How Important are Data Augmentations to Close the Domain Gap for Object Detection in Orbit?

Maximilian Ulmer[1,2] **Leonard Klüpfel**[1] **Maximilian Durner**[1,3] **Rudolph Triebel**[1,2]

***Abstract*—We investigate the efficacy of data augmentations to close the domain gap in spaceborne computer vision, crucial for autonomous operations like on-orbit servicing. As the use of computer vision in space increases, challenges such as hostile illumination and low signal-to-noise ratios significantly hinder performance. While learning-based algorithms show promising results, their adoption is limited by the need for extensive annotated training data and the domain gap that arises from differences between synthesized and real-world imagery. This study explores domain generalization in terms of data augmentations – classical color and geometric transformations, corruptions, and noise – to enhance model performance across the domain gap. To this end, we conduct an large scale experiment using a hyperparameter optimization pipeline that samples hundreds of different configurations and searches for the best set to bridge the domain gap. As a reference task, we use 2D object detection and evaluate on the SPEED+ dataset that contains real hardware-in-the-loop satellite images in its test set. Moreover, we evaluate four popular object detectors, including Mask R-CNN, Faster R-CNN, YOLO-v7, and the open set detector GroundingDINO, and highlight their trade-offs between performance, inference speed, and training time. Our results underscore the vital role of data augmentations in bridging the domain gap, improving model performance, robustness, and reliability for critical space applications. As a result, we propose two novel data augmentations specifically developed to emulate the visual effects observed in orbital imagery. We conclude by recommending the most effective augmentations for advancing computer vision in challenging orbital environments. Code for training detectors and hyperparameter search will be made publicly available.**

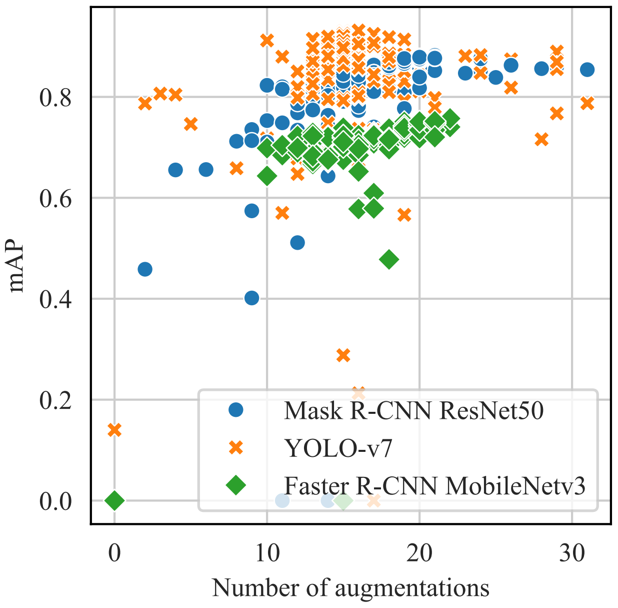

Figure 1: Impact of data augmentations on satellite object detection to bridge the domain gap. We observe that detectors predict more precise bounding box on real-world images when trained with more augmentation techniques applied during the training on synthetic images.

Table of Contents

1. Introduction 1
2. Related Work 2
3. Method 4
4. Experiment 6
5. Conclusion 10
Acknowledgments 10
References 10
Biography 12

1. Introduction

The application of autonomous systems in space is increasing as humans' presence in orbit expands and the sustainability of the orbital environment as well as its infrastructure is becoming increasingly important. Especially for highly complex, autonomous operations, such as on-orbit servicing, accurate and high-performance computer vision is a necessity. However, the orbit is an extremely challenging domain for camera-based algorithms due to hostile illumination, high contrast, and low signal-to-noise ratio. Such conditions can drastically deteriorate the performance of classical computer vision algorithms due to obscured color and geometric features. Learning-based algorithms have recently shown convincing capabilities to deal with such conditions, yet, the space-domain remains hesitant to widely adopt these methods. Besides the current lack of relevant hardware, one reason is the vast amounts of annotated training data that learning-based methods require to achieve good performance. To this end, recent work focuses on using rendered data for training and transferring the learned model to real data during inference [1]. This divergence between the training and inference data distributions is called the domain gap. The effects of the gap are especially drastic in the orbital domain, likely due to the difficulty of rendering tools to achieve a high degree of visual realism. Spacecrafts are often constructed from materials causing specular reflections that result in visual artifacts, such as a blooming and overexposure. Most objects in space are partially covered in a reflective insulation layer that can only be approximated in the 3D model and is challenging to render. Similarly, it is very expensive to synthesize the low signal-to-noise ratio and lack of atmospheric diffusion. These effects

[1] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany `<first>.<second>@dlr.de`
[2] Department of Informatics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
[3] Department of Computer Science, Technical University of Munich (TUM), 85748 Garching, Germany

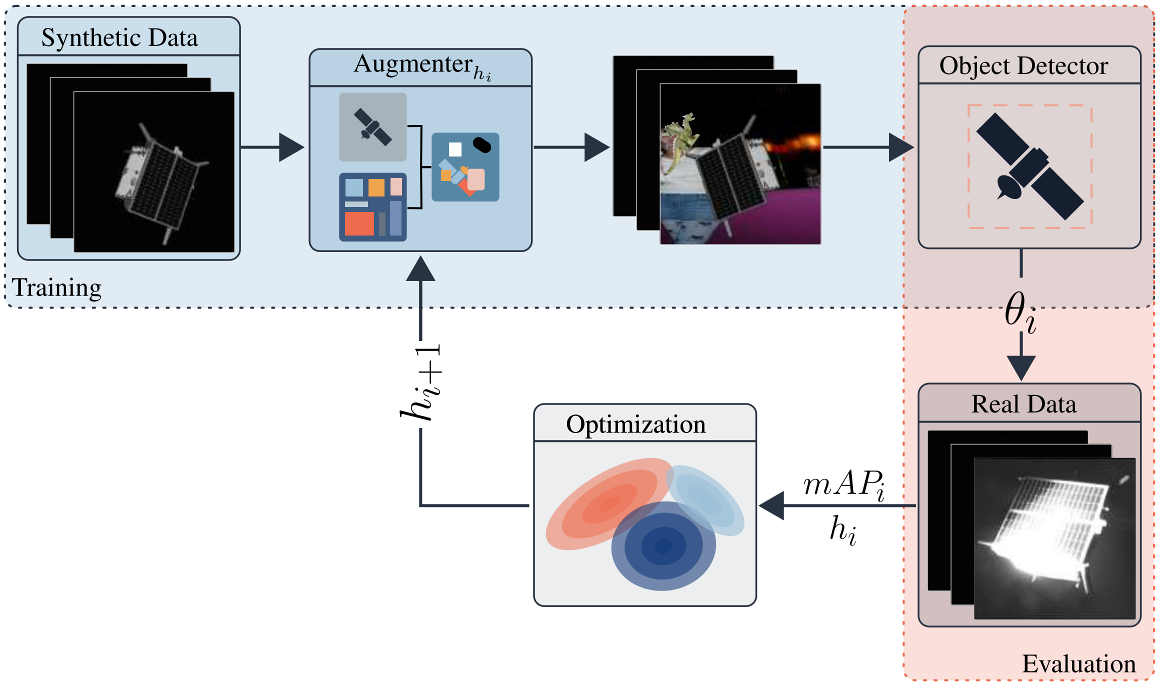

Figure 2: Overview on our hyperparameter optimization strategy for finding the set of data augmentations yielding the best precision when trained on synthetic and tested on real-world orbital satellite images. To this end, we train a detection model with varying augmentations techniques, report an evaluation score that informs the next set of augmentations to experiment with in the consecutive trial.

result in an especially wide domain gap for orbital imagery, making transferring a model more difficult [2].

Domain generalization is one of the primary approaches to deal with such a domain gap if the target domain is unknown or annotated target data is too costly [3]. It aims to augment the training procedure to learn domain invariant features or change the training data statistics to reduce the divergence between training and testing distributions. To achieve the latter, one of the most effective ways is to apply data augmentations during training. However, recent work has shown that the set of augmentations usually used for terrestrial applications is not sufficient for orbital imagery [4]. To this end, we conduct an extensive study on the efficacy of different data augmentations to close the domain gap for spaceborne computer vision. We explore a range of augmentations, ranging from simple color augmentations, geometric transformations, image mixing, kernel filters, information deletion, corruptions, to noise. In addition, we introduce two novel space augmentations that specifically aim to emulate visual conditions in orbit. To investigate the importance of each augmentation, we utilize the hyperparameter optimization framework Optuna [5]. The framework automatizes the process of selecting different hyperparameter configurations for individual trials and allows to implement efficient search strategies. Due to the sheer size of possible data augmentation combinations, we employ a Tree-structured Parzen Estimator (TPE) algorithm to sample the search space. As a reference task, we chose the field of 2D object detection as it is a well-studied and essential component of most spaceborne computer vision applications, such as pose estimation, inspection, and tracking. We train three very popular – yet very different – 2D object detectors: Mask R-CNN with a ResNet50 backbone, Faster R-CNN with a MobileNetv3 backbone, and YOLO-v7. We evaluate models and augmentations on the widely adopted SPEED+ dataset [1]. It's testing data is real recorded data from a real-world hardware-in-the-loop facility, hence, the focal point of the dataset is closing the domain gap. In addition, we compare the performance to a recent open set object detector, called GroundingDINO [6], that can be used out-of-the box and does not need to be fine-tuned or retrained. We evaluate all methods on an NVIDIA Jetson embedded system. Our findings demonstrate the importance of data augmentations to close the domain gap, leading to drastically improved performance and robustness to the challenging conditions of orbital imagery. At last, we present a set of augmentations that we found empirically to perform best to pave the way for more reliable and generalizable computer vision models for critical space applications.
To summarize, the contributions of this paper are: (i) we demonstrate the severeness of the domain gap for object detection in orbit, (ii) we conduct a large scale study involving 29 common augmentations and two novel space augmentations to find the best configuration for each detector, (iii) we explore the importance of the augmentations to each models and derive recommendations based on the collected data, (iv) we compare the performance to a modern open-set object detector and highlight differences in memory usage and inference speed on an NVIDIA Jetson embedded system.

2. Related Work

Methods based on deep neural networks have become widely adopted to solve computer vision tasks. One downside of such methods is their need of a large number of training samples to reach the desired performance [7]. The process of creating such datasets of real training data requires significant effort for terrestrial datasets [8] and is not practical for orbital perception. To this end, researchers started synthesizing images of 3D objects using computer graphics and train their

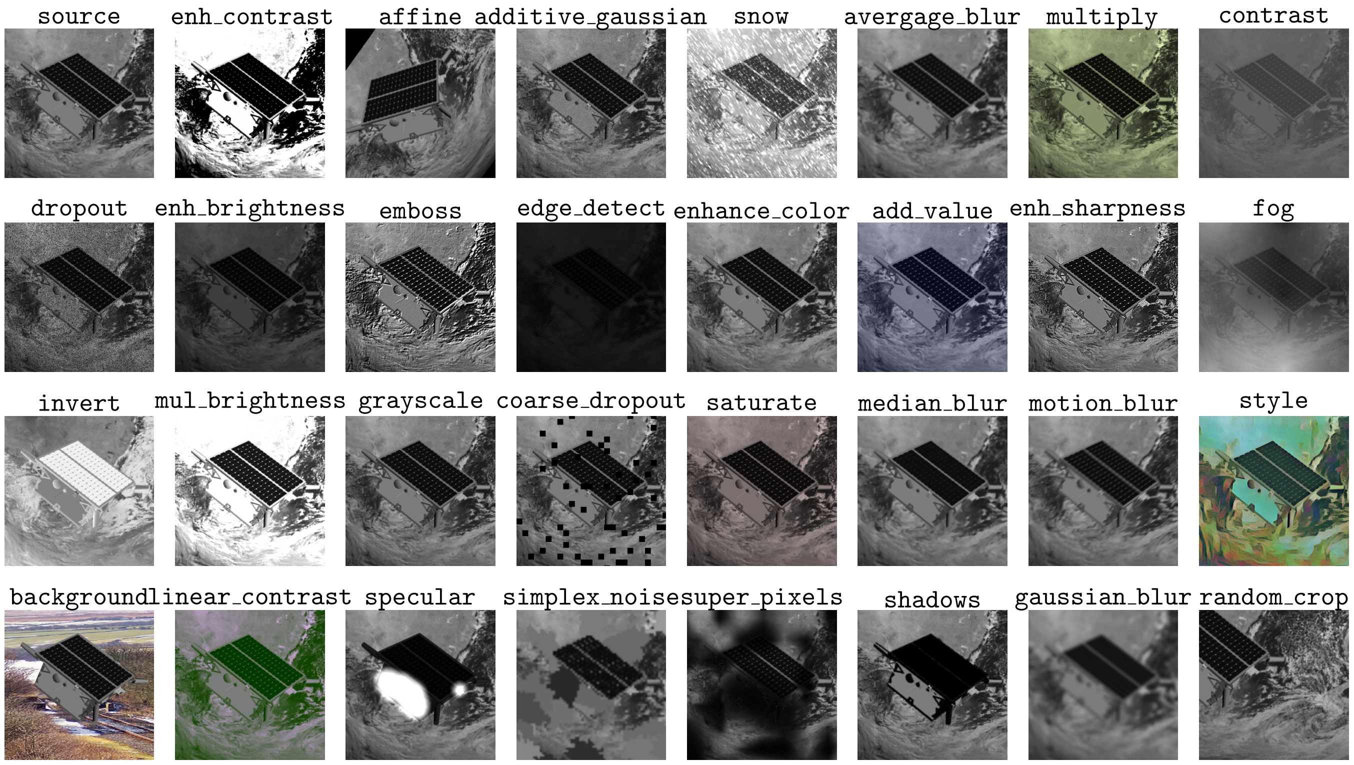

Figure 3: Overview on all our possible data augmentations, applied to the same source image (leftmost image, in the first row).

models on such synthetic samples [9]. This approach allows the creation of massive datasets with only minimal human effort. Yet, neural networks trained on such synthetic datasets demonstrated a significant degradation in testing performance on real data [10]. This effect has been coined the domain gap and closing it is a central tenet of many modern computer vision methods [11], [12].

The SPEED and SPEED+ datasets showed that the domain gap is especially dramatic for space imagery [1], likely due to the difficulty of rendering realistic orbital conditions. Many approaches were proposed to bridge this domain gap that can broadly be classified as domain adaption, domain randomization, and domain generalization.
Domain adaption techniques aim to adapt trained models to the target domain at test time without annotations [13]. This means, that they have access to the target domain, yet, have no training labels. For instance, Wang *et al.* [14] use images from the test set to train a generative adverserial network (GAN) to make training images more realistic. Park *et al.* [15] adapt the trained model to the target domain by unsupervised learning. The approach updates the parameters of normalization layers and affine transformations at test time by minimizing the shannon entropy of a segmentation head. At last, Pérez-Villar *et al.* [16] use a model trained on the source domain to generate pseudo annotation for the target domain.
Domain randomization is another popular approach to bridging the domain gap by randomizing rendering parameters during training data synthesis. This approach has been shown to be very beneficial for object detection and pose estimation models in terrestrial applications [17], [18]. However, to the best of our knowledge, such an approach has not been adopted by a published orbital dataset yet.
At last, *domain generalization* is the process of setting up the training of the model that it generalizes to the target domain without any prior knowledge of the target domain [3]. For instance, Li *et al.* [19] learn domain invariant latent features based on adverserial autoencoders. The most widely adopted implementation of domain generalization is data augmentation. The benefits can be manifold, such as increasing the training sample count, improving the training statistics variance, and reducing overfitting. Data augmentations have become a standard procedure in modern training of deep neural networks and most modern methods use it [11], [12], [20], [4], [21], [22].

Data augmentations for computer vision have been studied in the context of many different domains and tasks. In one of the earliest studies, Taylor *et al.* [23] study six classical data augmentations to improving performance on the Caltech101[2] image classification task. Mikołajczyk and Grochowski [24] focus on image classification of medical imagery and highlight a set of classical augmentations, texture transfer, and GAN-based augmentations. Shorten *et al.* [25] present an extensive study on classical and deep-learning-based data augmentations, yet, is primarily focused on image classification. Hendrycks *et al.* [26] benchmark neural network robustness to a list image corruption for image classification. In [27] these corruptions are tested on the task of object detection for autonomous driving. Recently, Yang *et al.* [28] evaluated classification, segmentation, and object detection methods on a fixed set of data augmentations and demonstrated that they improve inference performance.
Another interesting avenue of work to reduce the domain gap is to synthesize more realistic images. To this end, Hu *et al.* [20] use a physics-based renderer with ray-tracing to simulate orbital conditions in the SwissCube dataset.
To the best of our knowledge, there has been no study focused on data augmentations for orbital imagery. Yet, Ulmer *et*

[2]https://data.caltech.edu/records/mzrjq-6wc02

al. [4] show that the augmentations that work best for terrestrial applications are not sufficient in the space domain. In this work, we do not attempt to find the best set of data augmentation manually but automate the process using a hyperparameter optimization framework [5]. Moreover, we do not focus on a small set of augmentations, but optimize more than 30 different popular data augmentations. Due to this vast search space, we employ a search strategy that fits a Gaussian Mixture Model (GMM) to the experiment results and conduct hundreds of experiments to explore the importance of data augmentations for orbital perception.

3. Method

In this section, we first outline the task of object detection and the unique characteristics of the detectors that we deploy. Next, we discuss the set of data augmentation techniques which we investigate in the context of satellite object detection. Eventually, and for the means of our data augmentation investigation, we introduce the hyperparameter optimization strategy to find the best augmentation set. The complete optimization framework is shown in Fig. 2 and can be described as follows. The process runs for n_{trials} trials and a sampler chooses a set of hyperparameters h_i with $i \in \left[0, n_{\text{trials}} - 1\right]$. Each set h_i parameterizes a chain of augmentations that is applied to each image $\boldsymbol{I}$ during training $\hat{\boldsymbol{I}} = \texttt{augmenter}_{h_i}(\boldsymbol{I})$. The parameterization of an `augmenter` remains fixed for one full trial. In each trial, we train a $\texttt{detector}_{\theta_i}$ that is parameterized by its weights θ_i. After a fixed amount of training epochs is complete, we evaluate the $\texttt{detector}_{\theta_i}$ on the test dataset and calculate an evaluation score mAP_i. Then, we store the pair (h_i, mAP_i) in a database. For the initial $n_{\text{startup\_trials}}$ the sampler randomly chooses h_i to initialize the GMM. After the startup phase is complete, the sampler uses the database and the TPE search strategy to generate predicted high importance parameters h_i.

2D Object Detection

Given an image, $\boldsymbol{I} \in \mathbb{R}^{\text{w} \times \text{h} \times 3}$ with width w and height h, the goal of 2D object detection is to recognize all instances of an object class and localize them by their axis-aligned bounding boxes [29]. In this work, we investigate the performance of three different object detection groups in orbital visual conditions. First, the family of region-based detectors which predict objects and their bounding boxes by processing images in two distinct stages [30]. Second, and in contrast to the former category, single-stage detectors directly regress objects and their locations. These two approaches detect objects from a pre-defined object set on which they are specifically trained on. This differs to the third group, open-set detectors, which require no additional training on specific objects and thus can be deployed out-of-the-box.

Region-based Object Detection—These detectors consist of two stages: first a region-proposal stage which outputs regions of interest for the second stage, a regression model that estimates the actual object class and bounding box coordinates in the input image. In [30], Ren *et al.* introduce a learnable neural network for the region-proposal stage, predicting object scores and anchor boxes with varying dimensions to account for different object sizes and scales. Based on these regions of interest, the consecutive detection model learns to regress the actual object class labels and bounding boxes. In this work, we train two region-based detectors: Faster R-CNN [30] with a MobileNet [31] backbone and Mask R-CNN [32] with a ResNet50-based [33] feature extractor. The latter extends Faster R-CNN by simultaneously predicting segmentation masks for detected objects through expanding the architecture with an additional segmentation head. We train these two detectors with different backbones as this affects the final network size, making Faster R-CNN smaller than Mask R-CNN. Table 1 shows the exact model sizes as presented by the number of parameters. The rationale is that smaller sized networks require less compute and memory – a consideration relevant for the space domain. However, as we discuss in Section 4, the network size and parameter quantity influence its capacity to learn patterns and thus can impact performance.

Single-Stage Object Detection—This group of object detection networks mainly consists of You Only Look Once (YOLO) [34] models and its variants and extensions [35], [36], [37], [21]. Single-stage detection refers to the simultaneous and direct regression of bounding box coordinates and class probabilities from a given input image. Therefore, the image $\boldsymbol{I}$ is split into a grid of $s \times s$ cells and each cell predicts whether an object center falls into its cell [34]. Casting object detection as regression problem enables bounding boxes predictions at a high inference speed compared to region-based approaches. For our experiments, we train a YOLO-v7 [21] network with almost 8 times more trainable parameters (see Table 1) than the Faster R-CNN – and hence regarded as more powerful. This advanced YOLO architecture introduces model re-parameterization and model scaling, which improves the detection accuracy while maintaining its inference speed [21]. Additionally, by adjusting the training routine and adding an auxiliary loss, Wang *et al.* further increase the detection accuracy.

Open-Set Object Detection—Detectors of this category can identify unknown objects, i. e., unseen during the model's training phase, in addition to known, trained on objects. Therefore, recent models take as an additional input a descriptive text prompt about the objects that are supposed to be detected in an image [38], [39], [6]. Consequently, aligning visual and language modalities becomes necessary. In this work, we apply GroundingDINO [6], which is open-source and achieves competitive results on various detection benchmarks. GroundingDINO approaches the modality alignment through a large-scale pre-training phase on known datasets and objects. Furthermore, Liu *et al.*introduce an approach to improve the mapping of text features to specific image regions for a better modality fusion. In contrast to region-based and single-stage approaches, open-set detectors do not require further training, which poses an advantage. However, since it requires a text prompt as an additional input, the prompt selection directly impacts the detection performance – an observation we report and further elaborate in Section 4.

Data Augmentations

The application of data augmentations to the training dataset has long been a standard practice to train neural networks. Such models are very prone to overfitting and require proper regularization to generalize beyond the training data [22]. Besides regularization, another goal of augmenting the training data is to increase the variance, improving alignment between the inference and training distributions. In the context of orbital perception, simulating visual conditions such as specular reflections is very difficult and costly. Hence, synthetic images hardly entail realistic lighting conditions and models perform poorly on such samples which are clearly beyond the training distribution [4]. In this work, we study the impact of five commonly used classes [28] of image data augmentations: color, geometric, mixing, filters, and deletion. We further introduce specialized augmentations tailored to the distinctive characteristics of the space environment.

Color augmentations— change the statistics of the color channels of an image. `add_value` selects a random value in a set interval and adds this value to all or individual channels. `invert` inverts pixel values for all or individual channels. `multiply` selects a random value in a set interval and multiplies this value to all or individual channels. `multiply_brightness` convert an image to the HSV colorspace and multiplies a random value in a set interval to the brightness-related channel before converting the image back to the original colorspace. `enhance_color` changes the balance of the color channels of an image. `grayscale` converts the image to grayscale and overlays the gray version with the original by a random strength in a set interval. `contrast` changes the contrast of the image by a set severity [26]. `linear_contrast` modifies the contrast of an image linearly with a randomly chosen value in a set interval. `enhance_contrast` increases or decreases the contrast of an image randomly in a set interval. `saturate` changes the saturation of an image, making it randomly more or less colorful [26]. At last, `enhance_brightness` increases or decreases the brightness of an image randomly in a set interval.

Geometric augmentations—apply some geometric transformation to the image, fundamentally changing it. `affine` applies an affine transformation to the image, such as scaling, translation, rotation, and shearing. In this work, we only apply a random rotation in a set interval. `random_crop` calculates a random scale change and crops the image randomly such that a minimum area of the target object is still visible.

Image mixing—changes the contents of an image by mixing it with some other source. `background` uses the binary foreground segmentation to replace the background of the image with some randomly sampled image, often from another dataset [11], [40] . `snow` adds randomly generated snow layers with a set severity from random directions which obstruct visual features [26]. `fog` adds a randomly generate layer of fog to the image with a set severity.

Kernel filters—apply some transformation to the image by applying a kernel or convolution operation. `gaussian_blur` applies a gaussian kernel with a standard deviation randomly chosen in a set interval. `average_blur` computes means over a neighbourhood randomly chosen in a set interval. `median_blur` computes the median over a neighbourhood randomly chosen in a set interval. `motion_blur` convolves the image with a random kernel that simulates motion blur of a moving camera or object. `emboss` calculates an embossed version of the image that pronounces highlights and shadows and blends the result randomly with the original. `edge_detect` convolves the original with an edge-detection kernel and then randomly blends the result with the original. `enhance_sharpness` randomly increases or decreases the sharpness of an image. `additiv_guassian_noise` adds noise that is sampled from a normal gaussian distribution with random variance elementwise to the image. `style` augmentation is a neural network which is trained to transform an image such that it remains semantically valid but randomizes the style of the image [41].

Information deletion—erases data from the image by some function. `super_pixels` chooses a random number of super pixels per image and replaces the value with a randomly chosen probability by its average pixel color. `simplex_noise` creates a simplex noise mask with that is used to blend the original image with the result of an edge-detection operation. `dropout` sets individual pixel values at random locations to zero for a randomly sampled fraction of the image, for all or individual channels. `coarse_dropout` chooses rectangular areas to drop at random locations for a randomly sampled fraction of the image, for all or individual channels.

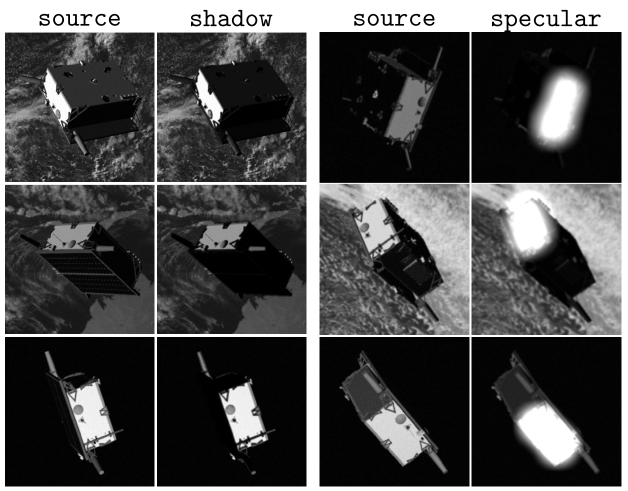

Figure 4: Visualization of our custom space data augmentations, simulating typical visual conditions encountered on objects in orbit: stark `shadow` effects and strong `specular` reflections. Source images displayed in the left, augmented ones in the right columns.

Space augmentations— attempt to emulate the special visual conditions of orbital perception. These augmentations are developed by Ulmer *et al.*[4] in the scope of the SPEED+ challenge to realize space specific data augmentations. `specular` simulates reflections which can occur on metallic surfaces and result in bloom-like visual artifacts that visually occlude features. Such effects are usually confined locally to the target satellite and the rest of the image is largely unchanged. To this end, the method uses the binary segmentation mask to sample the brightest region of the foreground and overlays a white bloom, bleeding into the rest of the image. `shadow` emulates the exceedance of the dynamic range of a camera by selecting the dark regions of the target, making the even darker while keeping the bright region constant. Examples of these augmentations are visualized in Fig. 4.

Hyperparameter Optimization

Our goal is to find a good set of hyperparameters in as little trials as possible. Finding the best set of hyperparameters in such a large space is very difficult due to time and resource constraints. Even if each parameter is only considered to be binary – either active or inactive – the search space quickly reaches millions of possible trials which makes an approach like grid search impossible. Historically, such an optimization has been conducted by humans in-the-loop and the effectiveness and results comes down to the experience of the researcher tweaking a few parameters at a time. However, modern software and hardware make it possible to automate the process and run many trials in parallel on compute clusters.

In this work, we use the Optuna framework [5] to conduct the exploration of the data augmentations search space. Modelling the dynamics of data augmentations is an inherently difficult task for any optimization process. As we will show, too little data augmentations result in poor performance to close the domain gap. Yet, too many augmentations can have detrimental effects on the performance due to degradation of information, often to the point that no training signal is retained in individual samples. This can lead to highly unstable gradients and catastrophic training dynamics. In addition, some augmentations have a strong relational component where combinations have a reinforcing or deteriorating effect.

To this end, we use the Tree-structured Parzen Estimator (TPE) [42] to sample the search space. TPE is a Bayesian optimization method that can handle tree-structured search spaces and uses a kernel density estimator. In short, the estimator fits one Gaussian Mixture Model (GMM) to the parameters correlated with the best objective results, and another GMM to the remaining hyperparameters. For more information, please refer to [43]. The estimator is initialized for $n_{\text{startup\_trials}}$ random trials for which the objective values are stored in a database. After the startup phase, the sampler draws a new hyperparameter configurations for each new trial based on its internal model.

4. Experiment

In this section, we first describe the dataset that is the basis for our results and the changes we employed to the original data. Next, we cover the setup of the training pipeline for the models and the hyperparameter optimization framework, before exploring the results of this study.

SPEED+ Satellite Pose Estimation Challenge

The Next-Generation Spacecraft PosE Estimation Dataset (SPEED+) was the subject of the 6D pose estimation challenge Satellite Pose Estimation Challenge (SPEC) organized by the European Space Agency (ESA) and Stanford University [1]. The purpose of the competition was to determine the current state-of-the-art performance on the challenging task of satellite pose estimation and closing the domain gap. The training and validation split of the dataset are 59960 computer-rendered images of the Tango satellite annotated with the 6D pose. For testing, the dataset contains 9531 hardware-in-the-loop (HIL) mockup-satellite images split into two subsets with distinct visual conditions. Concretely, the `lightbox` split simulates albedo conditions using light boxes with diffuser plates and the `sunlamp` split entails a high-intensity sun simulator to mimic direct homogeneous light from the sun [1].

In this work, we repurpose the dataset to 2D object detection. The original data does not include a 3D model of the target nor 2D detection or segmentation annotations for training and test set. To this end, we established a coarse wireframe model of the satellite by inspection and used this faulty model in combination with the given 6D pose to generate the missing annotations for the original training images, similar to [4]. In addition, we rectified each sample and reduce the resolution to 960 × 600 pixels.

Experiment Setup

We trained three very different object detectors for this study. Modern methods often employ a variety of techniques to improve learning performance and stability, such as Expo-

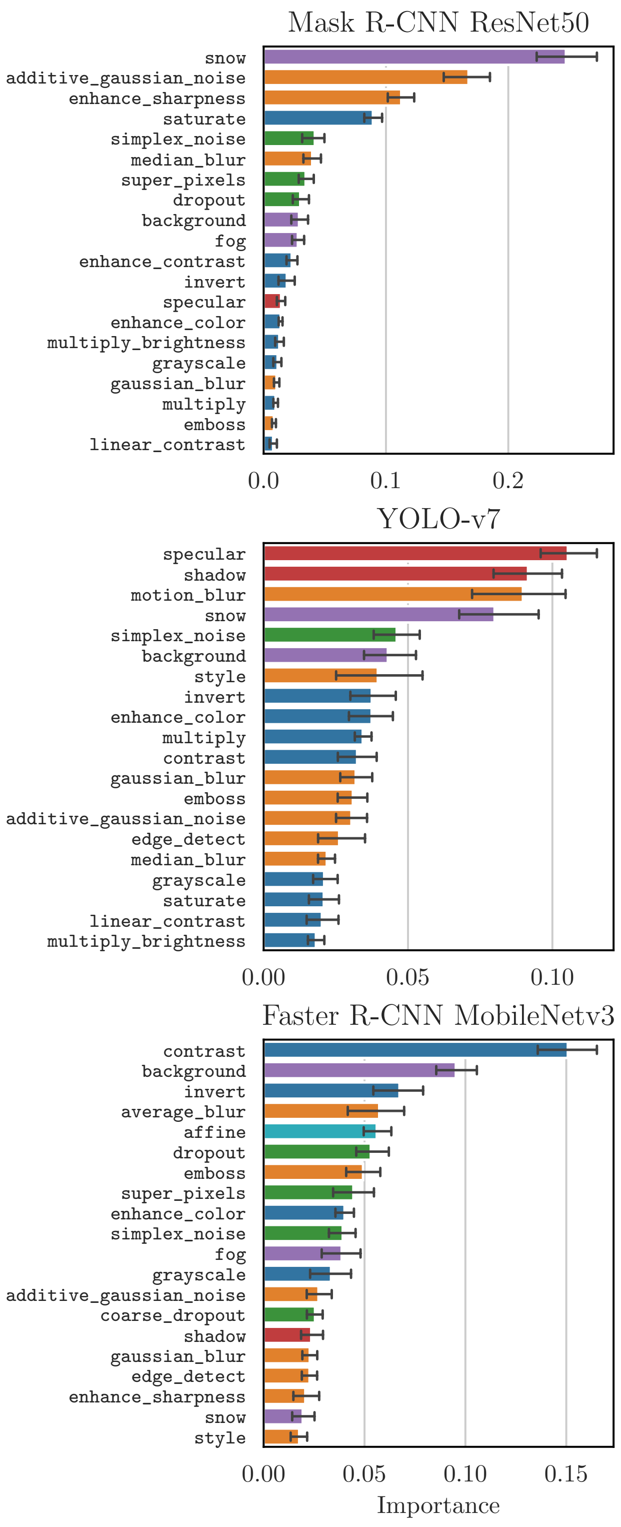

Figure 5: Importance and variance of individual data augmentations on the different models' precision performance. We observe varying key augmentations for each detection model. Furthermore, network size seems to impact whether severe image changing techniques benefit a model, with the larger YOLO-v7 being the only detector for which those augmentations are more influential.

Table 1: Training parameters of all models that were fixed for all training runs.

	Mask R-CNN ResNet50	YOLO-v7	Faster R-CNN MobileNetv3
epochs	20	20	20
avg. train walltime [h]	15.08	18.94	10.88
#parameters [M]	43.7	164.89	18.87
batch size	16	16	16
optimizer	SGD	SGD	SGD
learning rate	0.02	0.01	0.02
momentum	0.9	0.937	0.9
weight decay	1×10^{-4}	1×10^{-5}	1×10^{-4}
warmup steps	2000	2000	2000
annealing point	0.72	0.72	0.72

nential Moving Average (EMA) and elaborate learning rate schedules, that are fine-tuned to the specific model. To keep the results as comparable as possible we settled on a simple training schedule of 20 epochs with a linear learning rate warmup and cosine annealing to finish the training. We kept the batch size, learning rate and optimizer configuration as close as possible to the default values of the original works and references. A full list of all training parameters can be found in Table 1. All models were trained on a single NVIDIA A100 GPU with 80 GB of VRAM, however, the Mask R-CNN and Faster R-CNN would have fit on a smaller GPU.

For all three models we set up Optuna to conduct the hyperparameter optimization in an identical fashion. We initialize a database for each model and run $n_{\text{startup\_trials}} = 64$ trials to initialize the TPE. To keep the search space as small as possible, each augmentation probability is defined by a categorical distribution to be either active or inactive. If an augmentation is chosen to be active in a trial, we apply the augmentation to an image with a probability of 30%. For all data augmentations,we defined a default parameterization that is used whenever the augmentation is active that can be found in the github repository. For the objective metric we use the Mean Average Precision (mAP) calculated over the steps $[0.5, ..., 0.95]$ with step size 0.05.

Results

In this subsection we first explore each augmentation factor using the functional analysis of variance (fANOVA) framework to quantify the importance of individual hyperparameters, before presenting the best set of hyperparameters that we found for each individual model.

Hyperparameter Importance— First, we try to gain some insights into the relative importance of different data augmentations for each model. The fANOVA algorithm fits a random forest to the hyperparameter space that regresses the objective value, in our case the mAP. We run the algorithm with 64 trees in the forest and a maximum depth of 64. In addition, we repeat the algorithm 8 times to assess the variance in the results from the algorithm. The 20 most important augmentations, summing up to one, for each model are shown in Fig. 5.

For Mask R-CNN, the first four factors explain about 60% of the variance in the data we collected. These augmentations are `snow`, `additive_gaussian_noise`, `enhance_sharpness`, and `saturate`. The variance between different runs of fANOVA is low. This suggest that the random forest model can represent the data well and the TPE has converged to some local maximum. There is a second tier of augmentations that explain another 25% of the variance, containing `dropout`, `fog`, `enhance_contrast`, and `invert` among others. The augmentations that scored the worst in terms of importance, all below 0.5%, are `average_blur`, `motion_blur`, `style`, `enhance_brightness`, `coarse_dropout`, and `affine`.

Analysing YOLO-v7, the results of the fANOVA algorithm are more uncertain. The results of different random forests vary more than they did for Mask R-CNN. However, there are four factors which clearly express more variance than the rest, about 36%. Namely, `specular`, `shadow`, `motion_blur`, and `snow`. Notably, the `style` augmentation [41] varies extremely between different runs. We hypothesize, that `style`, which is a pretrained deep convolution network, has strong interactions with other data augmentations. In our experiments, the least important data augmentations for YOLO-v7 are `fog`, `average_blur`, and `enhance_contrast`, all with around 1% importance.

At last, the Faster R-CNN detector with the small MobileNetv3 backbone appears to react very differently to the augmentations than the previous two models. In our trials, `contrast`, `background`, and `invert` were the most important data augmentations explaining about 32.5% of the variance. The least important augmentations according to fANOVA are `multiply`, `median_blur`, and `multiply_brightness`, below 1% importance.

It is striking how much the importance of data augmentations differ from one model to another. One reason is most likely the very different model capacities. Our implementation of YOLO-v7 has more than 8 times the number of trainable parameters than Faster R-CNN MobileNetv3 with about 18 million parameters (see Table 1). In our data, only for the large YOLO-v7 the very aggressive space augmentations play an important role for the performance. Overall, the data suggests that model capacity correlates with the importance of augmentations that severely change the image or deteriorate information. Another likely contribution to this is the relatively small number of startup trials. The search space is greater by a few number of magnitude and hence, the initialization plays a pivotal role. However, there are still some key findings we can derive from the fANOVA

Table 2: Our satellite detection evaluation on SPEED+ images with varying data augmentations applied during training.

$\sum$	affine	shadow	specular	background	random_crop	fog	snow	emboss	invert	dropout	contrast	multiply	saturate	add_value	grayscale	edge_detect	median_blur	motion_blur	average_blur	super_pixels	enhance_color	gaussian_blur	simplex_noise	coarse_dropout	linear_contrast	enhance_contrast	enhance_sharpness	enhance_brightness	multiply_brightness	additive_gaussian	style	mAP$_{lightbox}$	mAP$_{sunlamp}$	IoU	mAP@75	mAP
Mask R-CNN ResNet50																																				
18	✓	✓	0	0	✓	0	✓	✓	0	0	✓	✓	✓	0	✓	0	✓	✓	0	0	✓	0	✓	✓	✓	0	✓	✓	0	✓	0	0.881	0.827	0.918	0.965	0.867
18	✓	✓	✓	0	✓	✓	✓	0	✓	✓	✓	✓	✓	0	✓	0	✓	0	✓	0	0	0	0	0	0	0	✓	0	✓	✓	✓	0.885	0.829	0.920	0.968	0.869
15	0	✓	0	✓	✓	✓	✓	0	✓	✓	0	0	0	0	✓	0	✓	✓	✓	0	0	0	✓	0	✓	0	0	✓	0	✓	0	0.884	0.832	0.918	0.968	0.869
20	✓	✓	✓	0	✓	✓	✓	✓	✓	✓	0	✓	✓	0	0	0	✓	0	0	0	✓	✓	0	0	✓	✓	✓	✓	✓	✓	0	0.888	0.823	0.920	0.968	0.869
20	✓	✓	✓	0	✓	✓	✓	0	✓	✓	0	✓	✓	0	✓	0	✓	✓	0	0	✓	0	✓	0	✓	0	✓	✓	✓	✓	0	0.886	0.841	0.922	0.967	0.873
19	✓	✓	✓	0	✓	✓	✓	0	✓	✓	0	0	✓	0	✓	✓	0	✓	0	0	✓	0	✓	0	✓	0	✓	✓	✓	✓	0	0.891	0.842	**0.925**	0.969	0.876
21	✓	✓	✓	0	✓	✓	✓	0	✓	✓	0	✓	✓	0	✓	✓	0	✓	0	0	✓	0	✓	0	✓	✓	✓	✓	✓	✓	0	0.891	**0.845**	**0.925**	0.968	0.877
20	✓	✓	✓	0	✓	✓	✓	0	✓	✓	0	✓	✓	0	✓	✓	0	✓	0	0	✓	0	✓	0	✓	0	✓	✓	✓	✓	0	0.892	0.843	**0.925**	0.971	0.877
21	✓	✓	✓	0	✓	✓	✓	0	✓	✓	0	✓	✓	0	✓	✓	✓	✓	0	0	✓	0	✓	0	✓	0	✓	✓	✓	✓	0	0.893	0.844	**0.925**	0.973	0.878
18	✓	✓	✓	✓	✓	✓	✓	0	✓	✓	0	0	✓	0	✓	✓	0	✓	0	0	✓	0	✓	0	0	✓	✓	0	0	✓	0	**0.895**	0.843	**0.925**	**0.977**	**0.881**
YOLO-v7																																				
14	0	0	0	✓	✓	0	✓	✓	0	✓	0	✓	0	0	✓	0	✓	0	0	✓	✓	0	✓	0	✓	✓	✓	0	0	0	0	0.918	0.873	0.935	0.957	0.904
16	✓	✓	✓	✓	✓	0	✓	0	0	✓	✓	0	0	0	0	✓	✓	0	✓	✓	✓	✓	0	0	0	✓	0	0	✓	0	0	0.926	0.850	0.934	0.958	0.904
17	0	✓	0	✓	✓	✓	✓	0	0	✓	✓	0	0	0	✓	0	✓	0	0	0	✓	0	✓	0	✓	✓	✓	✓	0	✓	✓	0.917	0.888	0.937	0.968	0.907
10	✓	0	0	✓	✓	0	✓	0	✓	0	✓	0	0	0	0	0	0	✓	0	0	✓	0	0	0	0	0	0	✓	0	✓	0	0.927	0.876	0.938	0.959	0.912
16	0	0	✓	✓	✓	0	0	✓	0	0	✓	✓	0	0	✓	✓	✓	0	0	✓	✓	✓	0	0	✓	✓	0	0	0	✓	✓	0.920	0.895	0.937	0.969	0.912
19	0	✓	0	✓	✓	✓	✓	0	✓	✓	0	0	0	0	✓	✓	✓	0	✓	✓	✓	0	✓	0	✓	✓	✓	✓	0	0	✓	0.924	0.890	0.940	0.968	0.914
18	0	✓	0	✓	✓	✓	✓	✓	0	✓	✓	0	0	0	0	0	✓	✓	0	✓	0	✓	✓	✓	✓	✓	✓	0	0	✓	0	0.941	0.858	0.940	0.969	0.918
17	0	✓	✓	✓	✓	✓	✓	0	0	✓	✓	✓	✓	0	0	✓	✓	0	0	0	✓	0	0	0	✓	✓	0	0	✓	0	✓	0.932	0.907	0.945	**0.979**	0.925
15	✓	0	0	✓	✓	✓	✓	0	0	0	✓	0	0	0	0	✓	0	✓	0	0	0	✓	✓	0	✓	✓	0	✓	0	✓	✓	**0.942**	0.900	0.947	**0.979**	0.931
16	✓	0	✓	✓	✓	✓	✓	0	0	0	✓	✓	✓	0	✓	0	0	0	0	0	0	0	✓	0	✓	0	✓	✓	0	✓	✓	0.937	**0.919**	**0.948**	**0.979**	**0.932**
Faster R-CNN MobileNetv3																																				
17	✓	✓	✓	✓	✓	0	0	✓	✓	✓	✓	0	0	0	0	✓	✓	0	✓	0	0	✓	✓	✓	0	0	✓	0	0	0	✓	0.726	0.730	0.858	0.850	0.726
13	0	0	✓	✓	✓	✓	0	✓	0	✓	✓	✓	✓	0	0	0	0	0	0	✓	0	0	0	0	✓	0	✓	0	0	✓	0	0.737	0.709	0.846	0.856	0.728
17	✓	0	✓	✓	✓	0	0	0	✓	✓	✓	✓	0	0	✓	✓	✓	0	✓	0	0	0	✓	✓	0	✓	✓	0	0	✓	0	0.732	0.723	0.860	0.858	0.729
18	0	0	✓	0	✓	✓	0	✓	✓	✓	✓	✓	0	0	0	✓	✓	0	✓	0	✓	0	0	0	✓	0	✓	✓	✓	✓	✓	0.740	0.714	0.852	0.846	0.731
18	✓	✓	0	✓	✓	✓	✓	✓	0	✓	✓	✓	0	0	✓	✓	0	✓	✓	0	0	0	✓	0	0	✓	✓	0	0	0	✓	0.741	0.713	0.858	0.856	0.732
18	✓	0	✓	✓	✓	0	✓	0	0	✓	✓	0	✓	0	✓	✓	✓	0	0	0	✓	0	✓	✓	✓	0	✓	0	✓	0	✓	0.736	**0.740**	0.861	0.862	0.737
15	0	✓	✓	0	✓	0	✓	0	✓	✓	0	0	0	0	0	0	0	✓	✓	✓	0	✓	✓	0	✓	✓	0	✓	0	✓	0	0.748	0.714	0.860	0.869	0.739
18	✓	✓	✓	0	✓	0	✓	✓	✓	0	✓	✓	0	0	✓	0	✓	0	✓	0	✓	0	✓	✓	✓	✓	0	0	0	✓	0	0.743	0.734	0.860	0.871	0.740
19	✓	✓	0	✓	✓	0	0	0	✓	✓	✓	0	✓	0	0	0	✓	✓	0	✓	✓	✓	✓	0	✓	0	✓	✓	✓	✓	0	0.753	0.720	0.862	0.872	0.743
19	✓	✓	0	✓	✓	✓	✓	✓	✓	✓	✓	0	✓	0	0	0	0	✓	✓	0	0	0	✓	0	✓	✓	0	0	✓	✓	✓	**0.755**	0.739	**0.865**	**0.883**	**0.750**

analysis. The augmentations invert, simplex_noise, enhance_color, and additive_gaussian_noise perform well for all three models. Over all runs and models, enhance_brightness, coarse_dropout, and linear_contrast were the least important augmentations.

Overall Performance.—First of all, the overall necessity for data augmentations in the scope of object detection in space environments is further demonstrated and derived through Fig. 1. Here, we observe that a detector, trained on synthetic images without any augmentation applied during, achieves very low performance as it is unable to bridge the domain gap. Both Mask R-CNN ResNet50 and Faster R-CNN MobielNetv3 overfitted to the training distributions resulting in 0.0mAP. YOLO-v7 was able to reach about 0.15mAP severely impeding the performance of the large model. However, the impact of adding only a few augmentations results in a substantial performance boost as measured by mAP. Yet, no model with less than 10 augmentation active come close to reaching their full potential. Furthermore, we derive that a certain number of augmentations, the testing performance starts to diminish, likely due to the loss of information if too many augmentations are added on top of each other. In our data, this effect is more pronounced for YOLO-v7 than it is for the other detectors. This suggests that the specific choice of data augmentations matter for object detection tasks.

The performance of the best 10 configurations of Mask R-CNN ResNet50, YOLO-v7, and Faster R-CNN MobileNetv3 can be seen in Table 2. We evaluate the performance according to two metrics: Mean Average Precision (mAP) and Intersection over Union (IoU). The IoU measures the overlap of the predicted box and the ground truth bounding box. In addition, mAP is a widely used metric for object detection, for instance in the popular COCO challenge [44]. The Average Precision (AP) is defined as the area under the precision-recall curve that is calculated at different IoU detection thresholds. Following the evaluation of the COCO challenge, we evaluate the AP for 10 IoU thresholds $[0.5, ..., 0.95]$ in steps of 0.05. Moreover, we report the mAP for the two parts of the test set lightbox and sunlamp.

In our experiments, YOLO-v7 performs best reaching a mAP of 0.932 and a very strong mAP@75 of 0.979. The second best model performs slightly better on the lightbox test data, however in the total mAP the difference is negligible. For YOLO-v7 the drop between the 10th best and very best model is significant, especially for the sunlamp test data. This suggests the importance of well tuned data augmentations.

The performance of the Mask R-CNN detector is worse. However, considering that the model entails four times less parameters, the performance is very convincing. The mAP@75 is only slightly worse at 0.977 and a total mAP of 0.881. The results of the ten best configurations are much closer together than they were for YOLO and suggest that the method is less sensitive to the choice of data augmentations.

At last, the smallest of the models performs significantly worse than the other two methods. Yet, the performance is still very good, considering the small model size and simplicity. The best augmentations configuration reached a mAP of 0.75 and mAP@75 of 0.883 that is easily good enough for most applications. Interestingly, similar to the other two methods, the best performance for sunlamp and lightbox is always achieved with a different set of augmentations.

Recommended Set of Augmentations—We have trained over 400 models across three different model families, namely Faster R-CNN, Mask R-CNN and YOLO. Our experiments suggest that there is no set of augmentations that performs best for all model sizes and methodologies. However, we found very good configuration for each model which are summarized in Table 2.

For training other models, there are some general recommendation that we can derive from the data we collected. Across all models, the augmentations invert, additive_gaussian_noise, enhance_color, emboss, and simplex_noise were important. For each model one blurring augmentations was important such as median_blur or motion_blur. The augmentations derived from [26], namely snow, fog, saturate, and contrast, were also important for the models and showed great potential as long as the severity is on the low end. If the model has a high capacity backbone, the space augmentations specular and shadow boost the performance drastically, which was also shown in [4]. At last, even though the results in our study were mixed, we suggest to use augmentations background, random_crop, and affine. These augmentations do not contribute to the catastrophic loss of information if too many augmentations are applied and their central contribution is to increase the size of the dataset. We hypothesize, that one reason for their little importance in our study is that the SPEED+ [1] dataset provides enough training variance in terms of geometric appearance.

Open-Set 2D Object Detection—Training any object detector requires a lot of time and resources. For instance, training YOLO-v7 on the SPEED+ dataset takes about 19 hours on a high-end GPU as the NVIDIA A100. If the data is not available, a dataset has to be generated that is sufficiently large and realistic. Moreover, as we demonstrated in this work, training such models requires careful consideration of hyperparameters and to find a satisfying set likely a multitude of models has to be trained

Motivated by this incredible effort and recent developments in open-set object detections, we conduct an experiment using GroundingDINO [6] that we briefly described in Section 3. Such a model can be used without training and reduces the cost and time required to a minimum. In this experiment, we ran one complete evaluation epoch and computed the same performance metrics as in our other experiments. In addition, we tested different text prompts as seen in Table 3. The best prompt – "satellite" – reaches a mAP of 0.602 and mAP@50 of 0.886. In other words, if the correct detection threshold is 0.5, it detects 88.6% of all samples correctly. GroundingDINO performs better on the sunlamp, which entails strong specularities, than on lightbox, which contains many samples that have very low signal-to-noise ratio. However, the model performance is not yet sufficient to deploy such a model to any critical scenario.

Inference Time and Memory Consumption—A very important consideration for orbital perception are the hardware requirements of the target system. Mission systems are extremely constrained in terms of computational resources, yet inference time plays a central role due to downstream actions such as robotic grasping. To this end, we evaluate the real GPU memory consumption during inference using the nvidia-smi tool. In addition, we run inference for 1000 samples and measure the time to process each sample

Table 3: Evaluation of the open-set detector GroundingDINO with different text prompts as required input.

Text Prompt	$mAP_{lightbox}$	$mAP_{sunlamp}$	IoU	mAP@50	mAP@75	mAP
”shiny rectangle with antennas”	0.454	0.562	0.742	0.822	0.527	0.483
”NASA satellite with antennas and solar panels”	0.509	0.567	0.763	0.865	0.581	0.523
”spacecraft with solar panels and antennas”	0.525	0.591	0.759	0.854	0.622	0.544
”metallic space object with foil, antennas and solar panel”	0.547	0.627	0.773	0.893	0.651	0.567
”metallic space object”	0.542	0.637	0.775	0.862	0.645	0.567
”satellite”	**0.583**	**0.655**	**0.786**	**0.886**	**0.683**	**0.602**

on the NVIDIA Jetson Orin[3] and a NVIDIA RTX A4000. The NVIDIA Jetson board family has recently received a lot of interest for future space missions[4][5]. Our results are shown in Table 4, all values are based on images of size 960×600.

Table 4: GPU Memory Requirements and Inference Time.

	VRAM usage	Jetson time	A4000 time
Mask R-CNN	1.4 GB	0.5s	0.09s
YOLO-v7	1.8 GB	0.4s	0.06s
Faster R-CNN	0.6 GB	0.06s	0.01s
GroundingDino	1.7 GB	1.1s	0.12s

The Faster R-CNN detector with the MobileNetv3 backbone has the overall smalles memory footprint and inference time that allows object detection above 10Hz on the Jetson. The YOLO architecture performs faster than Mask R-CNN, yet requires more GPU memory. At last, the large, transformer-based GroundingDINO is the slowest of the methods.

5. Conclusion

In this work, we have shown the importance of data augmentations to bridge the domain gap between synthetic and real-world images in the scope of computer vision applications in orbit. To this end, we investigated the impact of applying data augmentations during training on three different networks, Mask R-CNN, Faster R-CNN, and YOLO-v7 on the SPEED+ dataset for satellite object detection. We have performed a hyperparameter optimization over a set of 29 common augmentations, often used in terrestrial computer vision tasks, with additional, custom space augmentations, simulating common visual effects encountered on objects in space. The results of this search is a set of key and performance-boosting augmentation techniques for each detector class. To spark the use of data augmentation techniques in space, we share these implementation as well as the resulting list with the community. Additionally, we benchmark GroundingDINO, an open-set detector that does not need to be trained on satellite imagery, on the same real-world test data. This ablation and poor performance further underlines the unique visual conditions that computer vision models are exposed to in the space domain. To conclude, we hope that this work contributes to making object detectors, applied to the space domain, more robust and widely used.

[3]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[4]https://www.satellitetoday.com/technology/2024/06/10/planet-to-use-nvidia-ai-processor-onboard-pelican-satellite/

[5]https://www.space.com/ai-nvidia-gpu-spacex-launch-transporter-11

Acknowledgments

The authors gratefully acknowledge the computational and data resources provided through the joint high-performance data analytics (HPDA) project “terrabyte” of the German Aerospace Center (DLR) and the Leibniz Supercomputing Center (LRZ).

References

[1] T. H. Park, M. Märtens, G. Lecuyer, D. Izzo, and S. D’Amico, “Speed+: Next-generation dataset for spacecraft pose estimation across domain gap,” *arXiv:2110.03101 [cs]*, 2021.

[2] T. H. Park, M. Märtens, M. Jawaid, Z. Wang, B. Chen, T.-J. Chin, D. Izzo, and S. D’Amico, “Satellite pose estimation competition 2021: Results and analyses,” *Acta Astronaut.*, vol. 204, pp. 640–665, Mar. 2023.

[3] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. S. Yu, “Generalizing to unseen domains: A survey on domain generalization,” *arXiv [cs.LG]*, Mar. 2021.

[4] M. Ulmer, M. Durner, M. Sundermeyer, M. Stoiber, and R. Triebel, “6d object pose estimation from approximate 3d models for orbital robotics,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 749–10 756.

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[6] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” *European Conference on Computer Vision*, 2023.

[7] I. Goodfellow, “Deep learning,” 2016.

[8] T. Hodan, P. Haluza, S. Obdrzalek, J. Matas, M. Lourakis, and X. Zabulis, “T-less: An rgb-d dataset

for 6d pose estimation of texture-less objects," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2017.

[9] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Lecture Notes in Computer Science*, ser. Lecture notes in computer science. Cham: Springer International Publishing, 2019, pp. 682–697.

[10] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 801–814, Apr. 2019.

[11] M. Sundermeyer, Z. C. Marton, M. Durner, and others, "Implicit 3d orientation learning for 6d object detection from rgb images," *Proceedings of the*, 2018.

[12] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," *arXiv [cs.CV]*, Aug. 2020.

[13] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv [cs.CV]*, Feb. 2017.

[14] Z. Wang, M. Chen, Y. Guo, Z. Li, and Q. Yu, "Bridging the domain gap in satellite pose estimation: A self-training approach based on geometrical constraints," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 60, no. 3, pp. 2500–2514, Jun. 2024.

[15] T. H. Park and S. D'Amico, "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *Adv. Space Res.*, Mar. 2023.

[16] J. I. B. Pérez-Villar, Á. García-Martín, and J. Bescós, "Spacecraft pose estimation based on unsupervised domain adaptation and on a 3d-guided loss combination," in *European Conference on Computer Vision*. Springer, 2022, pp. 37–52.

[17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ieeexplore.ieee.org, Sep. 2017, pp. 23–30.

[18] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering," 2020.

[19] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018.

[20] Y. Hu, S. Speierer, W. Jakob, P. Fua, and M. Salzmann, "Wide-depth-range 6d object pose estimation in space," *arXiv [cs.CV]*, Apr. 2021.

[21] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[22] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[23] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," *arXiv [cs.LG]*, Aug. 2017.

[24] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. IEEE, May 2018.

[25] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, Dec. 2019.

[26] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[27] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv [cs.CV]*, Jul. 2019.

[28] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, "Image data augmentation for deep learning: A survey," *arXiv [cs.CV]*, Apr. 2022.

[29] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *arXiv [cs.CV]*, Apr. 2021.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv [cs.CV]*, Jun. 2015.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:12670695

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:206594692

[34] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:206594738

[35] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, and B. Lai, "Pp-yoloe: An evolved version of yolo," *ArXiv*, vol. abs/2203.16250, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247793126

[36] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *ArXiv*, vol. abs/2107.08430, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236088010

[37] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *Journal of Information Science and Engineering*, 2023.

[38] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *International Conference on Learning Representations*, 2022.

[39] W. Kuo, F. Bertsch, W. Li, A. J. Piergiovanni, M. T. Saffar, and A. Angelova, "Findit: Generalized localization with natural language queries," *European Conference on Computer Vision*, vol. abs/2203.17273, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247839259

[40] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[41] P. T. Jackson, A. Atapour-Abarghouei, S. Bonner, T. Breckon, and B. Obara, "Style augmentation: Data augmentation via style randomization," *arXiv [cs.CV]*, Sep. 2018.

[42] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Neural Inf Process Syst*, pp. 2546–2554, Dec. 2011.

[43] S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv [cs.LG]*, Apr. 2023.

[44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, ser. Lecture notes in computer science. Cham: Springer International Publishing, 2014, pp. 740–755.

BIOGRAPHY

Maximilian Ulmer *received his B.Sc. and M.Sc. in Electrical Engineering and Information Technology from the Technical University of Munich (TUM). He is currently a Ph.D. student at the Karlsruhe Institute of Technology (KIT) and a research scientist at the German Aerospace Center (DLR) in the department for Perception and Cognition at the Institute of Robotics and Mechatronics. His research focuses on closing the Sim2Real gap for orbital perception, 3D vision, and object-centric computer vision for robotic manipulation.*

Leonard Klüpfel *received his B.Sc. in Industrial Engineering from the Friedrich-Alexander University of Erlangen-Nürnberg and his M.Sc. in Robotics, Cognition, Intelligence from the Technical University of Munich (TUM). He is currently a Ph.D. student at the Karlsruhe Institute of Technology (KIT) and a research scientist at the German Aerospace Center (DLR) in the department for Perception and Cognition at the Institute of Robotics and Mechatronics. His research focuses on computer vision for semantic scene understanding in the context of robotic manipulation with a special interest in detection and tracking of articulated objects with physical considerations and inspiration.*

Maximilian Durner *is a research fellow at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR) since 2016 and a Ph.D. student at the Technical University of Munich (TUM). Before, he studied Electrical Engineering at the TUM, partially studying at the Politecnico di Torino, Italy, and the Universidad Nacional de Bogota, Colombia. Currently, he is leading a research group on semantic scene analysis in the Perception and Cognition Department at DLR, which is working on robotic perception providing semantic information to interact with the surroundings. This includes known, unseen, and unknown object detection, pose estimation, and tracking. In this context, Maximilian focuses on robust and continuously adapting object-centric perception for mobile manipulation.*

Rudolph Triebel *received his PhD in 2007 from the University of Freiburg in Germany. The title of his PhD thesis is "Three-dimensional Perception for Mobile Robots". From 2007 to 2011, he was a postdoctoral researcher at ETH Zurich, where he worked on machine learning algorithms for robot perception within several EU-funded projects. Then, from 2011 to 2013 he worked in the Mobile Robotics Group at the University of Oxford, where he developed unsupervised and online learning techniques for detection and classification applications in mobile robotics and autonomous driving. From 2013 to 2023, Rudolph worked as a lecturer at TU Munich, where he teached master level courses in the area of Machine Learning for Computer Vision. In 2015, he was appointed as leader of the Department of Perception and Cognition at the Robotics Institute of DLR, and in 2023 he was appointed as a university professor at Karlsruhe Institute of Technology (KIT) in "Intelligent Robot Perception".*