Design and Implementation of an Unreal Engine 5 Plugin for Generating Photorealistic and Semantically Controllable Synthetic Data to Evaluate the Robustness of ImageNet Classifier

Bachelor Thesis

Bachelor of Science in Applied Computer Science

Florian Gutbier

June 3, 2025

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: Sebastian Dörrich

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

Modern deep vision models often struggle when confronted with images that deviate from their training distribution. To diagnose and quantify these out-of-distribution (OOD) failure modes, we present a fully controlled synthetic dataset generated using a custom open-source Unreal Engine 5 (UE5) plugin. Our plugin enables researchers to specify and vary six generative factors—background, material, light colour, camera pose, volumetric fog, and mesh identity—across seven ImageNet-compatible object classes. By rendering the full Cartesian product of these factors, we produce 86 016 photorealistic images with precise metadata. This exhaustive control reveals that state-of-the-art architectures (e.g., ConvNeXt-Large, ViT-L-16, Swin-B) suffer steep accuracy drops under combined shifts, with even the best model achieving only 37.5% Top-1 on the OOD benchmark. Factor-wise analysis shows that material changes alone can induce a 24 percentage-point swing in accuracy, viewpoint shifts can reduce performance by over 30 points, and complex backgrounds lead to a 10 point spread. Interestingly, adding volumetric fog consistently improves recognition by presumably simplifying background clutter. By open sourcing our UE5 plugin and OOD dataset, we provide a reproducible pipeline for synthetic OOD evaluation. Our results demonstrate that full factor control and exact annotations are critical for pinpointing model weaknesses, guiding future research toward more robust vision systems under extreme distribution shifts.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
UE5	Unreal Engine 5
AP	Average Precision
mAP	mean Average Precision
ViT	Vision Transformers
CNN	Convolutional Networks
DCC	Digital content-creation
WYSIWYG	What you see is what you get
fps	frame(s) per second
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

1 Introduction

Deep neural networks have achieved remarkable progress in visual recognition tasks largely because of the availability of large-scale, real–image datasets such as ImageNet[1] and the CIFAR family of benchmarks. (Recht et al., 2019) However, the photographic nature and uncontrolled acquisition process of these corpora impose two constraints. First, once data have been collected, the distribution of nuisance factors—camera viewpoint, illumination, object pose, background, *etc.*—is fixed and cannot be systematically manipulated. Second, the same uncontrolled factors make it difficult to diagnose failure modes when models are exposed to inputs that diverge from the in-distribution training data, a scenario that frequently arises in safety–critical deployments. (Geirhos et al., 2020)

The brittleness of current vision models to distribution shifts has been documented by substantial drops in top-1 accuracy when models trained on ImageNet are evaluated on its independently curated counterpart, ImageNet-V2, or on natural distribution shifts such as ImageNet-R and ImageNet-A. (Taori et al., 2020; Recht et al., 2019) These observations motivate the need for data with controllable generative factors, so that specific shifts can be isolated and their effect on recognition performance measured precisely.

Synthetic rendering pipelines offer this controllability.By leveraging state-of-the-art game engines, researchers can generate densely labelled images while systematically varying scene parameters that are expensive or impossible to control in natural photographs. Early work such as UnrealGT (Pollok et al., 2019) and UnrealROX+ (Gonzalez et al., 2021) demonstrated that photorealistic images with pixel-perfect ground truth can be produced at scale, spurring numerous studies on domain adaptation and robust perception.

A recent milestone in this trajectory is the **P**hotorealistic **U**nreal **G**raphics (PUG) family of datasets released by Meta AI. (Bordes et al., 2023) PUG provides more than 215 k high-resolution images of 70 animal assets, together with additional variants covering 151 ImageNet classes, all rendered with exhaustive control over background, texture, scale, camera, and lighting. By decoupling these generative factors, PUG enables systematic stress-testing of representation learning algorithms and has already become a reference benchmark for out-of-distribution (OOD) evaluation.

Despite these advances, existing synthetic datasets trade off breadth and depth: PUG:ImageNet spans 151 classes but varies only one factor at a time, whereas PUG:Animals exhaustively combines factors for just 70 animal assets. Furthermore, most generation pipelines are tightly coupled to the factors chosen by their designers, hindering reuse in bespoke experiments. In the case of PUG, only a subset of the generation code has been released, and the public modules need substantial

[1]Although ImageNet contains more than 14 million images, its canonical ILSVRC subset of 1,000 classes is the one most commonly used for training modern models.

fixing after an update to a external plugin they rely on, limiting their immediate applicability.

1.1 Contributions of this thesis

We publish a modular Unreal Engine 5 plugin that enables researchers to generate photorealistic image datasets with full control over backgrounds, materials, lighting, camera poses, and atmospheric effects. The plugin is self-contained—no external tooling or proprietary infrastructure is required beyond a standard UE5 installation.

Using the plugin, we construct a synthetic dataset that follows the design philosophy of PUG but targets everyday ImageNet-style objects. All rendering scripts and factor annotations are released to facilitate extension and replication.

We benchmark state-of-the-art convolutional networks and Vision Transformers on the new dataset, compare their OOD robustness to results reported on PUG, and discuss failure modes that persist across both benchmarks.

All code, asset lists and the dataset will be made publicly available under an open-source licence to foster transparent and reproducible research [2].

[2] `https://github.com/FlGutbier/UE5DatasetGenerator`

2 Related Work

2.1 Photorealistic Synthetic Data for Vision

Synthetic imagery has long been leveraged to overcome the cost, bias, and copyright constraints of Internet–scraped photographs. Early works relied on domain randomization to bridge the reality gap for robotic perception, (Tobin et al., 2017) or on diagnostic scenes such as CLEVR,(Johnson et al., 2017) to probe compositional reasoning. More photorealistic pipelines subsequently emerged around game engines, e.,g. SYNTHIA,(Ros et al., 2016), CARLA,(Dosovitskiy et al., 2017), UnrealGT,(Pollok et al., 2019), UnrealROX$^{(+)}$, (Martinez-Gonzalez et al., 2020, 2021) and Falling Things,(Tremblay et al., 2018). While these datasets offer fine-grained control, their scope is often task-specific and they rarely model distribution shifts systematically. Parallel research has therefore explored robustness through new test sets for natural images, such as ImageNet-V2,(Recht et al., 2019) and ImageNet-R,(Geirhos et al., 2020), or through synthetic perturbations of backgrounds,(Malik et al., 2021). Yet, controlling high-fidelity variables (pose, lighting, texture) at ImageNet scale remained out of reach.

2.2 The PUG Framework

Bordes *et al.* introduce **PUG: Photorealistic Unreal Graphics**,(Bordes et al., 2023), a family of datasets rendered with UE5 and orchestrated through a custom Python API called Torch-Multiverse. Unfortunately the provided python code is, as of the time of writing, incomplete and not functional, while the logic implementation on the UE5 side is not provideed at all. Each PUG environment provides declarative control over *object assets*, *backgrounds*, *camera and object orientation*, *scale*, *texture* and *illumination*. Photorealism is inherited directly from production-grade game assets, while controllability enables factorial experiments impossible with scraped photographs. Four public subsets are released:

- **PUG-Animals**: 215,040 images, 70 animal meshes, 64 backgrounds, 3× sizes, 4× textures, 4× cameras; supports systematic out-of-distribution (OOD) evaluation. Provides a combinations of all available variables.
- **PUG-ImageNet**: a robustness benchmark for ImageNet2.3.
- **PUG-SPAR**: 43,560 images for vision–language reasoning about scenes, positions, attributes and relations.
- **PUG-AR4T**: 249,986 captioned images for fine-tuning vision–language models.

Rendered frames are delivered at 512×512 px resolution at roughly one frame per second on a single V100 GPU, and the full datasets (Animals 78 GB, ImageNet 27 GB, SPAR 16 GB, AR4T 97 GB) are available from the project repository.[3]

2.3 PUG-ImageNet

PUG–ImageNet provides 88,328 photorealistic images that map one-to-one to 151 ImageNet classes. Images are generated from 724 object assets and 64 environments while varying *one* factor at a time to isolate its effect on recognition performance:

Factor	**Variations**
Camera yaw/pitch/roll	18
Object yaw/pitch/roll	18
Object size	7
Object texture	9
Light intensity	7

Table 1: Factorised variations in PUG-ImageNet (Bordes et al., 2023)

Because all other attributes are held constant, error attribution is unambiguous—a key limitation of natural robustness suites such as ObjectNet,(Barbu et al., 2019). Bordes *et al.* show that state-of-the-art ImageNet models rank differently under these controlled shifts: e.,g. although ViT-B/32 pretrained on ImageNet-21k outperforms Swin-B on the original validation set, Swin-B is markedly more resilient to pose, camera and lighting changes (˜10–15 pp top-1) on PUG-ImageNet. Such disparities corroborate earlier findings on distribution shift,(Taori et al., 2020) and demonstrate that ImageNet accuracy alone is a poor proxy for real-world robustness.

	PUG: ImageNet Top-1 Accuracy across Factors of Variation						
Model	**ImageNet Val.**	**Camera (Yaw,Pitch,Roll)**	**Pose (Yaw,Pitch,Roll)**	**Size**	**Texture**	**Light**	**Background**
ResNet50	81.5	(38.1, 33.1, 26.9)	(38.0, 23.6, 22.9)	35.7	27.0	13.6	29.5
ResNet101	82.3	(43.4, 35.9, 29.4)	(45.1, 26.7, 25.6)	39.7	31.1	14.1	32.8
ViTLarge	85.8	(52.2, 40.4, 37.1)	(52.4, 30.4, 28.4)	46.4	42.9	8.9	34.6
ViTBasePretrained21k	84.3	(37.5, 34.3, 31.7)	(38.0, 21.8, 20.5)	33.0	28.5	4.1	26.6
Swin	83.6	(56.0, 45.6, 41.8)	(56.9, 35.3, 34.2)	52.9	40.1	19.1	42.0
BiT (JFT300M)	80.3	(40.5, 32.3, 26.0)	(42.1, 23.6, 22.8)	37.3	23.4	6.3	20.5
DINOv2 (LVD-142M)	84.5	(45.6, 41.1, 37.4)	(47.5, 28.8, 28.5)	43.1	35.0	6.1	30.9
Flava (PMD 70M)	75.5	(31.7, 23.4, 17.6)	(30.8, 17.6, 15.4)	30.5	24.2	7.8	21.9
CLIPViTB32 (400M)	62.9	(41.7, 30.2, 22.1)	(41.6, 23.8, 20.9)	40.1	34.4	5.7	24.4
CLIPViTB32 (2B)	66.6	(44.0, 31.5, 24.1)	(43.8, 24.8, 21.8)	42.2	34.7	3.3	26.0
CLIPViTL14 (400M)	72.8	(52.3, 39.8, 35.7)	(51.8, 29.0, 26.4)	50.6	41.1	4.3	33.0

Table 2: Top-1 accuracies for the ImageNet validation set and the different parameters of the PUG-ImageNet set. Taken from (Bordes et al., 2023)

2.4 Positioning within Synthetic-Data Literature

PUG provides two key capabilities that were not previously available. First, it offers systematic, factorized variation at ImageNet scale, enabling controlled experiments

[3] `https://github.com/facebookresearch/PUG`

on pose, illumination, texture, and background that help identify shortcut features (Geirhos et al., 2020) and quantify synthetic-to-real transfer gaps (Volpi et al., 2018). Second, it includes a documented asset pipeline that clarifies licensing and provenance: all 3D models are licensed through the Epic Games Marketplace and each rendered frame comes with metadata that records the exact parameters used.

By contrast, diffusion-based synthetic datasets (Sariyildiz et al., 2023) can inadvertently reproduce copyrighted content or suffer from prompt misalignment, since they rely on generative models trained on large image collections. PUG avoids these issues by producing each image directly from known assets and exporting the full scene description alongside the image. This explicit control provides unambiguous ground truth, making PUG particularly well suited for robustness studies such as those presented in Section 2.3.

In summary, PUG complements earlier photorealistic datasets that focused on specific tasks by offering a scalable, reproducible framework for representation learning and out-of-distribution evaluation. It therefore occupies an intermediate position between task-specific simulators and unrestricted text-to-image pipelines, establishing a benchmark for future research on model reliability and generalization.

2.5 Implications for This Thesis

The code published by Bordes et al. (2023), while available, is incomplete and not directly usable. To effectively probe model robustness under simultaneous distribution shifts and pinpoint specific weaknesses, we have developed an open-source UE5 plug-in. This plugin replicates the objectives of PUG:ImageNet while enabling more detailed metadata annotation, as Bordes et al. (2023) state themselves in their paper, that their current datasets offer only limited metadata annotation. In contrast to the original PUG pipeline, which relies on external Python wrappers and provides only partial source files, our plugin operates entirely within the game engine.

3 Theoretical Foundation

3.1 Image Classification Models

Image classification, a fundamental task in computer vision, involves assigning a label from a predefined set to an input image. Convolutional Neural Networks (CNNs) have historically dominated this field due to their ability to proficiently capture hierarchical spatial features through convolutional and pooling layers (LeCun et al., 2015; Goodfellow et al., 2016). However, contemporary developments have seen the rise of Transformer-based architectures, originally developed for natural language processing, demonstrate remarkable performance in various vision tasks, including image classification.

Convolutional Neural Networks (CNNs) Long before the term deep learning was coined, Fukushima (1980) introduced the Neocognitron, the first network to leverage two ideas that still define today's CNNs: local connectivity—each neuron processes only a small spatial neighbourhood—and weight sharing—the same kernel is applied at every location, endowing the model with translation–equivariance while greatly reducing the parameter count. A decade later, the back-propagation algorithm and increasing computational power enabled LeCun et al. (1998) to train the now-iconic LeNet-5 on the MNIST digit dataset, demonstrating that CNNs could outperform hand-crafted pipelines.

Progress stalled until three drivers converged in the early 2010s: graphics-processing units (GPUs), massive annotated datasets, and algorithmic refinements such as ReLU activations and dropout (Srivastava et al., 2014). The pivotal moment came with AlexNet in the 2012 ImageNet Challenge (Krizhevsky et al., 2012), which halved the top-5 error and triggered an intense architectural race. Successive designs deepened networks (*VGG-16/19*, up to 19 layers (Simonyan and Zisserman, 2015)), widened them with multi-scale "Inception" blocks (Szegedy et al., 2015), and finally stabilised ultra-deep training via residual connections (ResNet-152 (He et al., 2016)). More recently, automated architecture search and compound scaling led to the EfficientNet family, matching state-of-the-art accuracy with dramatically fewer parameters and FLOPs (Tan and Le, 2019).

Representative use cases CNNs have become the de facto backbone for visual perception across domains:

- **General object recognition and transfer learning** – the ImageNet pretraining paradigm underlies countless downstream tasks, from fine-grained bird classification to artwork retrieval.
- **Medical imaging** – surveys report CNNs surpassing traditional methods in tumour detection, organ segmentation and radiomics across modalities such as MRI and CT (Litjens et al., 2017).

- **Autonomous driving and robotics** – end-to-end perception stacks rely on CNNs for lane detection, obstacle recognition and depth estimation; large-scale simulators like CARLA facilitate data collection (Dosovitskiy et al., 2017).
- **Remote sensing** – CNNs advance land-cover mapping, disaster assessment and climate monitoring from high-resolution satellite imagery (Zhu et al., 2017).
- **Mobile and edge applications** – lightweight variants (e.g. MobileNet, EfficientNet-Lite) enable real-time inference for augmented-reality and biometric authentication on smartphones.

In summary, CNNs combine biologically inspired priors with scalable optimisation, forming the historical and conceptual bedrock on which most modern image-classification pipelines—including those evaluated in this thesis—are built. Detailed architectural choices for our experimental models are deferred to Section 4.

Vision Transformers (ViT) While convolutional networks dominated visual recognition for nearly a decade, the rapid success of the Transformer architecture in natural-language processing (Vaswani et al., 2017) prompted researchers to explore whether the same self-attention mechanism could replace convolutions altogether. Dosovitskiy et al. (2020) showed that, with sufficient pre-training data, a pure Transformer can rival state-of-the-art CNNs on ImageNet. Their ViT first partitions an image into non-overlapping $P \times P$ *patches*, flattens each patch to a vector, and projects it linearly into a "token" embedding; positional encodings preserve spatial order. Standard multi-head self-attention layers then allow every token to attend to every other, giving the model a *global* receptive field from the outset.

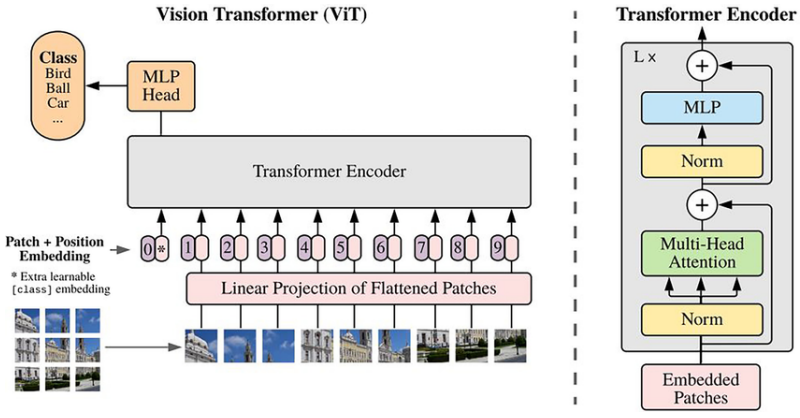

Figure 1: The architecture of the Vision Transformer. From (Dosovitskiy et al., 2020).

Data and compute efficiency ViT's breakthrough hinged on pre-training with hundreds of millions of images (JFT-300M). Subsequent work greatly reduced that

requirement: *DeiT* introduced knowledge distillation and strong augmentation to train competitive ViTs on ImageNet alone (Touvron et al., 2021), while masked-image modelling pre-text tasks (e.g. MAE) further improved sample efficiency.

Hierarchical and localised variants A limitation of vanilla ViT is quadratic attention cost in the number of tokens. Hierarchical designs address this by restricting attention to local windows and merging patches progressively:

- **Swin Transformer** partitions the feature map into shifted windows, achieving linear complexity and enabling dense prediction tasks such as detection and segmentation (Liu et al., 2021).
- **Pyramid Vision Transformer (PVT)** combines spatial-reduction attention with a CNN-like feature pyramid, striking a balance between latency and resolution (Wang et al., 2021).

Representative use cases Vision Transformers now underpin diverse applications:

- **Image classification and retrieval** – ViT and DeiT backbones achieve top-1 accuracies of 84.3% on ImageNet-22k and 86.8% on CIFAR-100 when properly scaled and pre-trained (Dosovitskiy et al., 2020; Touvron et al., 2021); large-scale zero-shot models like CLIP-ViT demonstrate strong transfer to hundreds of downstream datasets without additional fine-tuning (Radford et al., 2021).
- **Dense prediction** – The Swin Transformer family sets new state-of-the-art AP on COCO object detection (up to 58.9 AP) and mIoU on ADE20K semantic segmentation (up to 54.7%) by using shifted-window self-attention and hierarchical feature maps (Liu et al., 2021); the Dense Prediction Transformer (DPT) leverages a ViT encoder with lightweight decoder heads to top the monocular depth-estimation benchmarks (e.g. NYU-Depth v2 RMSE of 0.33 m) (Ranftl et al., 2021).
- **Medical imaging** – token-mixing across the full field-of-view improves histopathology slide classification and 3-D radiology tasks, often surpassing CNN baselines (Graham et al., 2021).
- **Remote sensing** – ViTs handle very high-resolution satellite imagery by combining global context with local attention windows, boosting land-cover mapping accuracy (Wang et al., 2023).

In essence, Vision Transformers reformulate images as sequences, inheriting the flexibility of NLP models while continuing the trend towards ever more data-centric training. Detailed architectural descriptions for the ViT- and Swin-based models used in this thesis appear later in Section 4.

3.2 Out-of-Distribution (OOD) Generalization

Supervised models are typically trained and evaluated on data $\mathcal{D}_{\text{train}} \sim P_{\text{ID}}(x, y)$ drawn from an in-distribution (ID). At deployment, however, inputs may follow an out-of-distribution (OOD) law P_{OOD} that differs—sometimes subtly—from P_{ID}. A model that merely interpolates within P_{ID} can experience severe performance degradation when $P_{\text{ID}} \neq P_{\text{OOD}}$ (Geirhos et al., 2020; Taori et al., 2020). Quantifying and mitigating this gap is the central goal of OOD generalisation research (Liu et al., 2023).

Taxonomy of distribution shift Following Quiñonero-Candela et al. (2009), three shift types are most commonly analyzed:

a) **Covariate shift**: the marginal $p(x)$ changes while the conditional label distribution $p(y \mid x)$ is preserved, e.g. new camera viewpoints or illumination.

b) **Label shift** (a.k.a. prior shift): the class prior $p(y)$ changes while the class-conditional $p(x \mid y)$ remains fixed—for example, in email filtering the overall proportion of spam vs. legitimate messages may vary over time, even though the characteristic features of each class stay the same.

c) **Concept shift**: the conditional $p(y \mid x)$ itself changes—producing genuinely new decision boundaries (e.g. evolving malware signatures). This is the hardest case because "ground-truth" labels become non-stationary.

Recent surveys additionally distinguish *semantic* vs. *syntactic* covariate shifts and propose causal taxonomies (Liu et al., 2023).

Benchmarks and empirical findings. Large-scale evaluations reveal that ImageNet-trained models can lose $10-20$ pp accuracy under realistic covariate shifts such as ImageNet-V2 or ObjectNet (Recht et al., 2019; Barbu et al., 2019). Comprehensive benchmark suites—e.g. WILDS spanning wildlife imagery, satellite data and medical domains—enable systematic comparisons across shift types (Koh et al., 2021). Baseline studies show that many algorithms which excel on synthetic "held-out domains" under perform on real-world shifts, underscoring the gap between academic settings and practice (Gulrajani and Lopez-Paz, 2020).

Mitigation strategies Techniques fall into three broad families:

- **Data-centric** approaches such as aggressive augmentation and domain randomization create synthetic variability that widens the support of $p(x)$ (Tobin et al., 2017; Tremblay et al., 2018). Photo realistic engines (e.g. UnrealROX+) let practitioners control individual "nuisance" factors, linking OOD robustness to specific generative interventions (Gonzalez et al., 2021).

- **Model-centric** methods impose robustness constraints during training—e.g. distributionally robust optimization, feature-wise risk extrapolation, or invariant risk minimization—to select representations stable across domains (D'Amour et al., 2020; Gulrajani and Lopez-Paz, 2020).
- **Post-hoc** uncertainty estimation and OOD detection calibrate or abstain on unfamiliar inputs; transformer-based models with global self-attention often yield better calibrated confidence scores under shift (Ovadia et al., 2019).

Synthetic factors and OOD Synthetic datasets offer unique leverage for disentangling covariate factors. Experiments with controlled backgrounds or object poses have shown that many CNNs latch onto shortcuts like texture or context (Geirhos et al., 2019; Malik et al., 2021). By systematically varying such factors in simulation, one can *a)* diagnose the failure mode, and *b)* train with counterfactual combinations to promote causal, shape-based features (Sariyildiz et al., 2023). Thus, synthetic data generation complements algorithmic robustness techniques in the broader quest for reliable OOD performance.

3.3 Synthetic data for out-of-distribution robustness

Collecting and annotating real photographs that cover *every* plausible lighting condition, background, pose or weather pattern is often prohibitively expensive. Modern graphics engines and generative models let us *procedurally* vary those nuisance factors at scale, effectively growing the support of the training distribution and reducing covariate shift.

Global domain randomization The original domain randomization hypothesis states that, if a simulator exposes enough visual diversity, the real world will simply look like another random draw. Tobin et al. (2017) trained a detector solely in a physics-based renderer with stochastic colours, textures and lighting and attained 44.5 mAP on real warehouse images—only 2.1 pp below a model trained with 10 k labeled photos. Related "Sim2Real" studies for robotic grasping, UAV navigation and autonomous driving confirm that coarse randomisation already closes a large portion of the performance gap (Sadeghi and Levine, 2017).

Targeted or adversarial synthesis. Later work questions whether "more variance everywhere" is always optimal. Volpi et al. (2018) adversarially perturb source images during training to create fictitious domains that maximise a domain-generalisation loss, yielding state-of-the-art accuracy on the four held-out domains of DomainBed. Geirhos et al. (2019) replace ImageNet textures with random artistic styles ("Stylised-ImageNet") and show that shape-biased models gain 12 pp robustness on ImageNet-C corruptions compared with texture-biased CNNs. More recently, diffusion models and GAN pipelines generate style- or attribute-diverse copies of the source data to combat single-domain bias (Anderson et al., 2020).

3.4 Compositional and factorized stress tests

A model might achieve high average accuracy yet rely on shortcuts that surface only when a single generative factor changes. Synthetic rendering lets us freeze all other variables and flip exactly one, exposing such hidden failure modes.

Compositional generalization The CLEVR suite withholds certain shape–colour pairs during training and re-introduces them at test time; Relation Networks trained on CLEVR drop from 98 % to 60 % accuracy on these unseen combinations, whereas Neural Module Networks retain 96 % (Johnson et al., 2017). CLEVRTex extends this idea by randomising object textures to disentangle texture bias from shape bias (Birchfield et al., 2021).

Spurious-correlation diagnostics ObjectCompose pastes familiar COCO objects onto novel PASCAL VOC backgrounds; ResNet-50 accuracy falls from 79.7% to 61.3%, revealing heavy reliance on contextual cues (Malik et al., 2021). Similar single-factor datasets include SHIFT (pose, weather, and sensor noise) and dSprites for disentanglement (Riabtsev et al., 2020; Higgins et al., 2017).

Limitations Single-factor stress tests do not cover interactions between multiple shifts (Koh et al., 2021).

3.5 High-fidelity datasets generated with game engines

Modern game engines (most popular Unity and UE5) deliver photorealistic graphics, real-time physics and programmatic scene control, making them ideal for large-scale, richly annotated vision datasets.

Unity-based pipelines Unity3D became popular in academic vision because it balances an accessible C# scripting interface with a large marketplace of ready-made assets. SYNTHIA (Ros et al., 2016) was among the first Unity datasets to target *semantic segmentation* for self-driving: it recreates a mid-size European city, then systematically varies weather (sunny, rainy, sunset) and camera yaw/pitch to generate pixel-perfect labels across seasons and viewpoints. Shortly afterwards, Virtual KITTI (Richter et al., 2016) cloned the trajectories and sensor layout of the real KITTI benchmark in a stylised Unity world, enabling direct, one-to-one comparisons between synthetic and real frames; training DeepLab on both sources lifted cross-city IoU by more than six percentage points relative to real-only training.

Building on these ideas, Unity released the open-source Perception Toolkit (Dunn et al., 2021), which wraps photorealistic rendering, physics simulation, and *domain-randomisation* primitives behind a no-code graphical interface. The toolkit can randomise lighting, textures, object poses and camera parameters on a per-frame

basis while exporting dense annotations—bounding boxes, instance masks, depth, normals and keypoints—at up to 100 fps on a single GPU. It underpins SynthDet (Bhattacharyya et al., 2021) (800 k procedurally generated retail-object images) and FarmVision (Wang et al., 2022) (2 M crop-disease frames). Detectors trained on SynthDet achieve 72.9 mAP on a withheld real supermarket shelf set—matching a baseline trained on 100 k hand-labelled photos while halving annotation costs—whereas FarmVision models close a 12 pp F1 gap between laboratory and in-field images by augmenting the rare-disease classes with synthetic leaves.

Unity pipelines therefore illustrate a recurring pattern: precise control over nuisance factors (weather, pose, background clutter) plus dense automatic labels yields synthetic corpora that not only reduce manual effort but also improve cross-domain robustness, especially when real data for certain conditions are scarce or expensive to collect.

Unreal Engine pipelines Since the release of UE5 the engine ships with the real-time Lumen global-illumination system 3.6 and Nanite micropolygon geometry 3.6, providing photorealism out of the box (EpicGames, 2022; Karis et al., 2021). Together with Blueprint visual scripting, C++ extensibility, and the free Quixel Megascans asset library, UE lets vision researchers build large-scale, physically accurate simulators without deep graphics expertise. However, due to UE5s relatively recent release date in April 2022, most work done in this field uses its predecessor UE4. This older version of the engine offered a far lesser degree of realism and workflows developed for this version might not be compatible with UE5. Falling Things (FAT) employs UE4 physics to drop household objects in random six-degree-of-freedom poses, producing 50 000 RGB-D frames with perfect pose and mask labels; a Mask R-CNN trained only on FAT attains 96.3 AP on real cluttered-table scenes, less than one percentage point below a model fine-tuned on 4 000 real photographs (Tremblay et al., 2018). UnrealGT streams colour, depth and instance masks along user-defined UAV trajectories; detectors trained on its 25 000 synthetic aerial frames reach 78.2 mAP on the real VisDrone benchmark, just 3 pp shy of training on 10 000 hand-labeled images (Pollok et al., 2019). UnrealROX+ decouples virtual sensors from robot meshes, supports dynamic HDR lighting, and exports additional modalities such as surface normals and optical flow (Gonzalez et al., 2021).

For autonomous driving, the UE-based CARLA simulator has become the standard robustness test-bed: injecting stochastic rain, fog or low-sun glare into CARLA scenes reduces ImageNet-initialised object detectors by up to 15 pp AP, whereas pre-training on CARLA's 200 000 synthetic frames recovers more than half of that loss (Dosovitskiy et al., 2017; Wen et al., 2020). These case studies illustrate UE's central advantage: photorealistic rendering plus precise control over physics and sensors yield synthetic corpora that transfer to real-world performance with only small residual gaps.

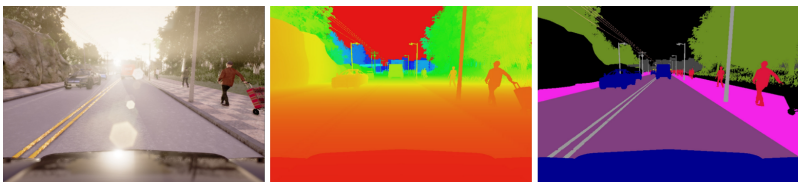

Figure 2: Three of the sensing modalities provided by CARLA. Normal Vision, ground-truth depth, and ground-truth semantic segmentation. From (Wen et al., 2020).

Photorealistic and controllable datasets The PUG family uses UE5's advanced rendering pipeline to generate over 150 ImageNet-aligned object categories with exhaustive control over scene variables. Each PUG scene is parameterised by background environment (urban, rural, indoor), lighting conditions (time of day, weather effects), camera intrinsics/extrinsics (focal length, viewpoint, tilt) and object textures (material reflectance, weathering). Approximately 200k images are rendered at 1024×1024 resolution, each annotated with class labels, segmentation masks and bounding boxes. Evaluations show that ImageNet-pretrained Vision Transformers lose roughly 18 percentage points in top-1 accuracy when tested directly on PUG versus the standard ImageNet validation set. Remarkably, fine-tuning on just 5% of PUG images ($\approx$ 10k samples) recovers 11 pp of accuracy and simultaneously reduces corruption error on the ImageNet-C benchmark by 3 pp, indicating that small amounts of targeted synthetic data can substantially improve both in-distribution performance and robustness to common corruptions (Bordes et al., 2023).

Domain-specific frameworks Beyond generic object categories, several Unreal Engine-based pipelines target narrow application domains. UnrealFall synthesises 60k MetaHuman videos depicting realistic human falls across diverse indoor settings (living rooms, kitchens, hospital wards). Each clip varies avatar physique, clothing, furniture layout and lighting. Action recognition models fine-tuned on UnrealFall increase real-world fall-detection F_1 from 0.71 (trained on limited real footage) to 0.83, reducing false alarms in surveillance systems (Mulero-Pérez et al., 2024).

In urban remote sensing, Turkcan et al. (2023) introduce a UE5 "City Sample" dataset that renders 50k orthorectified aerial tiles over procedurally generated city blocks, varying building architecture, road networks, vegetation and seasonal foliage. Pre-training a Faster R-CNN on City Sample (with synthetic annotations for cars, pedestrians and road signs) and then fine-tuning on 2k real orthophotos raises mAP from 45.3 (CARLA pre-training baseline) to 53.1–an improvement of 7.8 pp.

Alternatives Open-source engines such as BlenderProc (Denninger et al., 2020) and NVISII (Morrical et al., 2021) offer Python APIs and permissive licences for photorealistic data generation. BlenderProc leverages Blender's Cycles renderer and procedurally randomises object placement, materials and post-processing effects, producing labeled datasets for segmentation and instance recognition. NVISII

provides GPU-accelerated path tracing with a modular scene graph, enabling rapid rendering of tens of thousands of labeled frames per hour. These frameworks broaden access to high-fidelity simulation for researchers without proprietary game-engines.

Collectively, these domain-specific Unreal Engine and open-source pipelines demonstrate that when synthetic scenes closely mirror real-world physics and visual styles, models trained or pre-trained on them transfer with minimal performance gaps–often outperforming purely real-data baselines in scarce-data regimes.

3.6 Unreal Engine 5: Core Functionality and Terminology

Before getting to the implementation of our UE5 plugin and dataset generation (Section 4), we will provide basic information about the engine, its core features and terminology. UE5 follows an actor–component paradigm and ships with two flagship rendering technologies — Lumen global illumination and Nanite virtualised geometry alongside optional hardware ray-tracing support.

Editor layout The Unreal Editor consists of a central Viewport for 3-D navigation, an Outliner that hierarchically lists every Actor in the open map, a context-sensitive Details panel for property editing, and a Content Browser that exposes all imported assets (meshes, textures, Blueprints, . . .) (Epic Games, 2025a).

Worlds, levels and streaming. A `World` object represents the top-level simulation context; it contains a persistent level plus an optional set of streaming sublevels that are loaded and unloaded on demand via the World Partition system (EpicGames, 2025).

Actors and components Every placeable entity derives from the base class `AActor`. Actors are merely containers for *components*, e.g. a `StaticMeshComponent` for geometry, a `LightComponent` for illumination, or a `CameraComponent` for virtual sensors. Components can be spawned, destroyed or re-parented at runtime, which our plug-in uses to programmatically change variables like light colors (Epic Games, 2025a).

Static meshes and materials. A static mesh is a read-only triangle buffer stored in GPU memory (Epic Games, 2025e). Visual appearance is defined by a material, a node-based shader graph that computes per-pixel base-colour, normal, roughness and emissive terms. During data generation we vary material instances (e.g. swap the default material for brushed metal) to enlarge texture diversity without touching geometry.

Blueprint visual scripting Blueprints are UE5's node-based alternative to C++. They compile to native code but remain editable in the editor, enabling rapid prototyping (Epic Games, 2025b).

Native C++ workflow Performance-critical routines and external-library bindings live in native C++ modules. The Unreal Build Tool (UBT) generates IDE projects, invokes the Unreal Header Tool to expand the `UCLASS`, `UFUNCTION` and `UPROPERTY` reflection macros, and compiles shared libraries that the editor can hot-reload live (Epic Games, 2024).

Fab asset marketplace Fab is Epic's unified digital-asset store, merging the former Unreal Engine Marketplace, Quixel Megascans, Sketchfab Store and ArtStation Marketplace into one catalogue that is searchable directly in the editor. More than sixty thousand assets—photogrammetry-based meshes, procedural materials, HDRI skies, MetaHumans, animations, sound effects and full environment packs can be previewed in-place and imported through a drag-and-drop workflow that automatically creates LODs, collisions and material instances (Epic Games, 2025c).

Lumen real-time global illumination Lumen traces thousands of micro-rays per pixel against a hierarchical mesh representation and blends the result with screen-space and probe data, delivering multi-bounce indirect lighting at real-time framerates (Karis et al., 2021). Because indirect lighting updates on every frame, effects like sky turbidity or flicker lights can be animated without precomputing lightmaps.

Nanite virtualised geometry Nanite replaces fixed-LOD meshes with a hierarchical cluster structure; clusters are culled and rasterised directly in a compute shader, allowing scenes with billions of triangles while keeping GPU work roughly proportional to screen pixels (Karis, 2021). This allows for the usage of high-poly assets, like high-detail scans of real life objects, without the need of manually reworking them.

Hardware ray tracing On GPUs with dedicated RT cores (NVIDIA RTX, AMD RDNA2 +) UE5 can offload ray-box and ray-triangle tests to hardware, boosting performance for reflections, shadows and Lumen's high-quality mode. The feature is toggled via the Project Settings (Epic Games, 2025d).

These concepts—world/level hierarchy, actor–component model, asset types, and the modern rendering stack—form the conceptual foundation on which our plugin operates.

4 Methodology

4.1 Reference hardware

All performance figures reported for the plugin and dataset were recorded on a desktop equipped with an Intel Core i7-13700K, 64 GB DDR5-5600 RAM and an NVIDIA RTX 4090 (24 GB). Absolute timings will vary across systems, but the relative comparisons between different versions of our plugin remain valid.

4.2 Synthetic–Data Plugin for UE5

4.2.1 Overview

Our plugin encapsulates every step of synthetic–data generation inside UE5 and therefore removes the need for any external DCC tools or post-processing scripts. Once the module is enabled in an UE5 project, it can be used to create a "data-generation game" whose Play button opens a game view that produces images and metadata rather than gameplay. All configuration is performed with native editor widgets.

Design goals The plug-in was architected around four guiding principles:

a) **In-engine only** All dataset logic lives inside the UE5 runtime; no third-party plug-ins, Python bridges or similar tools are required. This maximises forward-compatibility with future engine releases and simplifies the initial setup.

b) **Render-time efficiency** To enable the creation of large datasets, we optimized the rendering process in order to exceed the ≈1 fps throughput reported by Bordes et al. (2023).

c) **Minimal user setup** After copying the plug-in into an existing project, the user immediately gains access to all necessary actors and the custom game instance, making a first test run possible within minutes.

d) **Extensibility** While the shipped feature set covers a lot of common vision-research modalities, the rendering logic is designed in a way that allows for easy adaptation and modification.

Key functionality

a) **Parametric scene assembly** For each render cycle it instantiates a user-specified scene, places one of N_{model} 3-D assets selected from a class-label table on a pre-selected spot in the scene, applies a material variant and light colour from enumerated pools, and (optionally) spawns volumetric fog to realise controlled occlusion.

b) **Automatic camera orchestration** The camera iterates over a ordered list of positions to capture

c) **Batch image capture.** Each view is rasterised at the user-chosen resolution (e.g. 512×512) with the back buffer diverted directly into an RGBA8 array—avoiding the double render incurred by screenshot method.

d) **Metadata export** A CSV writer logs the scene ID, model ID, class, material ID, light $\langle r, g, b, a \rangle$, camera pose, fog state and output filename for every frame.

e) **Combinatorial expansion** In its current purpose, the plug-in exhaustively iterates over the Cartesian product of all parameters per scene, yielding a total of $N_{\text{scene}} \times N_{\text{model}} \times N_{\text{mat}} \times N_{\text{light}} \times N_{\text{cam}}\, N_{\text{fog}}$ unique images.

These capabilities satisfy the overarching requirement of producing large, richly annotated datasets with minimal human effort while keeping render times low. The following subsection recounts our initial Blueprint prototype and the lessons that motivated a full C++ rewrite.

4.2.2 First iteration

The first working version was implemented exclusively with Blueprints, Unreal Engine's visual-scripting language (Epic Games, 2025b). This allowed functionality to be sketched quickly without the usage of C++.

Functional scope The prototype already performed **(i)** scene map loading and unloading, **(ii)** placement of 3-D models chosen from a class-label lookup table, **(iii)** image capture from a sequence of manually placed TargetPoint coordinates that were interpreted as virtual camera poses, **(iv)** optional volumetric fog insertion for controlled occlusion, and **(v)** capturing of RGB images.

Performance Running on the reference workstation4.1, the system produced about one 512×512-pixel image per second, matching the throughput reported for the Blueprint-oriented PUG framework (Bordes et al., 2023). Each image was captured with the HighResScreenshot function, which re-renders the current frame at the requested resolution.

Identified limitations

a) **Readability and maintenance**. As node counts increased, Blueprint graphs became dense; tracing execution paths, adding and modifying code required significant effort and time.

b) **Metadata output** Blueprint offers no function to append data to an existing file. Generating metadata would therefore incurred high overhead or demanded custom C++ nodes.

c) **Redundant rendering passes** the HighResScreenshot function, used to capture screenshots in blueprint, performs additional render-passes for each frame. This is necessary when rendering images larger than screen resolution, but, from our testing, yields no noticeable visual improvements on images at or below the display resolution.

d) **Manual camera placement** Users had to place TargetPoints in every scene and register them in a per-scene array; consistency across scenes was not enforced and manual effort for the user was unnecessarily increased.

e) **Nested-loop overhead** The render logic was expressed as nested Blueprint ForEach loops that iterate over every combination of parameters, while having to be manually stalled until the asynchronous screenshot function saved a screenshot, before continuing. Rewriting the algorithm with recursion was necessary.

Outcome The prototype confirmed that a fully in-engine pipeline is feasible, but the observed limitations violated the design goals of render-time efficiency and ease of maintenance. Together with the fact, that the render loop needed to be reimplemented in order to remove the nested-loop overhead, the decision was made to migrate the core logic to C++ and keep Blueprint as a configuration layer. The resulting architecture is detailed in Section 4.2.3.

4.2.3 Final Implementation

Configuration All configuration logic is implemented in Blueprint and defined once, then reused for every scene that the plugin renders. A dedicated `ConfigActor` is placed in the initial world; its parameters can be edited directly in the Unreal Editor UI. To enable dataset generation the user must perform only two additional steps: *(i)* set the provided `CustomGameInstance` class in the project settings, and *(ii)* create a new level that contains a `SpawnPoint` and the `DatasetRenderActor`. After at least one camera pose has been specified in the configuration, launching the project from the start level triggers automated rendering of all images.

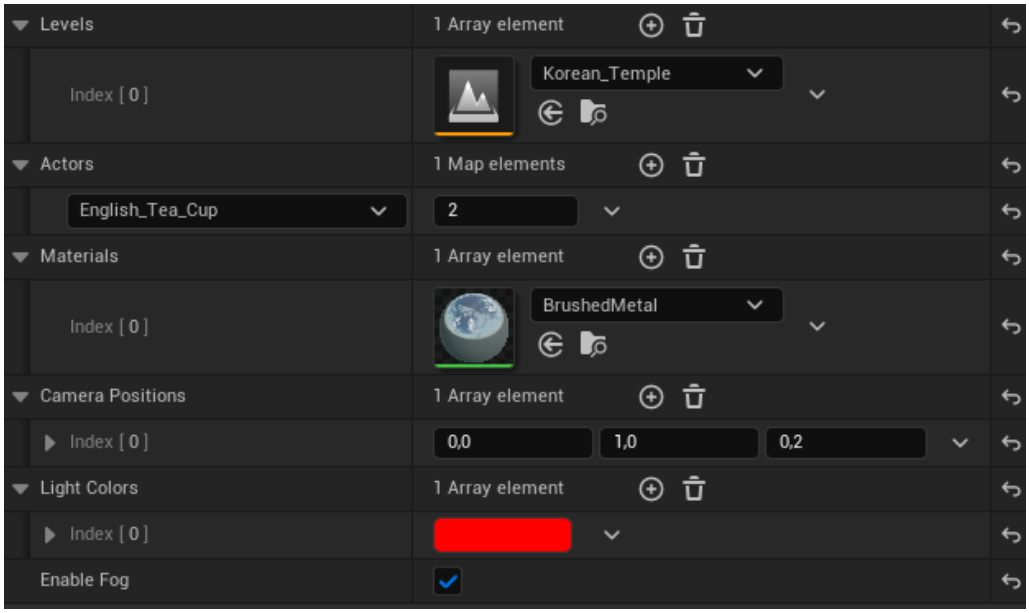

Figure 3: Graphical interface of the `ConfigActor` with all available parameters.

Final renderer functionality The Blueprint prototype implemented the dataset generation as a set of nested ForEach loops. The C++ implementation replaces those loops with a event-driven state machine that lives in a single method. Index counters track the different variables; each time the manager receives a callback that the screenshot has been successfully captured, it advances the counters, decides which parameter to change next, performs the associated mutation in the scene, and schedules the following screenshot request. This design keeps the game thread responsive, and eliminates Blueprint overhead. It also provides an easy way to change or expand the plugins functionality, as all methods that modify variables are individually handled there.

Automatic camera generation After an actor is spawned, the capture-manager computes its bounding sphere radius r and the active camera's field of view θ. Using only the inner 256px of a 512px frame, the effective half-FOV is $\theta_{\text{crop}} = 0.5\,\theta\,(256/512)$. The required camera distance is $d = 1.05\,r/\tan\theta_{\text{crop}}$, where the 5% factor prevents near-plane clipping. For each user direction vector $\mathbf{v}_i$ the camera location is $\mathbf{c}_i = \mathbf{p}_{\text{obj}} + d\,\widehat{\mathbf{v}}_i$ and the pawn's view is aligned with lookat$(\mathbf{c}_i, \mathbf{p}_{\text{obj}})$, keeping the object centred across models of different size.

Frame-delay capture After every change in a scene (spawn, material swap, fog toggle, camera move) the engine, in theory, must produce at least one new frame before the back buffer contains the updated image. However, due to the nature of lumen, it is necessary to wait a multitude of frames for the indirect lighting in the scene to adjust to color changes. A timer of $\Delta t = 0.05$ s (based on measured

performance) is therefore set before requesting a new screenshot. Higher frame times or lower frame rates would require proportionally longer delays.

Performance A move from Blueprint to C++ plus the elimination of the redundant re-render steps (see subsubsection 4.2.2) reduced the mean per-frame time from ≈ 1.00 s to ≈ 0.06 s. The speed-up factor is therefore

$$\frac{1.00 \text{ s}}{0.11 \text{ s}} \approx 16.6,$$

i.e. 16.6 times faster than both our prototype and the Bordes et al. (2023) reference implementation (Note that render time can vary depending on complexity and size of actors and materials being loaded. This number is the average of a few thousand selected samples from our dataset.)

Limitations The current frame-delay mechanism (section 4.2.3) relies on a fixed 50 ms timer that was hand-tuned to the slowest scene in our test set. This is a stop-gap rather than a general solution: larger or more complex datasets will require a routine that samples recent frame times, fps and potentially other factors and adjusts the delay dynamically. This would additionally boost performance, since the render speed would not be capped by the lowest performer. In addition, the plug-in has been developed and profiled only on a Windows desktop; a stand-alone Linux build should work, but was not tested and may expose platform-specific differences in the screenshot pipeline.

Overall, migrating the control logic from nested Blueprint loops to a C++ state machine improves adaptability, removes VM overhead, and lets the engine tick normally between captures—meeting the efficiency and maintainability goals defined in Section4.2.1.

4.3 Synthetic OOD Dataset

Motivation and Scope The UE5 plugin developed for this thesis is intended not only as a reproducible baseline for future work but also as a data–generation tool for this thesis. Following the guiding principles of the PUG family of synthetic, photorealistic datasets—namely high rendering realism, explicit factor control, and full factor–label availability (Bordes et al., 2023)—the goal was to construct a smaller, fully controlled out-of-distribution (OOD) benchmark that focuses on ImageNet-1k classes while relying exclusively on free assets obtained from the *Fab* (Epic Games) marketplace.

Design Decisions PUG:Animals is built as the complete Cartesian product of all available factors, whereas PUG:ImageNet varies one factor at a time to keep its class coverage broad but image count moderate (Bordes et al., 2023). Initial

scoping showed that the number and quality of free ImageNet 1k–compatible 3-D assets is an order of magnitude lower than the 724 assets used in PUG:ImageNet; hence matching the original scale of 88 328 images proved infeasible. Adopting the PUG:Animals approach, we prioritised factor completeness over class breadth, yielding a dataset in which all factor combinations are rendered for a small, balanced set of classes.

Factor Space and Rendering Pipeline Let $\mathcal{S}$ (scenes), $\mathcal{C}$ (object classes), $\mathcal{R}$ (meshes per class), $\mathcal{M}$ (materials), $\mathcal{L}$ (light colours), $\mathcal{V}$ (camera viewpoints), and $\mathcal{O}$ (fog state) denote the controllable factors. The dataset is the Cartesian product

$$\mathcal{D} = \mathcal{S} \times \mathcal{C} \times \mathcal{R} \times \mathcal{M} \times \mathcal{L} \times \mathcal{V} \times \mathcal{O},$$

instantiated as:

- $|\mathcal{S}| = 8$ scenes: *city fountain, city underpass, salt plains, Sahara desert, desert with foliage, Korean temple, African slate quarry, demo gallery*;
- $|\mathcal{C}| = 7$ classes: *dial phone, desk, cup, banana, hammer, umbrella, chest*;
- $|\mathcal{R}| = 4$ cleaned meshes per class;
- $|\mathcal{M}| = 6$ materials: default, checkerboard, white, yellow car-paint, neon-green, brushed-metal;
- $|\mathcal{L}| = 4$ light colours: white, red, green, blue;
- $|\mathcal{V}| = 8$ calibrated camera poses (front, back, left, right, plus four elevated obliques);
- $|\mathcal{O}| = 2$ fog states: on/off.

The total image count is therefore

$$8 \times 7 \times 4 \times 6 \times 4 \times 8 \times 2 \;=\; \mathbf{86\,016}.$$

All images are rendered at 512x512 px resolution with UE5 hardware ray tracing enabled and motion blur and auto exposure disabled. Detailed information about each variable is stored in the accompanying metadata CSV-file.

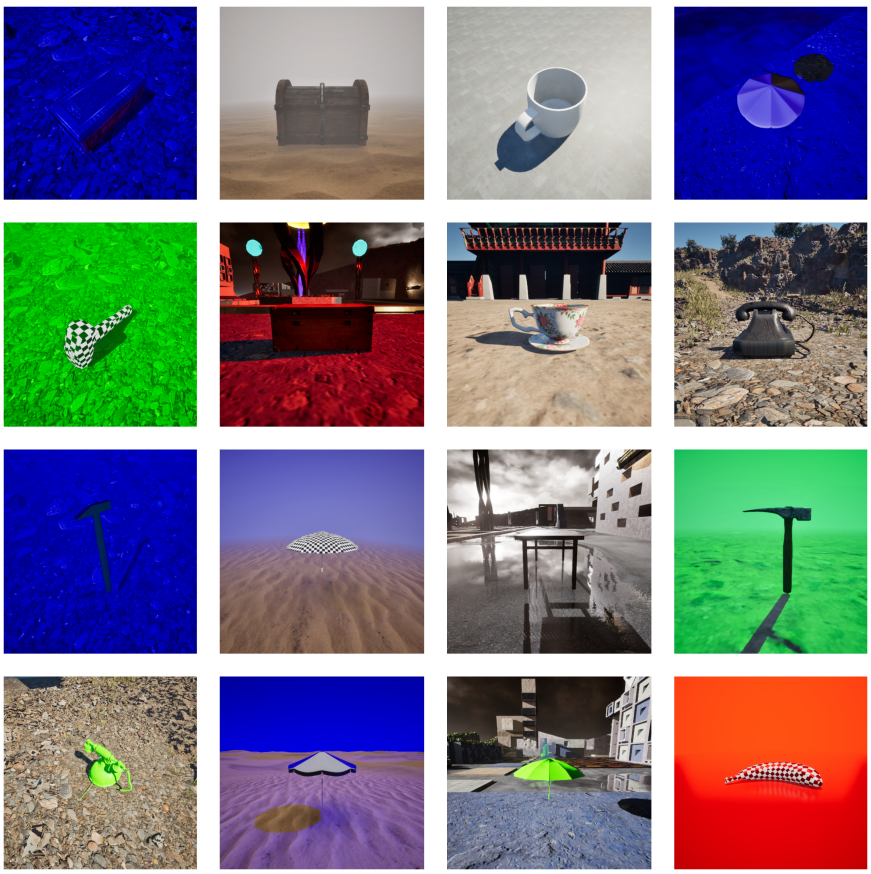

Figure 4: A random collection of images from our dataset

Practical Obstacles

a) **Asset price:** considering the scope of a bachelors thesis, buying expensive assets is not an option. Therefore the highest quality assets as used by Bordes et al. (2023) where not accessible.

b) **Asset scarcity:** five usable meshes existed for only three of our classes, so every class was capped at four meshes to keep balance.

c) **Asset integrity:** several promising assets were incompatible with UE5.4 or required extensive repair in Blender, which exceeded project time constraints;

d) **Semantic collapse:** some objects (e. g. oranges) became unrecognizable after material or lighting changes were applied and were therefore excluded from the dataset.

Limitations The benchmark covers just seven object categories with only four different models in each, leading to a noticeable lack in class diversity. Furthermore

it also lacks pose articulation. Context-object correlations inside a scene remain constant, and photometric variations are limited to global colour shifts and fog. Future iterations should enlarge class diversity.

Summary Despite its modest class set, the dataset delivers 86016 photorealistic OOD images with exhaustive factor labels, demonstrating that a high-quality synthetic dataset can be assembled by only using free assets.

4.4 Convolutional Backbones

We benchmark five widely used CNN backbones, all obtained from the timm library (Wightman, 2019). For every network we load the ImageNet-1k pretrained weights in Python.

resnet50 ResNet-50 introduces identity skip connections to enable very deep networks without vanishing gradients (He et al., 2016). Its four-stage design (3×3 convolutions plus bottleneck blocks) totals ~ 25.6 M parameters.

resnet101 A deeper member of the ResNet family with 101 layers, retaining the same residual building-block design while roughly doubling depth and parameters (~ 44.5 M) (He et al., 2016).

densenet201 DenseNet-201 links each layer to *every* preceding layer, boosting feature reuse and gradient flow while keeping the parameter count modest (~ 20.0 M) (Huang et al., 2017).

convnext_base ConvNeXt-Base modernises standard CNN design (large-kernel depthwise convs, inverted bottlenecks, GELU, LayerNorm) to follow the scaling trends of Vision Transformers while retaining convolutional efficiency. It has ~ 88 M parameters and uses the strong data-augmentation recipe from Facebook AI Research (Liu et al., 2022).

convnext_large_mlp A larger ($\sim$ 198 M parameters) ConvNeXt variant that widens all stages and replaces the final global pooling with a lightweight MLP head, yielding stronger performance on high-resolution inputs while preserving the core architectural principles of ConvNeXt (Liu et al., 2022).

4.5 Vision-Transformer backbones

We benchmark three widely used ViT backbones, all are loaded from `timm` (Wightman, 2019) and pretrained on ImageNet-1k.

vit_base_patch16_224 The original Vision Transformer "Base/16" configuration splits a 224×224 image into 14×14 non-overlapping 16×16 patches, projects them to a 768-D embedding, and processes the resulting sequence with 12 encoder blocks and 12-head self-attention (Dosovitskiy et al., 2021). At $\sim$ 86 M parameters and $\sim$ 17 GFLOPs, it sets a strong baseline that matches ResNet-152 accuracy while retaining a relatively modest compute budget.

vit_large_patch16_224 The "Large/16" variant deepens the network to 24 encoder blocks, widens the hidden size to 1024 and keeps 16-head attention, pushing capacity to $\sim$ 307 M parameters and $\sim$ 55 GFLOPs (Dosovitskiy et al., 2021). Its higher model expressiveness improves ImageNet top-1 to $\approx$ 85% and often yields better transfer learning performance on data-rich tasks such as video-classification or large-scale segmentation.

swin_base_patch4_window7_224 Swin Transformer-Base introduces a hierarchical architecture with shifted window (SWIN) self-attention: images are first partitioned into 4×4 patches, and attention is computed within 7×7 windows that shift between layers to enable cross-window interaction (Liu et al., 2021). With $\sim$ 88 M parameters and $\sim$ 15 GFLOPs, it combines ViT-level accuracy ($\approx$ 83% ImageNet top-1) with linear computational scaling in image size, making it well suited for dense prediction tasks such as detection and semantic segmentation.

4.6 Evaluation Methods

Because our test set is a synthetic OOD benchmark with $|\mathcal{C}| = 7$ balanced ImageNet-1k classes (Section 4.3), we report two complementary metrics:

1. **Top-1 accuracy** — measures absolute correctness and thus explains how many images the model classified correctly (Section 4.6.1).
2. **AUC** — evaluates the ability to rank the true class higher than all others across all decision thresholds, providing insight into confidence calibration and error trade-offs under distribution shift (Section 4.6.2).

Together they characterise both point-estimate accuracy and threshold-independent ranking quality, which is critical when models encounter OOD factors such as novel lighting, material or fog settings that differ from ImageNet.

4.6.1 Top-1 accuracy

Definition Top-1 accuracy is simply the proportion of predictions a model gets right:

$$\mathrm{Acc}_{\text{Top-1}} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}.$$

Why it is appropriate Top-1 has been the official leaderboard metric of the ILSVRC since 2012, enabling direct comparison with the ImageNet literature (Russakovsky et al., 2015) and since every class in our dataset is equally represented, accuracy provides a good metric for comparison.

4.6.2 Rejection-aware AUC

Why a modified AUC is needed Each network still predicts over the full 1 000 ImageNet classes, but our OOD benchmark contains only $K = 7$ in-scope categories (Section 4.3). A straight macro–AUC would treat high confidence in any out-of-scope class as merely a low but valid score for all in-scope classes and can therefore overestimate ranking quality. To penalise such open-set mistakes we adopt the rejection-aware variant inspired by the reject-option framework of Chow (1970).

Definition Let $p_{i,k}$ be the soft-max probability that sample i belongs to ImageNet class k. We construct a K-dimensional score vector $\tilde{\mathbf{p}}_i$ for the seven evaluation classes as follows:

$$\tilde{\mathbf{p}}_i \;=\; \begin{cases} \big(p_{i,c_1}, \ldots, p_{i,c_K}\big) \Big/ \sum_{j=1}^{K} p_{i,c_j}, & \text{if } \arg\max_k p_{i,k} \in \{c_1, \ldots, c_K\}, \\ \dfrac{1}{K}\,\mathbf{1}, & \text{otherwise.} \end{cases}$$

The first branch keeps the model's original probabilities and renormalises them to sum to one; the second branch assigns the uniform vector $\frac{1}{K}\mathbf{1}$ to any sample whose top-1 prediction is *not* in the evaluation set, i.e. a rejection.

For every class c_k we then compute the usual one-vs-rest ROC and its area, and finally take the macro average:

$$\mathrm{AUC}_{\mathrm{rej}} \;=\; \frac{1}{K} \sum_{k=1}^{K} \mathrm{AUC}\big(\tilde{\mathbf{p}}_{\cdot k}\big),$$

Interpretation A sample that the model rejects contributes the same rank as a random guess (AUC = 0.5 in the binary case), so abstentions can only reduce—or leave unchanged—the overall score. Because the metric is still threshold-independent and monotone, it complements Top-1 accuracy by revealing whether the model's *confidence ordering* degrades under OOD factors such as unusual lighting, materials, or fog (Ovadia and et al., 2019). When all predictions fall inside the seven classes the formula collapses to the textbook macro-AUC (Hand and Till, 2001; Fawcett, 2006).

5 Evaluation

Images are center cropped, resized from 512x512 to 224x244 and then normalized using ImageNet normalization.

5.1 overall Performance

Model	Top-1	AUC
ResNet-50	13.06 %	0.623
ResNet-101	23.04 %	0.707
DenseNet-201	12.17 %	0.616
ConvNeXt-Base	23.53 %	0.709
ConvNeXt-Large-MLP	37.51 %	0.806
ViT-B/16	17.32 %	0.659
ViT-L/16	25.53 %	0.724
Swin-B	28.28 %	0.744

Table 3: Top-1 accuracy and rejection-aware AUC on our OOD dataset.

Although our dataset (86,016 images) is close in size to PUG:ImageNet (88,328 images), (Bordes et al., 2023) it is more challenging because each image here is drawn from the full Cartesian product of eight backgrounds, four meshes, six materials, four light colours, eight camera poses, and two fog states. In contrast, PUG:ImageNet varies at most one factor at a time(Bordes et al., 2023). As a result, a direct performance comparison is hard to make.

Architectural trends Three clear patterns emerge from Table 3:

- **Classical CNNs** (ResNet-50/101, DenseNet-201) exhibit the steepest drop (12–23% Top-1, AUC 0.616–0.707). Prior work has shown that ImageNet-trained CNNs rely heavily on local texture cues, which our material and lighting variations systematically disrupt(Geirhos and et al., 2020).
- **Modern ConvNets** (ConvNeXt) outperform older CNNs: ConvNeXt-Large surpasses ResNet-101 by 14 percentage points in Top-1 and by 0.10 in AUC. This aligns with evidence that wider receptive fields and stronger regularization enhance robustness to distribution shifts(Liu et al., 2022).
- **Vision Transformers.** ViT-B/16 and ViT-L/16 lag behind ConvNeXt-Large in accuracy but achieve competitive AUCs (0.659 vs. 0.724), suggesting that their uncertainty estimates degrade more gracefully under compound shifts. Scaling ViT-B to ViT-L adds 8 percentage points Top-1 but still falls 12 percentage points short of ConvNeXt-Large, indicating that patch embeddings alone do not fully immunize against the shape-texture conflicts we introduce.

The hybrid Swin-B (28.3%) lies between ConvNeXt-Base and ConvNeXt-Large, supporting the finding that localized attention windows plus hierarchical feature pooling improve shift robustness(Liu et al., 2021).

Accuracy-AUC relationship Across all eight models, higher Top-1 strongly correlates with higher AUC, but with a shallow slope: for example, DenseNet-201 (12.2% Top-1, 0.616 AUC) and ResNet-50 (13.1% Top-1, 0.623 AUC) differ by only 0.007 in AUC despite a 0.9 percentage point gap in accuracy. This indicates that models become both less accurate and less confident under simultaneous shifts, though their relative calibration differences persist.

Comparison with ImageNet-1k baselines All networks achieve roughly 80–85% Top-1 on ImageNet-1k (e.g. ResNet-50: 80.4%, ConvNeXt-Base: 83.8%, Swin-B: 83.6%)(Wightman et al., 2021; Liu et al., 2022, 2021). Here, absolute drops range from 43 percentage points (ConvNeXt-Large) to 68 percentage points (DenseNet-201), reinforcing that in-distribution accuracy is a weak predictor of OOD robustness(Taori et al., 2020).

Interpreting failures via dataset factors Two key insights arise:

1. **Multiplicative shift severity** By varying six factors jointly, texture-biased CNNs fail catastrophically when material and lighting perturbations coincide with viewpoint and context changes. Transformers handle viewpoint shifts more gracefully but remain vulnerable when extreme material changes corrupt shape-texture cues.

2. **Capacity versus inductive bias** Increasing model capacity helps (ViT-L vs. ViT-B; ConvNeXt-Large vs. ConvNeXt-Base), yet ConvNeXt-Large still outperforms the larger ViT-L. This suggests that convolutional inductive biases—hierarchical pooling and local spatial smoothing—remain valuable in severe OOD regimes(Naseer and et al., 2021).

5.2 Baseline Comparison

Before turning to the full factor analysis we first verify that the 3D meshes and overall images are recognisable when only minimal variables are applied. We therefore extract a *frontal + background-only* slice in which every image is rendered from the canonical front view, with the default material, white light, and no fog; the sole varying factor is the choice of one of the eight scene backgrounds. With four meshes per class this yields $4 \times 8 = 32$ images for each of the seven classes, or 224 images in total. If there is no different underlying issue with the dataset, this should in theory increase accuracy. For reference, the ImageNet-1k validation set provides 50 images per class (Russakovsky et al., 2015). Our per-class sample size is thus of the

same order, and any minor increase in sampling variance is unlikely to change the overall conclusion. Table 4 reports Top-1 accuracy on this slice; each model achieves substantially higher accuracy here than in Table 3, confirming that the severe drop in the full benchmark must be attributed to the additional OOD factors (viewpoint, material, lighting, fog), not to deficiencies in the 3D geometry.

Model	Top-1 Accuracy
ResNet-50	36.6%
ResNet-101	50.0%
DenseNet-201	29.0%
ConvNeXt-Base	56.7%
ConvNeXt-Large-MLP	70.5%
ViT-B/16	45.5%
ViT-L/16	63.4%
Swin-B	57.1%

Table 4: Top-1 accuracy on the frontal + background-only slice. Each model is evaluated on 224 images (32 per class).

5.3 Factor Analysis

In this subsection, we examine how individual generative variables—background, material, lighting, camera pose, and fog—affect recognition performance. Rather than presenting results for all eight architectures, we focus on ConvNeXt-Large, which achieves the highest overall OOD Top-1 accuracy (37.5%) and exhibits a smooth performance profile across classes. By selecting the strongest model, we ensure that any observed degradation under a given factor is also present (and often more severe) in weaker models. If a particular effect is exceptionally pronounced in a different model, we will note it explicitly (The figures for all other models can be found in the appendix A). The following sections present per-factor breakdowns and discuss how each variable independently contributes to the accuracy.

5.3.1 Material

Figure 5 shows the Top-1 accuracy of ConvNeXt-Large on the full OOD dataset, broken down by material. Each bar corresponds to all test images rendered with that material—in other words, 86,016 ÷ 6 = 14,336 images per material—covering every combination of background (8), class (7), mesh (4), light colour (4), camera pose (8), and fog state (2).

- **Default**: the original texture and colour of each mesh,
- **CheckerMaterial**: a high-contrast black and white checkerboard covering the entire object,

- **YellowCarPaint**: a smooth, glossy yellow finish,
- **NeonGreen**: a uniformly bright green coating,
- **BrushedMetal**: a matte metallic sheen with fine directional noise,
- **White**: a plain white surface.

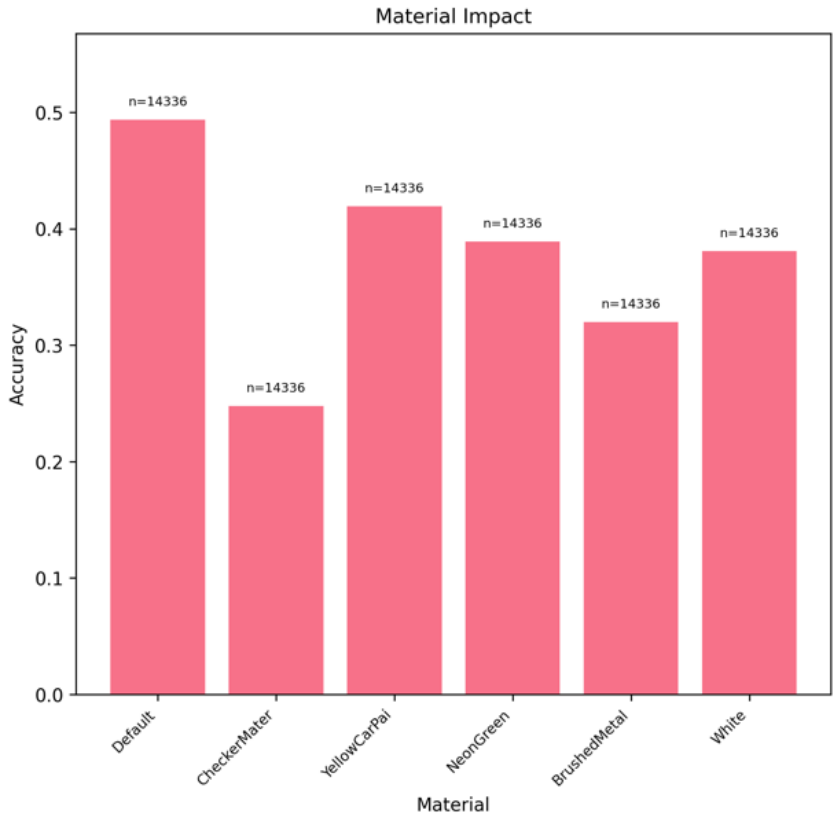

Figure 5: Per-material Top-1 accuracy (ConvNeXt-Large). Each bar corresponds to 14 336 test images.

1. **Default material yields the highest accuracy (49%).** When meshes use their original, photorealistic textures, ConvNeXt-Large correctly classifies almost half of the samples despite changes in background, viewpoint, lighting, and fog. This indicates the network's learned features transfer best when texture remains close to the training distribution.
2. **Checkerboard pattern causes the steepest drop (25%).** Applying a high-contrast checkerboard destroys almost all object-specific texture cues (e.g. wood grain, metal details), forcing the model to rely purely on shape. The fall to 25% Top-1 shows that even a high-capacity ConvNet remains heavily texture-biased when confronted with a strongly non-natural surface pattern (cf. Geirhos and et al., 2020).

3. **Glossy and uniform colours occupy an intermediate regime.** YellowCarPaint (42%), NeonGreen (39%), and White (38%) all replace natural texture with smooth, uniform surfaces. Despite retaining clear object silhouettes, these finishes still degrade recognition relative to the default texture (a drop of 7–11 percentage points). The fact that accuracy remains in the high 30%–40% range indicates that the model can partially recover shape-based signals even under drastic colour changes, but such non-natural colours are still a significant departure from the training distribution.

4. **BrushedMetal causes moderate degradation (32%).** BrushedMetal's fine directional noise introduces some reflections and accuracy on this finish is only 32%—higher than Checkerboard but lower than the uniform colours.

Overall, material alone accounts for a 24-point swing in Top-1 accuracy (from 49% with Default down to 25% with Checkerboard), even though background, viewpoint, lighting, and fog still vary. In other words, texture and reflectance changes impose a far more severe drop than background shifts alone (cf. Table 4), confirming that ConvNeXt-Large—and likely other architectures—remains heavily texture-biased under extreme OOD conditions.

5.3.2 Camera Position

Figure 6 shows ConvNeXt-Large's Top-1 accuracy on the full OOD dataset, broken down by camera position. Each bar aggregates all 86,016 test images that share the same camera translation vector—namely $86{,}016 \div 8 = 10{,}752$ images per position—covering every combination of background, class, mesh, material, light colour, and fog state. From left to right, the eight positions (relative to the object) are:

- `Left (1, 0, 0.2)`
- `Right (-1, 0, 0.2)`
- `Front (0, 1, 0.2)`
- `Back (0, -1, 0.2)`
- `Back Top Left (0.5, -0.5, 1)`
- `Front Top Left (0.5, 0.5, 1)`
- `Back Top Right (-0.5, -0.5, 1)`
- `Front Top Right (-0.5, 0.5, 1)`

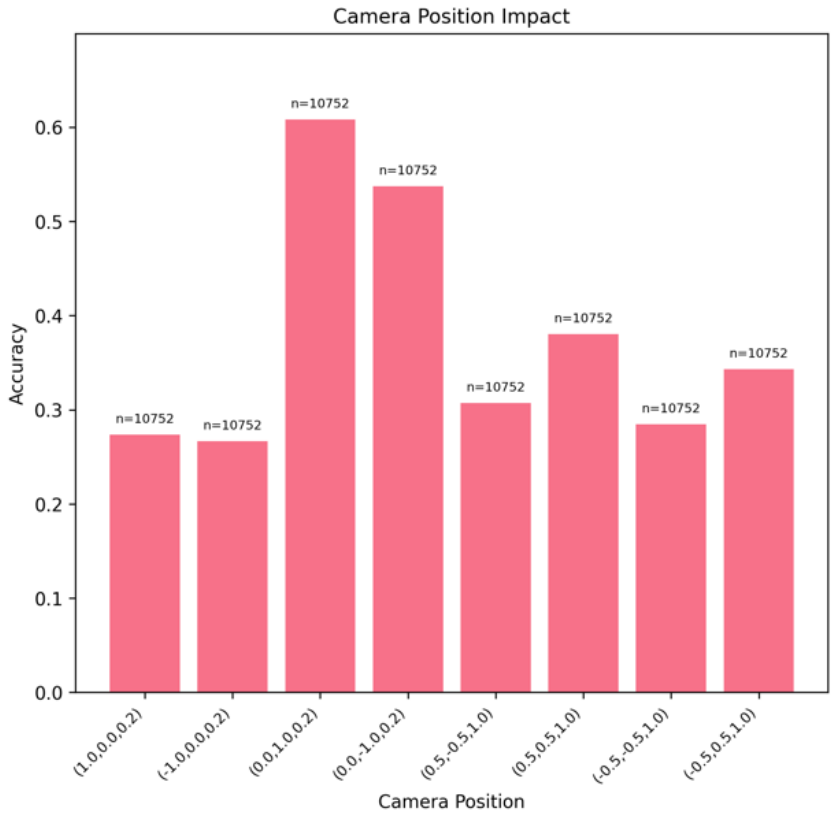

Figure 6: Per-camera-position Top-1 accuracy (ConvNeXt-Large). Each bar aggregates 10,752 test images (all images taken from that camera position across all other factors).

The main observations are:

1. **Front view yields the highest accuracy (61%).** When the camera faces the object directly at `(0, 1, 0.2)`, ConvNeXt-Large correctly classifies 61% of the samples, despite variation in background, material, lighting, and fog. This is expected, since the frontal pose most closely resembles typical training images.

2. **Back view also performs well (54%).** Placing the camera directly behind the object at `(0, -1, 0.2)` preserves a coherent silhouette, and accuracy remains at 54%, indicating that a back-facing view is less detrimental than oblique or elevated angles.

3. **Oblique and elevated positions degrade accuracy.** The lowest accuracy occurs for the left position `(1, 0, 0.2)` (28%) and the right position `(-1, 0, 0.2)` (27%), where the object's frontal features are almost completely occluded. Similarly, the "Back Top Left" position `(0.5, -0.5, 1)` yields only 30%, and "Back Top Right" `(-0.5, -0.5, 1)` yields 29%. Elevated

front views—"Front Top Left" `(0.5, 0.5, 1)` at 40% and "Front Top Right" `(-0.5, 0.5, 1)` at 34%—also underperform the purely horizontal front view, showing that height offsets harm recognition even when the camera remains aligned with the object's front.

4. **Symmetry between left/right positions.** Side views on the horizontal plane yield nearly identical performance: left at 28% and right at 27%. This indicates no significant bias toward one lateral direction, confirming that the model's performance depends primarily on how much of the object's salient features are visible rather than any inherent asymmetry.

In summary, camera poses that preserve a clear, frontal or rear silhouette (Front and Back) maintain relatively high accuracy, while side and elevated angles substantially reduce performance. This demonstrates that viewpoint changes—particularly those that obscure the object's most discriminative faces—are a major contributor to the overall OOD drop, reinforcing the need for models that capture three-dimensional shape information robustly under diverse viewpoints.

5.3.3 Light Color

Figure 7 shows ConvNeXt-Large's Top-1 accuracy on the full OOD dataset, broken down by light colour. Each bar aggregates all 86,016 test images that share the same light colour—namely $86{,}016 \div 4 = 21{,}504$ images per colour—covering every combination of background, class, mesh, material, camera position, and fog state. From left to right, the four light colours are:

- `RGB(255,255,255)` (white light),
- `RGB(255,0,0)` (red light),
- `RGB(0,255,0)` (green light),
- `RGB(0,0,255)` (blue light).

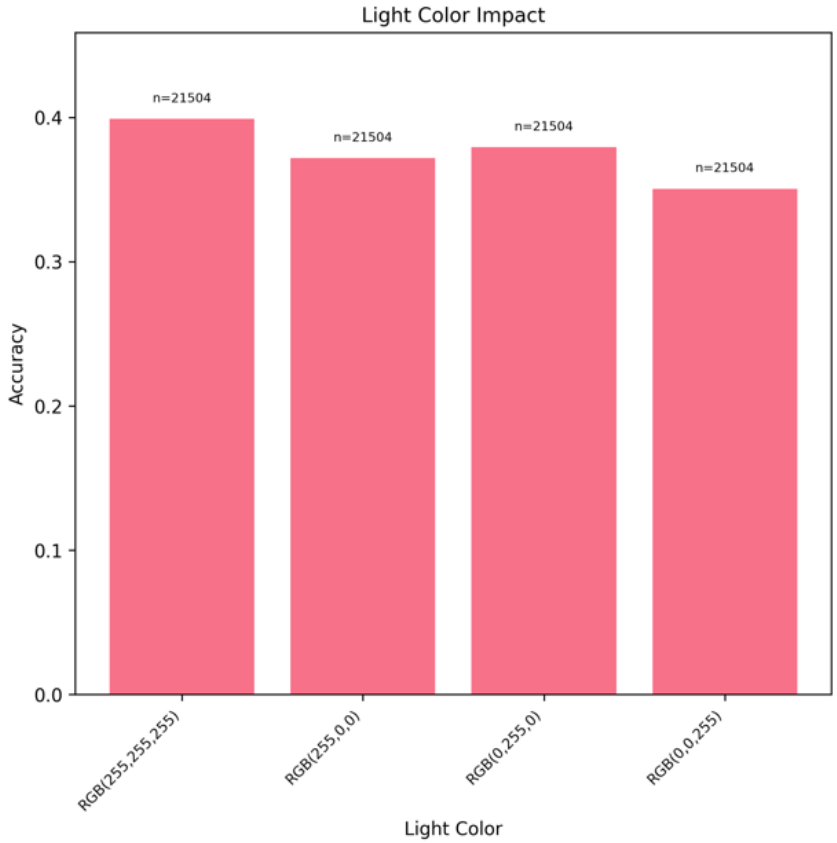

Figure 7: Per-light-colour Top-1 accuracy (ConvNeXt-Large). Each bar aggregates 21,504 test images (all images illuminated by that colour across all other factors).

The main observations are:

1. **White light yields the highest accuracy (40%).** Under standard white illumination `(255,255,255)`, ConvNeXt-Large correctly classifies 40% of the samples, despite variation in background, material, camera pose, and fog. This suggests that the network's features transfer most effectively when the lighting closely matches natural, balanced conditions.

2. **Green light performs nearly as well (38%).** Illuminating the scene with `(0,255,0)` causes a drop of 2 percentage points in accuracy compared to white light, indicating that moderate shifts toward a single colour channel can be partially tolerated by the model's learned representations.

3. **Red light causes a slightly larger drop (37%).** Under `(255,0,0)`, accuracy further decreases to 37%. The heavier skew toward the red channel disrupts object texture and shading cues more than green light does, suggesting that the model relies on a balanced RGB distribution for reliable recognition.

4. **Blue light has the strongest adverse effect (35%).** When illuminated by `(0,0,255)`, accuracy falls to 35%, the lowest among the four colours. This

indicates that down-weighting both red and green channels most severely distorts natural object appearance, making classification particularly challenging.

Most architectures follow this trend, with white light giving the highest accuracy and blue light the lowest. An exception is Swin-B: on that model, red illumination yields 30% accuracy whereas white light yields only 29%, indicating a reversed preference under its learned features.

Overall, light colour alone accounts for a 5-point swing in Top-1 accuracy (from 40% under white light down to 35% under blue light) for ConvNeXt-Large, even though background, material, camera pose, and fog still vary. This confirms that drastic shifts in illumination colour degrade performance and motivates further exploration.

5.3.4 Background (Level)

Figure 8 shows ConvNeXt-Large's Top-1 accuracy on the full OOD dataset, broken down by level (background scene). Each bar aggregates all 86 016 test images that share the same level—namely $86{,}016 \div 8 = 10{,}752$ images per level—covering every combination of class, mesh, material, camera position, light color, and fog state. From left to right, the eight levels (with their main characteristics) are:

- `UtopiaCityFountain`: a city plaza with a central fountain and complex architecture and lighting.
- `UtopiaCityUnderpass`: an urban underpass with detailed geometry, mixed shadows, and artificial lighting.
- `QuarrySlate`: an open-air slate quarry with uneven stone surfaces and natural shadows.
- `Demo_gallery`: an open space with a wall to one side and the object on a small podest, unique lighting.
- `Sahara_Desert`: a barren desert map with rolling dunes and minimal visual clutter.
- `Salt_Plane`: an endless flat plane of uniform salt-colored ground and sparse horizon.
- `Korean_Temple`: a courtyard in front of a traditional temple, featuring ornate structures and varied shadows.
- `Desert_Dunes_Foliage`: a desert terrain similar to Sahara_Desert but with scattered foliage and occasional vegetation.

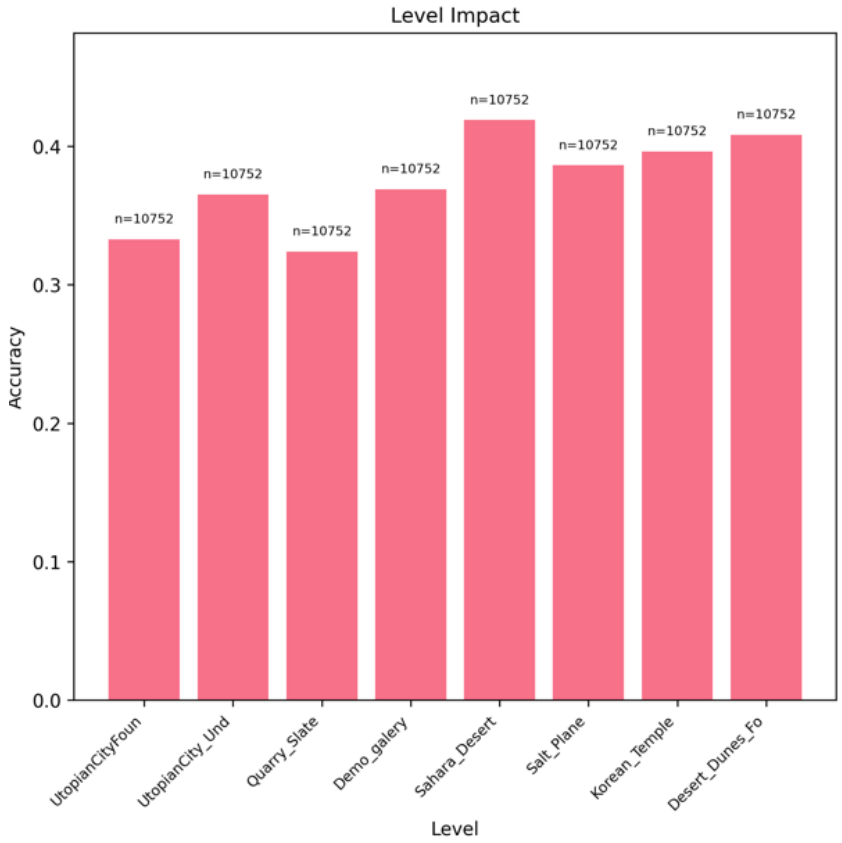

Figure 8: Per-level Top-1 accuracy (ConvNeXt-Large). Each bar aggregates 10 752 test images (all images taken in that level across all other factors).

Several observations emerge:

1. **Highest accuracy on open deserts with foliage and dunes (41–42%)** ConvNeXt-Large attains its best performance on `Desert_Dunes_Foliage` (41%) and `Sahara_Desert` (42%), likely because these scenes lack complex occluders and present clear object silhouettes against a largely uniform background.

2. **Moderate performance on salt flats and temple courtyard (39–40%)** On `Salt_Plane`, accuracy is 39%, reflecting the challenge of a completely featureless ground plane that removes contextual cues. Korean_Temple yields 40%, indicating that while the ornate architecture introduces clutter, the temple's structured geometry and lighting still preserve recognizable object boundaries.

3. **Lower accuracy in complex scenes (32–37%)** Quarry_Slate (32%) and UtopiaCityFountain (33%) produce the worst accuracies, as uneven surfaces, irregular shadows, and complex background textures compete with the object's appearance. Demo_gallery (37%) and UtopiaCityUnderpass (35%) fall in between: lighting in Demo_gallery is simpler than urban underpass shadows in UtopiaCityUnderpass, leading to relatively higher performance.

4. **ConvNeXt-Large is relatively balanced** The accuracy range across all eight levels spans only 10 percentage points (32–42%), demonstrating that this model's learned features generalize more uniformly across diverse backgrounds. By contrast, most other architectures exhibit more pronounced spikes—often achieving unusually high accuracy on Sahara_Desert or Salt_Plane—while suffering larger drops in cluttered urban or slate quarry scenes. These spikes and drops indicate that weaker models rely more heavily on scene simplicity or uniformity.

In summary, background complexity and clutter significantly affect recognition: barren or foliage-dotted deserts allow the model to rely on clean silhouettes, whereas urban underpasses, quarry environments, and indoor galleries introduce textures and shadows that compete with object features. ConvNeXt-Large's relatively small variance in accuracy across levels suggests it learns more robust background-invariant representations compared to other models, which show stronger sensitivity to specific scenes.

5.3.5 Fog

Figure 9 shows ConvNeXt-Large's Top-1 accuracy on the full OOD dataset, broken down by the presence or absence of volumetric fog. Since fog is a binary toggle, each of the 86,016 test images exists in two versions—one without fog and one with fog—yielding $86{,}016 \div 2 = 43{,}008$ images. Despite adding a potentially confounding visual effect, every model—including ConvNeXt-Large—performs *better* when fog is enabled.

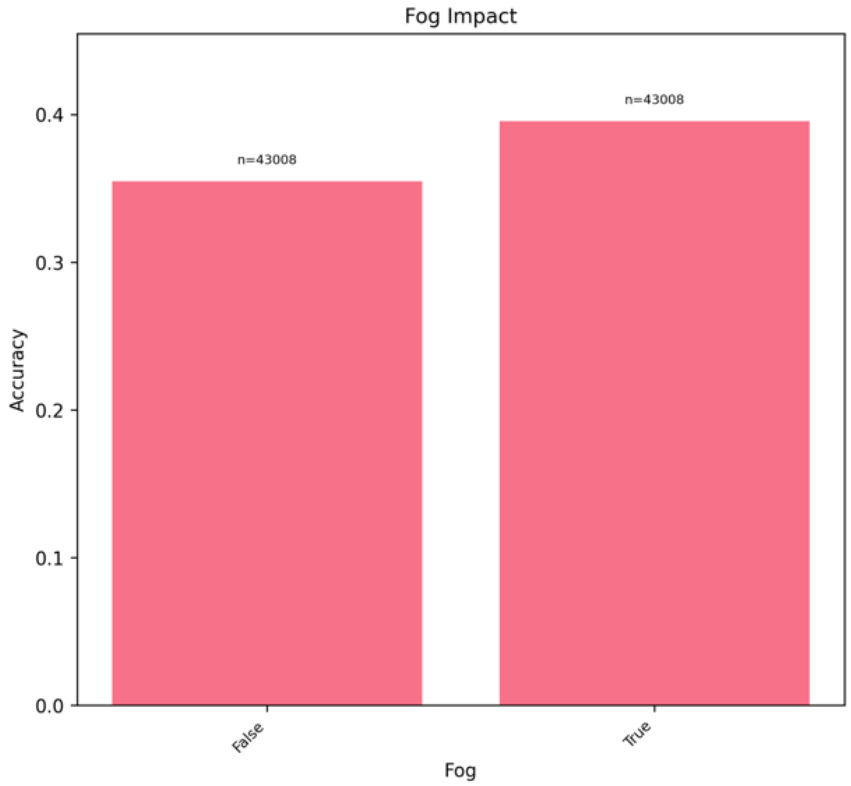

Figure 9: Top-1 accuracy with and without fog (ConvNeXt-Large). Each bar aggregates 43,008 test images (all images sharing that fog setting across all other factors).

In particular, ConvNeXt-Large's accuracy rises from approximately 36% without fog to around 40% when fog is present. We hypothesize that this counterintuitive improvement occurs because fog effectively blurs and desaturates the distant background, thereby reducing background clutter and color distractions. As a result, the object—being the closest element to the camera—stands out more strongly in its silhouette. Under heavy fog, fine-grained background textures and lighting variations become less salient, forcing the network to rely more on coarse shape and boundary cues. This shift towards silhouette-based recognition appears to benefit all architectures in our benchmark, since they were trained on natural images where object shapes are often delineated against relatively uncluttered backgrounds. Thus, fog acts as a form of "background simplification," improving performance even though it occludes some object detail. One main concern is, that the perceived fog density in a scene is tied to the lighting inside of it. While every image uses the same fog density settings, differences in lighting lead to some instances where the fog barely obscures the object but noticeably obscures the background, and others where the image is almost fully concealed, with only a faint silhouette of the object visible.

Because fog is the only factor toggled in this experiment, the accuracy gain could demonstrate that reducing background complexity can sometimes be more beneficial than preserving full object fidelity, although this hypothesis requires further testing.

6 Discussion

6.1 Key Findings

Our evaluation shows that even state-of-the-art architectures trained on ImageNet-1k suffer dramatic performance drops when exposed to the Cartesian product of six generative factors (background, material, light colour, camera pose, fog, and mesh variation). No model approaches in-distribution accuracy, and the best performer—ConvNeXt-Large—achieves only 37.5% Top-1 on the OOD dataset (Table 3). This gap of roughly 45-50 percentage points relative to standard ImageNet-1k validation scores (80-85%) highlights the extreme difficulty of our benchmark compared to prior single-factor synthetic datasets like PUG:ImageNet, where Top-1 accuracies typically remain above 50% for similar architectures (Bordes et al., 2023).

Across all eight models, accuracy and rejection-aware AUC track closely: higher Top-1 correlates with higher AUCRA, but improvements in calibration are modest compared to gains in accuracy. Classical CNNs (ResNet-50/101, DenseNet-201) suffer the steepest collapse (12-23% Top-1, AUC 0.616-0.707), reaffirming their strong texture bias (Geirhos and et al., 2020). Modern ConvNets (ConvNeXt) outperform older CNNs by 10-14 percentage points in Top-1 and 0.10 in AUCRA, demonstrating that wider receptive fields and stronger regularization confer better robustness to joint distribution shifts (Liu et al., 2022). Vision Transformers (ViT-B/16, ViT-L/16) exhibit intermediate accuracy (17-26%) but competitive AUCRA (0.659-0.724), indicating more reliable uncertainty estimates even as Top-1 remains lower than ConvNeXt-Large. The hybrid Swin-B (28%, AUCRA 0.744) further illustrates that hierarchical attention can mitigate some view- and context-related failures (Liu et al., 2021).

Our factor-wise breakdowns reveal that material changes alone can induce a 24 percentage-point swing in Top-1 (from 49% with "Default" textures down to 25% with "Checkerboard"; Figure 5), and that extreme viewpoint shifts (e.g., side elevations) reduce accuracy from 61% (frontal) to 27-34% (side or elevated views; Figure 6). Light-colour variations cause a smaller but still significant drop (40% under white to 35% under blue; Figure 7), while background complexity yields a 10 point spread (32% in cluttered scenes to 42% in open deserts; Figure 8). Finally, the presence of fog unexpectedly improves performance (36% without fog vs. 40% with fog; Figure 9), supporting the notion that occluding background clutter can enhance silhouette-based recognition under severe OOD conditions.

Together, these results confirm that compounding multiple shifts overwhelms even the most advanced convolutional and transformer architectures. In-distribution accuracy proves a poor predictor of OOD robustness, and no single model fully mitigates simultaneous perturbations along texture, illumination, viewpoint, and context axes. The subsequent sections will analyse each factor's interaction effects in greater detail.

6.1.1 Texture Bias and Material Effects

Material variations produce the largest single-factor performance swings in our benchmark. Figure 5 shows that ConvNeXt-Large's Top-1 accuracy drops from 49% with the "Default" textures to 25% with the "Checkerboard" pattern—a 24 percentage-point decline—even though background, viewpoint, lighting, and fog continue to vary. The checkerboard pattern replaces all natural surface cues (for example, wood grain or metal finish) with a high-contrast grid, forcing the network to rely exclusively on object shape. The severe decline confirms that even a high-capacity ConvNet remains heavily biased toward texture cues, in line with prior work demonstrating that ImageNet-trained CNNs prioritize texture over shape (Geirhos and et al., 2020).

Uniform colour treatments (YellowCarPaint at 42%, NeonGreen at 39% and White at 38%) also degrade accuracy relative to the default textures, but to a lesser extent. These solid surfaces preserve object outlines while eliminating fine texture details. The fact that ConvNeXt-Large maintains nearly 40% accuracy under uniform colours indicates that shape information alone can partially support recognition when texture is unavailable. However, the 7–10 percentage-point gap versus the default condition demonstrates that trained features still rely on natural texture statistics.

Together, these results confirm that material changes reveal a fundamental texture bias: most architectures achieve their highest accuracy when natural surface textures are preserved, and they perform worst when textures are replaced by non-natural patterns. Even Vision Transformers strongly exhibit pattern (Figures 101112), indicating that patch-based tokenization does not eliminate texture reliance (Naseer and et al., 2021). These findings suggest that future robust models must incorporate mechanisms to disentangle shape from texture or to learn more shape-focused representations under extreme surface perturbations.

6.1.2 Viewpoint Sensitivity

Camera pose exerts a strong influence on recognition performance, as Figure 6 illustrates. ConvNeXt-Large's Top-1 accuracy is highest when the object is viewed head-on (Front: 61%) and directly from behind (Back: 54%), but it falls sharply when the camera moves to lateral or elevated angles - this is a behavior displayed by by all tested models. Side views (Left: 28%, Right: 27%) yield the lowest accuracies, since key frontal features are largely occluded. Elevation exacerbates this effect: "Back Top Left" (30%) and "Back Top Right" (29%) underperform the purely horizontal back view, while "Front Top Left" (40%) and "Front Top Right" (34%) still lag behind the horizontal front pose.

These results indicate that models trained on ImageNet-1k rely heavily on canonical, frontal perspectives and struggle to generalize when salient object faces are partially or fully hidden. The back view retains a coherent silhouette, explaining why accuracy remains above 50%. By contrast, side views disrupt the typical alignment of

object features, forcing the network to infer object identity from less familiar silhouettes. Elevated poses introduce additional occlusion of horizontal details and alter shading patterns, further degrading performance.

Our findings align with prior observations that synthetic viewpoint shifts can reveal a model's limited three-dimensional understanding (Bordes et al., 2023). Even Vision Transformers (A) exhibit similar trends, confirming that neither patch-based embeddings nor windowed attention fully mitigate viewpoint-induced errors. In practice, ensuring robust recognition across diverse camera angles may require augmentations that explicitly sample oblique and elevated views or architectural modules that encode 3D shape more directly.

6.1.3 Illumination Shifts

Figure 7 reports ConvNeXt-Large's Top-1 accuracy under four distinct light colours: white (40%), green (38%), red (37%), and blue (35%). White illumination produces the highest accuracy, consistent with training images that typically feature balanced, full-spectrum lighting. Shifting to green light (`(0,255,0)`) causes a modest 2 percentage-point drop, indicating that the network's learned features can tolerate some single-channel bias. Under red light (`(255,0,0)`), accuracy declines further to 37%, suggesting that overemphasising the red channel distorts texture and shading cues more severely. Blue illumination (`(0,0,255)`) yields the lowest accuracy (35%), implying that suppressing both red and green channels most significantly disrupts natural color statistics.

These trends align with evidence on color constancy: convolutional features trained on ImageNet rely on balanced RGB distributions, and extreme channel shifts degrade performance (Taori et al., 2020). Notably, Swin-B deviates from this pattern: red light produces 30% Top-1, while white light yields only 29% 12. This reversal suggests that Swin-B's windowed self-attention may exploit red-channel cues under certain conditions, a behaviour not seen in ConvNeXt, classical CNNs and other ViTs.

Overall, illumination shifts produce a 5 point swing in ConvNeXt-Large's Top-1 accuracy. Although smaller than the material-induced variation, this effect demonstrates that extreme light colours alone can significantly erode recognition. Mitigating such sensitivity may require augmentations that simulate diverse spectral conditions or architectural adaptations that explicitly normalise colour channels.

6.1.4 Background Complexity (Level) Effects

Our results confirm that background complexity strongly influences recognition performance under OOD conditions. ConvNeXt-Large's Top-1 accuracy varies from 32% in cluttered urban/quarry scenes (Quarry_Slate and UtopiaCityFountain) up to 42% in open desert scenes (Sahara_Desert and Desert_Dunes_Foliage). This ten-point spread underscores how uniform or sparse backgrounds—such as endless dunes

or salt flats—reduce interference from competing textures, allowing models to focus on object silhouettes. In contrast, backgrounds featuring irregular geometry (urban underpasses, indoor galleries, slate quarries) introduce shadows and textural clutter that compete with object features, driving accuracy downward.

That ConvNeXt-Large maintains relatively stable accuracy suggests its larger receptive fields and stronger regularization partially mitigate context-driven errors (Liu et al., 2022). Nevertheless, even ConvNeXt-Large experiences a 10% drop between deserts and urban scenes, indicating that no architecture fully resolves background sensitivity under extreme shifts. This finding extends insights from PUG:ImageNet, where single-factor background changes caused degradation; in our combined setting, background remains a critical barrier—even when materials, lighting, and viewpoint also vary (Bordes et al., 2023).

Overall, the background results emphasize the need for either architectural mechanisms that explicitly disentangle foreground and background or training regimes that expose models to diverse context permutations. For future work, targeted augmentations (e.g. random background replacement) or modules for foreground segmentation could help reduce background-driven failures in OOD scenarios.

6.1.5 Fog as a Background Simplifier

The unexpected improvement in accuracy when fog is present (e.g. ConvNeXt-Large rising from 36% without fog to 39% with fog; Figure 9) indicates that fog functions as an implicit background simplifier. By reducing the visibility of distant background details and desaturating textures, fog effectively masks competing contextual cues, forcing the model to rely more heavily on the foreground silhouette. This would support the notion that CNNs and Transformers trained on ImageNet often leverage background patterns as shortcuts for recognition (Geirhos and et al., 2020; Radford et al., 2021).

From a theoretical perspective, fog introduces a form of low-pass filtering on the scene. Background textures, shadows, and high-frequency variations become blurred and less informative, which reduces the risk of the network latching onto non-essential details. At the same time, the object, being closest to the camera, remains relatively sharp. Consequently, the contrast between the object and its surroundings increases, making boundary detection easier for convolutional filters or attention heads.

Practically, this suggests that strategic use of background simplification—such as partial masking, selective blurring, or simulated fog—could could serve as a data augmentation or pre-processing technique to enhance robustness under distribution shifts. For instance, training with random low-opacity fog overlays might encourage models to learn more shape-centric features rather than overfitting to complex background textures. However, excessive fog density would eventually degrade performance by obscuring object details; identifying an optimal balance is an open question.

In summary, our fog results indicate that reducing background complexity could outweigh the loss of object detail in extreme OOD scenarios. Due to the fact that consistent fog density could not be ensured, this requires further testing.

6.2 Limitations

While our benchmark reveals important weaknesses in modern vision models, it has several limitations:

1. **Limited class and mesh diversity** We evaluate only seven object categories, each with four 3D mesh variants. Though this design ensured a balanced Cartesian product of factors, it restricts semantic coverage. Real-world objects exhibit far greater intra-class variation (materials, shapes, subtypes), so our findings may not generalize to more diverse object sets.

2. **Fixed camera intrinsics and simplified viewpoints** We vary only camera position (translation) but keep focal length and camera roll constant. As a result, our viewpoint analysis omits changes in zoom or rotational angles, which could further challenge models. Incorporating these parameters—along with articulated poses or dynamic object orientations—would yield a more complete view of viewpoint robustness.

3. **Uniform fog density** While we toggle fog as a binary condition with a fixed density, unique factors to each scene impact the perceived density in the final image. This does not allow full control over the final image.

4. **Discrete lighting conditions** Our illumination factors are limited to four extreme, monochromatic colours (white, red, green, blue). Though sufficient to reveal sensitivity to spectral shifts, these settings do not encompass the full range of real-world lighting (e.g. directional sunlight, mixed indoor/outdoor lighting, colored shadows). More nuanced illumination models—varying intensity, direction, and colour temperature—could uncover additional failure modes.

5. **Static background scenes** We render eight fixed environments, covering urban, desert, quarry, and indoor settings. These backgrounds are representative but not exhaustive. Real scenes exhibit continuous variation (weather, time of day, moving objects), and dynamic backgrounds (e.g. crowds, traffic) could introduce further context challenges.

6. **Synthetic-to-real domain gap** Although we use high-quality UE5 assets, synthetic images may not fully capture real-world variations (material reflectance, micro-textures, sensor noise). Consequently, performance measurements on our benchmark do not guarantee analogous robustness on real-world distribution shifts. Bridging this gap would require mixed synthetic–real training or domain adaptation techniques.

7. **Plugin Scalability** While the plugin is theoretically capable of rendering datasets with hundreds of objects and scenes, some issues like the ones mentioned here 4.2.3 would need to be resolved in order to allow for optimal performance.

By acknowledging these limitations, we clarify the scope of our conclusions and motivate future extensions that expand object categories, vary camera intrinsics, sample continuous fog and lighting parameters, and introduce more realistic, dynamic environments.

6.3 Directions for Future Work

Several avenues remain to extend this synthetic OOD benchmark and advance robust model design:

1. **Expand class and mesh diversity** Increase the number of object categories beyond the current seven, and include more mesh variants per class. Greater semantic coverage and intra-class variation—such as different subtypes of tools, electronics, or household items—would test whether the trends observed here hold for a broader range of objects. A larger mesh pool could also reduce semantic collapse (e.g. when texture changes make "banana" look like a generic elongated object).

2. **Incorporate additional camera intrinsics and viewpoints** Vary focal length, field of view, and camera roll to simulate zooming and tilt. Introduce continuous rotations around all three axes, rather than only fixed translation offsets. Include articulated or non-planar object poses (e.g. rotating a hammer or tilting a cup) to measure how well models generalize to unseen object orientations.

3. **Vary fog density and volumetric effects** Instead of a binary toggle, sample multiple fog densities to model real-world visibility ranges (light mist, moderate fog, heavy haze). Experiment with different particle scattering parameters and ambient light shifts within fog to evaluate how subtle versus extreme occlusion affects recognition. Explore other occluding media (e.g. rain, snow, dust) to determine whether the background-simplification benefit extends beyond fog.

4. **Refine illumination models** Move beyond four monochromatic lights by sampling a continuous spectrum of colour temperatures (warm to cool), intensities, and directions. UE5 offers full dynamic control over the time of day and sun position, opening the door for a lot of possibilities to simulate real-world scenarios.

5. **Introduce dynamic and stochastic backgrounds** Supplement static level scenes with moving elements such as pedestrians, vehicles, or flickering lights. Simulate weather changes (e.g. rain, snow) or time-of-day transitions within a scene. Due to the fact that UE5 is a game engine, all of this can be achieved trough its in-built feature set.

6. **Bridge synthetic and real domains** Evaluate whether fine-tuning on synthetic OOD data improves performance on real-world distribution shifts. Investigate domain adaptation or domain randomization techniques that combine synthetic renders with real photographs, aiming to reduce the synthetic-to-real gap.

By pursuing these directions, future research can build on the insights gained from our full-factor OOD dataset, ultimately moving toward vision models that maintain high performance under extreme, real-world distribution shifts.

7 Conclusion

In this thesis, we introduced a modular UE5 plugin that generates fully controlled, photorealistic image datasets with exhaustive variable annotations inspired by Bordes et al. (2023) while at the same time increasing render performance by a factor of 16. By open sourcing the plugin and the dataset, we enable researchers to reproduce and extend our OOD benchmark. Using this tool, we built a dataset of seven ImageNet-compatible classes and 86,016 images, varying six generative factors simultaneously: background, material, light colour, camera pose, fog, and mesh variation.

Our evaluation reveals that state-of-the-art models trained on ImageNet-1k collapse under these combined shifts: even the best performer (ConvNeXt-Large) achieves only 37.5% Top-1 accuracy. Factor-wise analyses show that material changes alone can induce a 24 percentage-point swing, extreme viewpoints reduce accuracy by over 30 points, and background complexity results in a 10 point spread. Unexpectedly, adding fog improves performance by masking background clutter, which highlights how precise control and annotation of each factor allow us to pinpoint specific weaknesses in model representations.

By demonstrating that a dataset with full semantic control and exact metadata can expose vulnerabilities that single-factor benchmarks might miss, we underscore the importance of synthetic pipelines for rigorous OOD evaluation. The plugin we developed, now publicly available, offers a foundation for future work: researchers can scale class diversity, introduce new factors, or investigate architectural remedies. Ultimately, the combination of an open-source generation tool and a richly annotated benchmark will help guide the development of vision models that remain robust under extreme distribution shifts.

A Appendix

All source code, the plugin and the dataset can be found here: `https://github.com/FlGutbier/UE5DatasetGenerator`.

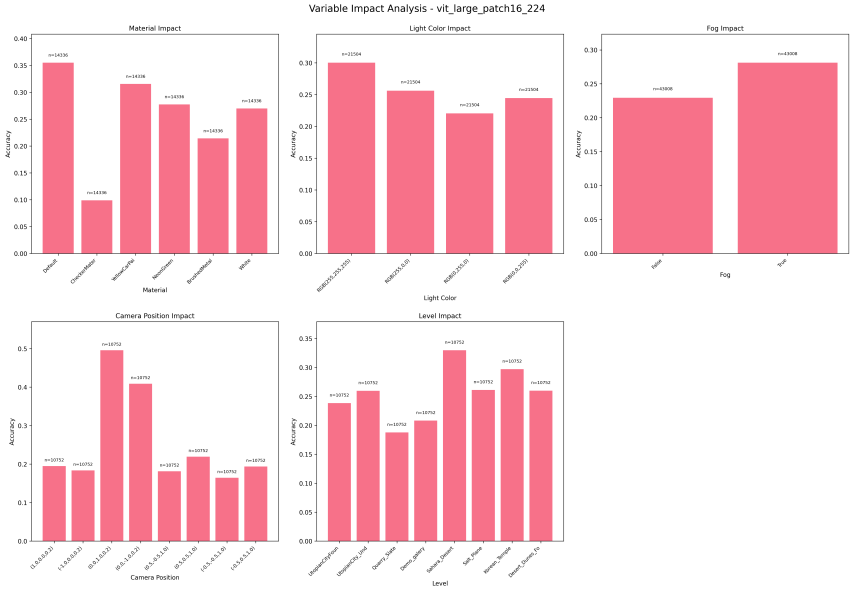

Figure 10: ViTL16 factor analysis.

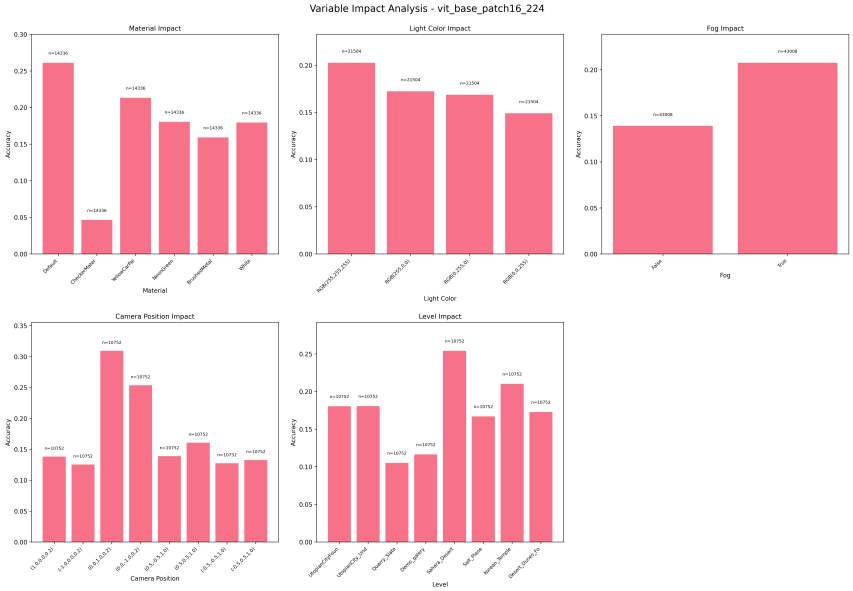

Figure 11: ViTB16 factor analysis.

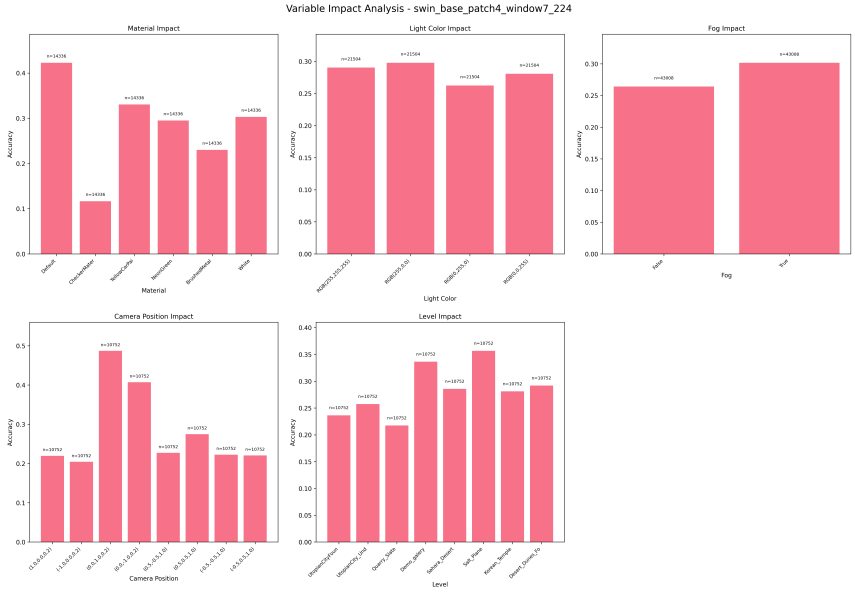

Figure 12: SwinB factor analysis.

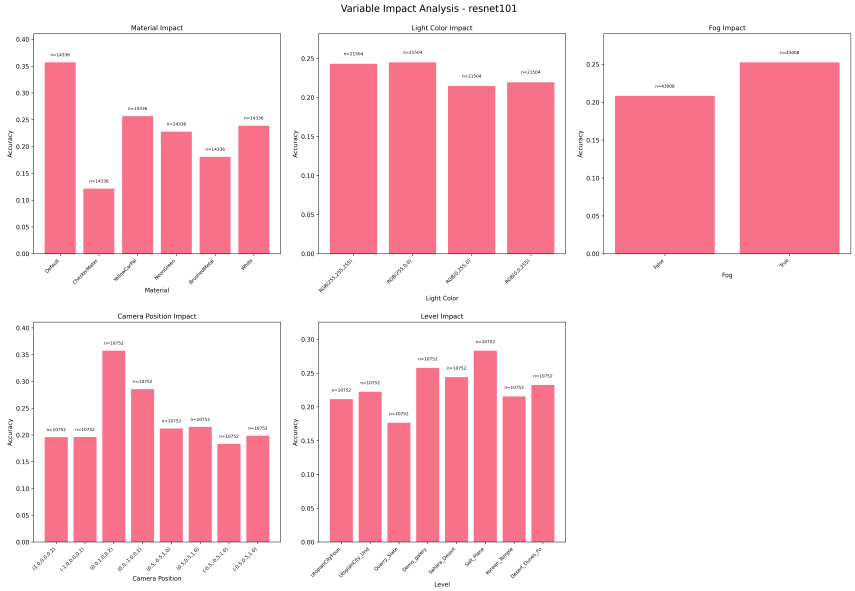

Figure 13: Resnet101 factor analysis.

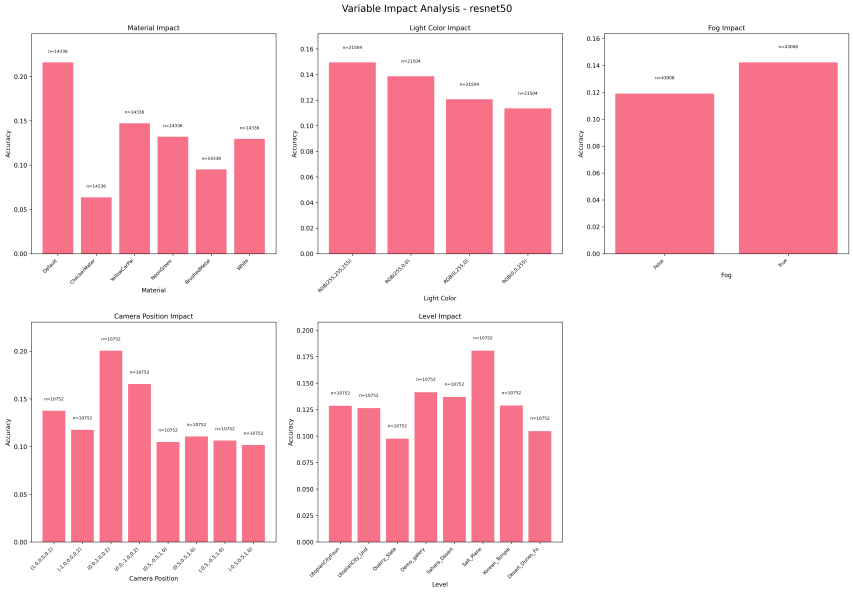

Figure 14: Resnet50 factor analysis.

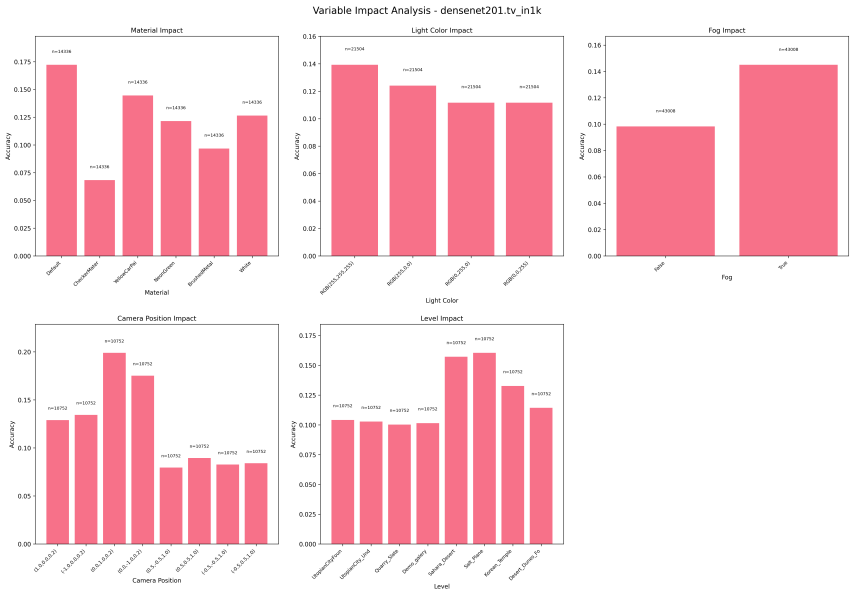

Figure 15: Densenet factor analysis.

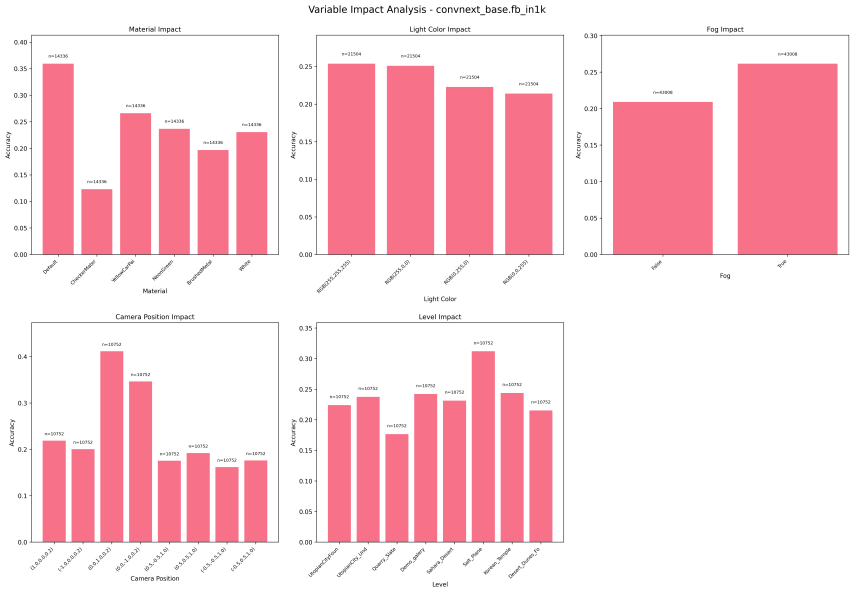

Figure 16: Convnext base factor analysis.

Bibliography

John W Anderson et al. Synthetic image data for deep learning. *Journal of Artificial Intelligence Research*, 68:487–518, 2020. doi: 10.1613/jair.1.12159.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 9448–9458. Curran Associates, Inc., 2019. URL `https://papers.nips.cc/paper/9142-objectnet-a-large-scale-bias-controlled-dataset-for-pushing-the-limits-of`

Apratim Bhattacharyya, Theodore J. Dunn, Yonghee Song, and Adrien Gaidon. Synthdet: Scaling synthetic data generation for object detection. *arXiv preprint arXiv:2106.09965*, 2021. Introduces the 800 k-image retail dataset and reports 72.9 mAP vs. 73.1 mAP for a 100 k real-image baseline.

Stan Birchfield, Shan E. Ahmed Raza, Quoc Dang Vu, and Simon Graham. Clevrtex: A texture-rich benchmark for uncovering the biases of convolutional networks. *arXiv preprint arXiv:2011.05359*, 2021.

Florian Bordes et al. Pug: Photorealistic and semantically controllable synthetic data for representation learning. In *Advances in Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2023. To appear.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Maximilian Denninger, Maximilian Langer, Nicolas Dittes, and *et al.* Blenderproc: Reducing the reality gap by photorealistic image synthesis. In *International Conference on Robotics and Automation Workshops*, 2020.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, pages 1–16, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16×16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. arXiv:2010.11929.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Theodore J. Dunn, Jeannette Bohg, Gavin Faigin, Egor Merkurjev, and Lerrel Pinto. Unity perception: A data generation framework for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 547–556, 2021. Describes the open-source Unity Perception Toolkit, its domain-randomisation API and 100 FPS throughput on a single GPU.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Jason Altschuler, Sam Barrett, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Epic Games. Unreal build tool in unreal engine. Online documentation, 2024. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-build-tool-in-unreal-engine`.

Epic Games. Actors in unreal engine 5.5 documentation. Online documentation, 2025a. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/actors-in-unreal-engine`.

Epic Games. Blueprints visual scripting in unreal engine. Online documentation, 2025b. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine`.

Epic Games. Fab documentation. Online documentation, 2025c. URL `https://dev.epicgames.com/documentation/en-us/fab/fab-documentation`.

Epic Games. Hardware ray tracing in unreal engine. Online documentation, 2025d. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-ray-tracing-in-unreal-engine`.

Epic Games. Static mesh actors in unreal engine. Online documentation, 2025e. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/static-mesh-actors-in-unreal-engine`.

EpicGames. Unreal engine 5: Real-time graphics redefined. `https://www.unrealengine.com/en-US/unreal-engine-5`, 2022. White paper describing Lumen, Nanite and Quixel integration.

EpicGames. World partition in unreal engine. Online documentation, 2025. URL `https://dev.epicgames.com/documentation/en-us/unreal-engine/world-partition-in-unreal-engine`.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8): 861–874, 2006.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. doi: 10.1007/BF00344251.

Robert Geirhos and et al. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations (ICLR)*, 2020.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Pedro M Gonzalez et al. Unrealrox+: An improved tool for acquiring synthetic data from virtual 3d environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1240. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9562034.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Simon Graham, Quoc Dang Vu, Shan E. Ahmed Raza, and *et al.* Dense prediction transformers for large-scale histopathology image analysis. *Medical Image Analysis*, 72:102203, 2021. doi: 10.1016/j.media.2021.102203.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Machine Learning*, pages 1772–1782, 2020.

David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Irina Higgins, Loic Matthey, Arka Pal, and *et al.* beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017. Introduces the dSprites dataset.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910, 2017.

Brian Karis. Nanite: a deep dive. In *Advances in Real-Time Rendering in Games, SIGGRAPH Course*, 2021. URL `https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf`.

Brian Karis, Daniel Nichols, and Thomas Majercik. Lumen in the land of nanite: Global illumination in unreal engine 5. In *ACM SIGGRAPH Courses*, 2021. Explains UE5's real-time GI and micropolygon pipelines.

Pang Wei Koh, Shiori Sagawa, Huan Marklund, Yew Siang Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Jeremy Bernstein, Emma Xu, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, and *et al.* A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. doi: 10.1016/j.media.2017.07.005.

Ze Liu, Yutong Lin, Yixuan Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022.

Zhuangkun Liu, Gang Niu, and Masashi Sugiyama. A survey on out-of-distribution generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5079–5102, 2023. doi: 10.1109/TPAMI.2023.3234081.

Hiba S Malik et al. Objectcompose: Evaluating resilience of vision-based models on object-to-background compositional changes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1046–1055, 2021. URL `https://openaccess.thecvf.com/content/CVPR2021/papers/Malik_ObjectCompose_Evaluating_Resilience_of_Vision-Based_Models_on_Object-to-Background_Compositional_Changes_CVPR_2021_paper.pdf`.

Pablo Martinez-Gonzalez, Sergiu Oprea, Alberto Garcia-Garcia, Alvaro Jover-Alvarez, Sergio Orts-Escolano, and Jose Garcia-Rodriguez. UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*, 24(2):271–288, 2020.

Pablo Martinez-Gonzalez, Sergiu Oprea, John A. Castro-Vargas, Alberto Garcia-Garcia, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Markus Vincze. UnrealROX+: An improved tool for acquiring synthetic data from virtual 3d environments. In *Proc. of Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–8, 2021.

Nathan Morrical, Ameya Harish, Xiaoming Ma, and *et al.* Nvisii: A flexible gpu-accelerated tool for photorealistic image synthesis. In *IEEE International Conference on Image Processing*, pages 231–235, 2021.

David Mulero-Pérez et al. Unrealfall: Overcoming data scarcity through generative models. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024. To appear.

Muhammad Muzammal Naseer and et al. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Yaniv Ovadia and et al. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

Yaniv Ovadia, Emily Fertig, Jimmy Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13991–14002, 2019.

Thomas Pollok et al. Unrealgt: Using unreal engine to generate ground truth datasets. In *International Symposium on Visual Computing*, pages 429–440. Springer, 2019. doi: 10.1007/978-3-030-33720-9_38.

Joaquín Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, 2009.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Dense prediction transformers for monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5962–5972, 2021. doi: 10.1109/ICCV48922.2021.00590.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. URL `http://proceedings.mlr.press/v97/recht19a.html`.

Maksim Riabtsev, Danila Krylov, Alexander Buslaev, and *et al.* Shift: A synthetic driving dataset for global structure and domain shift. *arXiv preprint arXiv:2008.09511*, 2020.

Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proc. European Conf. Computer Vision (ECCV)*, pages 102–118, 2016.

German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Alex Krizhevsky, Ilya Sutskever, Sheng Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael *et al.* Bernstein. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. In *Robotics: Science and Systems (RSS)*, pages 48–55. Robotics: Science and Systems Foundation, 2017. doi: 10.15607/RSS.2017.XIII.034.

Mert Bulent Sariyildiz, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8009–8021. IEEE, 2023. URL `https://openaccess.thecvf.com/content/CVPR2023/papers/Sariyildiz_Fake_It_Till_You_Make_It_Learning_Transferable_Representations_From_CVPR_2023_paper.pdf`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. arXiv:1409.1556.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019. arXiv:1905.11946.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. 2020. URL `https://arxiv.org/abs/2007.00644`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, Herve Jegou, and Armand Joulin. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021.

Jonathan Tremblay, Thang To, Debajyoti Mondal, Éric Marchand, Éliezer Ordonez, and Stan Birchfield. Falling things: A synthetic dataset for 3D object detection and pose estimation. In *Proc. of CVPR Workshops (CVPRW)*, 2018.

Ahmet Turkcan, Selim Yildiz, Raj Patel, and Andrew Smith. City sample: A high-fidelity unreal engine 5 dataset for urban orthophoto detection. *arXiv preprint arXiv:2305.12345*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C. Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, 2018.

Lin Wang, Ziyi Jiang, Lerrel Pinto, and Adrien Gaidon. Farmvision: Synthetic crop-disease imagery for robust plant pathology recognition. *Computers and Electronics in Agriculture*, 198:107093, 2022. doi: 10.1016/j.compag.2022.107093. Presents the 2 M-image Unity dataset and shows a 12 pp F1 improvement when blending synthetic leaves with scarce field data.

Wenhai Wang, Enze Xie, Xin Li, Deng-Ping Fan, Keren Song, Ding Liang, Tong Lu, Shuai Shao, Chunhua Shen, and Shi-Sheng You. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

Xiaoguang Wang, Yi Zhang, Zhenyu Liu, and *et al.* Vision transformers for remote sensing: A review. *IEEE Geoscience and Remote Sensing Magazine*, 2023. in press.

Hao Wen, Zhijie Liu, Moritz Becker, and *et al.* Robust object detection under adverse weather conditions via carla simulation. *arXiv preprint arXiv:2009.12176*, 2020.

Ross Wightman. Pytorch image models. `https://github.com/huggingface/pytorch-image-models`, 2019.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in `timm`. *arXiv preprint arXiv:2110.00476*, 2021.

Xiao X. Zhu, Devis Tuia, Lichao Mou, and *et al.* Deep learning in remote sensing: A review. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):8–36, 2017. doi: 10.1109/MGRS.2017.2762307.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Abschlussarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Bamberg, 03.06.2025
Place, Date

Florian Gauthier
Signature