EC4AS36

Scientific report on image augmentation with Generative AI

Document reference number for ANR

contact@confiance.ai | www.confiance.ai

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Bouchra Harnoufi	Airbus	AI engineer
Scientific responsible			
Co-authors	Nadira Boudjani	Valeo	AI engineer
	Bertrand Leroy	Renault	AI engineer
	Hassan Sleiman	Renault	AI engineer
	Gilles Foinet	Sirehna	AI engineer
	Antoine Jacquet	Jolibrain	AI engineer
	Julie Wang	Jolibrain	AI engineer
	Nicolas Winckler	Eviden	AI engineer

Document Control

Revision	Date	Commentary	Author
v1.0			

Contents

A. Introduction

A.1. General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness. . .).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.
  These are the challenges, among others, that the Confiance.ai program addresses.

A.2. Context and objectives

Data forms the foundation of machine learning models, with their performance significantly depending on both the quantity and the quality of the training datasets, particularly from a labeling perspective. Additionally, validating this performance with a sufficient level of confidence also necessitates access to a separate, large, diversified, and well-annotated validation dataset. However, acquiring, generating, and annotating real data comes with its own set of challenges. Synthetic and generated data can effectively address these challenges by providing a flexible and scalable solution that can be tailored to specific needs. It mitigates the issues associated with privacy and data security. Moreover, synthetic data can be generated to include rare events or edge cases that are difficult to capture in real datasets, thereby enriching the training process. As a result, synthetic data emerges as a valuable asset for enhancing the robustness and reliability of machine learning models.
In 2022, the Confiance.ai program initiated a series of investigations into the application of synthetic data for Computer Vision by Bosca et al. (2023). These early efforts primarily focused on utilizing GAN-based techniques and 3D-engine-generated data, particularly in validation settings. By 2023, the research extended to the exploration of diffusion models, as documented in a subsequent Confiance.ai study by Roussel et al. (2023), where these models were employed to enhance object detection datasets, with a focus on satellite imagery, through in-painting techniques. Additionally, a preliminary overview of Generative AI's role in Computer Vision was provided in a Confiance.ai methodological guideline Troya-Galvis et al. (2023).

The landscape of diffusion models has since evolved rapidly, with numerous new methods emerging, which offer enhanced control and generate higher-quality outputs. In this report, we present the experimentation and results of recent generative models for computer vision applications, with a focus on methods that generate annotated datasets for downstream tasks, such as bounding boxes for object detection. Specifically, we investigate three approaches: (i) **image editing models**, which allow modifications to images while preserving original labels, thereby maintaining image integrity. An application using the InstructPix2Pix editing model to augment foggy images in the BDD100k dataset is discussed in Section B; (ii) **image-to-image translation methods**, which diversify visual styles while keeping labels consistent. Applications utilizing GAN-based style transfer, CycleGAN-Turbo, and a CycleGAN-Turbo variant with CUT are presented in Sections C, D, and E, respectively. Results on BDD100k, Mapillary and rarePlanes datasets are presented ; and (iii) **layout-to-image models**, where images are generated from layouts that directly correspond to labels, ensuring precise control over scene composition. This includes inpainting methods and approaches based on ControlNet, where the diffusion process is conditioned on spatial layouts such as segmentation masks or bounding boxes. Application of these methods are presented in Section E and F.

B. Evaluation of Generated data using InstructPix2Pix Editing model and blending

B.1. Method

In the following, we explore the combination of edition model with blending as in BlenDA Huang et al. (2024) approach. BlenDA presents a method for unsupervised domain adaptation (UDA) that leverages the InstructPix2Pix Brooks et al. (2023) editing model to modify Cityscapes images (e.g., transforming clean weather scenes to foggy conditions) to better match the target domain. To avoid issues of obscured annotations and noisy labels from these changes, the authors introduce a blending technique that combines the original and modified images using a learnable mixing coefficient, δ. This blended approach creates an intermediate domain that preserves object visibility (see Fig B.1), which is then used to train the Adversarial Query Transform (AQT) Huang et al. (2022), a transformer-based object detector that uses adversarial learning to align source and target domains. Their experiments showed significant mAP gains (+6.3 for Foggy Cityscapes and +4.3 for BDD100K daytime) compared to using AQT alone.

Figure B.1: When applying edition on a source image ($\delta = 0$), such as adding fog or night, the resulting translated images ($\delta = 1$) may have their objects hidden or obscured. Using blending, e.g. with $\delta = 0.7$ allows to maintain the visibility of the objects, and hence enabling the original bounding box annotations. Courtesy from Huang et al. (2024).

B.2. Experiments and results

In our experiments, we have explored the usage of the InstructPix2Pix model with blending on the BDD100k clear to foggy weather domain adaptation. The clear-to-foggy scenario has been chosen due to the very small number of samples with fog in BDD100k, which amounts to a

Dataset	mAP@50	mAP@50:95	Run nb.
S_b (baseline)	61.29 ± 0.76	34.66 ± 0.4	7
$S_b + S_{0.7}$	61.44 ± 1.0	34.99 ± 0.5	6
$S_b + S_{0.9}$	59.53 ± 1.7	34.38 ± 0.7	3

Table B.1: Obtained results with YOLOv5m. S_b denote the real train set containing about 61k images. $S_{0.7}$ and $S_{0.9}$ denote the blended synthetic images with $\delta = 0.7$ and 0.9, respectively, each with 8k images. The Run nb. column indicates the number of run used for the obtained mean and standard deviation.

total of 142 images in the train and validation set. In our setting, we remove from the train set all images with fog resulting in a trainset of 61604 and build a validation set based on all 142 images with foggy conditions. Then, 8000 images from the train set are randomly selected and edited using the InstructPix2Pix model. The instruction prompt was set to "add some fog and grey sky" in the experiments. We conducted this process twice, for two blending coefficients $\delta = 0.7$ and $\delta = 0.9$. Visually good and bad generation examples are shown in Fig. B.2.

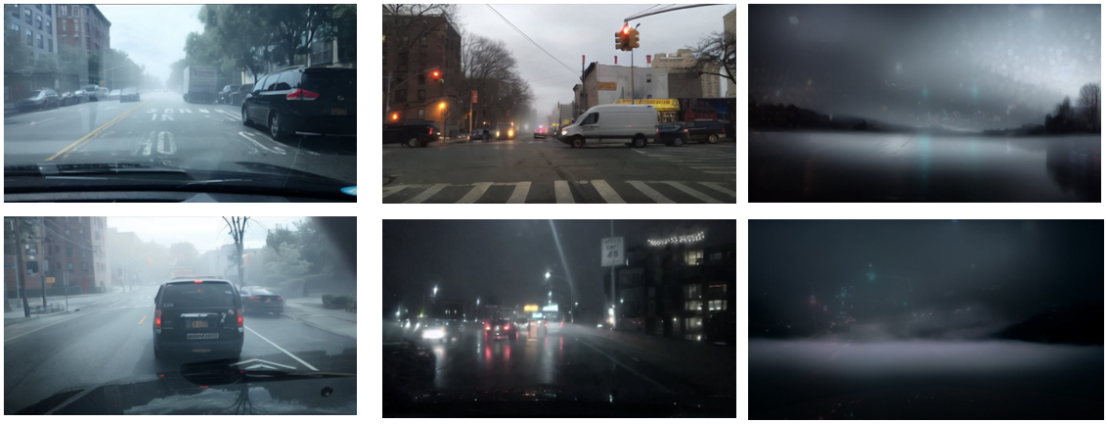


Figure B.2: Generation examples with $\delta = 0.9$. The two images on the right are examples of poor qualities generations, which, moreover, don't present much diversities.

All object detection models used are based on YOLOv5m and were trained for 120 epochs with a batch size of 16 and an image resolution of 640×640. The baseline model was trained solely on 61604 real images, with none depicting foggy conditions. Two additional configurations were evaluated: one trained on the same set of real images supplemented with 8000 blended synthetic images with a mixing coefficient $\delta = 0.7$, and another with the same setup but using a mixing coefficient of $\delta = 0.9$. For each setting, results are averaged over several training jobs and are presented in table **??**. We observe a slight improvement in the mean for a mixing coefficient of $\delta = 0.7$ but with a relative too large uncertainty to claim a real improvement.

B.3. Semantic analysis

To investigate the relevance and quality of the generated images we performed some analyses. We have first investigated whether the generated images are semantically aligned with real images. To do this we extracted train, validation and generated image embeddings from Dinov2 and

used Maximum Mean Discrepancy (MMD) to evaluate the distances between datasets. Dinov2 is a self-supervised vision transformer model that learns high-quality image representations without needing labeled data. It excels in capturing both global and local features of images, making it highly effective for understanding their semantic structure. The MMD quantify the degree of alignment between the different dataset distribution in a given space. We obtained an MMD of 0.33 between the two real train and valid set, while MMD of 0.92 and 1.02 are obtained for the synthetic vs valid set and for the synthetic vs train set, respectively. Despite being constructed from train set without fog, the synthetic data is closer to the target valid set with fog than the train set without fog. However, we observe that real train and valid set are still the closest. The Fig. B.3 shows a 2-dimensional UMAP projection of Dinov2 embeddings. We observe relatively good visual alignment between real and synthetic images, except at the center and bottom-right corner of Fig. B.3, where clusters of synthetic data are isolated. When inspecting these isolated synthetic data, it turned out that most of them were generation of bad quality images, by night and with a heavy horizontal fog, such as the two images shown on the right of Fig. B.2.

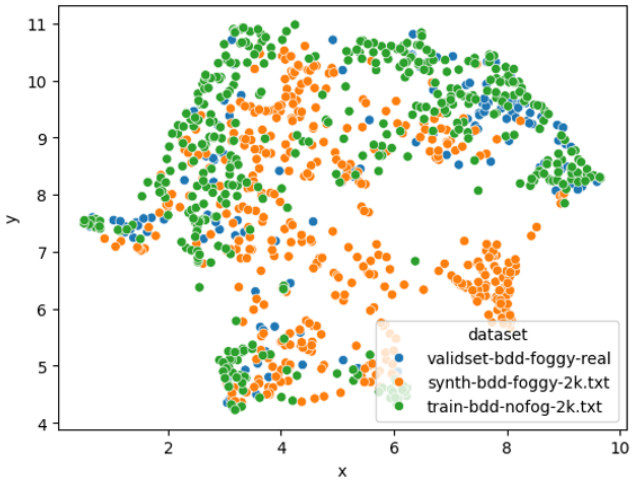


Figure B.3: 2-dimensional UMAP projection of Dinov2 embeddings for the valid set (blue) and 2000 images randomly sampled from the real train set (green) and synthetic set (orange). We observe synthetic data is rather well aligned with real data except for some data in the middle and for the cluster in the bottom right corner.

To better align with the target distribution, for each valid set image, we selected the $k = 10$ and $k = 20$ nearest neighbors from the generated set. The results did not improve the baseline results as shown in Table **??** and even degraded the results of the synthetic set with $\delta = 0.7$. One possible reason could be that the number of synthetic data have been significantly reduced, from 8000 to about 900 for $k = 10$ and to about 1600 for $k = 20$.

B.4. Distortion analysis

We have investigated other methods that could give other insight on the quality and relevance of the generated data. For example, we used ARNIQA, to analyze the quality in term of distortion. ARNIQA (leArning distoRtion maNifold for Image Quality Assessment) Agnolucci et al. (2024) is a self-supervised, no-reference image quality assessment (NR-IQA) method designed to model the image distortion manifold for robust feature representation of various distortion

Dataset	mAP@50	mAP@50:95	Run nb.
S_b (baseline)	61.29 ± 0.76	34.66 ± 0.4	7
$S_b + S_{0.7}^{k=10}$	60.71	34.92	1
$S_b + S_{0.7}^{k=20}$	61.24	34.43	1
$S_b + S_{0.9}^{k=10}$	60.40	34.71	1
$S_b + S_{0.9}^{k=20}$	59.63	34.55	1

Table B.2: Obtained results with YOLOv5m and *knn* filtering. S_b denote the real train set containing about 61k images. $S_{0.7}$ and $S_{0.9}$ denote the blended synthetic images with $\delta = 0.7$ and 0.9, respectively, each with 8k images. The k exponent indicates that the synthetic data have been filtered based on *knn* of given k. The Run nb. column indicates the number of run used for the obtained mean and standard deviation.

types and levels. The model is trained to cluster patches from different images that share the same type and level of distortion, thus capturing distortion characteristics across diverse content. After obtaining this distortion-aware representation, a linear regressor is trained to predict quality scores based on human judgments (mean opinion score). Compared to other NR-IQA methods, ARNIQA is more robust, as it effectively handles varying distortions across different image contents.

In our experiments, we have inspected the distribution of ARNIQA embeddings. It turned out that after UMAP projection, a large gap between real and synthetic images can be observed, as shown in Fig. B.4. This indicates that synthetic data have noise characteristic different from real data. Investigation using the KADID datasets, which is a dataset with different types of distortions, have not been conclusive, as no overlaps were observed between synthetic and KADID data. A preliminary analysis suggest that this noise is introduced after edition, that is, by the interpolation method used when rescaling the image to the original size. Indeed, in a similar approach with similar observations, original real images have been further downscale and then rescale to the original size and found to lay in the generated image region of the ARNIQA representation. Further analysis is required to evaluate if this type of distortions impact the detection model and if yes, how to mitigate them.

B.5. Summary and outlook

In this section, we explored an augmentation method based on the InstructPix2Pix editing model combined with blending to maintain the visibility of objects after editing. Training a YOLOv5 detection model on approximately 61k real images and 8k synthetic images showed no significant improvements in detection scores. Semantic analysis using DINOv2 revealed a generally good alignment with both the real training and validation images, except for some isolated clusters. Filtering using k-NN with $k = 10$ and $k = 20$ did not improve the results. Distortion analysis conducted with ARNIQA indicated a significant gap between real and synthetic data, suggesting the introduction of well-characterized noise.

Several aspects should be considered when interpreting the results of the experiments presented. First, the validation set of real BDD100k images with fog generally contains much lighter fog than that found in the synthetic images, which may indicate a domain shift. Adapting the prompt or reducing the blending may help address this issue. Second, the k-NN filtering was likely too stringent, leading to a significant reduction in the synthetic dataset size; using higher k values

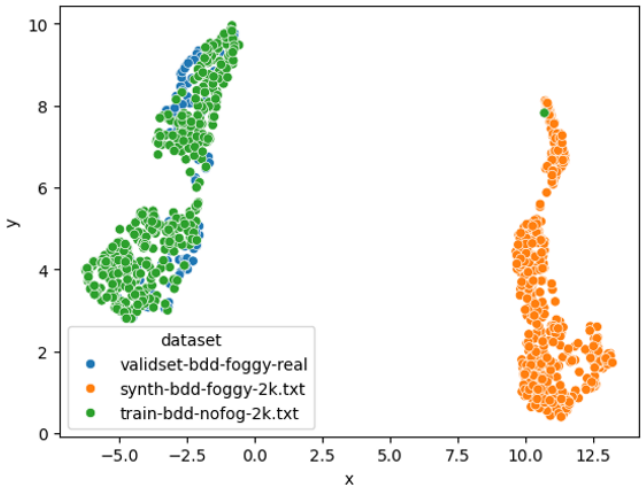


Figure B.4: 2-dimensional UMAP projection of ARNIQA embeddings for the valid set (blue) and 2000 images randomly sampled from the real train set (green) and synthetic set (orange). The proximity between two points denote the similarity of the type and level of distortion. We observe a large gap between synthetic and real images.

may yield better results. Furthermore, the observed gap in ARNIQA representation could impact the detection model. Preliminary analysis suggests that this noise occurs during the rescaling to the original image size due to the interpolation method. Further analysis is needed to determine if this noise impacts the detection model and, if so, to what extent. Finally, it is important to note that zero-shot or few-shot settings, which often yield significant improvements in the literature, were not explored in this study. Additionally, in our setup, combining the full real dataset with synthetic data, the inclusion of only 8k synthetic images may have been insufficient given the volume of real data. Previous studies have shown that increasing the ratio of synthetic to real data, typically above one, can improve model performance, whereas our study employed a much lower ratio of ≈ 0.13. This likely limited the potential benefits of the synthetic data. Future work could explore adjusting the synthetic-to-real ratio and incorporating few-shot settings to more effectively leverage synthetic data in similar scenarios.

C. Evaluation of Generated data using JoliGEN Style Transfer

C.1. Method

In this chapter, we present experiments on the use of images generated using a style transfer model from JoliGEN. This Experiments aim to improve the performance of an object detector. JoliGEN is a generative AI toolset developed by Jolibrain JoliGEN (2024). It allows training custom AI image -to-image models using techniques like GANs (Generative Adversarial Networks) and Diffusion models.
We used GANs with the CUT algorithm Park et al. (2020) for day-to-night transfer on the BDD100K dataset. The generated images were then used for training the YOLOv5 object detector. The goal is to improve object detection on night images, assuming we do not have any real night images available for training.

C.2. Experiments and results

In the first stage of our process, we generated 10,000 nighttime images using the GAN-CUT model, which was trained on the BDD100K dataset. The Figure C.1 shows some examples of day-to-night transfer.

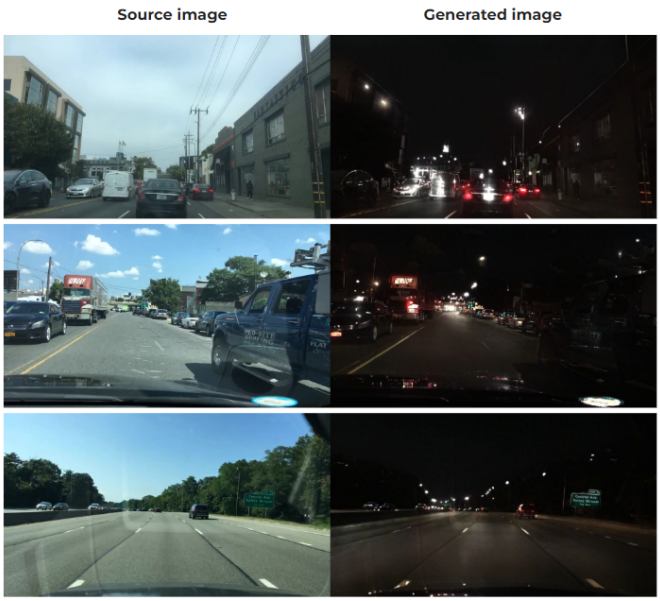


Figure C.1: Generated night-time image examples using JoliGEN GAN-CUT style transfer.

In the second stage, we conducted two experiments. The first involves training a YOLOv5s detection model exclusively on day-time images, with validation on a set of real day-time and night-time images, and testing on real night-time images. The second experiment involves training YOLOv5s on both daytime images and generated night images, with the same evaluation

and testing process as the first experiment (see Table **??**). The two models were trained for 250 epochs with a batch size of 300.

Experiments	Set	Day-time images	Real night-time images	Generated night-time images
Experiment 1	Train	40K	0	0
	Val	7K	7K	0
	Test	0	31K	0
Experiment 2	Train	40K	0	10K
	Val	7K	7K	0
	Test	0	31K	0

Table C.1: Experiments datasets

The validation KPIs for each experiment are detailed in Table C.2. Overall, the results are quite similar, with a difference of less than 1% in mAP50 favoring the experiment that used only daytime images (Experiment 1). However, there is a notable 3% improvement in mAP50 for the *bicycle* and *bus* classes. Performance in all other classes remains consistent.
The final test KPIs on real night-time images are presented in Table C.3. The results show an overall improvement of 2% in model performance, with a notable increase of 5% for the *bus* class, 4% for the *rider* class, 3% for the *motorcycle* class and 2% for the *truck* class, For the *pedestrian*, *car* and *bicycle* classes a slight increase of 1% is observed. The results show a slight degradation of 1% for the traffic light class and no improvement for the traffic sign class.

C.3. Summary and outlook

C.3.1 Data generation using JoliGEN GANs

Two versions of day-to-night transfer style GANs were analyzed: one from Confiance.ai batch 2 (second year of the program) and the other from Confiance.ai batch 4 (fourth year of the program). Overall, the nighttime images generated with these models are of good quality and appear realistic. However, Both versions have some limitations.
Day-to-night GAN batch 2:
We have identified two main limitations. The first limitation involves cases where the style transfer is poorly executed, with no visible change in the source images, as shown in Figure C.2 (a) . The second limitation pertains to cases where the style transfer is only partially successful: certain areas of the image remain overly lit, failing to convincingly represent a nighttime scene. An example of this can be seen in Figure C.2 (b).
Day-to-night GAN batch 4
Although the latest model performs better than the one from batch 2, the generated images still have some minor limitations. Specifically, reflections are particularly strong in certain images, especially in areas with light colors such as white and yellow. Additionally, the day-to-night transfer on images with snow results in excessive luminance in the snowy areas, which can make the images appear less realistic. Figure C.3 shows some examples.

Table C.2: Experiments validation KPI on real day-time and night-time images (BDD100K)

Experiments	Class	Images	Instances	P	R	mAP50
Experiment 1	all	14245	255498	0.731	0.45	**0.504**
	pedestrian		17983	0.721	0.474	0.552
	rider		911	0.733	0.349	0.431
	car		139906	0.779	0.675	0.748
	truck		5319	0.665	0.554	0.596
	bus		2263	0.703	0.544	0.601
	train		13	1	0	0
	motorcycle		610	0.688	0.37	0.451
	bicycle		1421	0.634	0.455	0.502
	traffic light		38679	0.659	0.523	0.545
	traffic sign		48393	0.727	0.557	0.617
Experiment 2	all	14245	255498	0.721	0.446	0.495
	pedestrian		17983	0.705	0.477	0.547
	rider		911	0.713	0.361	0.435
	car		139906	0.783	0.68	0.754
	truck		5319	0.653	0.539	0.567
	bus		2263	0.676	0.527	0.578
	train		13	1	0	0.00488
	motorcycle		610	0.665	0.387	0.448
	bicycle		1421	0.617	0.436	0.472
	traffic light		38679	0.671	0.508	0.54
	traffic sign		48393	0.73	0.545	0.607

C.3.2 Used of generated images

We used the generated night-time images to train the YOLOv5s model to enhance object detection in real night-time images. Surprisingly, even in the absence of night-time images, real and generated night-time images, the model performed quite well on real night-time images during the testing phase (Experiment 1). This is unexpected, as the model was trained exclusively with day-time images.

Looking at the performance of the model from Experiment 1, we observe that the model was not facing significant detection issues with night-time images. A global decrease of 8% in the model's validation KPIs compared to the test KPIs was observed. This does not reflect a significant drop in performance. Additionally, as we saw in the previous subsection, some images exhibit reflections in areas with light colors. These images may result in a degradation of the model's performance during training. Therefore, careful selection of the generated night-time images may be necessary to ensure optimal use of the generated images. Nevertheless, we observed an overall improvement of 2%, with a 5% increase for the bus class (Experiment 2).

Table C.3: Experiments test KPI on real night-time images (BDD100K)

Experiments	Class	Images	Instances	P	R	mAP50
Experiment 1	all	24879	395708	0.655	0.401	0.426
	pedestrian		16987	0.661	0.429	0.493
	rider		766	0.623	0.292	0.331
	car		214767	0.71	0.652	0.713
	truck		4519	0.616	0.455	0.474
	bus		2105	0.579	0.489	0.508
	train		23	1	0	0
	motorcycle		588	0.564	0.218	0.248
	bicycle		1264	0.506	0.423	0.416
	traffic light		75628	0.613	0.493	0.478
	traffic sign		79061	0.679	0.555	0.596
Experiment 2	all	24879	395708	0.661	0.419	**0.441**
	pedestrian		16987	0.641	0.462	0.505
	rider		766	0.622	0.345	0.37
	car		214767	0.734	0.657	0.726
	truck		4519	0.604	0.488	0.495
	bus		2105	0.616	0.523	0.553
	train		23	1	0	0
	motorcycle		588	0.532	0.274	0.276
	bicycle		1264	0.505	0.441	0.422
	traffic light		75628	0.641	0.461	0.469
	traffic sign		79061	0.715	0.536	0.598

(a) Example where style transfer is poorly executed on day-time images BDD100K

(b) Example where style transfer is partially executed on day-time images BDD100K

Figure C.2: Limitation examples images generation using day-to-night GAN batch 2.

Figure C.3: Limitation examples images generation using day-to-night GAN batch 4.

D. Evaluation of Generated data with img2img-turbo

D.1. Method

This chapter studies the impact of using single-step diffusion models for domain transfer on a detection task. A single-step diffusion model is trained on unpaired image datasets of synthetically generated satellite imagery on one hand, and real satellite imagery on the other hand, both from rarePlane Dataset Shermeyer et al. (2020)), to perform style transfer "from synthetic to real" (synth2real) and "from real to synthetic" (real2synth).

- A Single Step Detector is trained on synthetic images. The same detector is trained on the very same images processed with previously trained model (synth2real transformation). Detection performances are then compared on a test set of real images to measure the impact of style transfer for data augmentation with synthetic images.
- A Single Step Detector is trained on real images augmented with synthetic images, and real images augmented with processed synthetic images using previously trained model. Detection performances are then compared on a test set of real images to measure the impact of style transfer for data augmentation with synthetic images.

D.2. rarePlanes Dataset

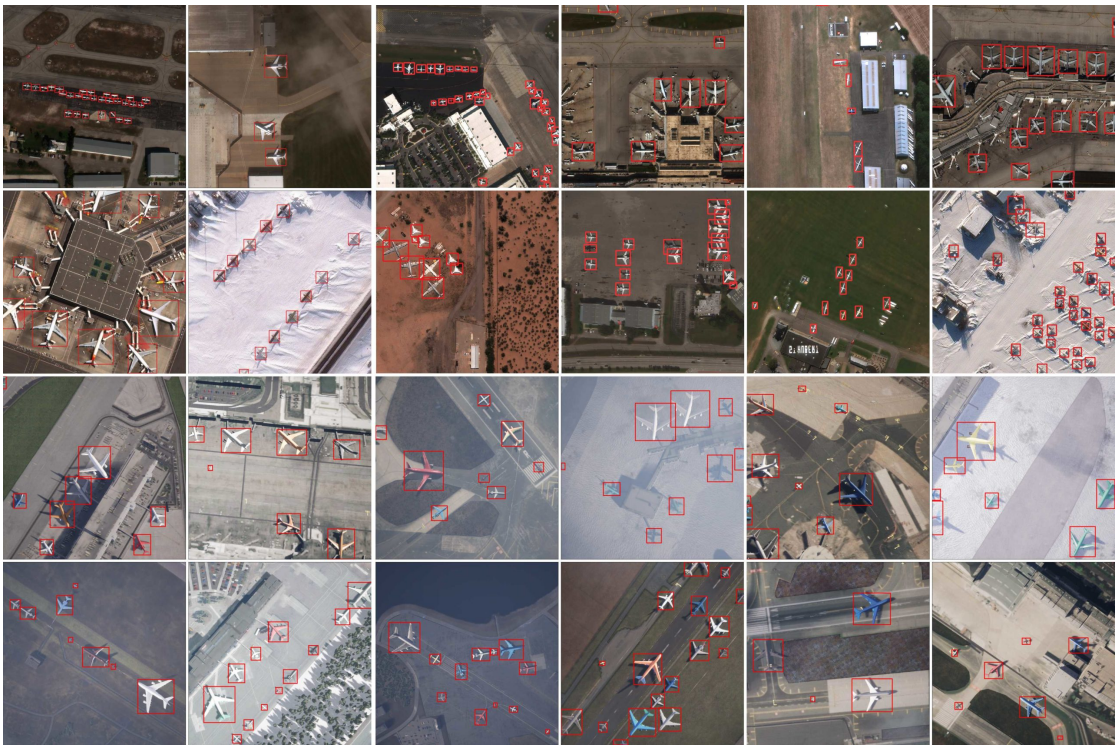


Figure D.1: rarePlanes dataset samples

Citation from Shermeyer et al. (2020): *'RarePlanes is a unique open-source machine learning dataset that incorporates both real and synthetically generated satellite imagery. The RarePlanes dataset specifically focuses on the value of synthetic data to aid computer vision algorithms in their ability to automatically detect aircraft and their attributes in satellite imagery. Although other synthetic/real combination datasets exist, RarePlanes is the largest openly-available very-high resolution dataset built to test the value of synthetic data from an overhead perspective. Previous research has shown that synthetic data can reduce the amount of real training data needed and potentially improve performance for many tasks in the computer vision domain. The real portion of the dataset consists of 253 Maxar WorldView-3 satellite scenes spanning 112 locations and 2,142 km^2 with 14,700 hand-annotated aircraft. The accompanying synthetic dataset is generated via AI.Reverie's simulation platform and features 50,000 synthetic satellite images simulating a total area of 9331.2 km^2 with approx. 630,000 aircraft annotations. Both the real and synthetically generated aircraft feature 10 fine grain attributes including: aircraft length, wingspan, wing-shape, wing-position, wingspan class, propulsion, number of engines, number of vertical-stabilizers, presence of canards, and aircraft role. Finally, we conduct extensive experiments to evaluate the real and synthetic datasets and compare performances. By doing so, we show the value of synthetic data for the task of detecting and classifying aircraft from an overhead perspective.'*

D.3. img2img-turbo

D.3.1 model description

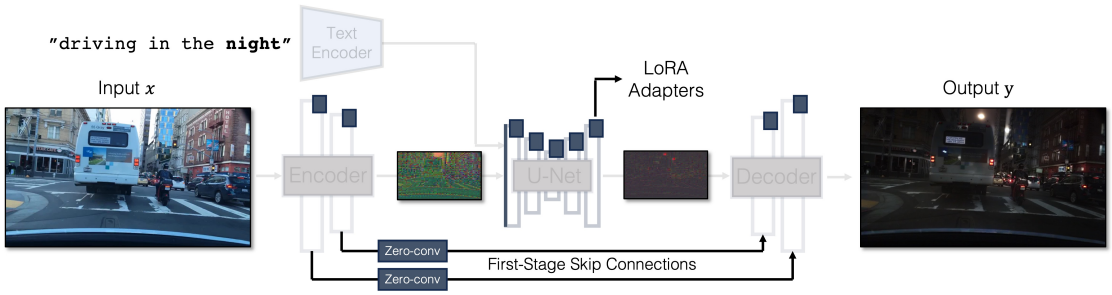


Figure D.2: Model architecture and example with "day to night" transform

Citation from Parmar et al. (2024): *In this work, we address two limitations of existing conditional diffusion models: their slow inference speed due to the iterative denoising process and their reliance on paired data for model fine-tuning. To tackle these issues, we introduce a general method for adapting a single-step diffusion model to new tasks and domains through adversarial learning objectives. Specifically, we consolidate various modules of the vanilla latent diffusion model into a single end-to-end generator network with small trainable weights, enhancing its ability to preserve the input image structure while reducing overfitting. We demonstrate that, for unpaired settings, our model CycleGAN-Turbo outperforms existing GAN-based and diffusion-based methods for various scene translation tasks, such as day-to-night conversion and adding/removing weather effects like fog, snow, and rain. We extend our method to paired settings, where our model pix2pix-Turbo is on par with recent works like Control-Net for Sketch2Photo and Edge2Image, but with a single-step inference. This work suggests that single-step diffusion models can serve as strong backbones for a range of GAN learning objectives.*

D.3.2 dataset preparation

The training scripts expect the dataset to be in the following format:

```
data
└── dataset_name
    ├── train_A
    │   ├── 000000.png
    │   ├── 000001.png
    │   └── ...
    ├── train_B
    │   ├── 000000.png
    │   ├── 000001.png
    │   └── ...
    ├── fixed_prompt_a.txt
    ├── fixed_prompt_b.txt
    ├── test_A
    │   ├── 000000.png
    │   ├── 000001.png
    │   └── ...
    └── test_B
        ├── 000000.png
        ├── 000001.png
        └── ...
```

A is real remote pictures of aircrafts, while B is synthetic remote pictures of aircraft. Trained model performs both transforms "A to B" ("a2b") and "B to A" ("b2a"). `fixed_prompt_a.txt` contains `"real remote picture of aircrafts"` and `fixed_prompt_b.txt` contains `"synthetic remote picture of aircrafts"` train_A contains 20.000 images of size 256 x 256, randomly cropped from real train dataset images (original size 512 x 512). test_A contains 200 images of size 256 x 256, randomly cropped from real test dataset images (original size 512 x 512). train_B contains 20.000 images of size 256 x 256, randomly cropped from synthetic train dataset images (original size 1920 x 1080). test_B contains 200 images of size 256 x 256, randomly cropped from synthetic test dataset images (original size 1920 x 1080).

(a) test_A/000000.png (b) test_A/000001.png (c) test_A/000002.png (d) test_A/000003.png

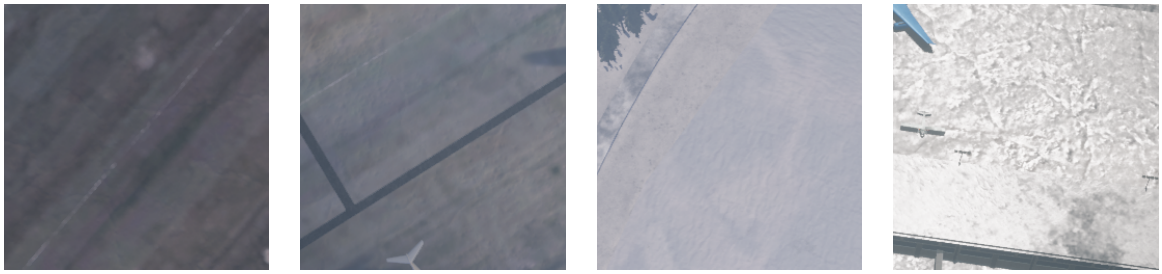
(e) test_A/000004.png (f) test_A/000005.png (g) test_A/000006.png (h) test_A/000007.png

(i) test_A/000008.png (j) test_A/000009.png (k) test_A/000010.png (l) test_A/000011.png

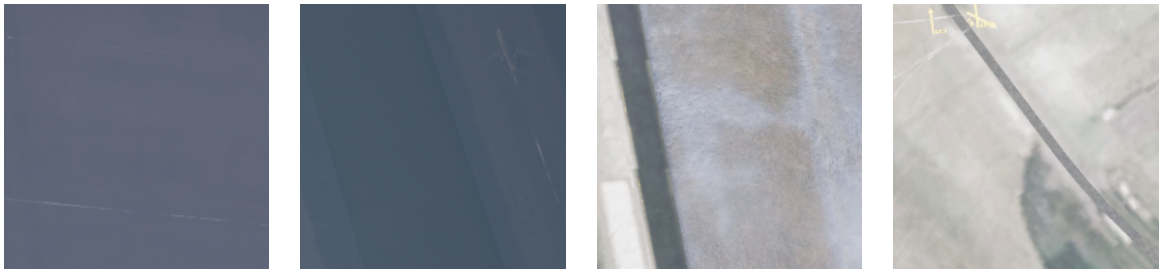
(m) test_A/000012.png (n) test_A/000013.png (o) test_A/000014.png (p) test_A/000015.png

Figure D.3: test_A samples (real images)

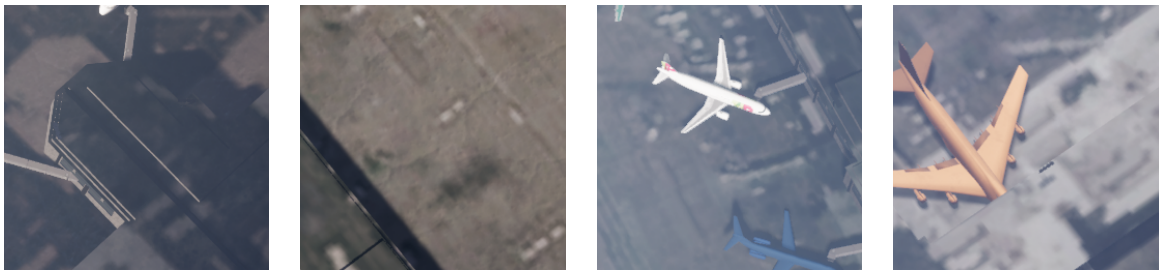
(a) test_B/000000.png (b) test_B/000001.png (c) test_B/000002.png (d) test_B/000003.png



(e) test_B/000004.png (f) test_B/000005.png (g) test_B/000006.png (h) test_B/000007.png



(i) test_B/000008.png (j) test_B/000009.png (k) test_B/000010.png (l) test_B/000011.png



(m) test_B/000012.png (n) test_B/000013.png (o) test_B/000014.png (p) test_B/000015.png

Figure D.4: test_B samples (synthetic images)

D.3.3 training

Main hyperparameters are:

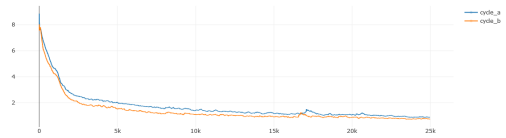
learning_rate	$1e^{-5}$
max_train_steps	25000
train_batch_size	1
gradient_accumulation_steps	1
validation_steps	250
lambda_gan	0.5
lambda_idt	1
lambda_cycle	1
enable_xformers_memory_efficient_attention	True
train_img_prep	"no_resize"
val_img_prep	"no_resize"

Hyperparameters value have been chosen to be similar to the "horse2zebra" example provided in the package. train_batch_size has been set to 1 to avoid CUDA out of memory issues during training, and image definition has been limited to 256 x 256.

training script is as following :

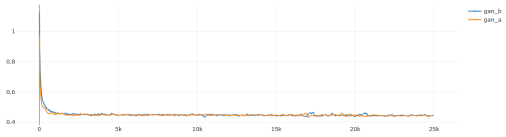
```
export NCCL_P2P_DISABLE=1
accelerate launch --main_process_port 29501 src/train_cyclegan_turbo.py \
    --pretrained_model_name_or_path="stabilityai/sd-turbo" \
    --output_dir="path/to/outputdir" \
    --dataset_folder "/path/to/dataset" \
    --train_img_prep "no_resize" --val_img_prep "no_resize" \
    --learning_rate="1e-5" --max_train_steps=25000 \
    --train_batch_size=1 --gradient_accumulation_steps=1 \
    --report_to "mlflow" --tracker_project_name "img2imgturbo_synth2real_planes" \
    --enable_xformers_memory_efficient_attention --validation_steps 250 \
    --lambda_gan 0.5 --lambda_idt 1 --lambda_cycle 1
```

Training duration was 18.5 hours, hardware used is one Nvidia GPU (RTX 3090) with 24GB of VRAM.

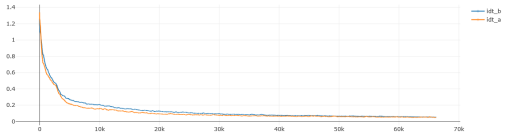


(a) train cycle losses (smoothed)

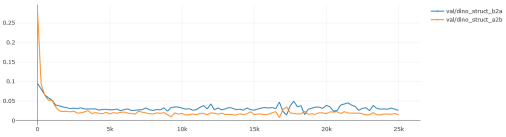
(b) train disc losses (smoothed)



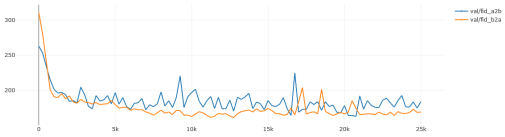
(c) train gan losses (smoothed)



(d) train idt losses (smoothed)



(e) validation dino struct losses (raw)

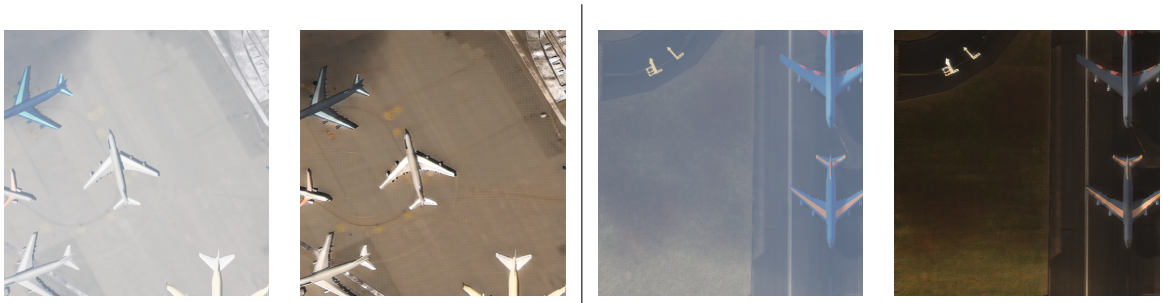


(f) validation fid losses (raw)

Figure D.5: Training and validation metrics

D.3.4 results

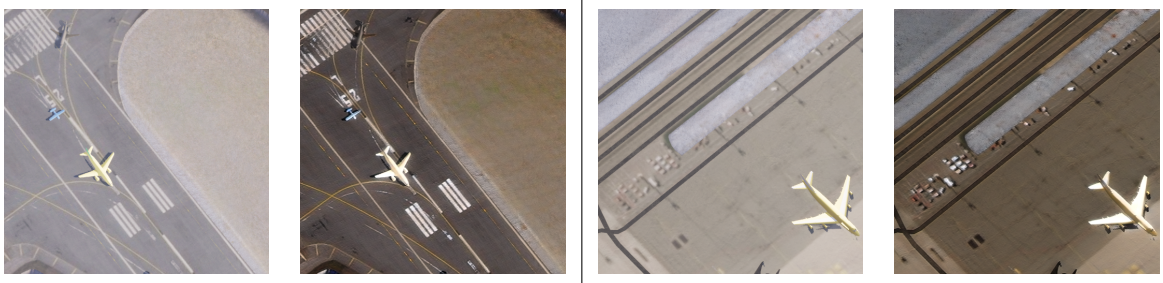
synth2real has been performed on images that don't belong to training or validation dataset, with greater definition of 512 x 512.



(a) real image 01 (b) synth2real image 01 (c) real image 05 (d) synth2real image 05

(e) real image 02 (f) synth2real image 02 (g) real image 06 (h) synth2real image 06

(i) real image 03 (j) synth2real image 03 (k) real image 07 (l) synth2real image 07



(m) real image 04 (n) synth2real image 04 (o) real image 08 (p) synth2real image 08

Figure D.6: synth2real results on 512x512 samples

D.4. evaluation on detection task

D.4.1 experiments description

D.4.1.1 model description

detection model is the Torchvision implementation of a Single-shot Detector with VGG16 backbone. No data augmentation is performed, pre-processing consists of scaling, resize to 300 x

300, removal of small bounding boxes (height or width <=5 pixels, area <= 10 pixels²), and tensorization.

D.4.1.2 data description

rarePlanes Dataset contains real images with annotations, in the form of tiles of size 512 x 512, and synthetic annotated images in the form of 1920 x 1080 images. Both real and synthetic images datasets are divided into train and test subsets.

- real train dataset contains 5815 images of size 512 x 512
- real test dataset contains 2710 images of size 512 x 512
- synthetic original train dataset contains 45000 images of size 1920 x 1080

From train synthetic images subset we extract 512x512 random tiles, bounding boxes are then filtered accordingly and a training dataset of synthetic images of size 512 x 512 is generated. images with no bounding box are dropped, resulting in 3537 synthetic tiles with a least one valid bounding box. synthetic tiles are then processed with our previously trained synth2real model to create a synth2real train dataset. that leads to:

- real train dataset contains 5815 images of size 512 x 512
- real test dataset contains 2710 images of size 512 x 512
- synthetic train dataset contains 3537 images of size 512 x 512
- synth2real train dataset contains 3537 images of size 512 x 512

D.4.1.3 experiment 1

in experiment 1 we compare detection performances on models trained on 3 different datasets:

- case 1: training set is real train dataset (5813 images)
  - train subset is composed of 4652 real images randomly chosen
  - validation subset is composed of 1163 real images randomly chosen
- case 2: training set is real train dataset plus synthetic train dataset (5815 + 3537 images)
  - train subset is composed of 4652 real images and 2829 synthetic images, randomly chosen
  - validation subset is composed of 1163 real images and 708 synthetic images, randomly chosen
- case 3: training set is real train dataset plus synth2real train dataset (5813 + 3537 images)
  - train subset is composed of 4652 real images and 2829 synth2real images, randomly chosen
  - validation subset is composed of 1163 real images and 708 synth2real images, randomly chosen

D.4.1.4 experiment 2

in experiment 2 we compare detection performances on models trained on 2 different datasets:

- case 1: training on synthetic data, validation on real data
  - train subset is composed of 3537 synthetic images
  - validation subset is rarePlane test dataset of 2710 real images
- case 2: training on synth2real data, validation on real data
  - train subset is composed of 3537 synth2real images

- validation subset is rarePlane test dataset of 2710 real images

D.4.2 performances

D.4.2.1 experiment 1 results

performances are evaluated on the rarePlane test dataset of real images (2710 images). For each case, best model is evaluated. Best model is determined by lowest validation loss. case 1 results are:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.273
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.559
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.235
```

case 2 results are:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.280
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.550
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.248
```

case 3 results are:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.280
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.550
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.255
```

synth2real transformation on training dataset shows no or no significant improvement of detection metrics ($AP_{50\%}$, $AP_{75\%}$, $AP_{50\%:95\%}$). It should be noticed that dataset augmentation (with synthetic images or synth2real images) doesn't improve detection metrics compared to real images only, when evaluated on real images. It might show better performances with test images that don't belong to the original training/testing dataset (better generalization).

D.4.2.2 experiment 2 results

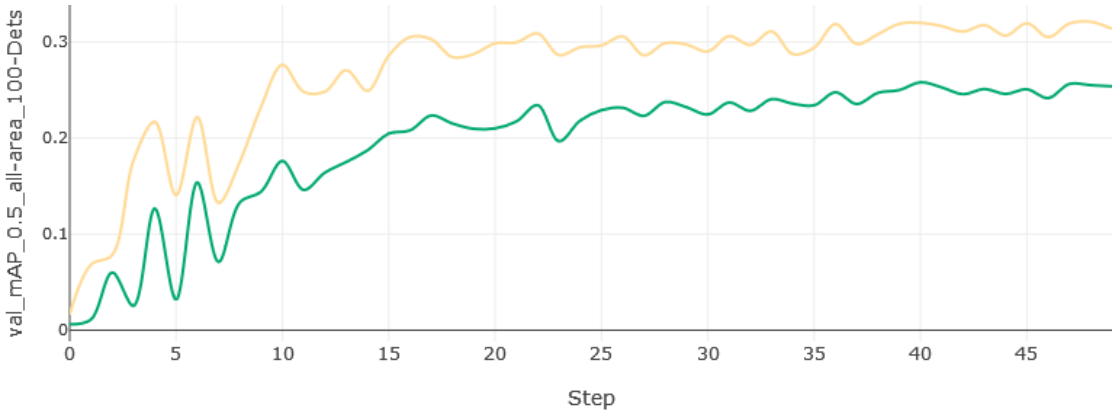


Figure D.7: Validation ($AP_{50\%}$ vs training epochs, with synthetic training set (green) and with synth2real training set (yellow)

D.5. Summary and outlook

- Training a cycleGan network for style transfer with unpaired data is pretty straightforward, though it requires high-end hardware. Training on a GPU with 24GB of VRAM was only

doable with images resolution of 256x256.

- Inference for data augmentation is fast as it is a one-step diffusion model (typ. 0.4 s for one 512 x 512 image with RTX3090 GPU)
- performances on a detection task, when training is performed on real data (62%) + synthetic data (32%) vs real data (62%) + synthetic data processed with synth2real transfer shows no difference when evaluated on real images.
- performances on a detection task evaluated on real images, when training data is only synthetic, are increased when training data is pre-processed with style transfer: synth2real

E. Evaluation of Generated data using JoliGen Style Transfer and Inpainting (temporary title)

E.1. Mapillary / Inpainting traffic signs

E.1.1 Mapillary / Training for traffic signs inpainting

We train a DDPM model to inpaint traffic signs conditioned by a Canny sketch using the Mapillary dataset as follows:

- During training, a random bounding box is sampled. A crop of the original image is extracted around the sampled bounding box.
- A Canny sketch using random thresholds is computed for the content of the bounding box. The content of the bounding box (original traffic sign) is masked.
- The model is trained to restore the original traffic sign conditioned by the Canny sketch.

Doing so, the model also learns to inpaint the background around the traffic sign to match the context around the bounding box.

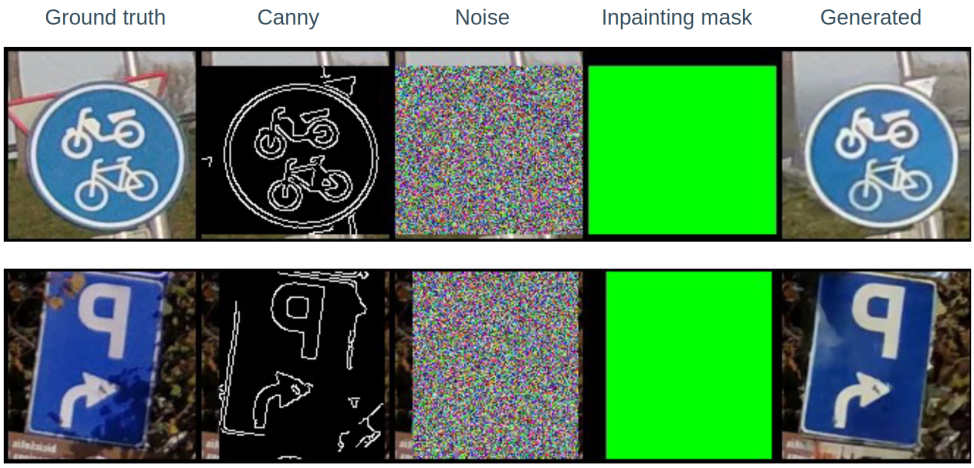


Figure E.1: A model is trained to reconstruct the inpainting zone, controled by a canny sketch. The model uses the remaining context pixels to fit the sign within the background.

E.1.2 Mapillary / Real vs Fake separability

Objective is to study the separability between real and fake traffic signs. The underlying hypothesis is that the lower the separability, the most useful the generated imagery should be for both testing and training purposes.
The methodology is as follows:

- We sample 1000 images from the Mapillary dataset that have not been seen during the training of the inpainting model.
- For each image, we do an inference conditioned by the Canny of the original traffic sign.

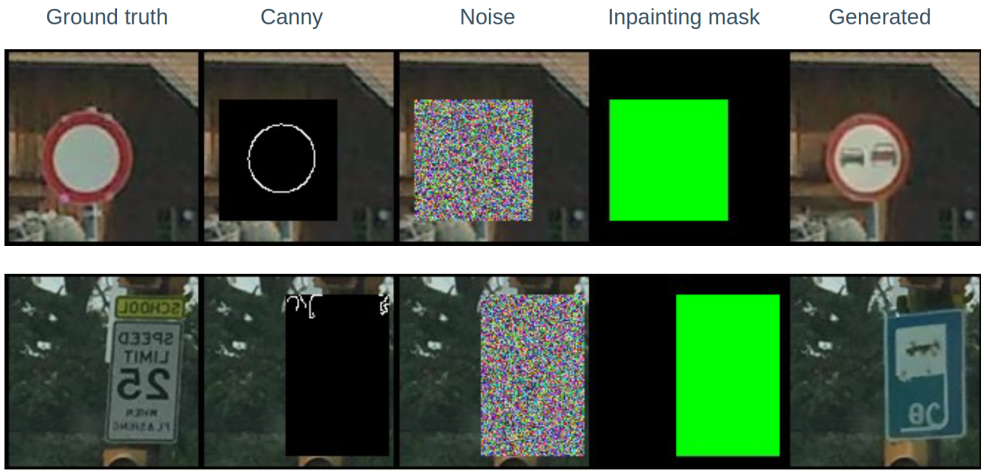


Figure E.2: Conditional canny sketches are randomized within thresholds. This figure shows two training samples with this sparse conditioning scheme.

Figure E.3: Inference on held-out test data using a canny from the real image.

This produces a set of generated traffic signs that should be as close as the real ones, as the model can do (since the canny is precisely that of the real image).
Note this is exactly the same task that the model has learned during training, but executed on a held-out image set. Thus we produce 1000 pairs of real / fake images. Only the crop are kept around every traffic sign.
Then we train a classifier to detect if the input image is a real image or a fake image. We tried different model architectures and sizes, only best model is shown below.

- Train set: 800 pairs
- Test set: 200 pairs
- Input size: 128x128
- Model: convnext-xlarge
- Best F1 = 0.801
- Best overall accuracy = 0.832
- Best accuracy (correctly detected) for real class = 0.835
- Best accuracy (correctly detected) for fake class = 0.955

Conclusions:

- As show on Figure E.3, generated imagery is almost identical to the real images, at least when looked at with naked eyes.
- However, separability with a classifier is easy with high accuracy, thus indicating that fake images bear low frequency patterns (e.g. from the generator network's deconvolutions) that are captured by a simple convolutional classifier.
- The classifier is far from perfect though, meaning some fake imagery may in practice be almost indistinguishable from real one.

In consequence, there's doubt that the generated fake imagery can be useful for training at large. This is because gradients may rely on the low-frequency unwanted patterns, instead of other features. However, since some imagery is almost indistinguishable from the real distribution, some use-cases may work better than others.

E.1.2.1 Mapillary / Pixel-level separability

In order to better analyze the fake vs real separability, we conduct two experiments:

1. **Fake pixels separability**: instead of classifiying the whole generated images, we train a segmenter to find out which pixels may be fake.
2. **Canny prediction**: we train a segmenter to re-generate the conditional canny from real and fake images and compare the results (section E.1.2.2).

Using the same dataset as in the previous, we train a Segmentation task to detect inpainted pixels, as follows (see Figure E.4:

- For real crops, the target mask is empty (no inpainted pixels).
- For fake crops, the target mask is the inpainting mask.

We obtain the following results:

- Train set: 800 pairs
- Test set: 200 pairs
- Input size: 128x128
- Model: Segformer-B5
- Test on real images: 0.81 accuracy and 0.81 F1
- Test on fake images: 0.96 accuracy and 0.86 F1

Results are somewhat equivalent to those of the initial per image classifier. The IoU metrics show that real pixels in both real and fake images (those real context pixels around the inpainting zone)

Figure E.4: Targets for pixel-level separability: empty for real images (no fake pixels), the inpainting zone for generated images.

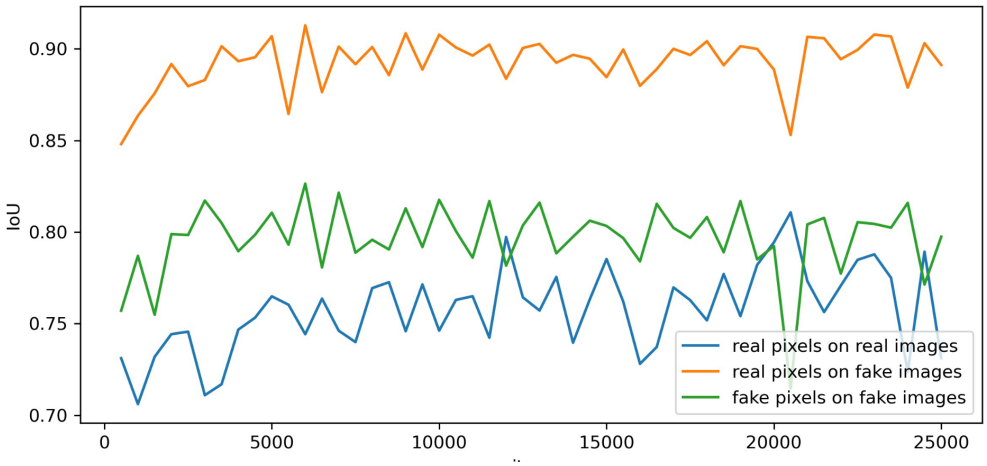


Figure E.5: Targets for pixel-level separability: per-class IoU.

are easier for the model to detect. Then fake and real pixels identification are close in difficulty, though fake pixels are detected more often (see Figure E.5.

E.1.2.2 Mapillary / Canny-level separability

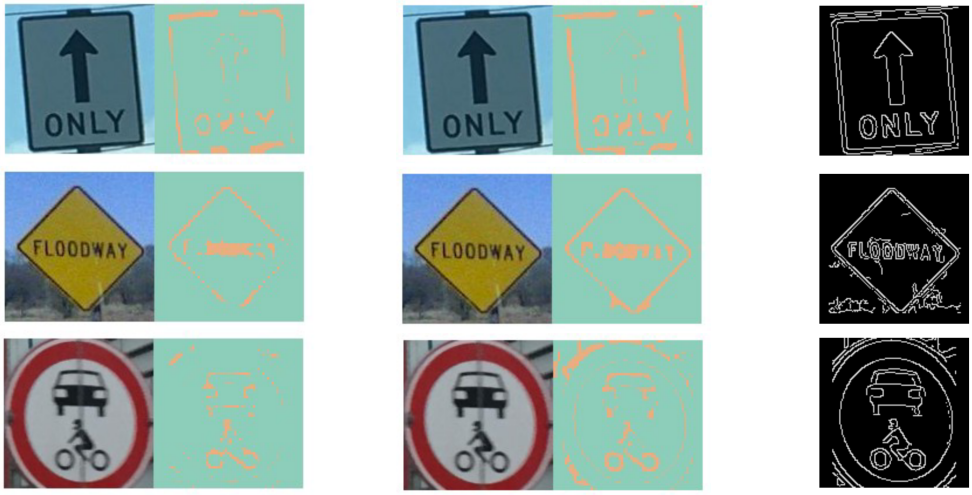


Figure E.6: Recovering the conditional canny on real and fake images.

Still using the same dataset, we know train a segmenter to predict the conditional canny. I.e. for a generated image, predict the canny of the real image.
For this we train two segmenters:

- Model real: trained using real images only
- Model fake: trained using fake images only

This allows using 'model real' on fake images as well as the opposite, see Figure E.6.
The best IoU for each class real and fake is around 0.57, which shows that recovering the canny is difficult. For us this is a signal that the model does significantly transform the canny and uses it for generating the fake traffic signs, and thus the canny is not leaked.

E.1.3 Mapillary / Improving metrics for minority class

The objective is to test whether generated traffic signs can be used to balance a training set and improve a classification accuracy on a minority class. By minority class it is meant a class for which the total number of images is so low compared to the volume of other classes, that it is almost ignored in loss by the classifier.
There are several ways to cope with very unbalanced datasets, and here we are testing one path that is to use diffusion-generated images to strenghen a minority class.
For this purpose, we focus on 6 classes of traffic sign from the Mapillary dataset:

- information—pedestrians-crossing–g1
- information—parking–g1
- information—disabled-persons–g1
- information—hospital–g1
- information—road-bump–g1

Figure E.7: Inference using the canny of the stairs minority traffic sign, pasted on other images (in place of other traffic signs)

- information–stairs–g1

Here *stairs* is a rare traffic sign in the Mapillary dataset.

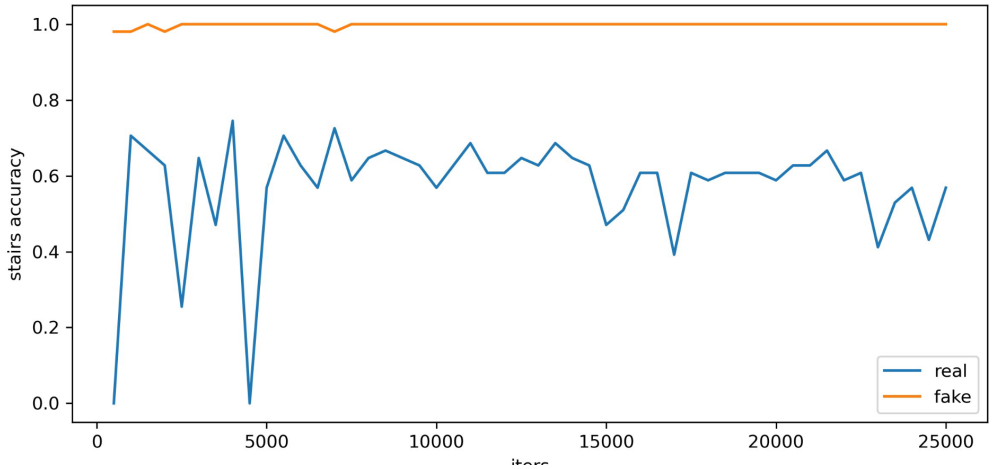


Figure E.8: Comparaison of accuracy for the stairs class (*real* is a model trained with real images only; *fake* is a model augmented with generated images).

We proceed as follows:

1. We train a classifier on real images crops, using only 3 instances of the stairs traffic sign. We do not train an object detector because the detection part of the traffic sign is straightforward and is independent of the sign class itself (i.e. training signal is common to all traffic signs). Thus only classification accuracy matters.
2. The model has a very good accuracy on all traffic signs expect the stairs, where the best accuracy is at 0.7451 for the stairs class. Note that oversampling the stairs traffic signs does not help.
3. We then generate fake stairs traffic signs, using Canny from real stairs bounding boxes and pasting them on other images, see Figure E.7. The generated images are sometimes of poor quality and sampling wrong colors, which we can explain since these traffic signs have not been seen a lot during the inpainting training.

4. We train a classifier on real images crops augmented by the fake images crops for the stairs traffic sign.

This allows to recover a near 1.0 accuracy on the stairs class, see Figure E.8.
We make the following observations:

- This shows that generated imagery can be useful in boosting minority classes.
- The generated imagery does not need to be visually perfect. We suppose that generation roughness and even errors are akin to data augmentation, thus building a halo of pixels around the minority class, securing separability from other classes during training.

Whether this technique can scale to multiple minority classes within a single dataset, or even to multiple close-looking minority classes is unknown at this stage.

E.2. BDD-100k / style transfer

In this section we train two style transfer models, on BDD-100k, and similarly to the work done on the Mapillary dataset, we study the separability of the generated images vs real ones.
The difference with Mapillary is that images are fully generated as opposed to partially inpainted, and thus the differences between real and fake more pronounced.

E.2.1 Clear2Snow: Training with cut-Turbo

First, Jolibrain has modified (Parmar et al., 2024) to use CUT(Park et al., 2020) along with stable-diffusion turbo model, that we denote *cut-turbo*.
Cut-turbo is trained as a GAN from a pretrained SDXL generative backbone, using Lora, and training the first convolutional layer after the VAE, from scratch. This allows to adapt to the image-to-image translation task at hand.
The reason to develop this model and integrate it into JoliGEN was to be able to train image-to-image tasks from high capacity foundation models. The underlying rational is that such models may have better capability than others in generating imagery that would be hard to distinguish from real ones. Thus bridging the gap of enhancing training datasets with generated data.
A model is trained on BDD-100k, from clear weather to snow. Figure E.9

E.2.2 Day2Night: Training with cut

For comparison we train a CUT generator from scratch with a mobile Resnet. The reason is for the model to act as witness to the cut-turbo model, even if the tasks are different (day to night).

E.2.3 Comparison

The table on Figure E.11 summarizes the number of parameters between the cut and cut-turbo models.

E.2.4 BDD-100k / Real vs Fake separability

We trained a CUT-Turbo model to do clear2snow style transfer.
We have proceeded as follows:

- Using 200 unseen clear images that have not been seen during the training, we produce 200 fake snow images. These are full BDD100K images in half resolution (640x360).
- We also have 200 real snow images, resized to the same resolution using the same interpolation.

Figure E.9: Generated samples from cut-turbo trained on BDD-100k (clear to snow).

Figure E.10: Generated samples from cut with a mobile Resnet trained on BDD-100k (day to night).

	CUT	Number of parameters	CUT-turbo	Number of parameters
G	mobile_resnet_attn	9.4M	stabilityai/sd_turbo	1300M
D	vision_aided	0.132M	vision_aided	0.132M
	basic	2.76M	basic	2.76M
	projected	12.67M	projected	12.67M

Figure E.11: Comparison between CUT and CUT-turbo models.

- Then we train a classifier to detect if the input image is a real image or a fake image.

Results are as follows:

- Train set: 160 fake snow + 160 real snow
- Test set: 40 fake snow + 40 real snow
- Model: convnext-tiny
- Best overall accuracy = 0.97500
- Best accuracy for real class = 1
- Best accuracy for fake class = 1

This shows that despite the finetuning of a foundational model, the separability is near-perfect. This shows that style-transfer is a difficult, undriven / semi-supervised task. The resulting imagery can possibly be too far from the target distribution, or bear patterns that are easily identified.

This implicitely shows that diffusion models, conditioned on precise information, are better candidates for generating data that is more useful for both testing and training.

F. Evaluation of Generated data using Layout-to-Image methods

Layout-to-image generation is a technique in data augmentation that uses predefined spatial layouts to create synthetic images. By defining object positions, sizes, and relationships within a scene, this approach allows for the generation of images that closely match specific compositions. This process offers significant advantages for augmenting datasets used in downstream tasks such as object detection and segmentation.
In this approach, the layout acts as a blueprint, guiding the generation of images that conform to precise spatial arrangements. This level of control enables the creation of diverse and contextually relevant synthetic data, which is especially valuable in scenarios with limited real-world annotations. By ensuring that generated images maintain the intended structure, layout-to-image models help produce datasets that enhance the training of models for tasks requiring accurate spatial arrangement.
Layout-to-image generation can be achieved using various techniques, including ControlNet-based methods Fang et al. (2024); Xiang et al. (2024), inpainting approaches Cho et al. (2024), and cross-attention-based methods Cheng et al. (2023); Zheng et al. (2023). In this section, we focus on two layout-to-image generation methods that leverage the ControlNet approach Zhang et al. (2023).

F.1. DODA

In this subsection, we explore DODA: Diffusion for Object-Detection Domain Adaptation in Agriculture Xiang et al. (2024). This paper presents a layout-to-image generation method based on ControlNet Zhang et al. (2023), using bounding box layouts to guide the image synthesis process. The authors introduce two setups: a multi-class method applied to the COCO dataset and a single-class method focused on domain adaptation in agricultural contexts. Both approaches build upon the Stable Diffusion 1.5 model and involve a three-stage training process: fine-tuning the Variational AutoEncoder (VAE), the Latent Diffusion Model (LDM), and training ControlNet with bounding box layouts.

F.1.1 Method

The VAE, specifically a VQGAN Esser et al. (2021), is trained in a self-supervised manner on source images (from the COCO Lin et al. (2014) or GWHD David et al. (2021) datasets) and is responsible for encoding and decoding the output of the LDM. In the single-class setup, the LDM is trained with a modified Stable Diffusion model, where the text feature extractor is replaced with an image feature extractor. Specifically, they utilize a ViT-B model trained via masked image modeling He et al. (2022) to better capture texture and high-frequency features from target domain images. The LDM is conditioned on these features using the cross-attention blocks that were originally used for text embeddings. The model is thus trained on source-target image pairs instead of text-image pairs. In contrast, the multi-class setup uses the standard text-to-image LDM, with prompts constructed from label annotations, such as "a photograph with 1 sink, 1 potted plant, 5 oranges, 1 banana, 3 chairs, 1 oven, 1 dining table, 1 refrigerator."

Finally, the ControlNet is trained either on image-text-layout triplets or source-target-layout triplets. For the multi-class setup, distinct colors were assigned to each class when training ControlNet, with input layouts (condition) color-coded accordingly. Multiple instances of the same class were represented using the same hue but with slightly reduced brightness for differentiation. To minimize occlusion in cases of overlap, the layouts were drawn in descending order of area.
The authors have evaluated their single-class augmentation approach for object detection task on a wheat target domain dataset and show significant improvement in YOLOX detection.

F.1.2 Experiments and Results

In the following we present the experiments performed using the multi-class approach, applied on BDD100k dataset.

F.1.2.1 Dataset preparation

We used the BDD100k training set to train the VAE, LDM and ControlNet.

Prompts In addition to bounding box annotations we exploited BDD100k metadata to build the prompt that describes the image. This includes the *Scene* (tunnel, residential, parking lot, undefined, city street, gas stations, highway), the *weather conditions* (rainy, snowy, clear, overcast, undefined, partly cloudy, foggy) and the *time of day* (daytime, night, dawn/dusk, undefined). An example of resulting prompt can be "a photograph of a city street scene with a clear weather during night, where we see 5 traffic lights, 4 cars, 3 pedestrians, 3 traffic signs".

Layouts Bounding box layout for training the ControlNet are built as in the original method (see above) using bounding box annotation, with assinged color for the different object classes, with slighted modified birghtness for multiple instance and drawn in descending order of area in case of overlaps.

F.1.2.2 Training

VAE The VAE training is self supervised on the training set images. The VAE encoder takes as input a resized image to 256×256, which is further downscaled to 64×64 in latent space and reconstructed back to 256×256 by the decoder. The training involves horizontal flip augmentation and an optional cropping augmentation method. We performed two trainings of 160k steps each, one including the cropping method and one without. In both cases, the initial learning rate was $2.5 \cdot 10^{-6}$.We compare reconstruction scores with the default pretrained models from CompVis' taming GitHub repository as shown in Table F.1.

Model	PSNR	SSIM	FID
pretrained	32.20	0.926	34.92
no cropping augm.	32.29	0.922	31.68
cropping augm.	32.33	0.924	32.06

Table F.1: Reconstruction preformance metrics for the VAE

We observe in Table F.1 that fine tuning with or without cropping augmentation improve the PSNR and FID scores and slightly decrease the SSIM score. The cropping augmentation provide

better results for PSNR and SSIM score but lower value for the FID. We rely on the FID score to select the VAE for training the LDM as it assesses how closely the generated data matches the target data distribution (BDD100k). We thus have selected the fine tuned VAE without cropping augmentation.

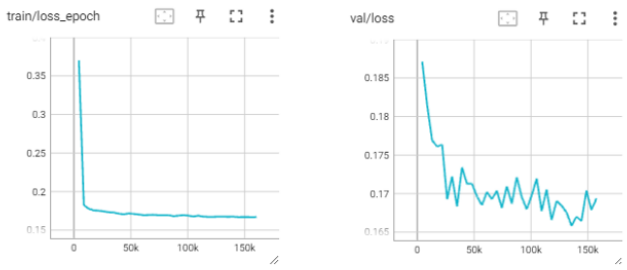


Figure F.1: LDM loss for training and validation data as a function of the iteration steps.

LDM The Latent Diffusion Model is trained on image-text pairs, and apply the diffusion process on the VAE latent space. We fine tuned the LDM from stable diffusion 1.5 with the BDD100k trainset and associated constructed prompts for 160k epochs with an initial learning rate of 10^{-5}. The Fig. F.1 shows the total loss for training and validation data as a function of the iteration steps. The total loss is, in both cases, dominated by the *simple loss* Ho et al. (2020) , which reflect the ability of reconstructing the input from the denoising process.

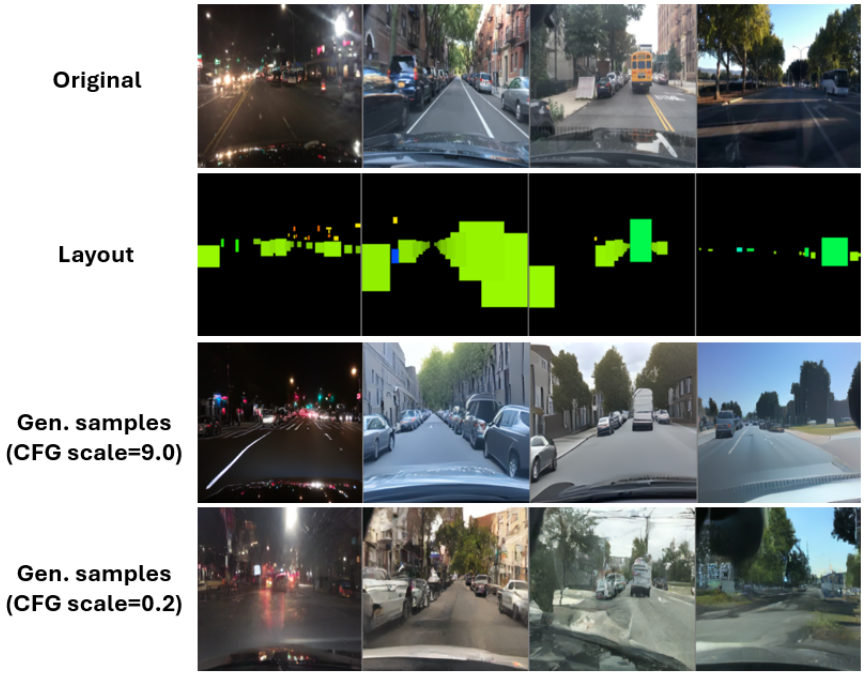


Figure F.2: From top to bottom: (i) images from training set, (ii) corresponding (flipped) layout, (iii) Generation samples with guidance scale of 9.0, (iv) Generation samples with guidance scale of 0.2. Note that horizontal flip is randomly performed on the layout

ControlNet Once the LDM is trained, we further train a ControlNet, which consists in fine tuning (i) a trainable copy of the LDM U-Net encoder with an extra condition (the bounding box layouts), and (ii) additional weights called zero-convolutions, which connects the trainable encoder and middle blocks to the frozen decoder blocks. The ControlNet has been trained for 80k steps on image-text-layout triplets with an initial learning rate of 10^{-5}.

F.1.2.3 Results

Fig. F.2 and Fig. F.3 illustrate a comparison between images from the training set, their corresponding (flipped) bounding box layouts, and samples generated using both text and layout conditioning with varying guidance scales. On one hand, we observe that lower guidance scales tend to introduce more artifacts, but seems to better capture the textures and characteristics of the target domain. On the other hand, higher guidance scales result in cleaner and sharper images with fewer artifacts, as the model adheres more strictly to the control inputs, producing outputs that are more polished and faithful to the provided conditions.

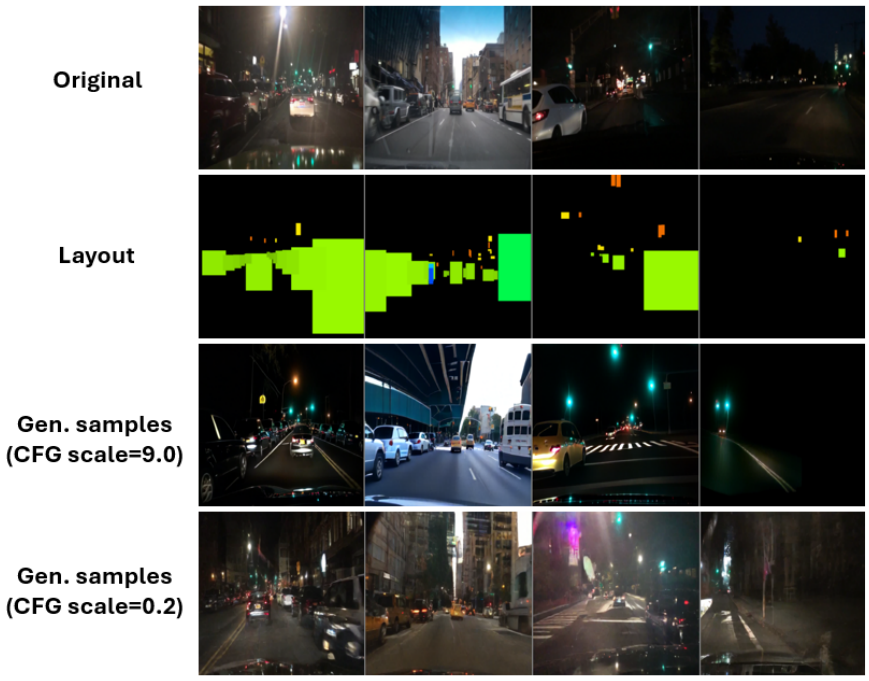


Figure F.3: Other generation examples. From top to bottom: (i) images from training set, (ii) corresponding (flipped) layout, (iii) Generation samples with guidance scale of 9.0, (iv) Generation samples with guidance scale of 0.2. Note that horizontal flip is randomly performed on the layout

F.2. SDXL, ControlNet and IPAdapter

While the generation method proposed by DODA shows promise, it requires a three-stage training process and significant computational resources, particularly for high-resolution images. Additionally, the domain adaptation method in DODA is not directly applicable to multi-class object detection. Therefore, we explored a fine-tuning-free approach that is similar to DODA

but conditions the model on both text and target image features.

F.2.1 Method

The method consists in combining three components: (i) Stable Diffusion XL (SDXL) Podell et al. (2023), a large foundation model capable of generating high-resolution images; (ii) ControlNet models Zhang et al. (2023) conditioned on Canny edges and depth maps; and (iii) IPAdapter Ye et al. (2023), which enables image prompting through decoupled cross-attention mechanisms. The core idea of this pipeline is to extract Canny edges and depth maps from a source image using the Canny algorithm and a pretrained depth estimation model. These extracted layouts are then used to condition the image generation process via pretrained ControlNet models, ensuring that the spatial structure of the source image guides the generated content, allowing the preservation of the original labels.
Additionally, by employing the IPAdapter on a reference target domain image, we can capture essential target domain features, such as textures and lighting conditions. This helps steer the generation process toward better alignment with the target domain, enhancing the relevance of the generated images.

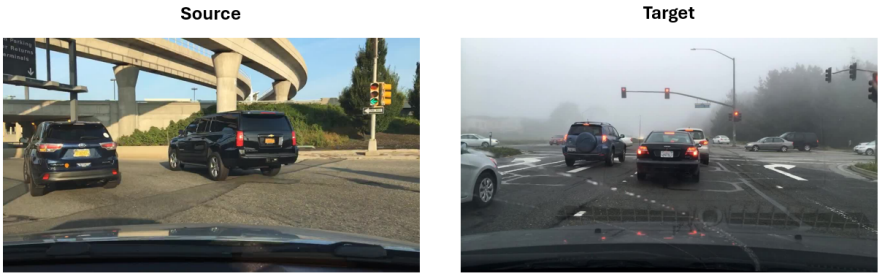


Figure F.4: Example of a source (clean weather) and target (foggy weather) images from BDD100k dataset.

F.2.2 Experiments and Results

We conducted a preliminary exploration of this approach using the ComfyUI framework for domain adaptation from clean to foggy weather, using BDD100k images. In Fig. F.4, the source image on the left is used only to extract Canny edges and depth maps for conditioning the ControlNet, while the image on the right, representing the target domain (foggy weather), serves as input to the IPAdapter module. Using these extracted conditions and the target image, Fig. F.5 illustrates three generation examples based on varying textual prompts. We observe that when no prompt is provided, the overall structure of the scene is preserved, and key aspects of the target domain, such as fog and changes in lighting, are captured. For instance, the generated foggy forest in the background and the adjusted lighting conditions align with the target domain. However, some objects have been altered or removed, such as the color change of the cars and the disappearance of the road sign in the top left corner and the traffic sign on the right. When using a simple prompt like 'A foggy day,' the generation is further guided toward a denser fog. In this case, the road sign class in the top left corner is preserved, though slightly altered, while the traffic sign on the right is still replaced with a road sign. A more detailed prompt can help preserve object classes more effectively, as shown in the text-image pair at the bottom of Fig. F.5. In thi example, the prompt was constructed in two steps. First, we used the source image as

input to an image-to-text model, specifically ChatGPT 4.0, which generated the following textual description: "*Daytime suburban street with a road stretching into the distance. Trees line both ides of the road. A road sign is visible on the left. Several vehicles are on the road. The scene is calm and well maintained*". We then manually modified the prompt by adding, "*The weather is foggy, and ...*" at the end of the last sentence, and introduced an additional sentence: "*There is a traffic light on the right*". This modification helped preserve all object classes in the generated image. It is worth noting that further improvements could be made using more advanced prompt engineering and incorporating a proper visual question answering (VQA) model, which would enhance the overall results and practicality of the method to produce entire datasets.

Figure F.5: Generation examples as a function of input text prompt.

F.3. Summary and outlook

In this section, we explored two layout-to-image methods. First, we tested the DODA approach in multi-class mode, using textual prompts without domain adaptation. This involved a three-stage training process for the VAE, LDM, and ControlNet. Our experiments suggest that low guidance scales (close to unconditional generation) produce images with artifacts but still resemble the training set distribution, while higher guidance scales result in cleaner, sharper images that diverge more from the training set, giving them a slightly synthetic appearance.
The second method combined SDXL, ControlNet, and IPAdapter. This approach utilizes pre-trained models and requires no additional training. In our preliminary experiments, IPAdapter effectively captured target domain elements, while ControlNet, conditioned on depth maps and Canny edges, helped preserve the overall structure of the scene. We also demonstrated that more detailed textual prompts led to better preservation of object classes in the generated images.

However, to fully automate generation for downstream tasks, a proper VQA model and refined prompt engineering would be necessary to produce annotated datasets. Additionally, using or combining other ControlNet input modalities, such as segmentation maps, HED, or Scribbles, could further improve results. Fine tuning a ControlNet on bounding box layout as in DODA could also be an option.
While both methods show promise for data augmentation in object detection, further investigation is needed to demonstrate their effectiveness in providing meaningful and efficient data augmentation.

G. Conclusion

Previous actions carried out within Confiance.AI have highlighted the interest of generative AI in improving confidence in the validation of object detection models. This action focused on the training phase and attempted to evaluate the performance gains that could be achieved using generative AI.
Within this action, several studies have been carried out across diverse use cases involving classification or object detection tasks. To generate new data, different generative AI models has been tested, either based on Generative Adversarial Networks (GANs) or on diffusion models (coupled or not with text-to-image models).
Based on these experiments, no definitive conclusion could be drawn. In some tests, the generated data has significantly improved the detection performance, while in other scenarios, the improvement was marginal. Overall, no negative impact has been identified.

G.1. Issues related to training on generated data

Based on the conducted experiments, several hypotheses can be proposed to explain why, in some cases, generating training data does not have a significant effect on the performance of classification or detection models.

Separability between real and generated data

From the perspective of connectionist models dedicated to classification or object detection, real and generated data are not indistinguishable. To support this observation, it has been demonstrated that a model trained to classify real and generated images, achieves high performances in separating real images from generated images. Furthermore, a distortion analysis conducted using the ARNIQA method reveals a significant gap between real and synthetic data, suggesting the introduction of specific noise by generative pipelines.

Artefacts and hallucinations

One of the lessons learned from the experiments is that it is necessary to clean the generated datasets before using them for training purposes. Whether using GANs or diffusion-based methods, a portion of the generated data consists of artifacts or "hallucinations" that heavily impact the performance of models if these data remain present in the training dataset.

Proportion of real and generated data

The marginal performance gains observed when augmenting real data with generated data can also be explained by an inadequate proportion of real/generated data within the training dataset. Based on the experiments, it appears that achieving a significant improvement may require a large proportion of generated data. However, it has not been possible to determine the optimal proportion, and whether it exists independently of the models and datasets.

G.2. Observed significant contribution of generated data to model performance

In some cases, substantial improvements have been achieved through the use of generated data. These are cases where the training dataset is significantly unbalanced (i.e., certain data classes are greatly underrepresented). Adding generated data for these classes, in proportions significantly larger than the real data, greatly enhances performance for both classification and detection tasks.

Bibliography

Agnolucci, L., Galteri, L., Bertini, M., and Del Bimbo, A. (2024). Arniqa: Learning distortion manifold for image quality assessment. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 189–198.

Bosca, A., Deveze, L., Leroy, B., Randon, Y., and Winckler, N. (2023). Ec5as12 guidelines on synthetic data usage in machine learning context. In *Confiance.AI - 2023 reports*.

Brooks, T., Holynski, A., and Efros, A. A. (2023). Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402.

Cheng, J., Liang, X., Shi, X., He, T., Xiao, T., and Li, M. (2023). Layoutdiffuse: Adapting foundational diffusion models for layout-to-image generation. *arXiv preprint arXiv:2302.08908*.

Cho, J., Li, L., Yang, Z., Gan, Z., Wang, L., and Bansal, M. (2024). Diagnostic benchmark and iterative inpainting for layout-guided image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5280–5289.

David, E., Serouart, M., Smith, D., Madec, S., Velumani, K., Liu, S., Wang, X., Espinosa, F. P., Shafiee, S., Tahir, I. S., et al. (2021). Global wheat head dataset 2021: more diversity to improve the benchmarking of wheat head localization methods. *arXiv preprint arXiv:2105.07660*.

Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883.

Fang, H., Han, B., Zhang, S., Zhou, S., Hu, C., and Ye, W.-M. (2024). Data augmentation for object detection via controllable diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1257–1266.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Huang, T., Huang, C.-C., Ku, C.-H., and Chen, J.-C. (2024). Blenda: Domain adaptive object detection through diffusion-based blending. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4075–4079. IEEE.

Huang, W.-J., Lu, Y.-L., Lin, S.-Y., Xie, Y., and Lin, Y.-Y. (2022). Aqt: Adversarial query transformers for domain adaptive object detection. In *IJCAI*, pages 972–979.

JoliGEN (2024). Style transfer on bdd100k.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Park, T., Efros, A. A., Zhang, R., and Zhu, J.-Y. (2020). Contrastive learning for unpaired image-to-image translation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 319–345. Springer.

Parmar, G., Park, T., Narasimhan, S., and Zhu, J.-Y. (2024). One-step image translation with text-to-image models. *ArXiv*, abs/2403.12036.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. (2023). Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Roussel, L., Bosca, A., and Boyer, J.-P. (2023). Scientific contribution on synthetic data generation using diffusion processes. In *Confiance.AI - 2023 reports*.

Shermeyer, J., Hossler, T., Van Etten, A., Hogan, D., Lewis, R., and Kim, D. (2020). Rareplanes: Synthetic data takes flight.

Troya-Galvis, A., Adjed, F., Bosca, A., Leroy, B., and Winckler, N. (2023). Methodological guideline for synthetic data generation, domain gap characterization and mitigation. In *Confiance.AI - 2023 reports*.

Xiang, S., Blok, P. M., Burridge, J., Wang, H., and Guo, W. (2024). Doda: Diffusion for object-detection domain adaptation in agriculture. *arXiv preprint arXiv:2403.18334*.

Ye, H., Zhang, J., Liu, S., Han, X., and Yang, W. (2023). Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*.

Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847.

Zheng, G., Zhou, X., Li, X., Qi, Z., Shan, Y., and Li, X. (2023). Layoutdiffusion: Controllable diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22490–22499.

Confiance ai

Title: Title in english

Keywords: Keyword 1, keyword 2

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Our partners:

Air Liquide AIRBUS Atos cea Inria
NAVAL GROUP GROUPE RENAULT SAINT EXUPÉRY SAFRAN sopra steria
SystemX THALES Valeo