AI Playground: Unreal Engine-based Data Ablation Tool for Deep Learning

Mehdi Mousavi[1][0000−0001−8948−8011], Aashis Khanal[1][0000−0002−0164−2465], and Rolando Estrada[1][0000−0003−1607−2618]

Department of Computer Science, Georgia State University, Atlanta GA 30303, USA
{smousavi2,akhanal1}@student.gsu.com, restrada1@gsu.edu

Abstract. Machine learning requires data, but acquiring and labeling real-world data is challenging, expensive, and time-consuming. More importantly, it is nearly impossible to alter real data post-acquisition (e.g., change the illumination of a room), making it very difficult to measure how specific properties of the data affect performance. In this paper, we present AI Playground (AIP), an open-source, Unreal Engine-based tool for generating and labeling virtual image data. With AIP, it is trivial to capture the same image under different conditions (e.g., fidelity, lighting, etc.) and with different ground truths (e.g., depth or surface normal values). AIP is easily extendable and can be used with or without code. To validate our proposed tool, we generated eight datasets of otherwise identical but varying lighting and fidelity conditions. We then trained deep neural networks to predict (1) depth values, (2) surface normals, or (3) object labels and assessed each network's intra- and cross-dataset performance. Among other insights, we verified that sensitivity to different settings is problem-dependent. We confirmed the findings of other studies that segmentation models are very sensitive to fidelity, but we also found that they are just as sensitive to lighting. In contrast, depth and normal estimation models seem to be less sensitive to fidelity or lighting and more sensitive to the structure of the image. Finally, we tested our trained depth-estimation networks on two real-world datasets and obtained results comparable to training on real data alone, confirming that our virtual environments are realistic enough for real-world tasks.

Keywords: Synthetic data · Deep learning · Virtual environment

1 Introduction

The remarkable success of deep learning in recent years would not have been possible without large, high-quality datasets [8]. Deep neural networks have thousands or even millions of parameters, which require vast numbers of training examples to tune. However, producing a high-quality dataset of real data is very challenging. First, one has to acquire the raw data, an often laborious task. Second, the training data must either be labeled manually—which is slow, subjective

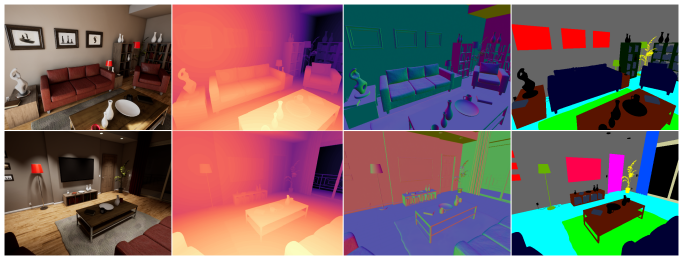

Fig. 1: **Virtual environments:** Sample screenshots from our annotated virtual environments. From left to right: depth, surface normals, and semantic labels

and may require significant expertise—or with expensive, specialized equipment. Finally, errors can occur in both the acquisition and labeling phases.

Real data also has an additional, more subtle limitation: it is very difficult to control before acquisition and nearly impossible to change afterwards. For instance, once an image has been taken, one cannot change its illumination from day to night or replace one object for another[1]. The only way to achieve these effects is by manipulating the source of the data before acquisition; however, this approach requires a controlled environment and precise measurements. For example, to change the color of a couch one would need to swap out two otherwise identical couches and place them in the same, exact location. Aside from its difficulty, this approach is not feasible for natural scenes or crowd-sourced data.

The above limitation makes it is very difficult to isolate the impact of individual features on a system's performance. For example, imagine that we want to assess how an object's texture affects our system's ability to segment it. In this case, we would need to compare our system's output across different objects and hope that the impact of other features, e.g., lighting or shape, cancels out across the samples. As such, data ablation studies are rare in machine learning. Most ablation analyses add/remove either (1) components of the model [10] or (2) secondary features computed from the data [9]. The latter is close in spirit to data ablation but is more limited, since secondary features are dependent on the raw, unchangeable data.

To address this gap, we developed AI Playground, a user-friendly tool based on the Unreal Engine (Epic Games, USA) [4] that supports data ablation studies in computer vision.[2] Our system allows researchers to easily capture synthetic data from fully customizable virtual environments; this data can then be used to train or test an AI system. Virtual data is free from acquisition errors or labeling bias and is ideal for the data ablation studies discussed above, e.g., capturing the same image under multiple lighting conditions. More importantly, as

[1] Photo-manipulation techniques can be used to alter images, but their effects are either non-specific (e.g., reducing brightness) or introduce unwanted artifacts. They also require significant human effort.

[2] Source code, documentation, supplementary images, and high-definition figures are available on our GitHub page: `https://git.io/JJkhQ`

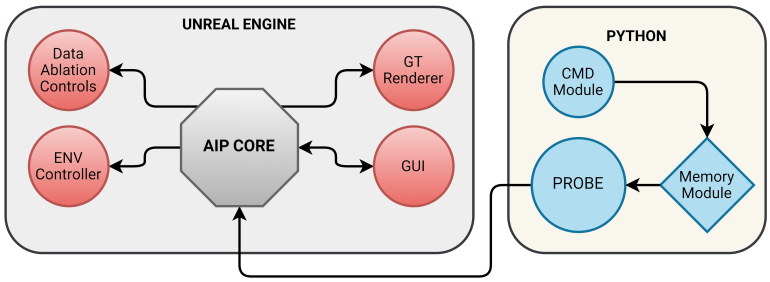

Fig. 2: **AI Playground:** Our tool has two main modules: the *AIP Core* within UE4 and *Probe*, a Python module that communicates with the Core. Probe receives instructions generated by the Command module, and saves its state in its own dedicated memory. This allows changing settings inside the engine while AIP is running. Manually changing components is also possible via the GUI.

our experiments confirm, today's high-resolution computer graphics are realistic enough to be used for training deep neural networks on real-world tasks.

As we detail in Sec. 3, AI Playground is an open-source UE project with four main components: (1) a set of high-resolution environments; (2) multiple ground-truth annotations (e.g., depth, surface normals, etc.); (3) built-in tools for data ablation (e.g., for adjusting lighting, polygon resolution, etc.); and (4) a user-friendly, graphical interface. Users can either run our system as a pre-built application or import it as a regular UE project. In the latter case, users can extend their local version of AIPlayground or copy parts of it (e.g., scripts) for use in their own projects. It is easy to add custom environments or ground-truth annotations without writing any code. And we provide sample code and the necessary documentation to add new forms of data ablation to AIPlayground. Figure 2 provides a flowchart of our tool.

To validate its usefulness, we used AIP to carry out a series of data ablation studies. As detailed in Sec. 4, we trained and tested deep networks on (1) monocular depth estimation, (2) surface normal estimation, and (3) semantic segmentation. AIP allowed us to draw novel insights about feature importance (Sec. 5), and we also confirmed that networks trained on depth estimation via AI Playground achieve good performance on real-world datasets.

2 Related Work

Data-hungry models like DCNN (Deep Convolutional Neural Networks) have generated newfound interest in virtual data [5, 6, 13]. One popular approach is to use modded old video games (e.g., Atari games [11]). However, this approach lacks customizability and photo-realism. This data cannot be customized to fit a more specific problem and using old video games introduces a lack of photo-realism that has been proven beneficial for virtual data [10, 18]. In contrast, Veeravasarapu *et al.* [18] used probabilistic generative models to create random environments in Blender (Blender Foundation, The Netherlands) [2]. However,

these probabilistic models need to be manually adapted for each type of desired environment. For example, the probabilistic model of an outdoor street scene varies significantly from one of an interior environment. Also, while randomness is useful for quickly creating novel environments, these environments may not be faithful to reality. For example, a random probabilistic model might decide to put a couch on a table, which never happens in the real world. Furthermore, depending on hardware, rendering an image in Blender using ray-tracing can take up to a minute or more; the same level of fidelity can be achieved in game engines in real-time. As mentioned in [18], generating a Path-traced image in Blender takes up to 9 minutes (547 seconds), and ray-tracing based rendering for a single image can take 20 seconds or more.

In another study, researchers used 3D reconstruction to generate a photo-realistic 3D scene that allows limited interaction such as walking around [15]. This method requires expensive equipment and complex calculations to generate the pixel-wise ground-truth for tasks like *depth estimation* and *surface normal estimation*. The generated ground-truth and 3D environment are subject to artifacts and estimation errors appearing as black spots in the images. Also, these environments are extremely hard to expand as they require costly specialized equipment for measurement.

The work most similar to our proposed system is UnrealCV—an Unreal Engine 4 (UE4) plugin that has been used in a number of research projects. UnrealCV provides an interface to communicate with the Unreal Engine for computer vision and robotics research [19]. However, UnrealCV requires command-line-based interaction and C++ coding. As such, it has a high barrier of entry and can be discouraging for computer vision researchers who are unfamiliar with game engines. It also lacks intuitive dials and knobs for dynamic interaction with the environment. More importantly, it is not built for data ablation; any systematic changes in fidelity, lighting, etc. have to be coded from scratch by the researcher.

In contrast, our goal is to reduce the skill level need to obtain virtual pixel-perfect data. Our approach is accessible, user-friendly, and has many intuitive ways to interact with the environment. We use the high quality renderer integrated in Unreal Engine to produce lifelike synthetic images, and AIP does not require any knowledge of UE4 programming. As we detail in the following section, our companion Python module (Probe) communicates with the UE4 application to control the environment and take samples while keeping a record of every step for image re-creation.

3 AIPlayground

AIPlayground is a UE4-based tool for data ablation studies in computer vision. Unreal Engine is the engine of choice for video games with high-resolution, real-time 3D graphics. It is free for both commercial and non-commercial use and its source code is publicly available (though not fully open source). As illustrated in Fig. 2, our system has four components: (1) high-resolution 3D environments;

Fig. 3: Sample images captured by Probe. Left to right: Brown Room Day, Brown Room Night, Blue Room Day, Blue Room Night (All high settings)

(2) multiple ground-truth annotations; (3) data ablation controls; and (4) a user-friendly, graphical interface. As we discuss further below, we use Blueprint, Unreal's visual scripting language, for the ground-truth annotations and data ablation controls. We use a separate Python interaction module—Probe—for data collection, which is also publicly available.

3.1 Three-dimensional environments

In addition to being a game engine, Unreal Engine provides powerful tools for realistic architectural visualization. As such, we developed two environments based on UE4's built-in "Realistic Rendering" scene, dubbed Brown Room and Blue Room in our experiments. Each environment has two general lighting profiles, Day and Night, as illustrated in Fig. 3. To mimic existing real-world datasets, the environments are static (i.e., no movement of the components aside from the probe character). AIP currently uses static (i.e., baked) lighting to illuminate the scene. Baking light-maps is a commonly used method to simulate high-fidelity lighting on lower-capacity hardware. It uses ray-tracing to determine dark and light spots in the scene and paints the textures on those areas to look accordingly. The result is a very realistic environment that is rendered rapidly with little to no extra computation required at run-time. This means AIP supports very high frame-rates, which allow for fast data acquisition. We can switch between different ground-truth annotations in fractions of a second without causing artifacts such as blur, fuzziness on the edges, or motion-blur.

3.2 Ground-truth annotations

One of the main advantages of virtual environments is that obtaining ground-truth annotations is trivial relative to real-world environments. Specifically, we use Unreal Blueprint (an internal scripting language) to calculate the ground-truth properties listed below. AIP includes Blueprint scripts for estimating depth, surface normals, and object classes, and can be readily extended by adding additional scripts. We use post-processing shaders, called materials in UE4, to overlay these properties over the image, enabling pixel-perfect alignment between the data and the ground-truth labels (see Fig. 1 for examples).

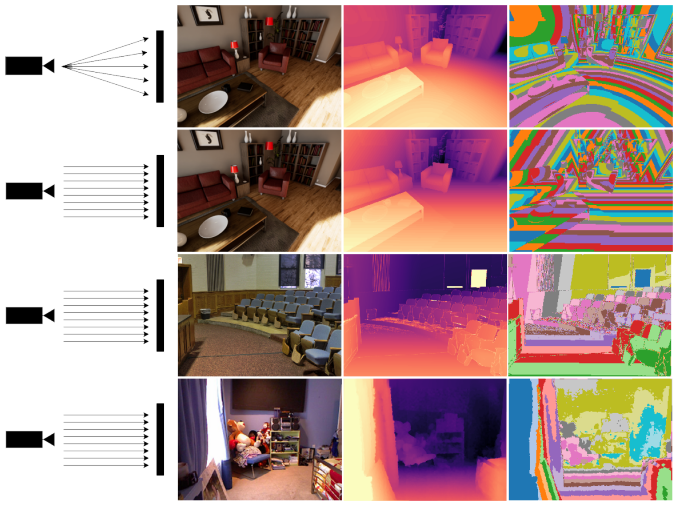

Fig. 4: **Depth estimation:** AIP uses perspective projection (first row), which is more accurate than orthographic projection (second row). The third column uses color banding to highlight the differences between these two approaches. The bottom rows show examples from the DIODE and NYUv2 datasets. Note the lack of artifacts in the virtual ground truth.

Depth estimation: We calculate the normalized distance between each pixel that belongs to a specific object and the camera. We set the real-life range of depth to 10 meters, which covers the entire environment and does not clip between any corners of the room. We define the depth using *perspective projection* relative to the viewer's POV, which is significantly more accurate than orthographic methods. In perspective depth, each light ray is traced to the exact pixel from the object its coming from; in orthographic depth, on the other hand, light-rays are assumed to be coming from *infinity* (see Fig. 4). In real-world datasets, e.g., NYUv2 [12] and DIODE [16], depth is registered based on orthographic projection because of physical limitations in the sensor.

Surface normals: We estimate the normal vector w.r.t to each 3D surface, then color each pixel to indicate the vector's direction. We use 6 main colors to show 6 axis of direction (positive and negative xyz, as shown in Fig. 1).

Semantic segmentation: In UE4, it is easy to map visible pixels to their corresponding 3D objects. Our Blueprint script uses this mapping to overlay pixel-perfect semantic labels on the various objects in the scene (e.g., couch, table, lamp, etc.).

3.3 Data ablation controls

Similar to the ground-truth, we use Blueprint to dynamically alter properties of the environment. We can access and isolate specific properties in different

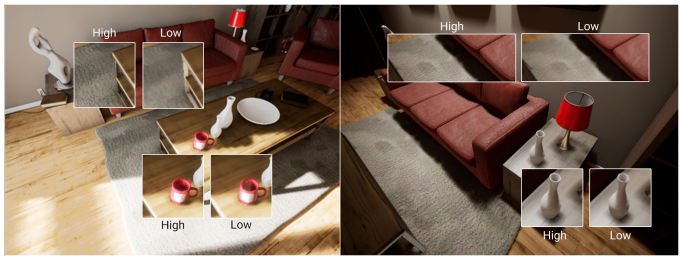

Fig. 5: **Fidelity Comparison:** Left: Day(High Fidelity), Right: Night(High Fidelity). Each image snippet of Low Fidelity indicates the difference in Texture resolution, Reflections quality, Render Scaling and Shadow quality. The amount of change in each of these settings is customizable through AIP's Core.

objects. For example, we can isolate metallic objects or rough surfaces with a pixel-perfect binary ground truth. We can also change the fidelity of reflections, lighting, mesh level of detail (LOD), render resolution (either localized to an object or globally), anti-aliasing algorithms (or toggle on and off), or render scaling. Figure 5 illustrates the same scene rendered under different fidelity settings. Our scripts are reusable, in the sense that they do not require adaptation to other environments and are also easily portable to other UE projects.

3.4 User interface

The AIP Core can be opened as a project in UE4, giving access to all its assets and scripts. Alternatively, we provide a pre-compiled version which can be run as an independent program. AIP has intuitive user menus and keyboard shortcuts. Our Python Probe script uses the latter to collect data (see Sec. 4 for details).

Table 1: Scenarios used in experiments[a]

Default	**Settings**		
Maps	***Lighting***	***Fidelity***	***Anti-Aliasing***
Brown Room	Day	High	Temporal AA
Brown Room	Night	High	Temporal AA
Brown Room	Day	Low	Temporal AA
Brown Room	Night	Low	Temporal AA
Blue Room	Day	High	Temporal AA
Blue Room	Night	High	Temporal AA
Blue Room	Day	Low	Temporal AA
Blue Room	Night	Low	Temporal AA
Abstract Shapes	Day	High	Temporal AA
Unlit[b] Brown Room	N/A	High	Temporal AA
Unlit Blue Room	N/A	High	Temporal AA

[a]shows settings used, not indicative of all settings available.[b]diffuse shading.

4 Experiments and Results

We carried out multiple experiments to validate the usefulness of our proposed system. Specifically, we tested AIP in two ways. First, we verified its viability

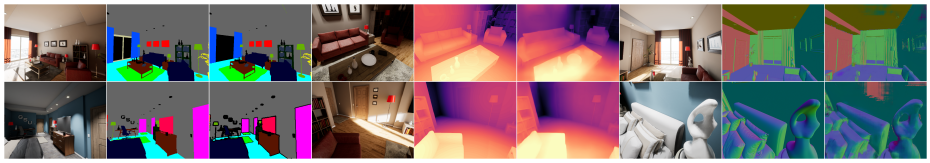

Fig. 6: **Sample results:** Sample images, ground truth, and predictions for semantic segmentation (first three columns), depth estimation (middle columns), and surface normal estimation (last three columns). Figure best viewed onscreen.

as a data ablation tool. As we detail below, we captured the same images under different fidelity and lighting settings (which we refer to as a *scenario*), then trained deep neural networks on each scenario to assess the impact of the various environmental features. We carried out both same- and cross-scenario testing (e.g., a Brown/Day/High network on Brown/Night/High). Table 1 summarizes the scenarios used. For each scenario, we tested our networks on (1) monocular depth and (2) surface normal estimation, as well as (3) semantic segmentation.

Second, to validate that our virtual data is realistic enough, we tested networks trained with AIP on real-world depth-estimation datasets, achieving results comparable to training on real data alone. Below, we first detail our experimental setup, then discuss each experiment.

4.1 Experimental Setup

Hardware: We conducted all our experiments in a Dell Precision 7920R server with two Intel Xeon Silver 4110 CPUs, two GeForce GTX 1080 Ti graphics cards, and 128 GBs of RAM.

Image acquisition: Our Probe script can control the viewpoint by simulating keystrokes. It can move and look freely (yaw and pitch) in the environment. Probe can also send specific commands and can gather images with high overlap (in groups) or low overlap (completely random). Probe's step size, look sensitivity, randomness of image acquisition (group capture), and number of images to gather are all customizable and can be saved for reproduction across all different scenarios. For our depth estimation experiments, we randomly collected 8265, 640×480 synthetic color images. We collected the same images, by replicating the same camera positions and rotations, across different lighting and fidelity scenarios (Tbl. 2). We split these images into 80% for training, and 20% for testing. Similarly, for semantic segmentation and surface normal estimation, we gathered 3000 images for each scenario and split in the same ratio.

Deep neural networks: We used the encoder-decoder architecture, and loss function from [12] for depth estimation, and an implementation of U-net [14] from [7] for surface normal estimation and semantic segmentation. We use smooth L1 loss function for Surface Normal Estimation, and Cross-Entropy loss for segmentation task. We use a *mini-batch size* of 16, *learning rate* of 0.001, and trained for 51 *epochs* for all experiments.

Table 2: **Depth estimation:** Data ablation test results. Metrics are threshold accuracy ($\delta_i < 1.25^i$), average relative error (REL), root mean squared error (RMS), and average (log10) error. Arrows indicate if higher or lower values are better. For space, we included only some of the conducted experiments; results shown are indicative of the behavior of the trained models in other scenarios. **SC:** Sanity Check. **L:** Change in Lighting. **M:** Change in Maps. **F:** Positive Change in Fidelity

Training Scenario / Fidelity	**Test / Fidelity**	**Goal**	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	**REL↓**	**RMS↓**	**log10↓**
Brown / Day / High	Brown/ Day / High	**SC**	0.7992	0.9113	0.9474	0.1426	0.0278	0.0740
Blue / Day / Low	Blue / Day / Low	**SC**	0.7609	0.8980	0.9278	0.1643	0.0366	0.0858
Brown / Night / High	Brown / Night / High	**SC**	0.8333	0.9248	0.9509	0.1327	0.0278	0.0689
Brown/ Day / Low	Brown/ Day / Low	**SC**	0.7719	0.8945	0.9388	0.1544	0.0289	0.0798
Brown/ Day / High	Brown/ Night / High	**L**	0.7616	0.8928	0.9315	0.1711	0.0398	0.0875
Brown/ Night / High	Brown/ Day / High	**L**	0.7366	0.8942	0.9420	0.1904	0.0351	0.0939
Blue / Day / Low	Blue / Day / High	**F**	0.7817	0.9062	0.9329	0.1587	0.0370	0.0822
Brown/ Day / Low	Brown/ Day / High	**F**	0.8010	0.9113	0.9475	0.1426	0.0273	0.0731
Brown/ Night / High	Blue / Night / High	**M**	0.5959	0.8632	0.9079	0.3415	0.0671	0.1193
Brown / Day / High	Blue Day / High	**M**	0.6420	0.8528	0.9223	0.2220	0.0433	0.1067

Table 3: **Depth estimation:** Results on real-world datasets.

Train / Fidelity	**Test**	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	**REL↓**	**RMS↓**	**log10↓**
Brown / Day / High	NYUv2	0.3666	0.6012	0.7586	0.5044	0.2014	0.1938
Brown / Day / Low	NYUv2	0.3720	0.6062	0.7627	0.5010	0.2010	0.1921
Brown / Night / High	DIODE	0.3563	0.5948	0.7945	0.7659	3.6897	0.2148
Brown / Night / Low	DIODE	0.3163	0.5647	0.7345	0.7743	3.7898	0.2149
Brown / Night / High	DIODE - Filtered	0.6546	0.7725	0.8371	0.6608	2.9765	0.1458
Brown / Day / High	NYUv2 - Filtered	0.5996	0.8405	0.9308	0.2835	0.1232	0.1054
DIODE/Indoor [17]	DIODE/Indoor	0.4919	0.7159	0.8256	0.3306	1.6948	0.1775
NYUv2 [1]	NYUv2	0.895	0.980	0.9960	0.1030	0.390	0.0430
NYUv2 [1]	DIODE/Indoor	0.2869	0.5097	0.6730	0.6599	2.8854	0.2573

4.2 Monocular depth estimation experiments

Data ablation: Table 2 shows a representative sample of the data ablation experiments we conducted using our depth ground truth. For these experiments, we initialized our deep networks using the weights from a network trained on NYUv2. For evaluation, we used the same metrics as those used in [3]: average relative error (REL), root mean squared error (RMS), average log10 error, and threshold accuracy ($\delta_i < 1.25^i$ for $i = [1, 2, 3]$). As we discuss further in Sec. 5, models trained in higher fidelity data generally tend to yield higher scores, even on lower-fidelity scenarios.

Real-world validation: To demonstrate the transferability of learned features from a synthetic dataset, we tested our best-performing models on the real-world DIODE and NYUv2 datasets. In addition to the full test set, we also evaluated our networks on a filtered subset that only contained scenes structurally similar to our virtual environments, i.e., indoor scenes of a living room, with objects such as couches, beds, TVs, etc. As Tbl. 3 shows, our high-fidelity trained model had better threshold accuracy on DIODE than a model trained only on NYUv2 [17],

Table 4: **Surface normal estimation:** Metrics are percentage of pixels that differ by 11.5°, 22.5°, and 30° from the true normal, and mean and median errors. Mean and median are higher than [20] because our loss function did not implement hybrid measures to reduce them. This wasn't necessary since our ground-truth data does not suffer from the problem mentioned in [20].
SC: Sanity Check. **L:** Change in Lighting. **M:** Change in Maps. **F:** Positive Change in Fidelity

Scenario / Fidelity	**Test / Fidelity**	**Goal**	11.5° ↑	22.5° ↑	30° ↑	**Mean↓**	**Median↓**
Brown / Day / High	Brown / Day / High	**SC**	0.9014	0.9566	0.9727	24.4575	88.2878
Blue / Day / Low	Blue / Day / Low	**SC**	0.9274	0.9746	0.989	30.5607	94.9516
Blue / Night / High	Blue / Night / High	**SC**	0.865	0.9224	0.9401	28.2409	69.2181
Brown / Day / Low	Brown / Day / Low	**SC**	0.8883	0.9443	0.961	25.3718	81.4871
Brown / Day / High	Brown / Night / High	**L**	0.052145	0.2238	0.3464	106.70	121.26
Brown / Night / High	Brown / Day / High	**L**	0.050291	0.2135	0.4253	115.82	119.86
Blue / Day / Low	Blue / Day / High	**F**	0.195269	0.2683	0.3015	97.832	113.57
Brown / Day / Low	Brown / Day / High	**F**	0.028247	0.2102	0.368	109.14	118.08

confirming that the features learned on our environments are transferable to real-world data. In addition, our model trained on Night lighting, high-fidelity settings achieved 31% δ_1 vs 28% δ_1 of NYUv2 model — 59% δ_2 vs 50% δ_2 of NYUv2 model — 79.4% δ_3 vs 67.3% of δ_3 of NYUv2 model. These results further confirm that our photo-realistic data can match and even exceed real-life training. Furthermore, these models achieved a much higher score in our filtered test set, suggesting that depth estimation is more sensitive to the structure of the input image than to lighting or fidelity. We also believe our models would have performed even better had DIODE used perspective depth (Fig. 4).

4.3 Surface normal estimation experiments

We carried out a similar set of data ablation experiments as above, but using surface normal data as the ground truth. Here, we trained each model from scratch, i.e., without pre-trained weights, and used the same evaluation metrics as in [20]: mean (average L1 loss), median (average L2 loss), and percentage of pixels that differ by 11.5°, 22.5°, and 30° relative to the true surface normal. Surface normal estimation is a promising use case for AIP because it is very challenging to capture surface normal information for real scenes. One needs expensive equipment to measure the angles, and these sensors are extremely hard to calibrate. As Tbl. 4 shows, we can successfully train deep networks using AIP (see Fig. 6). Overall, surface normal models seem to be less sensitive to photo-realistic features and higher fidelity settings compared to depth estimation or segmentation. Models trained on high fidelity settings perform 2% better than ones trained on low fidelity, a point we discuss further in Sec. 5.

4.4 Semantic segmentation experiments

Semantic segmentation involves assigning a class label to every pixel on the image. The built-in environments in AIP have fifteen classes, all of which corre-

Table 5: **Semantic segmentation:** Mean intersection over union (IOU) of all classes for different scenarios. Higher values are better.
SC: Sanity Check. **L:** Change in Lighting. **M:** Change in Maps. **F:** Positive Change in Fidelity

Scenario / Fidelity	**Test / Fidelity**	**Goal**	**Global IOU↑**
Brown / Day / High	Brown / Day / High	**SC**	0.8984
Blue / Day / Low	Blue / Day / Low	**SC**	0.4119
Blue / Night / High	Blue / Night / High	**SC**	0.8335
Brown / Day / Low	Brown / Day / Low	**SC**	0.4714
Brown / Day / High	Brown / Night / High	**L**	0.6932
Brown / Night / High	Brown / Day / High	**L**	0.6418
Blue / Day / Low	Blue / Day / High	**F**	0.3862
Brown / Day / Low	Brown / Day / High	**F**	0.4188

sponds to regular household objects, e.g., *wall*, *couch*, *table*, *TV*, *plant*, etc. We use a label of *other* for miscellaneous items. As with the surface normals, we trained different networks from scratch on each scenario. We used mean intersection over union (IOU) of all classes as our evaluation metric. As we can see in Tbl. 5, model performance is directly linked to a scenario's fidelity (see Fig. 6). Semantic segmentation seems to depend heavily on the render scaling and resolution. At lower settings, borders of the objects are blurry, as is their texture. This causes the model to label them as *other* since it cannot surely ascertain their object class, thus lowering the global IOU (see Fig. 7 for an example).

5 Discussion

Below, we discuss some insights from our data ablation experiments that serve as examples of the kind of analyses that AIP makes possible.

Sensitivity to lighting: Changes in lighting are a result of the environment, so they cannot be "fixed" by a better acquisition device. As such, a general-purpose model should be robust to them. However, objects can appear in drastically different ways under different lighting conditions, which did affect performance across all experiments. More specifically, segmentation models are particularly sensitive to differences in lighting. In Fig. 7 both models labeled the top part of the *TV* as *Wall* since they have almost the same color. However, the model trained on a Day setting was much less accurate on the Night image than its counterpart, presumably because the Night setting is darker overall and has more pronounced reflections. The opposite effect is visible in the reverse case (bottom Fig. 7), where the reflection in the lamp confused the model because that level of reflection from sunlight does not exist in the Night lighting.

Our surface normal models are also sensitive to changes in lighting. However, for depth estimation, performance drops only slightly when the lighting is changed, suggesting that local contrast is less important for this problem.

The impact of fidelity on surface normals vs. segmentation: Semantic segmentation is very sensitive to changes in fidelity. When objects are blurred

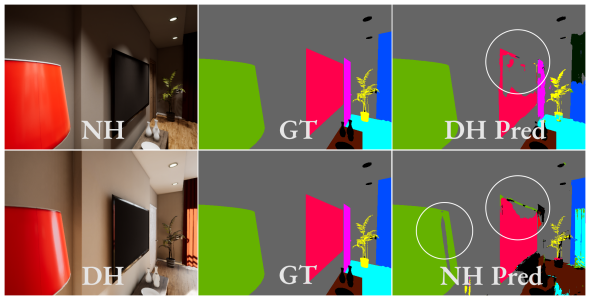

Fig. 7: **Semantic segmentation:** (Image, Ground Truth, Prediction). Top: A model trained on Brown Day High (DH) images segmenting a Brown Night High (NH) image. Bottom: a model trained on Brown Night High tested on Brown Day High. Note the impact of lighting on the final result.

due to lower rendering resolution and lower texture clarity, the model appears to be indecisive about picking an object's class in its border regions. As shown in Fig. 6, we see that the model incorrectly classified border regions as *Other*.

In contrast, surface normal estimation is more robust to these kinds of changes. This difference between these two problems highlights the importance of using data ablation tools. Previous studies, e.g., [5,6,18], mainly focus on the effects of fidelity on their segmentation experiments. Our findings with surface normals, on the other hand, suggest that fidelity as a general feature of the image might not be enough to draw conclusions about the quality of the data. AIP's tools allow us to study other aspects of data, such as texture, structure complexity, lighting and more.

Perspective vs orthographic depth: Orthographic depth projection is when light-rays coming to the camera are assumed to be coming from *infinity*. In calculating the depth ground-truth, this simplification introduces errors to the measurement. We have seen the effects of this assumption on the NYUv2 and DIODE dataset (Fig. 4). Specifically, our models' performance on DIODE was lower in part due to them being trained on perspective depth, which is different from the GT used in DIODE. Although orthographic measurements are currently widely used, we argue that perspective depth, which AIP supports, is the *correct* way to measure depth.

Impact of fidelity on depth estimation: Generally, the performance of models trained on higher fidelity settings are better than those trained in lower fidelity settings (Table 2). However, one exception is when the lower fidelity setting in training better matches the features of the target domain. In Tbl. 3, our low fidelity model does slightly better on NYUv2 than the high-fidelity one. We argue this is due to the blur present in NYUv2, which is also present in our low fidelity settings training set due to its lower render settings, making them visually similar. The DIODE dataset, on the other hand, is much more recent, so the depth ground truth was measured with a more accurate sensor. Due to

the lack of blur and fuzz on the ground-truth, we did not observe the same kind of performance gain on this dataset.

6 Conclusion & Future work

In this work, we presented AI Playground, a data creation and ablation tool for machine learning. Using AIP, we generated different image datasets and conducted experiments that are nearly impossible with real data, thus demonstrating that AIP is a viable tool for data ablation studies in computer vision. We also verified that our high-fidelity trained models can match or exceed the scores achieved by training with real-data. As suggested by other studies [5,6,13,18], we found that higher-fidelity data is linked to better performance in segmentation, but we also found that sensitivity to scene structure, fidelity and lighting scenario of training data varies from task to task. For example, our surface normal and depth estimation models were not as sensitive to fidelity as our segmentation models were. AIP enables us to change individual features, e.g., quality of shadows, quality of reflections, quality of lighting or resolution of textures, and assess their impact on different models based on the current task. More generally, AIP can help researchers find sensitive points in their models and aid them in creating high-quality data for training neural networks for a specific computer vision task.

We are currently working to add more environments to AIP to widen its usage range. These environments include more indoor scenes, outdoor scenes and fully interactive environments allowing individual interaction with objects. Additionally, we'll be providing support for reinforcement learning studies and real-time ray-tracing. There are still many other possible experiments that remain to be explored. For example, UE4 allows the fast change of lighting profile by using HDRI maps. This opens the possibility of adding more specific lighting scenarios like rainy, overcast and foggy. In our future updates, we'll be adding support to introduce intentional camera artifacts such as chromatic aberration, penumbra, lens flares and distortions to help study the effects of using small sensors in capturing data. This is especially useful in robotics since consumer-grade robots rarely come with expensive capture equipment; fine-tuning training to the exact specifications of the camera is a very exciting avenue for future work. Furthermore, we are refining our ground-truth options, including removing texture and changing colors and properties of shaders. These enhancements will enable us to manipulate the scene even further, e.g., changing the pattern in a fabric or changing smoothness of a stone. We believe that AIP will open new and exciting avenues in synthetic data and machine learning.

References

1. Alhashim, I., Wonka, P.: High quality monocular depth estimation via transfer learning. arXiv e-prints **abs/1812.11941**, arXiv:1812.11941 (2018), `https://arxiv.org/abs/1812.11941`

2. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), http://www.blender.org
3. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. CoRR **abs/1406.2283** (2014), http://arxiv.org/abs/1406.2283
4. Epic Games: Unreal engine, https://www.unrealengine.com
5. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis (2016)
6. Haltakov, V., Unger, C., Ilic, S.: Framework for generation of synthetic ground truth data for driver assistance applications. In: GCPR (2013)
7. Khanal, A., Estrada, R.: Dynamic deep networks for retinal vessel segmentation. CoRR **abs/1903.07803** (2019), http://arxiv.org/abs/1903.07803
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (May 2015). https://doi.org/10.1038/nature14539
9. Merrick, L.: Randomized Ablation Feature Importance. arXiv e-prints arXiv:1910.00174 (Sep 2019)
10. Meyes, R., Lu, M., Waubert de Puiseau, C., Meisen, T.: Ablation Studies in Artificial Neural Networks. arXiv e-prints arXiv:1901.08644 (Jan 2019)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning. arXiv e-prints arXiv:1312.5602 (Dec 2013)
12. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
13. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) European Conference on Computer Vision (ECCV). LNCS, vol. 9906, pp. 102–118. Springer International Publishing (2016)
14. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
15. Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
16. Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F.Z., Daniele, A.F., Mostajabi, M., Basart, S., Walter, M.R., Shakhnarovich, G.: DIODE: A Dense Indoor and Outdoor DEpth Dataset. CoRR **abs/1908.00463** (2019), http://arxiv.org/abs/1908.00463
17. Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F.Z., Daniele, A.F., Mostajabi, M., Basart, S., Walter, M.R., Shakhnarovich, G.: Diode: A dense indoor and outdoor depth dataset (2019)
18. Veeravasarapu, V., Rothkopf, C., Visvanathan, R.: Model-driven simulations for computer vision. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1063–1071 (2017)
19. Weichao Qiu, Fangwei Zhong, Y.Z.S.Q.Z.X.T.S.K.Y.W.A.Y.: Unrealcv: Virtual worlds for computer vision. ACM Multimedia Open Source Software Competition (2017)
20. Zeng, J., Tong, Y., Huang, Y., Yan, Q., Sun, W., Chen, J., Wang, Y.: Deep surface normal estimation with hierarchical RGB-D fusion. CoRR **abs/1904.03405** (2019), http://arxiv.org/abs/1904.03405