Generating Synthetic Image Data with Blender to Address Data Scarcity in Military Applications: Leveraging the RF-DETR Model

Julian Cornel Berndt[1], Tobias Frisenborg Christensen[2], Lars Würtz Jochumsen[3]

[1] Systematic A/S, Søren Frichs Vej 39, 8000 Aarhus - jcb@systematic.com
[2] Systematic A/S, Søren Frichs Vej 39, 8000 Aarhus - tfchr@proton.me
[3] Systematic A/S, Søren Frichs Vej 39, 8000 Aarhus - lwj@systematic.com

Keywords: Object Detection, Computer Vision, Synthetic Data, Military Vehicles, 3D Models.

Abstract

Military vehicle recognition faces critical data scarcity due to operational security constraints and prohibitive collection costs. Classification of vehicles demands extensive training data rarely available in defence contexts. We propose a hybrid approach combining limited real-world data with scalable synthetic generation. Our methodology comprises: (1) a Blender-based pipeline generating high-resolution synthetic images with domain randomization across 3D models, lighting, and camera angles; (2) training transformer-based RF-DETR detectors on real-world and synthetic data, respectively; (3) an in-depth evaluation of the trained networks to determine the effect of synthetic data. Our approach utilizes a baseline RF-DETR detector trained on real-world imagery to compare against. Then we utilize the custom-made synthetic data generation pipeline to create an equally large synthetic dataset. This generated data is added to real data subsets, thus creating a mixed datasets containing varying percentages of real data. We created five datasets containing 5%, 10%, 25%, 50%, and 100%, respectively. With these new mixed datasets we train another set of RF-DETR detectors. Afterwards we evaluate the influence of the synthetic data by comparing the detectors across computer vision metrics.

1. Introduction

Modern object detection models have achieved a high level of performance across a range of benchmarks (Ciaglia et al., 2022, Sapkota et al., 2025), but this is only possible due to large-scale annotated datasets (Shao et al., 2019, Xia et al., 2019). Many studies (Sun et al., 2017, Hestness et al., 2017, Wang et al., 2025) assume access to thousands or tens of thousands of labelled images. Finding, preprocessing, and annotating data can be a costly endeavour (Zheng et al., 2025). In some areas, such as traffic surveillance or vehicle detection, there exists a lot of available labelled data, either open-source or paid versions (Unidata, 2026, Wu, 2025).

In many other real-world applications, this abundance of labelled data cannot be transferred, as they are limited by data scarcity (Nandy et al., 2022, Alzubaidi et al., 2023). Domains such as defence, medicine, industrial inspection, and remote sensing fall into this category, as you can face the need for domain experts, a general lack of data, and security and privacy concerns (Upadhyay and Bhandari, 2024, Kim et al., 2019, Rettore et al., 2024, Wang et al., 2015).

So in this work, we focus on the regime of data scarcity, where only a few hundred real images are available and collecting more is costly or impractical. Instead of asking how much data is needed before more images bring little benefit, we address the central question of whether synthetic data can be used to improve the performance of object detection models in data-scarce domains. We investigate whether and how synthetic data can substitute for, or augment, scarce real data. Specifically, we study whether synthetic RGB imagery, generated under controllable, variable conditions, can achieve this improvement in object detection performance.

The focus of this paper will be military vehicle detection, as it is a relevant field that is constrained by data scarcity. We create a training pipeline based on the transformer-based RF-DETR models (Robinson et al., 2026) with a DINOv2 (Oquab et al., 2024) backbone. We combine a limited set of hand-annotated real images with an equal amount of synthetic images created in Blender (Blender, 2018). These images come with randomized models, locations, camera parameters, lighting scenarios, and environmental conditions. We do this to increase variety and reduce overfitting to specific configurations. This setup allows us to systematically examine how synthetic data influences model performance in comparison to real data.

To examine the effects, we compare real-only detectors and mixed detectors (real data + synthetic data) across computer vision metrics. These can be measured when testing the detectors on a real-world test dataset. We simulate increasing data scarcity, starting from the full real dataset and downsampling progressively to 50%, 25%, 10%, and 5%. The overall training budget (number of optimization steps) is fixed across all experiments. This design isolates the impact of data composition (real vs. synthetic) from differences in training compute. By holding the training schedule constant and varying only the mix and amount of real data, we can directly assess when synthetic data is beneficial and when it might have opposite effects.

2. Methodology

The following section explains the methodology for synthetic data generation and the object detection algorithm. The first part describes the generation of synthetic image data with Blender, while the second part covers training the models for object detection using both real and synthetically generated images.

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLIX-B2-2026-633-2026

2.1 Synthetic Image Generation Pipeline with Blender

In order to test whether synthetic data can have a positive impact on object detection models, especially in data-scarce domains such as military applications, we create a pipeline that can generate images and test them in comparison to real-world data.

This is particularly acute for specialized domains such as vehicles in diverse environments and weather conditions, where it is difficult to control real-world factors like scene layout, lighting, and climate.

To address this, we implement a fully automated synthetic data generation pipeline built on Blender.

The pipeline renders 3D vehicle models into randomly selected 3D environments under varied viewpoints, backgrounds, and optionally weather conditions, while simultaneously producing COCO-formatted (Lin et al., 2014) bounding-box annotations.

The flow of the pipeline can be seen in Figure 1 and explained as follows:

2.1.1 Configuration The whole pipeline is controlled by variables that allow for variance in the generation of the images. The most important ones are the number of different positions in the 3D environments, defined as P_v, the number of images generated at each position, defined as I_p, where each image is generated with a random distance and angle to the vehicles, and the number of 3D vehicles V. The equation for the total number of generated images I is as follows:

$$I = V \cdot I_p \cdot P_v, \tag{1}$$

2.1.2 Asset management In this project there are three different groups of assets that need to be handled. The first one contains the 3D vehicles, which include texture and mesh. For this we have 10 high-resolution military vehicle models. They comprise different vehicle classes, ranging from military trucks and main battle tanks to surface-to-air missile systems. The second asset class that is necessary for the creation of synthetic images is the scenery files. These are also .blend files that contain the terrain the vehicles are to be placed upon, as well as cosmetic surroundings such as boulders, bushes, trees, etc. The scenery is also coupled with textures that contain information for all the included assets for the scenery. The last assets that need to be taken into account before everything can be arranged are the high dynamic range image (HDRI) assets. An HDRI contains a wider range of brightness levels and more information than a regular RGB camera image. It allows for accurate depiction of different lighting scenarios. It is used as the backdrop of the whole scene and as a light source in order to get accurate shadow casting from the vehicles and the scenery.

2.1.3 Model processing and normalization Initially, in the model processing, we load the scenery or world file and apply preprocessing to ensure the same coordinate system for the loaded world and the vehicle models that get placed later on. Furthermore, we delete the most often delivered camera that is included in the world files, as we create our own later on. For the vehicles, each of the models from the respective asset directory will go through multiple steps before it is placed in the world. The model loading is the first step, where Blender loads the vehicle's .blend file and then the textures are applied. Afterwards there is a clean-up process, which ensures that there is no unwanted scenery included in the vehicle's model file. Anything from floors, grids, and cameras included gets excluded from the vehicle model. Afterwards a specially created group in the model is used for turret randomization. On a military vehicle, the turret is the rotating armoured structure mounted on top of the hull that holds the guns. This ensures that with each pass the model gets placed into the scene with a different outer appearance – it is a sort of augmentation that is baked into the image creation pipeline. This is necessary, as military vehicles typically can change a lot, i.e. when the turret is turning on a main battle tank. When all meshes are positioned correctly, they get merged into one object which can later be used for the placement in the scenery. For this merged object we calculate a bounding box which contains the full object in 3 dimensions.

2.1.4 Scene and Camera Setup After the model loading we initialize a camera that is later to be used. This initialization comes with a few settings that go into the rendering in the end:

- Render Engine: Cycles
- Resolution: $1920px \cdot 1080px$
- Render device: GPU (NVidia RTX 4060)
- Samples: 128
- Denoising: OpenImageDenoise

Blender has two render engines, EEVEE and Cycles (Blender, 2018). We used Cycles, which is a physically based path tracer. It simulates how light bounces around the scene: reflections off metal surfaces, soft shadows, realistic ambient occlusion, volumetric fog. This produces a more photorealistic output, which is critical for synthetic training data.
Each pixel in a Cycles render is computed by shooting random light rays into the scene. The samples value sets how many rays there are per pixel. More samples equal a less noisy image, but the rendering gets slowed down.
The render resolution was set to $1920px \cdot 1080px$, as full-HD images are a standard for many airborne cameras. There would have been the possibility of opting for a higher resolution, but the render time gets significantly higher. The picked resolution is a trade-off between realism and render times.

2.1.5 Ground-plane detection and object placement In order to place the vehicle model on the surface of the scenery we need to figure out what part of the scenery is surface level and what is other meshes. For that we create a ground-plane inventory, which is a name-based, filtered subset of all the contained meshes. As a fallback the default ground plane is assumed to be at $z = 0$.
Afterwards a random position for the placement of the vehicle is calculated that lies within the bounds of the scenery. The placeable area is shrunk to 80 percent of the original scenery in order to not place it out of bounds. A single vertical ray is cast from high above the candidate downward against the terrain. Among the hits (multiple meshes overlapping), the one with the highest Z is selected (the topmost surface at that XY) and its normal vector is used as the local surface normal. Adding to that, we have introduced a slope threshold, as a terrain angle of more than 30° is highly unrealistic for most military vehicles.

If the slope is too steep, the location is rejected and another random spot is sampled.
As the scenery is not a flat plane but consists of multiple sloped meshes, the vehicle model needs to adapt to the terrain. For that we use the underside of the vehicle's 3D bounding box, created in step three, and in a 5×5 pattern perform raycasting onto the scenery. This yields 25 sample coordinates. The intersection height and surface normal from all valid hits are averaged and normalized to obtain an approximate mean terrain normal under the footprint. This normal is returned so that, after the Z placement is fixed, a tilt can be applied to slightly align the model with the terrain without disturbing the contact height. After all these steps the model is successfully placed on the scenery surface.

2.1.6 Camera distance, view framing, and randomness

To place a camera, the vehicle's 3D bounding box dimensions (X, Y, Z) are used. The smallest of these is interpreted as the vehicle height, which is used as the dimension that should occupy a target fraction T_f of the image. Using the camera's horizontal field of view ϕ_h and the image aspect ratio α, it derives the vertical FOV ϕ_v

$$\alpha = \frac{I_w}{I_h}, \tag{2}$$

where I_w is the width of the image and I_h is the height of the image.

$$\phi_v = 2 \arctan \left(\frac{\tan\left(\frac{\phi_h}{2}\right)}{\alpha} \right) \tag{3}$$

Then it converts the target object size in pixels f_{px},

$$f_{px} = \frac{\frac{H}{2}}{\tan\left(\frac{\phi_v}{2}\right)}, \tag{4}$$

where H is the render height

$$h_{px} = H \cdot T_f \tag{5}$$

and solves for the camera–object distance D via the pinhole camera model

$$D = S \cdot \frac{f_{px}}{h_{px}}. \tag{6}$$

This yields a baseline distance such that the object's smallest dimension occupies a prefixed variable fraction of the image height. For each vehicle load, the pipeline determines how many images to render with that model and samples a list of random distances within preset bounds. This list allows for shots ranging from relatively close to substantially farther, for broad scale variation. For each of the chosen distances the pipeline does the following:

- Selects a random azimuth uniformly from 0° to 360° around the object and a random elevation within predefined bounds (e.g. low to moderate angles).
- Computes camera coordinates on a spherical shell around the object's center at the chosen distance and elevation, and positions the camera there.
- Applies a random framing offset so the vehicle is not always centered, so as not to train bias into the model later on.

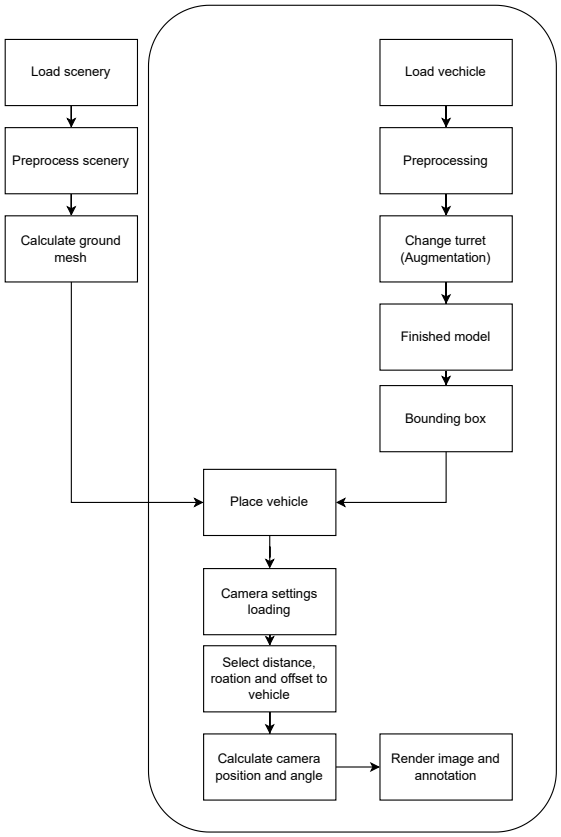

Figure 1. Synthetic data pipeline.

2.1.7 Rendering and COCO annotation

After all the previous steps have been resolved successfully, the pipeline calls Blender to render the image and place it in a predefined output folder. For every rendered image there needs to be an annotation file. Due to the fact that the pipeline has information about all variables such as the position of the object, the camera placement, the FOV, etc., the bounding box can be calculated on the fly. The pipeline stores annotations in the COCO format (Lin et al., 2014), a widely used JSON-based standard for object detection and segmentation datasets that organizes images, categories, and annotations (e.g., bounding boxes) in a consistent, model-friendly structure. In this implementation, a COCO dictionary is first initialized. When an existing annotations file is present, the script loads it, advances image id and annotation id to avoid collisions, and reuses the existing category definitions so that multiple runs can safely append to the same COCO dataset. After each image is rendered and annotated, the updated COCO structure is written back to disk, ensuring annotations remain consistent and recoverable across runs.

2.1.8 Class-balanced generation and runtime control

All vehicle models are assigned to semantic classes (e.g., military truck, main battle tank), and the generator targets a fixed number of images per class rather than per individual model. For each class, the pipeline defines a class-specific image budget and distributes this budget evenly across all models belonging to that class by cycling through them in round-robin fashion. If each class contains the same number of models, each model is rendered equally often; if some classes have fewer mod-

els, the corresponding models are reused more frequently, with multiple camera distances and poses, until the per-class image quota is met. This strategy cannot create as much intra-class geometric diversity as adding more distinct models would, but it does enforce balanced class frequencies in the final dataset while making maximal use of the available assets. In the end we were able to create images such as the one in Figure 2.

Figure 2. Example: Synthetic image.

To compare it against a real-world example, please see Figure 3.

Figure 3. Example: Real-world image.

2.2 Training Workflow

To address the central research question, whether synthetic imagery provides measurable benefits for object detection in data-scarce settings, all experiments employ RF-DETR (Robinson et al., 2026) with a DINOv2 (Oquab et al., 2024) backbone as the base architecture, specifically the RFDETRMedium model (RF-DETR-docs, 2026). The available training data comprise 420 real, manually annotated images and 420 synthetically generated images with automatic annotations. This way both base datasets are equally large.

As an initial step, three baseline models were trained: one on real images only, one on synthetic images only, and one on a mixed dataset. To avoid dependence on the number of epochs, all models were trained for a fixed number of iterations, corresponding to 25,000 randomly sampled images (with replacement) from the respective training sets. These three baselines provide an initial comparison between models trained exclusively on real data, exclusively on synthetic data, and on a balanced mixture.

Augmentation	Probability	Details
Geometric augmentations		
Horizontal flip	50%	Mirror left/right
Shift + Scale + Rotate	60%	Shift ±5%, scale 80%–120%, rotate ±15°
Affine shear	30%	±10°
Photometric – brightness group (one picked)		
One of the following	70%	
Brightness/Contrast	33%	±30%
Random gamma	33%	Gamma correction 0.7–1.3
CLAHE	33%	Adaptive histogram clipping
Photometric – colour		
Hue/Saturation/Value	50%	Hue ±15, saturation ±30%, value ±20%
Colour jitter	30%	Brightness ±0.1, contrast ±0.1, saturation ±0.2, hue ±0.05
Channel shuffle	5%	Randomly swaps RGB channels
Blur/Noise group (one picked)		
One of the following	30%	
Gaussian blur	33%	Kernel 3–7 pixels
Motion blur	33%	Kernel 3–7 pixels
Median blur	33%	Kernel 1–5 pixels
Gaussian noise	30%	Standard deviation 1%–5% of pixel range
Synthetic–real domain bridge		
Compression	30%	Adds compression artefacts
Downscale + Upscale	20%	Scale to 50%–80%, then back up (adds resolution degradation)
Random shadows	20%	1–3 random overlays
Random flare	10%	1–3 random overlays

Table 1. Augmentation overview.

Subsequently, we systematically investigated the effect of real data scarcity. A series of models was trained using only real images, with the available real dataset downsampled to 100%, 50%, 25%, 10%, and 5% of the 420 manually annotated images. In parallel, for each of these five real-data subsets, a corresponding mixed-data model was trained by combining the reduced real subset with the full set of 420 synthetic images. For all these experiments, the total number of training iterations again corresponded to 25,000 randomly sampled images, ensuring a consistent training budget across conditions.

It is important to note that each image that gets picked in a random manner has an augmentation applied to it to increase variety in image data and reduce bias.

The augmentation pipeline consists of a sequence of geometric and photometric transforms, each activated with a specified probability.

We have used the library Albumentations (Buslaev et al., 2020). The configuration parameters can be seen in Table 1.

Since images can be revisited multiple times during training, each pass through the dataset produces a different combination of flips, geometric distortions, colour shifts, blur, and noise for the same underlying scene, providing strong regularization and helping to bridge the domain gap between clean synthetic renders and real-world imagery.

For some example augmentation results, see Figure 4.

2.3 Metrics

We use several metrics in order to evaluate the model performances produced in this work.

Mean IoU (mean Intersection over Union) measures the average overlap quality between predicted bounding boxes and their matched ground-truth boxes, focusing on localization accuracy for correctly detected objects (Schlosser et al., 2024). For a single prediction–ground-truth pair, IoU is defined as $IoU = A_i/A_u$, where A_i is the area of intersection and A_u is the area of union, where the intersection area is the overlap region of the two boxes and the union area is the combined area of both boxes minus their intersection. A higher mean IoU indicates that, for the objects the model does detect, the bounding boxes more closely align with the actual object shapes and positions.

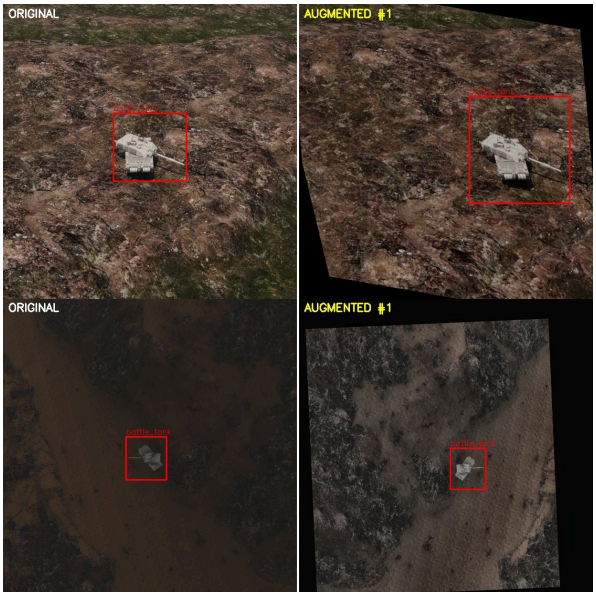

Figure 4. Example: Augmentation preview.

True positives (TP) are the correctly detected objects, where a model's predicted bounding box successfully corresponds to a ground-truth object of the same class with sufficient overlap (Schlosser et al., 2024). Formally, a prediction is counted in TP if its IoU with a ground-truth box is greater than or equal to a chosen threshold (e.g., 0.5) and the predicted class label matches the ground-truth class; each ground-truth box is usually allowed to match at most one prediction, typically the highest-confidence one. The total number of TP is then used directly in metrics like precision and recall, and it reflects how many ground-truth objects were successfully found and correctly localized according to the evaluation criteria.

False positives (FP) are predicted bounding boxes that are not correct detections, representing instances where the model claims an object is present but that claim is wrong (Schlosser et al., 2024). A prediction is counted as an FP if it does not match any ground-truth box with IoU above the threshold, if it matches a ground-truth box of the wrong class, or if it is an extra prediction on an object that has already been matched by a higher-confidence prediction (duplicate detection). The total number of FP enters directly into the precision formula. Having many FP lowers precision and indicates that the model frequently hallucinates objects or over-segments the scene.

False negatives (FN) are ground-truth objects that the model fails to detect, representing the misses in detection (Schlosser et al., 2024). A ground-truth box is counted as an FN if there is no predicted box of the correct class with IoU above the threshold that can be matched to it; in other words, no prediction satisfies the criteria to become a true positive for that object. The total number of false negatives appears in the recall formula. A large number of FN means low recall, indicating that the model often overlooks objects that are actually present in the images.

Precision in object detection measures the proportion of predicted bounding boxes that are correct detections, and it focuses on the reliability of the model's positive predictions (Schlosser et al., 2024). Formally, it is defined as $P = \frac{TP}{TP+FP}$. In practice, a predicted box is counted as a true positive if it matches a ground-truth box of the correct class with IoU above a chosen threshold; otherwise, it contributes to the FP count. A high precision means that when the model predicts an object, it is usually correct, while low precision indicates that many of the predicted boxes do not correspond to real objects or correct classes.

Recall in object detection measures how completely the model finds the objects that are actually present in the dataset, focusing on coverage rather than reliability (Schlosser et al., 2024). It is defined as $\rho = \frac{TP}{TP+FN}$. To compute recall, each ground-truth object is checked to see whether there is at least one prediction of the correct class overlapping it with IoU above the threshold; if not, that object contributes to FN. High recall means the model detects most of the objects in the images, whereas low recall indicates it misses many objects even if some of its predictions are accurate.

The **F1 score** combines precision and recall into a single metric that balances both correctness and completeness of detections (Schlosser et al., 2024). It is defined as the harmonic mean of precision and recall, $F1 = 2\frac{P \cdot \rho}{P+\rho}$. Because it uses the harmonic mean, F1 becomes high only when both precision and recall are relatively high; if one of them is very low, the F1 score drops significantly. In object detection, F1 is often evaluated at a chosen confidence threshold and IoU threshold, and it is useful for selecting an operating point where the trade-off between avoiding false positives and avoiding false negatives is acceptable for a given application.

Mean confidence refers to the average confidence score the model assigns to its predicted bounding boxes and describes how “certain” the model tends to be in its detections (Schlosser et al., 2024). If γ_i is the confidence score of the i-th prediction and there are N_{pred} predictions, then the mean confidence over all predictions can be expressed as $\mu = \frac{1}{N_{\text{pred}}} \sum_{i=1}^{N_{\text{pred}}} \gamma_i$. Sometimes this average is computed only over specific subsets, such as TP or FP, to analyze calibration (e.g., whether high-confidence predictions are indeed more likely to be correct). While mean confidence itself does not measure accuracy, comparing it with the corresponding TP and FP rates helps assess whether the model's confidence scores are well calibrated.

3. Results

To produce results with the trained RF-DETR networks, we used a test dataset that contains 320 annotated real-world images and has not been part of the real training data.

All results are measured with the metrics from Subsection 2.3.

As explained in Subsection 2.2, we first obtained results for a direct comparison between real, synthetic, and mixed data. For comparability with the other results, we discarded the network that was purely trained on synthetic data. The results can be seen in Figure 9.

Furthermore, we obtained results for the networks trained under increasing data scarcity. Please refer to Figures 5, 6, 7, and 8.

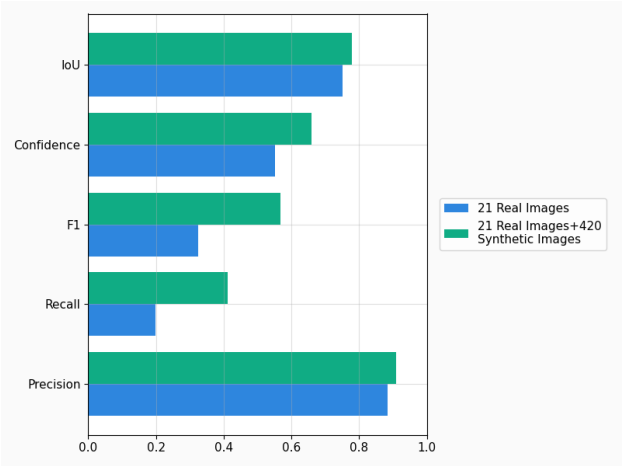

Figure 5. Real data scaled down to 5%.

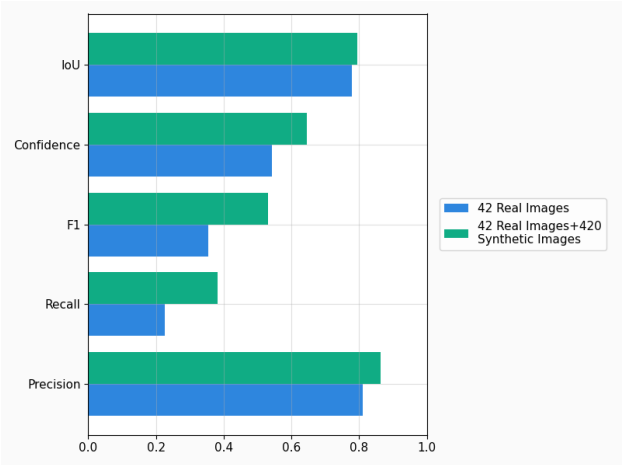

Figure 6. Real data scaled down to 10%.

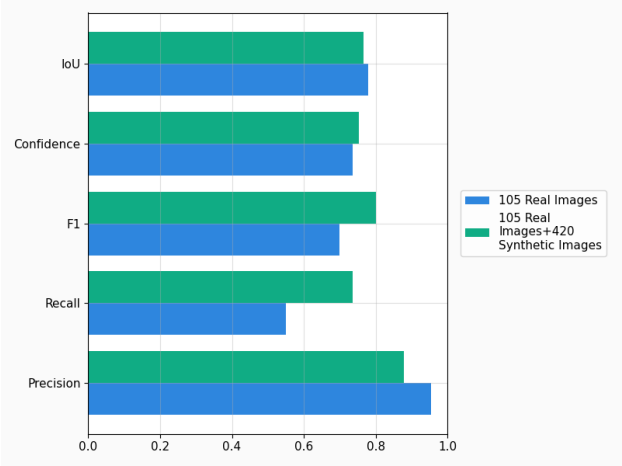

Figure 7. Real data scaled down to 25%.

4. Discussion

To quantify the impact of synthetic imagery under different levels of real-data scarcity, we compare, for each real-data fraction (5%, 10%, 25%, 50%, 100%), two models trained with an identical training budget: one using only the corresponding subset of real images and one using the same real subset augmented with the full synthetic dataset. All models are evaluated on the same real-world test set.

When 5% of the real data is available, adding synthetic images leads to a substantial performance gain (see Figure 5). Precision increases from about 88% to around 91%, recall more than doubles from roughly 20% to about 41%, and the F1-score rises from around 32% to approximately 57%. The number of true positives more than doubles (from roughly 210 to about 450), while false positives increase moderately and false negatives decrease markedly. Mean confidence and mean IoU also improve, indicating that the model not only detects more objects but also does so with higher confidence and better localization quality. This regime clearly illustrates that synthetic data can compensate for severe shortages of real annotations.

When 10% of the real data is available, the same qualitative behaviour is observed (see Figure 6). Added synthetic data increases precision from roughly 81% to about 86% and recall from around 23% to approximately 38%, resulting in an F1-score improvement from about 35% to roughly 53%. True positives increase markedly, with only a modest rise in false positives and a notable reduction in false negatives. Mean confidence and mean IoU both improve. Compared to the 5% setting, the

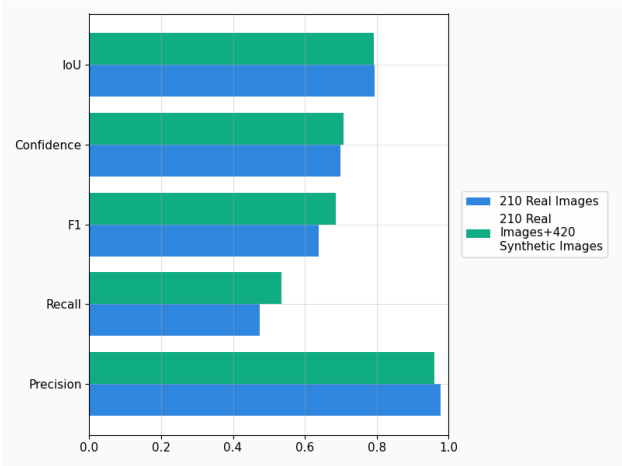

Figure 8. Real data scaled down to 50%.

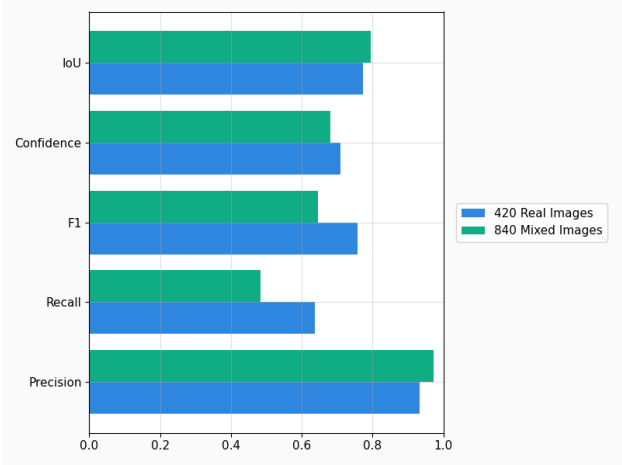

Figure 9. 100% real data training compared to a mix of 100% real and 100% synthetic data.

real-only model is already stronger, but the synthetic data still provides a large relative increase in overall effectiveness.

When 25% of the real data is available, the real-only model becomes highly precise but conservative: precision reaches about 95%, with recall at roughly 55% and an F1-score of around 70% (see Figure 7). Adding synthetic data shifts this operating point: precision decreases to around 88%, but recall rises strongly to approximately 74%, yielding a higher F1-score of about 80%. True positives increase considerably, while both false positives and false negatives change in line with this shift. Mean confidence increases slightly, whereas mean IoU decreases marginally. This illustrates a clear precision–recall trade-off: synthetic data makes the detector less conservative, significantly increasing coverage and overall F1 at the cost of more false alarms.

When 50% of the real data is available, the incremental benefit of synthetic data is smaller but still consistent (see Figure 8). The real-only model achieves very high precision (around 98%) but moderate recall (roughly 47%), with an F1-score of about 64%. With synthetic data included, precision remains high at approximately 96%, recall increases to around 53%, and F1 improves to circa 69%. The number of true positives rises, false positives roughly double (though remain low in absolute terms), and false negatives decrease. Mean confidence improves slightly, while mean IoU remains nearly unchanged. This suggests that once a moderate amount of real data is present, synthetic data still yields measurable gains, but the relative impact is reduced compared to the low-data regimes.

When 100% of the real data is available, adding synthetic augmentation shifts the model’s behaviour notably toward higher precision but at a significant cost to recall (see Figure 9). The real-only model achieves solid precision (around 93%) with moderate recall (roughly 64%), yielding an F1-score of about 76%. With synthetic data, precision rises to approximately 97%, but recall drops considerably to around 48%, and F1 decreases to circa 65%. The number of true positives falls from 688 to 521, false positives decrease sharply (from 49 to 15), and false negatives increase from 391 to 558. Mean confidence decreases slightly, while mean IoU improves marginally. This suggests that at full real-data availability, synthetic data makes the model more conservative. It detects fewer objects overall but with greater certainty, indicating that the synthetic examples may be reinforcing stricter decision boundaries rather than broadening the model’s ability to detect challenging cases.

Taken together, these experiments show that the value of synthetic data is strongly tied to real-data scarcity. In the low-data regimes, 5% to 10%, synthetic imagery improves recall and overall detection quality while maintaining or even increasing precision, yielding substantial F1 gains. In the intermediate regimes, 25% to 50%, synthetic data still improves F1 by boosting recall, though at the cost of a moderate reduction in precision. However, at full real-data availability, 100%, this trend reverses - adding synthetic data increases precision but substantially reduces recall, resulting in a lower F1-score than the real-only model.

5. Conclusion

Overall, the experimental results discussed above provide evidence that synthetic data can be used to improve the performance of object detection models in data-scarce domains. Across all examined levels of reduced real-data availability, inflating real datasets with synthetic imagery consistently increased F1-score compared to training on real data alone, indicating a net gain in overall detection quality. This improvement is primarily driven by substantial gains in recall, while maintaining high precision.

The effect is most pronounced in the extreme low-data regimes, where adding synthetic images more than doubles recall and F1, and also improves mean confidence and IoU, demonstrating better coverage, higher certainty, and improved localisation of detections. Even when moderate amounts of real data are available, the addition of synthetic data still raises F1 by trading a modest reduction in precision for a substantial gain in recall, thereby yielding more effective detectors overall. However, at full real-data availability, this trend reverses: added synthetic data increases precision but substantially reduces recall, resulting in a lower F1-score than the real-only model.

This reveals a pattern: the more data-scarce a domain is, the more impactful synthetic data becomes. As real data grows abundant, synthetic images offer diminishing and eventually negative returns, suggesting they may constrain the model’s ability to generalise to harder detection cases rather than expand it.

These findings support an affirmative answer to the research question: synthetic data can indeed be used to improve the performance of object detection models in data-scarce domains. Synthetic imagery is particularly valuable when real annotations are severely limited, where it enhances detection coverage and robustness, and it continues to provide measurable benefits even once a moderate amount of real data is available.

References

Alzubaidi, L., Bai, J., Al-Sabaawi, A., Santamaría, J., Albahri, A. S., Al-dabbagh, B. S. N., Fadhel, M. A., Manoufali, M., Zhang, J., Al-Timemy, A. H., Duan, Y., Abdullah, A., Farhan, L., Lu, Y., Gupta, A., Albu, F., Abbosh, A., Gu, Y., 2023. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1), 46. https://doi.org/10.1186/s40537-023-00727-2.

Blender, O. C., 2018. Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A. A., 2020. Albumentations: Fast and Flexible Image Augmentations. *Information*, 11(2), 125. http://dx.doi.org/10.3390/info11020125.

Ciaglia, F., Zuppichini, F. S., Guerrie, P., McQuade, M., Solawetz, J., 2022. Roboflow 100: A rich, multi-domain object detection benchmark.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., Zhou, Y., 2017. Deep learning scaling is predictable, empirically.

Kim, D.-E., Gourbesville, P., Liong, S.-Y., 2019. Overcoming data scarcity in flood hazard assessment using remote sensing and artificial neural network. *Smart Water*, 4(1), 2. https://doi.org/10.1186/s40713-018-0014-5.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., Dollár, P., 2014. Microsoft COCO: Common objects in context.

Nandy, A., Duan, C., Kulik, H. J., 2022. Audacity of huge: overcoming challenges of data scarcity and data quality for machine learning in computational materials discovery. *Current Opinion in Chemical Engineering*, 36.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P., 2024. Dinov2: Learning robust visual features without supervision.

Rettore, P. H. L., Zißner, P., Alkhowaiter, M., Zou, C., Sevenich, P., 2024. Military Data Space: Challenges, Opportunities, and Use Cases. *IEEE Communications Magazine*, 62(1), 70-76.

RF-DETR-docs, 2026. Run an RF-DETR Object Detection Model. https://rfdetr.roboflow.com/develop/learn/run/detection/.

Robinson, I., Robicheaux, P., Popov, M., Ramanan, D., Peri, N., 2026. Rf-detr: Neural architecture search for real-time detection transformers.

Sapkota, R., Cheppally, R. H., Sharda, A., Karkee, M., 2025. Rf-detr object detection vs yolov12 : A study of transformer-based and cnn-based architectures for single-class and multi-class greenfruit detection in complex orchard environments under label ambiguity.

Schlosser, T., Friedrich, M., Meyer, T., Kowerko, D., 2024. A Consolidated Overview of Evaluation and Performance Metrics for Machine Learning and Computer Vision.

Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., Sun, J., 2019. Objects365: A large-scale, high-quality dataset for object detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Sun, C., Shrivastava, A., Singh, S., Gupta, A., 2017. Revisiting unreasonable effectiveness of data in deep learning era.

Unidata, 2026. Real-Time Traffic Video Dataset. https://unidata.pro/datasets/real-time-traffic-and-environmental-video-dataset/.

Upadhyay, A. K., Bhandari, A. K., 2024. Advances in Deep Learning Models for Resolving Medical Image Segmentation Data Scarcity Problem: A Topical Review. *Archives of Computational Methods in Engineering*, 31(3), 1701-1719. https://doi.org/10.1007/s11831-023-10028-9.

Wang, L., Ma, Y., Zhang, J., Zhang, X., Liu, Y., 2015. Uncertainty Quantification and Structural Reliability Estimation Considering Inspection Data Scarcity. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 1(2), 04015004. https://ascelibrary.org/doi/abs/10.1061/AJRUA6.0000818.

Wang, X., Alabdulmohsin, I., Salz, D., Li, Z., Rong, K., Zhai, X., 2025. Scaling pre-training to one hundred billion data for vision language models.

Wu, J., 2025. Traffic Scene Analysis Dataset for Vehicle Detection and Distance Estimation. https://dx.doi.org/10.21227/akgz-xg04.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., Zhang, L., 2019. Dota: A large-scale dataset for object detection in aerial images.

Zheng, F., XingMing, L., JuYing, X., MengYing, T., BaoJian, Y., Yan, S., KeWei, Y., ZhiKai, L., Cheng, H., KeLan, Q., XiHao, C., WenFei, D., Ping, H., RunYu, W., Ying, Y., XiaoHui, B., 2025. Significant reduction in manual annotation costs in ultrasound medical image database construction through step by step artificial intelligence pre-annotation. *PLOS Digital Health*, 4(6), 1-17. https://doi.org/10.1371/journal.pdig.0000738.