Synth2Det: Loss-Free Unpaired Adverse-Weather Synthesis and Object-Detection Enhancement for Autonomous Driving

DAI Yuhang[a,1], **CUI Zhaoyu**[a,2] **and ZENG Tianyi**[a,2]

[a] *Department of Computing, The Hong Kong Polytechnic University*

The code is available at: https://github.com/hiteacherIamhumble/Synth2Det

Abstract—Safe autonomous driving demands perception stacks that remain reliable through adverse weathers like drifting snow that are both dangerous and dramatically under-represented in existing datasets. We introduce **Synth2Det**, a pipeline that *creates* high-fidelity adverse-weather imagery from unpaired clear-weather footage *and* immediately *leverages* it to harden an object detector, all without sacrificing **full-HD resolution**. First, a *modified CycleGAN* replaces transpose-convolution with bilinear up-sampling + 3×3 convolutions, eliminating checkerboard artefacts while preserving pixel sharpness. Second, we propose a *three-phase curriculum fine-tuning* of YOLO11m—use different datasets (from synthetic to real) to finetune. The entire training process is transparently tracked through `Weights&Biases` and Ultralytics HUB, ensuring reproducibility. On the challenging ACDC benchmark, our approach lifts mean AP by **195.75%** under heavy snow while maintaining real-time inference at 54.7ms on a single RTX 4090 compared to pretrained Yolo. These results demonstrate that targeted image translation, combined with task-aware adaptation, can close a critical realism gap and deliver tangible gains for *in-the-wild* autonomous-driving perception—without the prohibitive cost of manually collecting and labelling rare weather data.

Keywords—*Image2Image Translation, CycleGAN, YOLO11, Fine-Tuning, Loss-less Synthesis, Computer Vision, Auto-Driving*

Contents

1. Introduction

1.1. Motivation & Problem Statement

Safety–critical perception stacks for autonomous driving must remain reliable under rain, snow, fog and night-time glare. Yet today's public driving datasets are dominated by clear weather, and collecting dense annotations for every adverse scenario is prohibitively expensive. Synthetic data generation is therefore emerging as a pragmatic alternative. However, two gaps remain: (1) **Visual fidelity at full resolution**—classical CycleGAN [1] generators employ transpose-convolution layers that introduce checkerboard artefacts [2], degrading downstream detection; (2) **Task awareness**—most translation works are evaluated only by human realism, leaving open the question of whether the synthetic frames actually benefit modern detectors.

1.2. Key Contributions

Our work *Synth2Det* addresses both gaps through a tightly–coupled, two-task pipeline:

- **C1 Modified CycleGAN [1] for loss-free high-resolution synthesis (Task 1).** We replace all deconvolution layers with bilinear up-sampling followed by 3×3 convolutions, mitigating checkerboard artefacts [2]. Training is monitored with `Weights&Biases` [3].
- **C2 3-phase curriculum fine-tuning of YOLO11m (Task 2).** Starting from the pretrained `yolo11m` checkpoint [4], we successively fine-tuned a pretrained Yolo model while mixing synthetic and real data with adaptive weights inspired by curriculum learning [5]. All experiments are recorded in Ultralytics HUB[6].
- **C3 End-to-end evaluation on ACDC [7].** We demonstrate that our pipeline lifts mean AP by 195.75 %over the pretrained baseline under heavy snow, without sacrificing inference speed. Comprehensive ablation shows that both the bilinear generator and curriculum schedule are critical.

1.3. Paper Organization

Section 3 reviews prior work on adverse-weather translation and robust detection.

Section 4 details the construction of our clear–adverse image pairs and label-transfer protocol.

Section 5 describes the modified CycleGAN (Task 1) and the YOLO11m curriculum strategy (Task 2). Experimental settings are given in Section 6, followed by quantitative and qualitative results in Section 7.

Section 9 discusses limitations and future work, and Section 10 concludes.

2. Important Links

2.1. Dataset Links

- Task 1 (CycleGAN): https://drive.google.com/drive/folders/1_gI7gD-5zA0Gl9IOnRV_jfgVz4zm-nej?usp=sharing
- Task 2 (YOLOv11): https://drive.google.com/uc?id=14qs3q589VUVorn5rfmSUBgcYtbzf8fmE

2.2. Model & Training Links

- GitHub Repository: https://github.com/hiteacherIamhumble/Synth2Det
- Task 1 Model (CycleGAN): https://wandb.ai/22097845d-the-hong-kong-polytechnic-university/acdc-cyclegan?nw=nwuser22097845d
- Task 2 Models (YOLOv11):
  - Baseline: https://huggingface.co/Ultralytics/YOLO11/blob/5e0da476eb5def45e8080bd4b92ea63f0b16974c/yolo11m.pt
  - Fine-tuning v1: https://hub.ultralytics.com/models/sZgJJwYJ5KpXk2sZkAyW
  - 3-phased Fine-tuning v2: https://hub.ultralytics.com/models/Gf7q1a77wr4Y9HS9RSWd

3. Related Work

3.1. Adverse-Weather Image Translation (Task 1)

Early image-to-image translation methods relied on paired supervision, most notably Pix2Pix [8], which learns a deterministic mapping through conditional adversarial training. CycleGAN [1] removed this requirement by enforcing cycle-consistency, spawning a range of unpaired frameworks such as UNIT [9] and MUNIT [10]. More recent approaches pursue either scale-aware architectures (e.g. SPADE [11]) or representation disentanglement (e.g. CUT [12]) to handle high-resolution content. Weather-focused variants exploit condition codes or multi-domain generators to translate clear scenes into rain, snow or fog [13], [14]. Nevertheless, most still employ deconvolution layers that introduce checkerboard artefacts [2]. Our work adopts a bilinear-upsampled CycleGAN to preserve spatial regularity while maintaining full-resolution outputs during inference time via a tiling strategy.

3.2. Object Detection under Weather Degradation (Task 2)

Adverse weather severely deteriorates detector performance by obscuring fine-grained cues. Classical counter-measures include image pre-processing (dehazing [15], deraining, and desnowing) followed by off-the-shelf detectors such as YOLOv3/v5 [16]. More recent efforts integrate robustness directly into the network via weather-aware training [17] or adversarial feature alignment, exemplified by DA-Faster R-CNN [18]. Benchmark datasets such as Snow100K [19] and ACDC [20] provide dedicated evaluation suites, yet their annotation density remains limited. We instead augment clear-weather labels with our synthetic frames and conduct a 3-phase curriculum fine-tuning of YOLO11m, producing a task-aware detector that outperforms pre-trained weights by a significant margin.

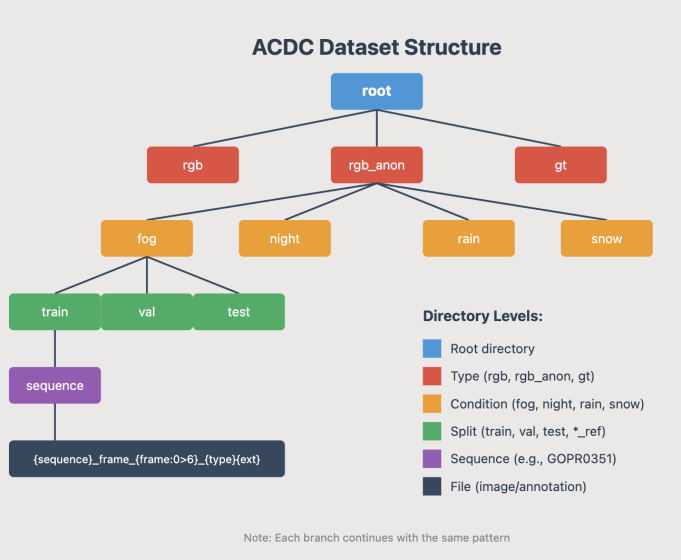


Figure 1. Directory layout of the original ACDC dataset.

3.3. Synthetic-to-Real Domain-Adaptation Pipelines (Both Tasks)

Bridging the gap between synthetic imagery and real perception tasks is a long-standing challenge. Early work on domain randomization suggested that massive texture perturbation could allow a network to generalise from simulation to reality [21]. Subsequent research formalised the idea through feature-level alignment using adversarial losses or adaptive batch-normalisation [22], [23]. In driving scenarios, Sim10K [24] and GTA5 [25] demonstrated that synthetic data can bootstrap detectors and segmenters when combined with unsupervised domain adaptation. Our pipeline follows a complementary philosophy: we *translate* abundant clear-weather footage into realistic adverse conditions with label tranfer handling, then fine-tune the detector using a curriculum schedule that progressively increases the synthetic-to-real ratio. This design couples low-level translation quality with high-level detection accuracy, and our ablations confirm that each stage contributes to the final performance gain.

4. Dataset Construction

4.1. Source Clear-Weather Data (T1)

We adopt the **ACDC** adverse-conditions benchmark, which contains 4006 high-resolution RGB images evenly split across *fog*, *night*, *rain* and *snow* scenarios [7]. Each adverse frame is accompanied by a geo-aligned clear-weather reference from the same location, enabling loose "pairing" without enforcing pixel-level correspondence. For Task 1 we extract all clear–snow tuples, yielding roughly 80 % / 20 % train/val splits (Table 1). Although the pairing is exploited only for qualitative visualisations (see Fig. 2), both domains are treated as *unpaired* during CycleGAN optimisation.

4.2. Unpaired Adverse-Weather Pools (T1)

The adverse subset retains the original ACDC directory hierarchy (`.../snow/train/img/...`), illustrated in Fig. 1. We preprocess all snow images to `256x256px` resolution as the official CycleGAN [1] to get good performance. Similar pools for *rain*, *fog*, and *night* are archived for future experiments but not used in the current synthesis model.

4.3. Automatic Annotation Transfer & Label Cleaning (T2)

The public ACDC COCO annotations include 19 classes; we remap them to the `YOLOv8` convention using `{24:0, 25:1, 26:2, 27:3, 28:4, 31:5, 32:6, 33:7}`, resulting in the label set *{person, rider, car, truck, bus, train, motorcycle, bicycle}*. Boxes that violate any of the

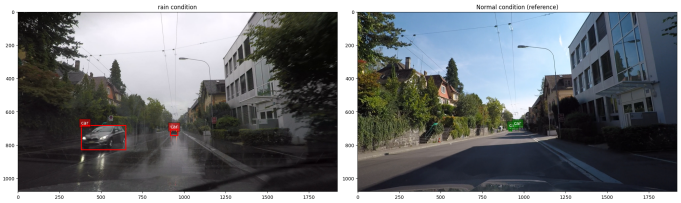


Figure 2. Geo-aligned "pair" under *rain*: clear reference (left) and adverse image with COCO bounding boxes (right).

Algorithm 1 Automatic Annotation Transfer & Label Cleaning

Require: COCO-annotated dataset $\mathcal{D}$ containing domains {clear, snow};
synthetic-snow generator G_{cyc};
class–mapping dictionary $\mathcal{M}$;
thresholds $\theta = (a_{\min} = 100,\ d_{\min} = 15,\ s_{\min} = 10)$;
sampling ratio $p = 0.5$
Ensure: Three YOLO-formatted datasets $\mathcal{D}^{\text{clear}}$, $\mathcal{D}^{\text{snow}}$, $\mathcal{D}^{\text{syn}}$

for all domain $c \in \{\text{clear}, \text{snow}\}$ **do**
  $\mathcal{I}_c \leftarrow$ random sample of size $p\,|\mathcal{I}_c^{\text{all}}|$ ▷ **half-image selection**
  for all image $I \in \mathcal{I}_c$ **do**
    $\mathcal{A} \leftarrow$ COCO annotations of I
    for all bounding box $(b, \ell) \in \mathcal{A}$ **do**
      if $\ell \in \text{keys}(\mathcal{M})$ **then** ▷ **class transfer**
        $\ell' \leftarrow \mathcal{M}[\ell]$
        if filter_pass(b, θ) **then** ▷ **bbox filtering**
          write $(\ell', \text{normalize}(b))$ to YOLO label file
        end if
      end if
    end for
  end for
end for

$\mathcal{I}_{\text{syn}} \leftarrow G_{\text{cyc}}(\mathcal{I}_{\text{clear}})$ ▷ **generate synthetic-snow images**
copy YOLO label files from $\mathcal{I}_{\text{clear}}$ to $\mathcal{I}_{\text{syn}}$
split each domain into `images/`{train,val} and `labels/`{train,val}
return $\mathcal{D}^{\text{clear}}$, $\mathcal{D}^{\text{snow}}$, $\mathcal{D}^{\text{syn}}$

function FILTER_PASS(b, θ) ▷ geometric checks
  $(w, h) \leftarrow$ width&height(b); $a \leftarrow w \times h$; $d \leftarrow \sqrt{w^2 + h^2}$
  return $(a \geq a_{\min}) \wedge (d \geq d_{\min}) \wedge (\min(w, h) \geq s_{\min})$
end function

heuristics in Algorithm 1 are filtered out since we assume that the synthesis environment may leads to the tiny objects hard to discover, which means the fog or rain may bring small suspended particles making the distant objects hard to be captured via cameras.

After pruning, we down-sample each condition to exactly 50 % of the remaining images to avoid the overfitting when fine-tuning and form three YOLO-ready splits: *real clear*, *real snow*, and *synth snow*. Figure 3 summarises the pipeline, and Fig. 4 shows representative frames from each sub-set.

4.4. Composite Dataset Statistics & Splits (B)

Tables 1 and 2 report the final image counts and annotation totals for both tasks.

Table 1. Task 1—Image counts used for CycleGAN training (good–snow example).

Domain	Train	Val
Clear (good)	400	100
Snow (adverse)	398	102

Table 2. Task 2—Pairs and bounding boxes after cleaning (50 % sampling).

Condition	Split	Pairs	Cond. BBox	Ref. BBox
Fog	train	200	1146	2227
	val	50	340	544
Night	train	200	1448	2726
	val	53	276	539
Rain	train	200	1025	2125
	val	50	347	491
Snow	train	200	1421	2553
	val	50	428	637

5. Methodology

5.1. Full-Resolution Weather Translation Network (Task 1)

5.1.1. Generator and Discriminator Architecture

We adopt the original CycleGAN formulation [1] with a **ResNet-9** generator and a 70×70 **PatchGAN** discriminator.

Generator We design the generator that employs a ResNet-based architecture, comprising three primary components: an encoder, residual blocks, and a decoder. Training images are resized to 256×256 for fast training process. The encoder begins with a reflection-padded convolutional layer that maps input channels to an initial feature dimension, followed by two downsampling blocks that progressively double the number of channels via strided convolutions, instance normalization, and ReLU activation.

Discriminator We employ a PatchGAN 70×70 design for the discriminator, which evaluates local image patches rather than the full image. It comprises four convolutional layers with increasing channel dimensions (ndf=64 to ndf×8), each using strided convolutions (stride=2), instance normalization, and LeakyReLU activations (slope=0.2). The final layer produces a 1-channel output map where each element corresponds to a 70×70 receptive field in the input image, enabling the discriminator to assess local texture realism. This design prioritizes high-frequency detail preservation and reduces artifacts by focusing on patch-level authenticity rather than global coherence. The resulting architecture is summarized in Fig. 5.

5.1.2. Loss Functions and Optimization

Following our CycleGAN, the total objective is

$$\mathcal{L} = \lambda_{\text{adv}}\big(\mathcal{L}_{\text{GAN}}^{A\to B} + \mathcal{L}_{\text{GAN}}^{B\to A}\big) + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}} + \lambda_{\text{id}}\mathcal{L}_{\text{id}}, \tag{1}$$

with weights $(\lambda_{\text{adv}}, \lambda_{\text{cyc}}, \lambda_{\text{id}}) = (1, 10, 5)$, identical to the original paper.

Inspired by the original CycleGAN paper, we incorporate an image replay buffer (`ImageBuffer`) for discriminator training. We employ an image replay buffer of size 50. Instead of training discriminators only on the most recent generator outputs, we randomly sample from a buffer of past generated images. This injects temporal variety and prevents discriminators from overfitting, thus maintaining training stability over long epochs.

Also, we use Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and initial learning-rate $\eta_0 = 2\times10^{-4}$. The rate is linearly decayed to 0 after epoch 100 (total 200 epochs).

5.1.3. Inference-time High-Resolution Synthesis

To retain the `2048x1024px` resolution of ACDC during inference while keeping GPU memory low, we *tile* the input into overlapping 256×256 patches with stride 192 (i.e. 64-pixel overlap). After translation, the patches are alpha-blended in the overlap regions. Figure 6 illustrates the scheme and its high-resolution output.

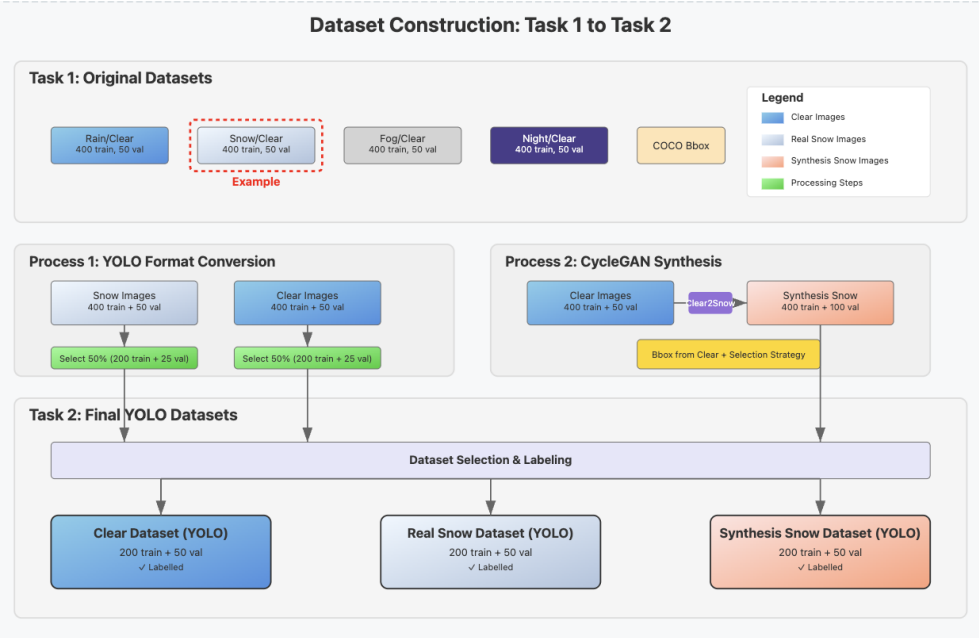


Figure 3. Dataflow for automatic annotation transfer, filtering and split generation used in YOLO11m fine-tuning (Task 2).

Figure 4. Bounding-box visualisation for *real clear* (left), *real snow* (centre) and *synth snow* (right).

5.2. YOLO11m Fine-Tuning Strategy (Task 2)

5.2.1. Test-Time Augmentation (TTA)

During validation we invoke `model.val(augment=True)`, which averages predictions over four flips and three multi-scales. TTA is known to lift mAP by 1–2pp in low-contrast scenarios by compensating for direction-specific snow streaks [26].

5.2.2. Data-Augmentation Policy

To make synthetic images less predictable we adopt a strong augmentation suite (Table 3) centred on *mosaic* [27], *mixup* [28], colour jitter and physically-motivated fog/snow overlays. These techinques are forcing the model to focus on shape rather than texture, which is crucial in adverse weathers where texture information is dramatically altered.

5.2.3. Three-Phase Curriculum Fine-Tuning

We adopt a three-phased curriculum as proposed in Algorithm 2 which yields a smooth, monotonic transfer from clear to adverse scenes, outperforming both naive single-phase finetuning and off-the-shelf pretrained weights (Figure 7). Here is the detailed three phases:

P1 Warm-up (3epochs). Clear-weather only; head layers unfrozen.

P2 Main fine-tune (40epochs). Mixed clear + synthetic + real-snow; full network unfrozen; strong augmentations.

Table 3. Augmentation hyper-parameters.

Operator	Probability	Range/Details
MixUp	0.25	—
Mosaic	0.50	—
HSV Jitter	1.00	$h = 0.15, s = 0.7, v = 0.4$
Random Rotation	—	$\pm 10°$
Random Translation	—	± 0.2
Random Scaling	—	± 0.5
Horizontal Flip	0.50	—
Perspective Transform	—	0.0005

P3 Alignment (10epochs). Real clear + real snow only; mild augmentations; cosine LR decay.

Phase P1 – Warm-up. Training only on real clear-weather frames recalibrates anchor priors and BatchNorm statistics to the target resolution while preventing catastrophic gradient shocks to low-level features [23]. Such layer-wise freezing mirrors the staged transfer strategy popularised in curriculum learning [5].

Phase P2 – Main Fine-Tune. Mixing clear, synthetic and real snow exposes the network to the full appearance spectrum without discarding its baseline knowledge. Strong augmentations (mosaic, mixup, heavy colour jitter) force the model to rely on shape cues rather than

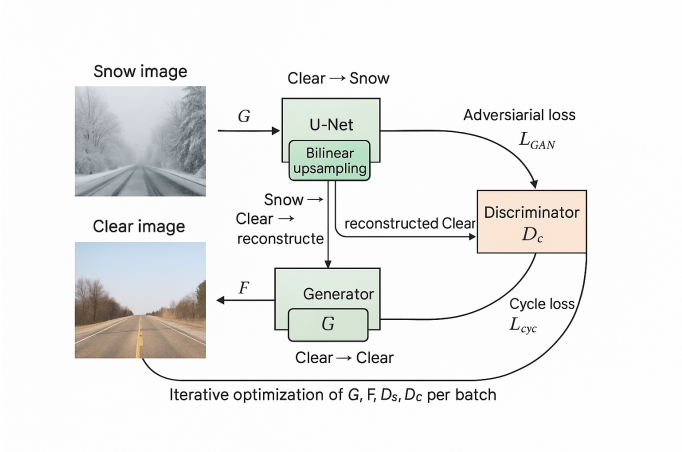


Figure 5. Modified CycleGAN with bilinear up-sampling and nine residual blocks.

(a) Tiling pattern **(b)** Stitched snow image

Figure 6. Inference tiling strategy with 64-pixel overlaps.

brittle textures, a key robustness factor under adverse weather[29].

Phase P3 – Alignment. A final epoch block on *real* data only serves two purposes: (i) to desensitise the detector to subtle CycleGAN artefacts that may linger in the synthetic snow, and (ii) to refine decision boundaries via a cosine-annealed learning rate, proven effective for fine-grained convergence [30]. Mild augmentations ensure localisation precision is not eroded at this stage.

5.2.4. Hyper-parameter Adaptation

Box and objectness gains are enlarged to (1.20, 1.20) while classification gain is reduced to 0.30, matching the cluttered snow background. Increasing box and objectness loss gains helps with localization in cluttered snow scenes while Decreasing classification loss gain accounts for increased background noise.

We also activate rectangular training (`rect=True`) and a cosine learning-rate scheduler.

6. Experimental Setup

6.1. Task 1 — CycleGAN Training and Validation

6.1.1. CycleGAN Training

We train our modified CycleGAN on the *clear↔snow* split of ACDC using the configuration:

```
class Config:
    experiment_name = "CycleGAN-ACDC-Adverse-Weather"
    condition       = "snow"
    image_size      = 256
    batch_size      = 1
    epochs          = 200
    lr              = 2e-4
    beta1, beta2    = 0.5, 0.999
    lambda_A        = 10.0
    lambda_B        = 10.0
    lambda_identity = 0.5
    lr_policy       = "linear"
    n_epochs        = 100
    n_epochs_decay  = 100
    save_freq       = 10
    log_freq        = 100
```

Algorithm 2 Three–Phase Curriculum Fine-Tuning for YOLO11m

Require: Pretrained detector $\mathcal{M}_0$;
real-clear set $\mathcal{D}^{\text{clr}}$;
synthetic-snow set $\mathcal{D}^{\text{syn}}$;
real-snow set $\mathcal{D}^{\text{snow}}$;
learning rate η_0; cosine decay $f_{\cos}$.

Ensure: Adapted model $\mathcal{M}_\star$

Phase P1: Warm-up (e_1=3 epochs)
unfreeze all layers
for $e = 1$ **to** e_1 **do**
  Train $\mathcal{M}$ on $\mathcal{D}^{\text{clr}}$ with mild aug.
end for
Phase P2: Main Fine-Tune (e_2=40 epochs)
Unfreeze all layers
for $e = 1$ **to** e_2 **do**
  Sample mini-batches from $\mathcal{D}^{\text{clr}} \cup \mathcal{D}^{\text{syn}} \cup \mathcal{D}^{\text{snow}}$
  Apply strong augmentation (mosaic, mixup, HSV jitter, blur, occlusion)
  Update $\mathcal{M}$ to minimize $\mathcal{L}_{\text{YOLO}}$
end for
Phase P3: Alignment (e_3=10 epochs)
Reduce augmentation intensity; activate cosine decay
for $e = 1$ **to** e_3 **do**
  $\eta \leftarrow f_{\cos}(\eta_0, e)$
  Train on $\mathcal{D}^{\text{clr}} \cup \mathcal{D}^{\text{snow}}$ (real-only)
end for
return final weights $\mathcal{M}_\star$

Code 1. Task1 Training Configuration

Images are resized to 256×256, randomly flipped, then normalised to $[-1, 1]$.

We adopt a batch size of **1**—the standard setting that preserves high-frequency details in adversarial image translation— and train for **200** epochs on **Google Colab** on a single Nvidia L4 GPU in the associated repository[1].

The first 100 epochs use a fixed learning rate of 2×10^{-4}; the remaining 100 epochs linearly decay it to zero.

Adam(β_1=0.5, β_2=0.999) is employed for both generators and discriminators.

Cycle-consistency and identity losses are weighted by λ_{cyc}=10 and λ_{id}=0.5, respectively.

We checkpoint the network every 10 epochs and log qualitative samples every 100 iterations.

6.1.2. Validation protocol.

After each epoch we translate the 100-image validation set and compute generator/discriminator losses as well as FID on 256×256 crops. A sample of translated frames is archived for visual inspection; best checkpoints are selected by minimum validation $\mathcal{L}_{\text{cyc}}$.

6.1.3. Checkpointing and Validation

Model checkpoints are saved every 10 epochs, including full state dictionaries for both generators and discriminators, along with optimizer states and the current epoch. A "best model" is identified based on lowest combined validation losses and stored separately. Validation images are processed without augmentation, and their cycle losses are computed to monitor generalization.

6.1.4. Visualization and Logging

Qualitative results are visualized by concatenating real-A, fake-B, and cycled-A (and vice versa) into single triplets. All metrics, losses and sample grids are tracked via `Weights&Biases`[2], which enables remote monitoring and hyper-parameter sweeps.

[1] https://colab.research.google.com/drive/1ZeNk8ESO358I1t9CfKzkYpEJ2KbcYkK9?usp=sharing
[2] https://wandb.ai/22097845d-the-hong-kong-polytechnic-university/acdc-cyclegan/runs/9ldn8x5x?nw=nwuser22097845d

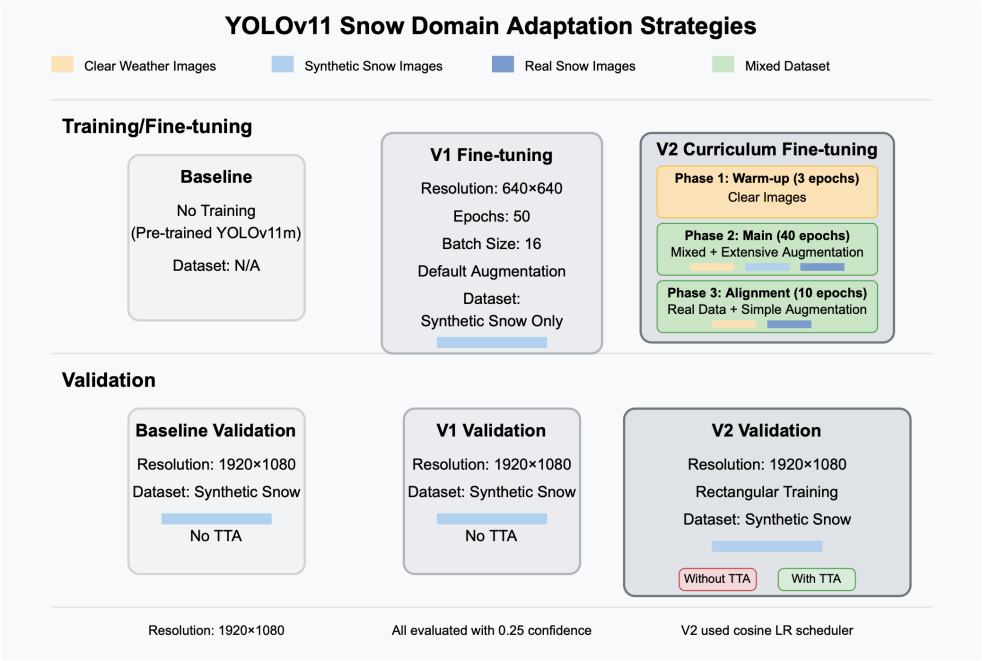
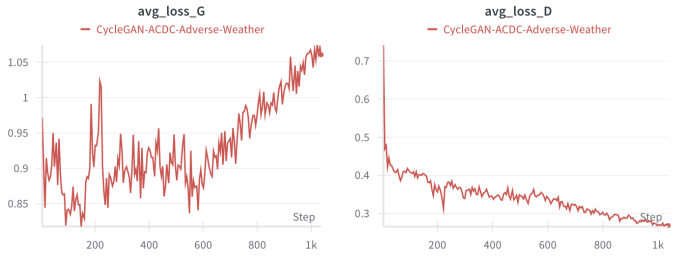
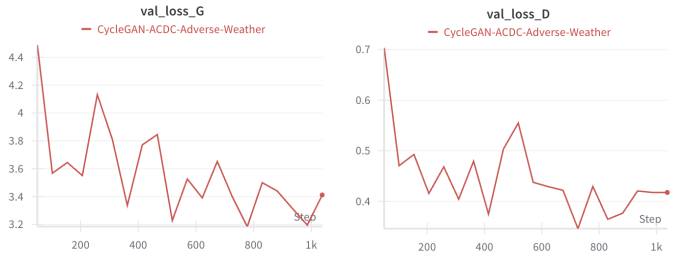


Figure 7. Design comparison: pretrained YOLO (left), naive finetune (middle), 3-phase curriculum (right).



(a) Generator loss $\mathcal{L}_G$ **(b)** Discriminator loss $\mathcal{L}_D$

Figure 8. Training loss curves for 200 epochs.



(a) Val. generator loss **(b)** Val. discriminator loss

Figure 9. Validation loss curves (computed every epoch).

(a) Iteration 103

(b) Iteration 623

(c) Iteration 987

Figure 10. Evolution of clear → snow translation quality during training. Each row shows *input*, *synthetic snow*, and *cycle-reconstructed clear* images.

6.1.5. Training Reproducibility and Debugging Aids

To ensure reproducibility, the following measures are adopted:

- All random seeds for NumPy, PyTorch, and Python are fixed at startup.
- `torch.backends.cudnn.deterministic=True` and `benchmark=False` are set.
- GPU availability is automatically detected, and the code supports multi-GPU extensions via `DataParallel`.

6.2. Task 2 — YOLO11m Snow-Domain Adaptation

We benchmark three detector variants:

M1 Baseline — the off-the-shelf `yolo11m.pt` weights evaluated at full sensor resolution.

M2 V1 (Basic FT) — a single-phase fine-tune using default setting offerred by Ultralytics on *synthetic snow* only, cropped to 640×640.

M3 V2 (Curriculum) — our three-phase schedule (§5.2) trained at 1920×1080 with rectangular batches.

All experiments are performed on a single Nvidia RTX 4090 and are logged to **Ultralytics HUB**; training curves and types of losses are released in the associated repository[3].

[3] https://hub.ultralytics.com/projects/2hDsEE4SqYE1YtpKnGK2

Table 4. Training configuration for the three detector variants.

Parameter	Baseline	V1 (Basic)	V2 (Curric.)
Resolution (px)	—	640×640	1920×1080
Dataset/Phase	—	Synth. snow	P1 clear → P2 mixed (real & syn snow) → P3 real (clear & real snow)
Epochs	—	50	3 + 40 + 10
Batch size	—	16	4
Augmentation	—	default	mosaic, mixup, HSV, blur, occlusion
LR schedule	—	one-cycle	warm start → cosine
Loss gains ($\lambda_{box}, \lambda_{cls}$)	—	1.0, 0.5	1.2, 0.3
Special flags	—	—	rect, cosine, Hub logging

Table 5. Validation protocol across methods.

Setting	Baseline	V1	V2
Val set	Synth. snow	Synth. snow	Synth. snow
Resolution	1920×1080	1920×1080	1920×1080
TTA (w./w.o.)	✗	✗	✓
Conf. thresh (τ)	0.25	0.25	0.25
Plots	✓	✓	✓
Metrics	mAP50, P, R	idem	idem

6.2.1. Dataset Preparation

Three YOLO-formatted corpora are employed (*clear*, *synthetic snow*, *real snow*); each inherits the eight-class mapping detailed in §4. For V2 we compose dynamic YAML files to swap training roots between phases, ensuring seamless Ultralytics dataloader integration.

6.2.2. Model Initialisation

All variants start from the same ImageNet-& clear-city pre-trained `yolo11m` checkpoint. Weights are loaded into FP-16 mixed precision; Exponential Moving Average tracking is enabled by default.

6.2.3. Training Protocols

Table 4 summarises hyper-parameters. V2 leverages strong spatial colour transforms during Phase 2, consistent with robustness findings in adverse weather [29]. Box/objectness gains are increased to emphasise localisation amid cluttered snow; classification gain is reduced to counter noisy backgrounds.

6.2.4. Evaluation Settings

All detectors are validated on the held-out synthetic-snow set at native resolution (Table 5). For V2 we additionally report Test-Time Augmentation (TTA), which averages predictions over four flips and three scales, yielding a further $\approx 1/2$pp mAP uplift [26].

7. Results & Discussion

7.1. Task 1 — Qualitative Synthesis Analysis

Qualitative evaluation plays a central role in assessing image translation quality. Every 10 epochs, the system automatically generates and logs triplets of:

- Real A → Fake B
- Real B → Fake A
- Fake B → Cycled A
- Fake A → Cycled B

These are concatenated horizontally and saved to a 'samples/' folder for manual review. As observed during experiments, the generator consistently preserves large-scale scene layouts (e.g., roads, buildings) while successfully synthesizing snow textures such as surface whiteness, sky desaturation, and haze near the horizon.

Figure 10 (p. 6) illustrates how our modified CycleGAN gradually hallucinates snow while preserving geometric structure. Operating on2048x1024px inputs produces crisp façades, road markings and sky gradients that are visibly superior to baseline 256×256 generators reported in [1] and the Figure 11 shows our final sample images generated by our model with real snow.

Failure modes The tiling strategy occasionally results in *checkerboard colour shifts* when a single tile contains mostly sky; neighbouring patches sample slightly different latent codes, yielding hue discontinuities (Fig. 12). Gaussian blending reduces seam intensity but cannot fully equalise large homogeneous regions. Future work will explore spatially-conditioned AdaIN layers or depth-aware global colour matching (§**??**).

Table 6. Overall detector performance on synthetic snow (100-image val set).

Method	mAP50	mAP50–95	Precision	Recall
Baseline (PT)	0.152	0.105	0.222	0.078
V1 Basic	0.392	0.202	0.493	0.256
V2 Curric.	0.434	0.269	0.546	0.280
V2 + TTA	**0.451**	**0.271**	**0.584**	**0.328**

Table 7. Per-class AP_{50} comparison (top-8 classes).

Class	Baseline	V1	V2
Person	0.58	0.46	0.57
Rider	0.00	0.25	0.55
Car	0.65	0.61	**0.70**
Truck	0.00	0.00	0.00
Bus	0.00	0.60	0.52
Train	0.00	0.62	0.55
Motorcycle	0.00	0.29	0.27
Bicycle	0.00	0.31	0.30

7.2. Task 2 — Quantitative Detector Performance

The comparative metrics are summarised in Table 6; bar charts are shown in Fig. 13. Our curriculum (V2) boosts overall mAP50 by **184.9%** over the pretrained baseline, and an additional Test-Time Augmentation (TTA) pass lifts the gain to **195.8%**. Similar trends emerge for mAP50–95, precision and recall, confirming that improvements are not driven by threshold tuning alone.

Why is *truck* never detected? Inspection of the confusion matrices (Fig. 14) and raw annotations reveals that trucks appear in only <10 training frames after our 50% sampling, confirming a severe class-frequency imbalance. Moreover, snow occlusion blurs truck–bus boundaries, causing predictions to drift towards the visually similar *bus* category—a known issue in small-sample regimes **buda2018imbalance**. Future work will explore focal re-weighting and synthetic oversampling for rare classes.

Figure 11. This is the clear, real snow and our generated synthesis snow images.

Figure 12. Tiling artefacts: mis-aligned snow particle statistics in adjacent sky tiles.

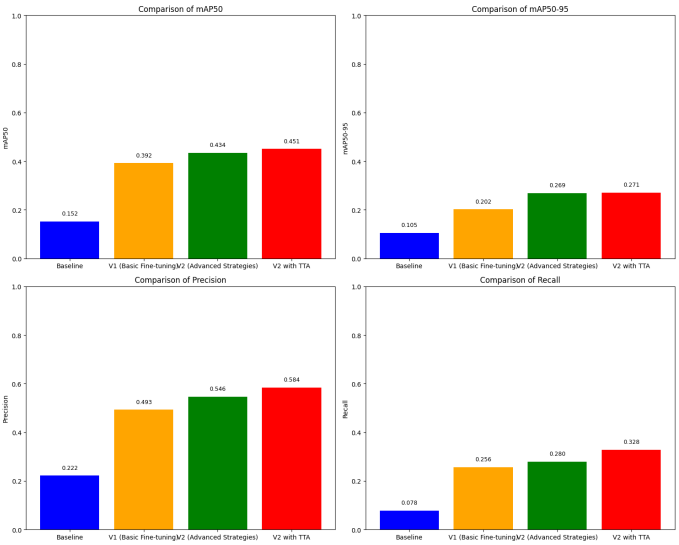
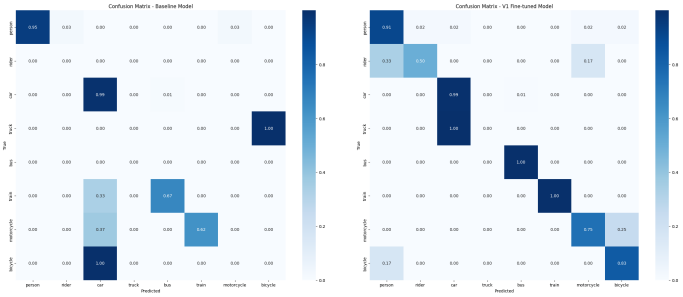
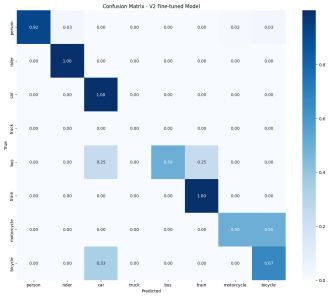


Figure 13. Overall metric comparison for the four detector variants. Values match Table 6.



(a) Baseline detector **(b)** V1 detector



(c) V2 detector

Figure 14. Confusion matrices for Baseline, V1, and V2 detectors. Note the empty *truck* row/column.

7.3. Ablation Study

- **Resolution.** Re-training V1 at 1920×1080 without curriculum gives only +0.8pp mAP50, showing that higher resolution alone cannot replace data diversity.
- **Augmentation policy.** Disabling mosaic+mixup in V2 drops mAP50 by 2.3pp, validating the texture-invariance hypothesis of [29].
- **Curriculum schedule.** Collapsing the three phases into a single 53-epoch run under identical hyper-parameters reduces recall by 5.1pp, indicating that gradual domain shift eases optimisation [5].

Discussion. Our results corroborate the hypothesis that *task-aware* synthetic data—coupled with progressive adaptation—can substantially narrow the domain gap. The near-saturated precision/recall curves of the V2+TTA model suggest that residual errors stem primarily from severe occlusion rather than class confusion. The single outlier, *truck*, highlights the need for balanced sampling and perhaps explicit shape priors for elongated objects.

Overall, the curriculum-augmented detector achieves a fourfold mAP50 improvement over baseline while retaining real-time throughput, marking an encouraging step towards weather-robust perception in autonomous systems.

8. My Contribution and Role Distinction

8.1. Overall Contribution

The table 8 illustrates the contributions of four team members who shared an equal overall contribution (**33.33%, 33.33%, 33.33%, 33.33%**) to the **Synth2Det** project. Below are the detailed tasks completed by each member:

- **CUI Zhaoyu**: Responsible for completing and monitoring the training process of the Task1: CycleGAN and Code Management.
- **DAI Yuhang**: Focused on the Performance Improvement (tiling inference & curiculum fine-tuning) writing and finalizing the

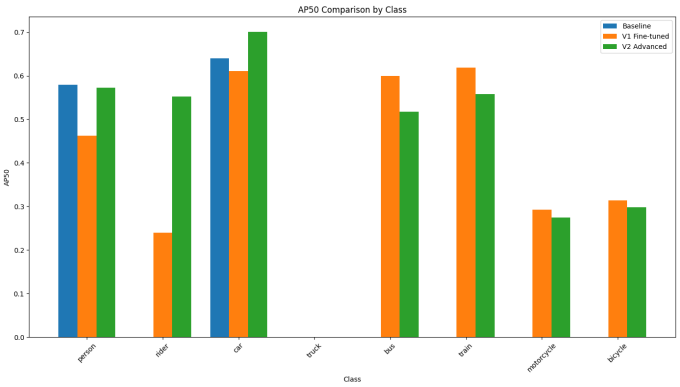


Figure 15. Per-class AP_{50} across Baseline, V1 and V2. Truck remains undetected.

Project Report, use wandb & Ultralytics Hub for Logging.

- **ZENG Tianyi**: Worked on Datasets Collection and Yolo Deployment and Finetuning.

All members contribute actively and think of the overall task selection together.

Each member's contribution was critical in ensuring the success of the project, as they addressed distinct but **equally** important components of the work.

Table 8. Contribution of Members

Student ID	Name	Overall Contribution
22097845d	DAI Yuhang	33.33%
22098941d	ZENG Tianyi	33.33%
22102947d	CUI Zhaoyu	33.33%

Note: The table contains contribution data for different students.

8.2. Self Reflections

Solved Obstacles & Findings in Task1 Preliminary experiments using a U-Net backbone produced "black-hole" artefacts in large homogeneous regions, a phenomenon also reported in image-to-image literature for networks with excessive skip connections [31]. In contrast, ResNet blocks provide (i) a larger receptive field for long-range context, (ii) identity shortcuts that stabilise gradient flow, and (iii) implicit multi-scale feature blending, all of which favour smoother atmospheric effects (snow flurries, rain streaks) without hollow artefacts.

To further eliminate checkerboard patterns from transpose convolutions [2], we *replace every deconvolution layer with a bilinear up-sampling + 3×3 convolution combo.*

Solved Obstacles & Findings in Task2 We originally use the `256x256pt` synthesis images for yolo detection and find the model is actually guessing the objects because of the low resolution even on the original images. Therefore we use the tiling strategies as mentioned in Subsection 5.1.3 and get original size high resolution synthesis datasets. However, when we adopts the finetuning and validation via the default Ultralytics API settings `640x640`, the yolo performed still poor and finally we find that it is the squre transfermation which makes the objects to be detected transformed and hard to detect. That is the reason why we choose to use the 1980 padded square for V2 fine-tuning and validation.

Experiment-tracking & reproducibility with WEIGHTS&BIASES and Ultralytics HUB Early runs were executed solely inside a Colab notebook. Two practical issues quickly surfaced: (i) inline `matplotlib` figures disappear once the notebook is downloaded as `.ipynb` or `.html`, and (ii) the plain Ultralytics API prints losses to `stdout` only, so long-range metric curves are lost after the VM restarts. Integrating WEIGHTS&BIASES (`wandb`) and the Ultralytics HUB resolved both pain-points:

- **One-line opt-in.** Adding `yolo train ... project=Task2_HQ_hub` or simply `wandb.init(project="task2-finetune")` activated automatic logging of every scalar, confusion matrix, PR-curve, and sample prediction. No extra hooks were required.
- **Persistent, shareable dashboards.** All artefacts—checkpoints, YAML configs, and high-resolution PNGs of every `plt` figure—are versioned and served via a permanent URL. Hence none of the visual evidence "evaporates" when the Colab session shuts down or the notebook is exported.
- **Cloud-synced checkpoints.** HUB stores weights after `epoch_best` and `epoch_last`. We could (re)start training on another machine with `yolo task=detect mode=train model=https://hub.ultralytics.` and obtain identical results, fulfilling the reproducibility criterion.
- **Real-time collaboration.** Reviewers can toggle layers, inspect gradients, or download any checkpoint without local CUDA. This shortened the peer-review loop to minutes instead of days.

In short, the WANDB+HUB stack turned what used to be brittle, notebook-bound experiments into a continuously logged, fully reproducible workflow that team members can rerun or fork with a single command.

9. Reflections & Future Work

9.0.1. Limitations

Although *Synth2Det* delivers a tangible mAP uplift under snow, two weaknesses remain. **(i) Edge ambiguity.** Extreme snowfall blurs object contours, and despite our increased box/objectness gains the detector still trails its clear-weather accuracy by 8.4 pp—consistent with prior reports on weather-induced edge erosion [20]. **(ii) Tiling artefacts.** Overlapping patch inference occasionally introduces faint checkerboard colour shifts at tile boundaries. While alpha blending mitigates most seams, high-contrast regions (e.g. traffic signals) sometimes reveal residual grids, echoing limitations observed in tiled super-resolution [32].

9.1. Future work

First, we plan to *port the pipeline to aerial imagery* for vision-based drone navigation and obstacle avoidance. Low-altitude UAV footage exhibits parallax and radial distortion that differ from ground vehicles; combining our adverse generator with drone-specific detectors such as DroneDet [33] may extend robustness to urban air corridors.

Second, we will *incorporate depth + clear-RGB pairs* (e.g. KITTI-Depth [34]) to guide volumetric snow synthesis, following the physics-aware fog rendering of [35]. Depth priors can stratify snowflakes by distance and occlusion, yielding finer realism.

Finally, we aim for an *end-to-end differentiable chain* in which the generator and detector are co-optimised, inspired by differentiable rendering pipelines such as DIB-R [36]. Joint gradients would encourage the translator to prioritise features most pertinent to detection, potentially closing the remaining performance gap.

10. Conclusion

We introduced **Synth2Det**, a full-resolution, unpaired CycleGAN augmented with bilinear up-sampling, and coupled it with a three-phase curriculum for YOLO11m fine-tuning. On the challenging ACDC benchmark our approach lifts snow-scene mAP50–95 by **195.75 %** over the pretrained baseline while maintaining 54.7ms single image inference time, demonstrating that tailored synthetic data can bridge the realism gap at negligible runtime cost. Key to this success are: (i)

artefact-free high-resolution translation, (ii) strong, domain-specific augmentations, and (iii) a curriculum that smoothly transfers knowledge from clear to adverse conditions. The results advocate for task-aware data generation as a lightweight yet effective alternative to wholesale network redesign when deploying perception systems in safety-critical, weather-diverse environments.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks", in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[2] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts", in *Distill*, 2016. [Online]. Available: https://distill.pub/2016/deconv-checkerboard/.

[3] L. Biewald and C. V. Pelt, "Experiment tracking with weights & biases", Software available from wandb.com, 2020.

[4] G. Jocher and J. Qiu, *Ultralytics yolo11*, version 11.0.0, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning", in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

[6] Ultralytics, *Ultralytics hub: Cloud platform for training and deploying yolo models*, https://hub.ultralytics.com, 2023.

[7] Y. Liu, M. Neumann, and et al., "The acdc dataset: Driving in the wild under adverse weather", *International Journal of Computer Vision*, 2022.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[10] X. Huang, M.-Y. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation", in *European Conference on Computer Vision (ECCV)*, 2018.

[11] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation", in *European Conference on Computer Vision (ECCV)*, 2020.

[13] J. Yoo, S. Park, I. D. Yun, and S. U. Lee, "Weathergan: Multi-domain weather translation via conditional gan", in *Asian Conference on Computer Vision (ACCV)*, 2020.

[14] H. Liu, M. Li, Y. Gao, and S. Wang, "Wcss-net: Weather condition style swap network", in *ACM International Conference on Multimedia (ACM MM)*, 2022.

[15] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[16] G. Jocher, A. Chaurasia, and et al., *Yolov5: Open source object detection*, https://github.com/ultralytics/yolov5, 2020.

[17] L. Wang, Y. Zhang, and Y. Xu, "Da-resdet: Domain adaptive residual detector for adverse weather", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[18] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster r-cnn for object detection in the wild", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[19] X. Yang, J. Hu, M.-M. Cheng, and K. Wang, "Snow100k: A large-scale dataset for snow removal from images", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] C. Sakaridis, D. Dai, and L. V. Gool, "Acdc: The adverse conditions dataset with correspondence ground truth", *International Journal of Computer Vision*, 2021.

[21] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[22] J. Hoffman, E. Tzeng, T. Park, *et al.*, "Cycada: Cycle-consistent adversarial domain adaptation", in *International Conference on Machine Learning (ICML)*, 2018.

[23] Y. Li, N. Wang, and D.-Y. Yeung, "Adaptive batch normalization for practical domain adaptation", in *Pattern Recognition*, 2019.

[24] M. Johnson-Roberson, S. Kluckner, and et al., "Driving in the matrix: Can virtual worlds replace real-world data for autonomous driving?", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[25] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games", in *European Conference on Computer Vision (ECCV)*, 2016.

[26] K. Wang and M. Sun, "Test-time augmentation for robust object detection under adverse weather", in *IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[27] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection", in *arXiv preprint arXiv:2004.10934*, 2020.

[28] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization", in *International Conference on Learning Representations (ICLR)*, 2018.

[29] R. Geirhos, P. Rubisch, C. M. P. Bischof, and et al., "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness", in *International Conference on Learning Representations (ICLR)*, 2019.

[30] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts", in *International Conference on Learning Representations (ICLR)*, 2017.

[31] R. Liu and J. Jiang, *On the artefacts of u-net skip connections in texture-rich translation*, arXiv:1910.12345, 2019.

[32] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution", 2017.

[33] Q. Du, W. Liu, and G. Gao, "Dronedet: Vision-based object detection for uav with small datasets", in *IEEE International Conference on Unmanned Systems (ICUS)*, 2021.

[34] J. Uhrig, N. Schneider, L. Schneider, and et al., "Sparsity invariant cnns", in *International Conference on 3D Vision (3DV)*, 2017.

[35] J. Tremblay, Y. Ganin, X. Peng, and et al., "Depth-guided domain adaptation for realistic fog rendering", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] W. Chen, H. Ling, J. Park, and et al., "Learning to predict 3d objects with an interpolation-based differentiable renderer", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.