A-NDFT and baseline using the same batch size and learning rate as NDFT.

When training the baseline by setting all γs as 0, we obtained an AP of 45.59% (using IoU threshold = 0.7), similar to 45.64%, which is reported in [35]. Since this model is equivalent to the standard Faster R-CNN model, it is indicated as the baseline.

5.2. Detection Performance of A-NDFT

We perform the experiments on UAVDT for object detection. First, we study the impact of just using each nuisance type (**A**, **V**, and **W**) and then combine those into two or three nuisance types. As [35] reported, applying $\gamma = 0.01$ results in the best performance; we also apply all the values of γ_i as 0.01 in this experiment. Table 1 shows the overall results by incrementally adding adversarial losses to training. **A**, **V**, and **W** are when the branch corresponding to flying altitude, camera view, and weather condition nuisance is used for training. For example, **A** is when set γ_V, γ_W to 0 and γ_A not to 0. **A+V+W** represents the experiment when all three nuisance branches are used in training, which means all γs are not 0.

When training with only one nuisance module, the performance was the best when using **W**, and in the case of two branches, the performance was the best when using **A+W**. Their improvements over the baseline are +1.23% and +1.41%, respectively. When all the branches were used, we could see that the performance was the best in all cases. For a detailed comparison, we added the performance of NDFT from [35], and a reproduced NDFT to the rightmost side of the table. The discrepancy with the value in [35] could arise from the differences in the data preprocessing as mentioned in 5.1. Precisely, the number of frames belonging to each attribute may not match [35]'s setting, so there are some differences in per-nuisance performances. However, the overall case has a similar result to the performance of [35] because it trains using all data without considering attributes. Consequently, the performance difference between A-NDFT and NDFT is not significant, which means that A-NDFT maintains the performance of NDFT. Figure 4 shows some visual examples for a qualitative comparison.

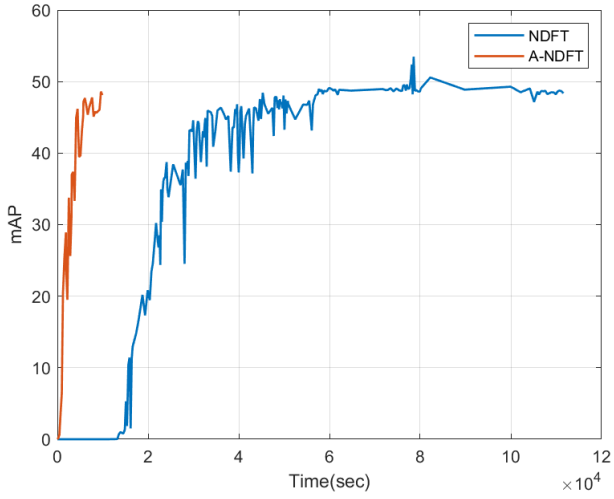


Figure 5: A comparison of mAP over training time for A-NDFT and NDFT. Note that the NDFT was trained for three epochs, whereas our A-NDFT was trained for five epochs.

5.3. Convergence Speed: A-NDFT versus NDFT

In this section, we give a comparison of convergence speeds of A-NDFT and NDFT. We use the same architecture specified in 5.1. In training NDFT, based on the implementation of [35], we increased the batch size by 10x and reduced the epochs proportionally to make the most of GPU resources and cut down on the learning time. NDFT took about 31 hours to train, while A-NDFT finished in about 3 hours. We visualize their performance over time in Figure 5. By utilizing the feature replay and the slow learner, A-NDFT significantly reduces training time and speeds up convergence than NDFT.

6. Conclusion

In this paper, we present A-NDFT, which uses a feature replay and a slow learner to accelerate the training of NDFT. The proposed method saves ten times as much time as and still maintains performance analogous to NDFT. We conducted vehicle detection on the UAVDT dataset, a large-scale benchmark, and showed that it performs better than baseline, not using nuisance factor predictions, and performs comparably with NDFT. In the future, we intend to carry out further analyses of various remote-sensing datasets such as and VisDrone [44] and use the other backbone architectures.

Acknowledgements

This work was supported by the Defense Challengeable Future Technology Program of the Agency for Defense Development, Republic of Korea.

References

[1] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018. 1

[2] Wojciech Czaja, Neil Fendley, Michael Pekala, Christopher Ratto, and I-Jeng Wang. Adversarial examples in remote sensing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 408–411, 2018. 2

[3] Xueqing Deng, Hsiuhan Lexie Yang, Nikhil Makkar, and Dalton Lunga. Large scale unsupervised domain adaptation of segmentation networks with adversarial learning. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 4955–4958. IEEE, 2019. 2

[4] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–386, 2018. 1, 2, 4, 6

[5] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 364–380, 2018. 1

[6] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017. 1

[7] Farzan Farnia and Asuman Ozdaglar. Do GANs always have Nash equilibria? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3029–3039. PMLR, 13–18 Jul 2020. 4

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, 07–09 Jul 2015. PMLR. 2, 4

[9] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 3

[10] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 5

[11] Yusuke Iwasawa, Kei Akuzawa, and Yutaka Matsuo. Stabilizing adversarial invariance induction from divergence minimization perspective. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1955–1962. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track. 2

[12] Ayush Jaiswal, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed, and Premkumar Natarajan. Invariant representations through adversarial forgetting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4272–4279, 2020. 2

[13] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 480–490, 2019. 1

[14] Benjamin Kiefer, Martin Messmer, and Andreas Zell. Leveraging domain labels for object detection from uavs. *arXiv preprint arXiv:2101.12677*, 2021. 2

[15] Yohei Koga, Hiroyuki Miyazaki, and Ryosuke Shibasaki. A method for vehicle detection in high-resolution satellite images that uses a region-based object detector and unsupervised domain adaptation. *Remote Sensing*, 12(3):575, 2020. 2

[16] Hyungtae Lee, Sungmin Eum, and Heesung Kwon. Me r-cnn: Multi-expert r-cnn for object detection. *IEEE Transactions on Image Processing*, 29:1030–1044, 2019. 1

[17] Yujia Li, Kevin Swersky, and Richard Zemel. Learning unbiased features. *arXiv preprint arXiv:1412.5244*, 2014. 1

[18] 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*, pages 21–37. Springer, 2016. 3

[19] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Multi-task adversarial network for disentangled feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[20] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Multi-task adversarial network for disentangled feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3743–3751, 2018. 4

[21] Xiaoqiang Lu, Tengfei Gong, and Xiangtao Zheng. Multisource compensation network for remote sensing cross-domain scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(4):2504–2515, 2019. 2

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 4

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 3, 4, 7

[24] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. 4

[25] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018. 4

[26] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017. 4

[27] Shaoyue Song, Hongkai Yu, Zhenjiang Miao, Qiang Zhang, Yuewei Lin, and Song Wang. Domain adaptation for convolutional neural networks-based remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 16(8):1324–1328, 2019. 2

[28] Onur Tasar, Alain Giros, Yuliya Tarabalka, Pierre Alliez, and Sébastien Clerc. Daugnet: Unsupervised, multisource, multitarget, and life-long domain adaptation for semantic segmentation of satellite images. *IEEE Transactions on Geoscience and Remote Sensing*, 2020. 2

[29] Onur Tasar, Yuliya Tarabalka, Alain Giros, Pierre Alliez, and Sébastien Clerc. Standardgan: Multi-source domain adaptation for semantic segmentation of very high resolution satellite images by data standardization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 192–193, 2020. 2

[30] Joshua B Tenenbaum and William T Freeman. Separating style and content. *Advances in neural information processing systems*, pages 662–668, 1997. 1

[31] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, Ambrish Tyagi, James M Rehg, and Visesh Chari. Learning to generate synthetic data via compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 461–470, 2019. 1

[32] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9898–9907. PMLR, 13–18 Jul 2020. 2

[33] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. In *International Conference on Machine Learning*, July 2020. 4

[34] Nicholas Weir, David Lindenbaum, Alexei Bastidas, Adam Van Etten, Sean McPherson, Jacob Shermeyer, Varun Kumar, and Hanlin Tang. Spacenet mvoi: a multi-view overhead imagery dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 992–1001, 2019. 1, 2

[35] Zhenyu Wu, Karthik Suresh, Priya Narayanan, Hongyu Xu, Heesung Kwon, and Zhangyang Wang. Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1201–1210, 2019. 2, 3, 4, 6, 7, 8

[36] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 606–624, 2018. 2, 3, 4

[37] Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. Controllable invariance through adversarial feature learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[38] Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. Controllable invariance through adversarial feature learning. *arXiv preprint arXiv:1705.11122*, 2017. 2

[39] Minghao Xu, Hang Wang, Bingbing Ni, Qi Tian, and Wenjun Zhang. Cross-domain detection via graph-induced prototype alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12355–12364, 2020. 1

[40] Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in gan training, 2019. 5

[41] Ye Yuan, Wuyang Chen, Tianlong Chen, Yang Yang, Zhou Ren, Zhangyang Wang, and Gang Hua. Calibrated domain-invariant learning for highly generalizable large scale re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3589–3598, 2020. 2, 4

[42] Han Zhao, Shanghang Zhang, Guanhang Wu, José M. F. Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[43] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 3

[44] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, and Haibin Ling. Vision meets drones: Past, present and future. *arXiv preprint arXiv:2001.06303*, 2020. 1, 2, 8