A3D: Studying Pretrained Representations with Programmable Datasets

Ye Wang
Autodesk, Inc.
ye.wang@autodesk.com

Norman Mu
UC Berkeley
thenorm@berkeley.edu

Daniele Grandi
Autodesk, Inc.
daniele.grandi@autodesk.com

Nicolas Savva
Autodesk, Inc.
nicolas.savva@autodesk.com

Jacob Steinhardt
UC Berkeley
jsteinhardt@berkeley.edu

Abstract

Rendered images have been used to debug models, study inductive biases, and understand transfer learning. To scale up rendered datasets, we construct a pipeline with 40 classes of images including furniture and consumer products, backed by 48,716 distinct object models, 480 environments, and 563 materials. We can easily vary dataset diversity along four axes—object diversity, environment, material, and camera angle, making the dataset "programmable". Using this ability, we systematically study how these axes of data characteristics influence pretrained representations. We generate 21 datasets by reducing diversity along different axes, and study performance on five downstream tasks. We find that reducing environment has the biggest impact on performance and is harder to recover after fine-tuning. We corroborate this by visualizing the models' representations, findings that models trained on diverse environments learn more visually meaningful features.

1. Introduction

Large datasets of naturally occurring images have been critical for computer vision [3, 4, 17]. However, natural data is limiting for some scientific questions, as researchers cannot exert fine-grained control over specific aspects of the image distribution (e.g. diversity of materials, lighting, etc.). To address these limitations, researchers have used rendered images to create controlled experiments, studying inductive bias [31], model debugging [14], and transfer learning [19].

Most rendered datasets constructed for these purposes are small-to-medium scale, comprising either synthetic shapes [31] or a limited collection of realistic shapes [14, 29]. Recently, Mishra et al. [19] constructed a pipeline to generate large rendered datasets for pretraining, backed by $2.3k$ object models and 140 materials. We complement this work with a larger collection of realistic object models and materials—$48.7k$ models with 563 materials and 480 environments. Using optimized renderers, we produce a system—A3D—that can generate a dataset of $200,000$ images in under a day on a small CPU cluster.

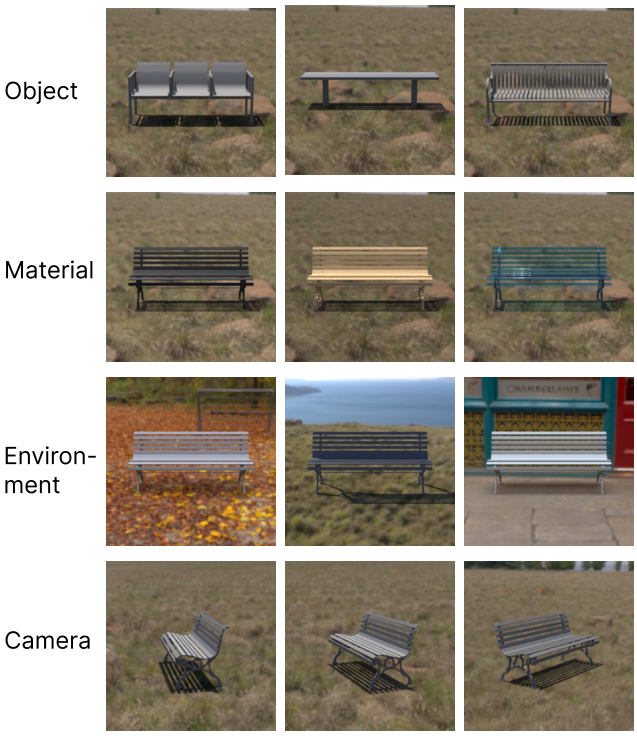

Figure 1. Axes of variation for A3D.

We first check that our rendered images are diverse and realistic enough to be used for pretraining. We generate a base dataset of $200k$ rendered images, and compare to pretraining on $200k$ images from ImageNet. On average across 5 downstream tasks, our rendered images provide

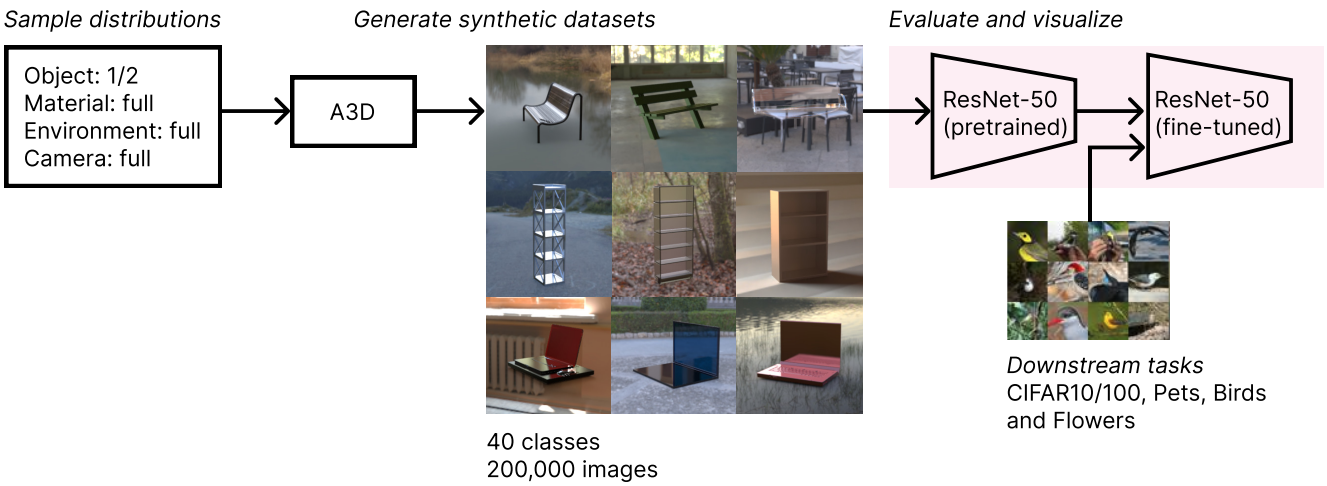

Figure 2. The pipeline used for the experiments. In this example, the dataset is programmed to reduce half the object diversity, while keeping the full diversity from material, environment and camera.

Dataset Name	Objects	Materials	Envir.	Classes
A3D (Ours)	48,716	563	480	40
Task2Sim [19]	2,322	500	—	237
3DB [14]	19	7	408	16

Table 1. An overview of the diversity of unique objects, materials, environments, classes, and total number of images in A3D and other synthetic image datasets.

85% of the performance boost that ImageNet does (relative to a randomly initialized model). This suggests that rendered images induce meaningful learned representations for pretrained models.

We then leverage the *programmable* nature of our dataset to measure the effects of different training distributions on the visual representations learned by neural network models. We render multiple training datasets with different data distributions by reducing the data diversity along each axis: object, environment, material, and camera angle. By training the same architecture on these different distributions, we can isolate the effect that diversity along each axis has on downstream task performance.

We compare our base dataset—which is maximally diverse along each axis of variation—to several less diverse datasets that contain only a subset of camera angles, materials, environments, or object models. Unsurprisingly, models trained on these less diverse datasets generalize poorly to the base distribution.

More surprisingly, training on less diverse data often does not affect the usefulness of pretrained representations. For instance, while removing camera angle variability reduces validation accuracy by over 30%, it reduces the average downstream task accuracy by less than 0.7%. The main exception is environment diversity, which does affect downstream task performance.

To further investigate these findings, we visualize the learned representations of each model. While most models produce similar visualizations, those trained on undiverse environments differ and are typically less semantically meaningful. Together, our findings illustrate the advantage of programmable datasets—by creating well-controlled experiments, we can better understand important phenomena such as the role of data in pretraining.

To summarize, 1) we developed a programmable pipeline for generating high quality rendered images that can be used for pretraining; 2) we generated large datasets of rendered images with rich data on object diversity, materials, environments and camera angles; and 3) we systematically studied how these characteristics affect model behavior. Beyond our current study, programmable datasets are a promising approach to study other scientific topics in machine learning, such as how the training data affects out-of-distribution generalization or reliance on spurious cues.

2. Related Work

Representation learning. Usually operationalized as "pretrain then fine-tune", representation learning has driven significant progress in deep learning. To improve our understanding of pretraining, recent work has focused on quantifying dataset quality and diversity, its effect on learned representations, and the importance of certain dataset features for specific tasks [10,26,30]. Most previous work uses naturally occurring data and is therefore only able to vary the number of classes and overall dataset size (since fine-grained attributes like texture are not annotated). Hendrycks

et al. construct a natural image dataset annotated with camera angle and other metadata, but use it to study OOD robustness rather than pretraining [8]. The most closely related work is Mishra et al. [19], who also use synthetic data to study pretraining. Our work mainly differs by providing more diverse objects and materials, and systematically studying the effect of these on the learned representations.

Synthetic image datasets and pipelines. Given that large image datasets are costly to collect, synthetic image pipelines have proven to be a useful alternative for various computer vision tasks, and several frameworks for photorealistic scene synthesis and multi-modal physical simulation have been developed [6, 16, 29]. Prior work has created large-scale datasets of indoor scenes, by taking 3D models of furniture and varying their location within room environments [15, 18, 25]. Our work expands the availability of photorealistic images by introducing new classes beyond furniture (such as consumer products and vehicles), and utilizing more diverse and numerous environments and textures.

Probing model behavior with synthetic images. Researchers have exploited synthetic image pipelines to exert fine-grained control over image attributes such as backgrounds, lighting, textures, etc. [14, 19]. Prior work has shown that models trained on real images are often biased towards some specific image cues, such as textures, colors, and object pose [1, 7]. These biases have been used to construct adversarial attacks on models [5, 11]. Among these, our work is the first to systematically study the effect of fine-grained attributes such as materials, object models, and environments on pretrained representations[1].

3. A3D: Data Collection and Rendering

In this section, we walk through the A3D rendering pipeline and the data sources for our experiments.

3.1. Rendering

The A3D renderer is based on an unbiased physically-based path tracing CPU renderer [24]. It can be deployed in the cloud, allowing us to easily scale up the generation of large synthetic image datasets. A3D produces an image from 3D objects in mesh formats with specified physically based materials, lighting environments, and camera angles. We describe each axis further below.

Objects models. Similar to other rendering pipelines, our 3D object models are represented as polygonal meshes. We use 48,716 objects in 40 classes from ShapeNet. The models are preprocessed to avoid artifacts as described in Section 3.2. The full list of classes and total number of objects in each class can be found in supplemental materials.

[1] In principal, it should be possible to study this with the pipeline in [19], but they do not appear to do so.

Materials. We curated a large library of physically-based material presets, commonly used in industry products such as Autodesk Fusion 360 and Revit. The material library consists of 563 materials across the categories of metal, wood, plastic, stone, fabric, glass and paint, suitable for the object classes in ShapeNet.

Environments. We use a collection of 480 high dynamic range environment map images from Poly Haven (polyhaven.com). The environments include diverse indoor and outdoor backgrounds as well as different lighting conditions.

Cameras. We set a fixed distance from the camera to the object and center the object. The renderer supports changes of camera azimuth angles.

3.2. Source of Objects and Mesh Repair

Our 3D object models come from ShapeNet core v.2 [2]. We chose this dataset as it has a large number of realistic 3D models, representing many different classes of objects, aligned to have the same orientation. Out of 56 classes in ShapeNet Core v.2, we selected the 40 most numerous classes to use in our experiments. These include models of various man-made objects, such as furniture (chairs, table, sofas, etc.), consumer products (clocks, bottles, laptops, etc.), and vehicles (trains, airplanes, motorcycles, etc.). In total, we use 48,716 different object models.

ShapeNet is curated from public online repositories and existing research datasets. As such, many of the 3D models in the dataset violate simple physics (manifold constraints, etc) and cannot be properly rendered with path tracing engines. Some models do not have the current physical dimensions, which creates additional challenges when adding materials as many materials depend on the scale of the object. The following section discusses in more detail the process used to address the above issues.

Mesh Repair. The original ShapeNet meshes are not suitable for physically-based rendering as they have flipped triangles, overlapping triangles, non-manifold edges, and other common issues with mesh models. Using the meshes as provided results in broken renderings, as shown in Figure 3a. These issues are not specific to ShapeNet and affect other datasets as well [28].

To fix these mesh issues, we process all meshes with ManifoldPlus, a tool for generating clean surface meshes [9]. ManifoldPlus uses a voxelization approach that fixes all aforementioned issues, as shown in Figure 3b. This mesh repair step alters object geometry in some minor ways, especially in areas with sharp corners, but the changes are generally not visible in the final rendering. To address the issue of missing units of measurement, we normalize the scale of all models to a 1m cube. This ensures that rendered textures do not contain unpredictable artifacts and remain

(a) Original ShapeNet meshes

(b) Fixed meshes

Figure 3. Example renderings made using the original ShapeNet meshes (a), compared to the same meshes rendered after mesh repair (b).

(a) Original (b) Scaled

Figure 4. Example renderings showing a mesh with no scaling factor (a), compared to the same mesh rendered after scaling (b), which accentuates the texture of the wooden material.

within a reasonable scale range. The effect of this scaling is shown in Figure 4.

4. Programmable Datasets

A key advantage of rendered data is that we can flexibly control the data distribution by exploiting metadata information. To showcase these advantages, we generated a base training set and validation set with full variation, as well as 4 sets subsampling along each of the four axes of variation, and 1 singleton set.

4.1. Axes of Variation

We defined four axes of variation for our dataset, corresponding to the four main rendering parameters consumed by the A3D pipeline: camera (azimuth) angle, material, environment, and object model. By controlling the range of these four axes we can programmatically render datasets with varying levels of diversity. We first set aside 10% of the models in each class for use in the validation set, which is sampled according to the same distribution as the base training set. We then construct a base training set and multiple subsampled training sets as described below.

4.2. Base Training Set and Validation Set

The base training set samples from the full range of values across all four axes: -90° to 90° for camera angles, all 563 materials across 7 categories (see Section 3.1 for details), all 480 environments, and all $48,716$ models. This allows for billions of possible combinations, from which we sample 5000 from each of the 40 classes. The training set contains $200,000$ images in total.

The validation set is sampled separately from the same distribution as the base training set, with 500 images from each class, and $20,000$ in total.

4.3. Subsampled Training Sets

To study the influence of the four axes of variation, we created 20 subsampled training distributions, each also containing $200,000$ rendered images. We create each dataset by reducing the range of eligible values in one particular axis down to $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and single. We subsample hierarchically, so the environments in the $\frac{1}{4}$-environment condition are a strict subset of those in the $\frac{1}{2}$-environment condition. The specific subsampled distributions are as follows:

Object models. Object models are sampled uniformly at random in each class. For example, the table class has $8,436$ object models. The base training set samples uniformly among $8,436$ objects. The $\frac{1}{2}$-object training set randomly chooses $4,218$ objects, and uniformly samples among them. The other axes (materials, environments, and camera angles) are sampled the same way as in the base training set.

Material. To generate a material, we first choose a category uniformly at random, then select a material within that category. Each subsampled datasets contains all 7 categories of materials (fabric, glass, metal, paint, plastic, stone, and wood), but subsamples materials within each category by a factor 2, 4, 8, or 16.

Environment. We subsample the full set of environments by factors of either 2, 4, 8, or 16.

Camera angles. Camera angles are uniformly sampled from the range $-(90/N)°$ to $(90/N)°$, for $N \in \{2, 4, 8, 16\}$.

4.4. Singleton Training Sets

In addition to subsampling, we consider a more extreme restriction where an axis is fixed to a *single* value. For camera, the angle is fixed at $0°$. The single environment is fixed to kloppenheim from Poly Haven, a field with the sky during the day. The material is light gray plastic. The single object training set uses one object from each class for rendering.

Dataset	Train	Test	Classes
CIFAR-10 [13]	50000	10000	10
CIFAR-100 [13]	50000	10000	100
Pets [23]	3680	3669	37
Flowers [20]	1020	6149	102
Birds [27]	5994	5794	200

Table 2. The number of classes and images in the train and test splits of the datasets used for fine-tuning.

5. Data Characteristics and Model Behavior

In this section, we design experiments to study how different data dimensions influence the model behavior after pretraining and fine-tuning. We then visualize different models and discuss the results.

5.1. Experiments

As described in Section 4, we use our pipeline to render the base training set, 16 subsampled training sets along the four axes of variations, object, material, environment and camera angle, and 4 singleton training sets. Each training set contains $200,000$ images and 40 classes.

Pretraining. We pretrain a ResNet-50 and ResNet-18 on each of these 21 datasets for the task of image classification. For comparison, we also pretrain on a $200,000$-image subset of ImageNet, and also construct an untrained, randomly initialized network. These serve as upper and lower bounds, respectively, for the performance of models trained on our rendered images. All models are trained for 100 epochs with standard data augmentation at 224×224 px and a cosine learning rate schedule using SGD with momentum. We also compute validation accuracy, as measured on our $20,000$-image validation set. This helps benchmark the severity of different subsampled data distributions.

Fine-tuning. To evaluate our models' representations, we select five downstream tasks for evaluation: CIFAR-10 [13], CIFAR-100 [13], Oxford Pets [23], Oxford Flowers [20], and Caltech-UCSD Birds 2011 [27]. Table 2 shows the number of classes, as well as the number of images used for training and testing during fine-tuning. For comparison, we also fine-tuned on 2% of the A3D base training set (4000 images), which is similar in size to the Birds, Pets and Flowers training sets. For the downstream tasks, we train all models by sweeping over learning rates of $\{0.1, 0.3\}$ and weight decays of {5e-4, 1e-3}, and report the best result.

5.2. Model Performance

Performance gain from pretraining. Among the five downstream tasks, on Pets, Birds and Flowers, we see a significant improvement of performance due to pretraining. Training ResNet-50 from scratch with random initialization gives 48.2% accuracy on average for the three datasets. Using a network pretrained on the A3D base set improves performance by 30%, less than a 5% difference from pretaining on ImageNet-200K, which contains 1,000 classes and (unlike A3D) includes classes of pets, birds and flowers. See Table 3 for detailed numbers.

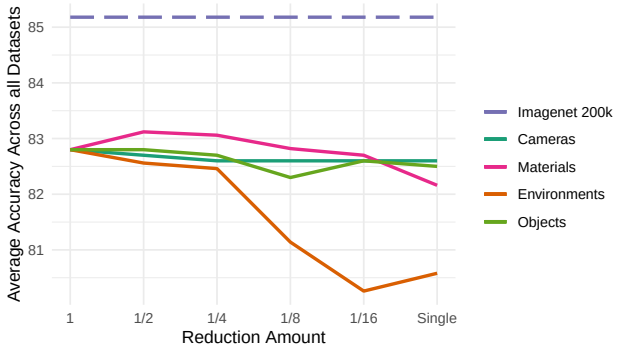

Figure 5. ResNet-50 results, showing how accuracy is most affected by reduction of environment diversity.

On CIFAR-10 and CIFAR-100, for all models we observe only limited differences between the randomly initialized network and pretrained ones. This may be because the CIFAR training sets are large compared to Birds, Flowers, and Pets, so benefit less from the inductive bias of pretraining (though the inductive bias should still help in the limit [12]).

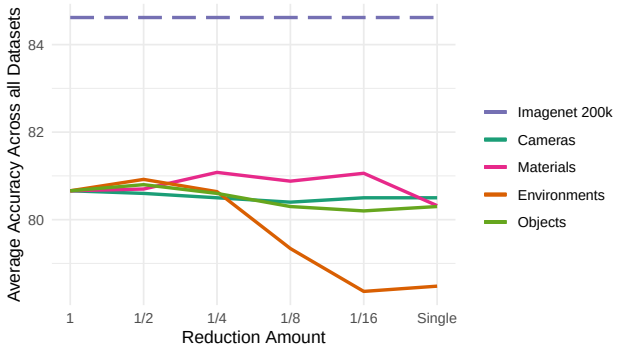

Figure 6. ResNet-18 results, showing how accuracy is most affected by reduction of environment diversity.

Effect of diversity. Along the object model, material and camera angle axes, reducing diversity has little effect ($< 1\%$ drop in average accuracy) even when we consider the most restricted subsampled training sets (see Figure 5). The exception is environment diversity, where we observe a 3% drop in average accuracy (across Pets, Flowers, and Birds) as we reduce environment diversity seen during pretraining. The reduction is consistent across these three tasks,

Train Data		Val	A3D	Pets	Flowers	Birds	C-10	C-100
Random		—	—	54.1	50.1	40.5	95.0	78.9
IN-200K		—	—	84.0	90.3	76.1	95.7	79.8
A3D Base	full	83.2	—	81.7	87.7	69.4	95.5	79.7
Camera	1/2	73.5	79.9	81.7	87.7	69.1	95.4	79.6
	1/4	61.8	78.1	82.7	87.1	67.8	95.3	79.9
	1/8	54.2	76.5	82.7	87.0	68.2	95.5	79.7
	1/16	50.5	75.6	82.0	86.9	68.8	95.4	79.7
	single	46.8	75.1	82.5	87.2	68.0	95.4	79.7
Material	1/2	83.4	83.3	83.1	87.8	69.7	95.4	79.6
	1/4	82.9	83.0	82.4	88.2	69.2	95.7	79.8
	1/8	82.2	83.0	82.2	88.4	68.6	95.3	79.6
	1/16	79.9	82.2	81.8	87.9	68.8	95.4	79.6
	single	71.5	79.0	81.3	86.0	68.8	95.2	79.5
Environment	1/2	82.9	82.5	82.1	86.7	68.9	95.2	79.9
	1/4	82.2	82.8	80.9	88.3	68.0	95.4	79.7
	1/8	67.4	74.3	79.4	84.8	65.9	95.5	80.1
	1/16	53.3	75.6	77.2	84.1	65.0	95.2	79.8
	single	20.4	71.6	79.3	82.2	66.6	95.0	79.8
Object	1/2	81.6	81.6	82.0	86.7	70.0	95.6	79.8
	1/4	79.2	79.6	81.6	86.7	69.9	95.4	79.9
	1/8	75.4	79.6	81.5	86.8	68.6	95.2	79.3
	1/16	70.7	78.8	82.1	86.6	69.0	95.4	79.7
	single	46.9	74.3	82.3	86.7	68.0	95.5	79.9

Table 3. ResNet-50 Performance. The Val column reports accuracy of the pretrained model on the A3D base validation set. Other columns report fine-tuning performance.

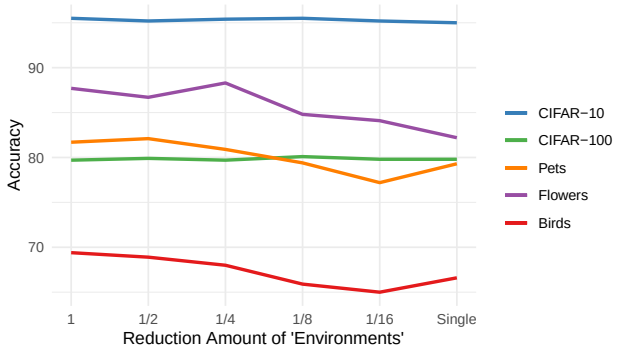

Figure 7. Model performance decreases as reducing environments. Different tasks have different sensitivity to environment reduction.

and the Flowers task is most sensitive: accuracy decreases from 87.7% to 82.2% (See Figure 7 for details).

Comparison on A3D Base. Despite the small changes in fine-tuned model performance, the pretrained models behave differently—the 'Val' column in Table 3 shows that the base validation accuracy drops significantly when we decrease diversity along any axis, with environments again having the largest effect.

As a further check, we fine-tune our pre-trained models on the full A3D base distribution and find that this significantly reduces the observed drop in most cases, but does

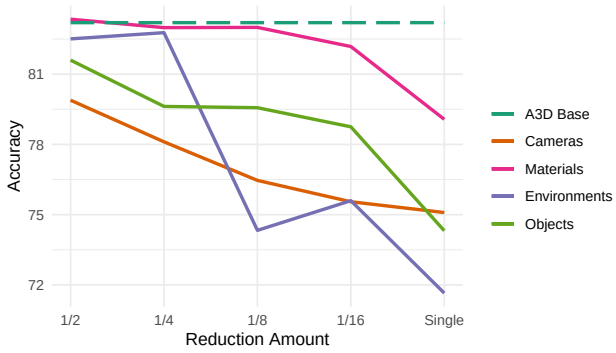

Figure 8. Model performance on A3D validation set after fine-tuning with a subset of A3D Base.

not always remove it. After fine-tuning, material reduction influences the performance the least, and environment reduction influences the performance the most (see Figure 8). With $\frac{1}{8}$ of the environments, performance decreases close to 10%. In comparison, $\frac{1}{8}$ of materials and camera angles have nearly no influence on accuracy, and $\frac{1}{8}$ of objects has a difference of less than 4%.

Figure 9. Five evaluation images for model visualization.

5.3. Model Visualization

To further investigate how models are influenced by the reduction of data diversity along each axis, we use model visualization based on Olah et al. [21, 22] to build intuition for what is happening inside the models. We use the Lucent library, a PyTorch adaption of Lucid [22], for visualization. We evaluated different pretrained ResNet-50 models before and after fine-tuning on Pets on 5 unseen images (Figure 9). We select 6 different positions (layers) in ResNet-50 for evaluation. Half of the positions are after convolutional layers and half are after residual connections. We find consistent results across different images, so in the figures below, we use the bird picture as the main one to demonstrate how the network changes after fine-tuning and after reduction of data diversity along different axes. Due to the limits of pages, we choose one convolution layer in Stage 2 and one after residual connection position in Stage 3 to demonstrate the results. More visualization examples can be found in supplemental materials. The full set of results can be ex-

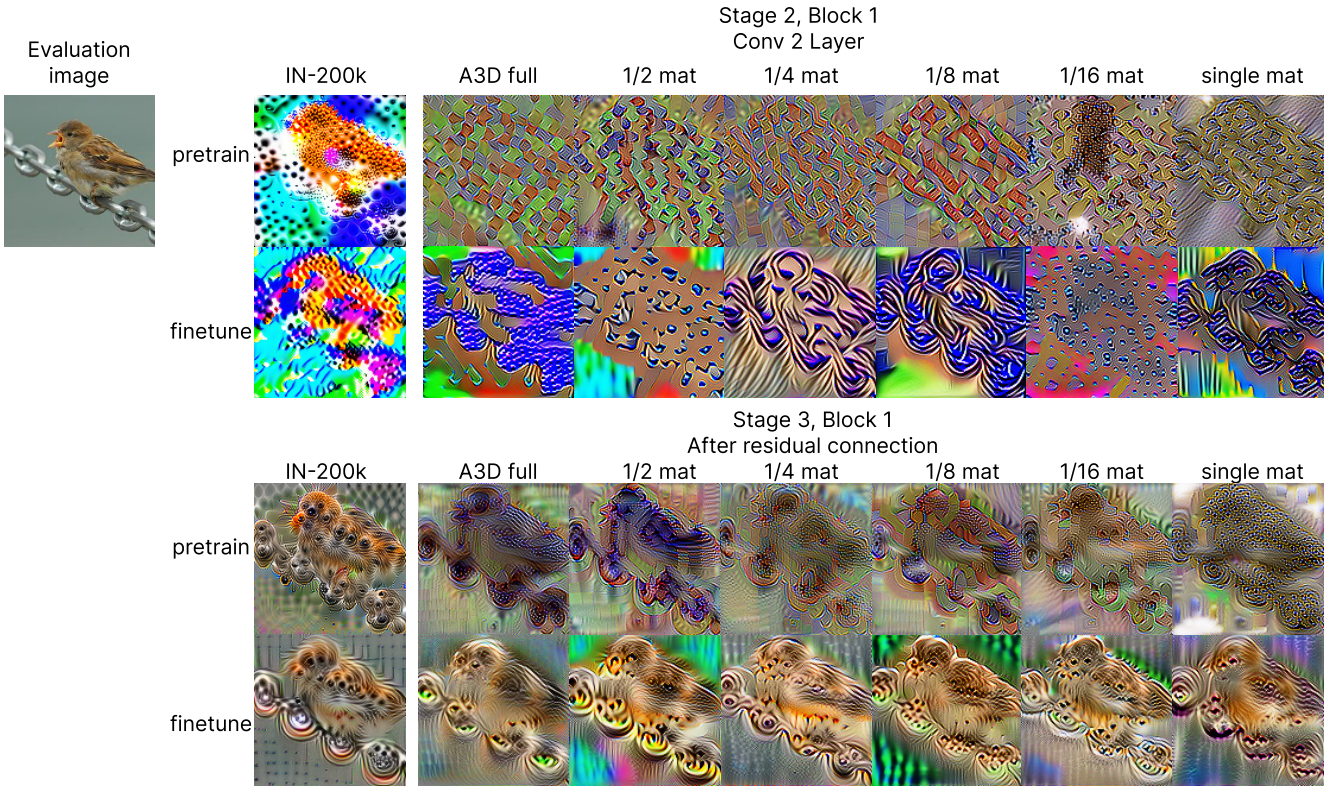

Figure 10. Visualizing how model features change with material reduction.

plored as an online app [2].

A3D and ImageNet-200K. We compared the models trained on ImageNet-200K and A3D Base before and after fine-tuning (Figure 10). Before fine-tuning (see the "pretrain" rows of images in Figure 10), a visual difference between ImageNet and A3D Base can be seen. In the earlier convolutional layer (Stage 2, Block 1, Conv 2 Layer), the bird cannot be distinguished from the background, whereas after the residual connection (Stage 3, Block 1, After residual connection), the model trained on A3D can distinguish the foreground, but cannot pick up animal features, such as feathers or eyes, which are instead present in the ImageNet model visualization. This is likely due to A3D having fewer classes than ImageNet-200K, and lacking any animal classes. After fine-tuning (see the "finetune" rows of images in Figure 10), the visualization from the A3D model becomes significantly more similar to ImageNet-200K, especially for the later layers.

Material Reduction. The reduction of materials result in significant visual differences on pretrained models, although these differences become unnoticeable after fine-tuning. Taking the single material model as an example (Figure 11), before fine-tuning, the model produces dotted patterns with the same size across the visualization. After fine-tuning, the dotted patterns disappear and instead, we see curves and lines present also in models with more materials (Figure 10). This is consistent with what we find in model performance (Section 5.2) – the reduction of materials does not have significant impact on model performance if the model is then fine-tuned on other data.

Environment Reduction. The models trained on diverse environments learn more visually meaningful features. After reducing environment diversity to $\frac{1}{8}$, we observe significant visual differences before and after fine-tuning (Figure 12). Once again we observe that the earlier convolutional layers suffer more from reduction in diversity compared to later layers.

6. Limitations and Future Work

We only visually inspect the model visualization results with our current approach. Future efforts can increase the total number of evaluation images and quantify the visualization differences as in previous work [21].

The axes of variation in A3D could be extended. For materials, A3D currently only supports assigning a single

[2] https://github.com/whyzcandy/a3d/

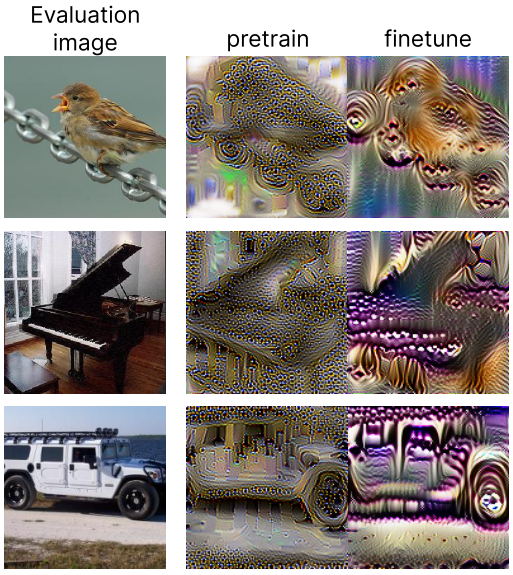

Figure 11. Comparing single material models before and after fine-tuning after residual connections in Block 1, Stage 3 of ResNet-50.

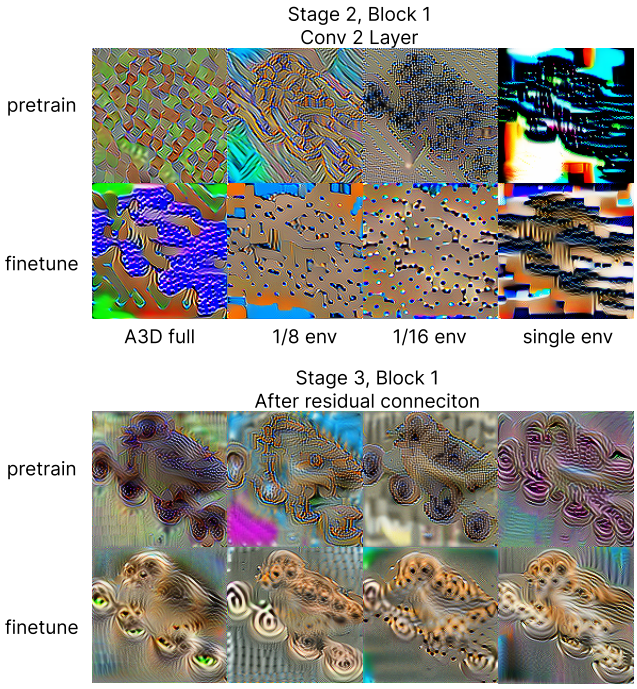

Figure 12. Visualizing how model features change with environment reduction.

material per object due to the complexity of segmenting and processing the 3D meshes. In the future, modifying materials for individual components could increase material diversity. For cameras, the camera manipulation could be extended beyond azimuth angles and fixed distance. For environments, lighting and backgrounds could be separated to have finer-grained control over the datasets.

7. Conclusion

In this work, we develop the A3D pipeline for generating large programmable datasets with specific object, material, environment, and camera diversity. Using several unique datasets generated with the pipeline, we study how data diversity along four axes influenced pretrained representations. As an outcome, we find that the models trained on diverse environments learn more visually meaningful features than the models with diverse materials, objects or camera angles, pointing to the importance of environments for tasks that require synthetic image generation.

Acknowledgements

The authors thank Jean-Stanislas Denain for helping with the model visualizations. Special thanks to the Autodesk Graphics Platform group for providing valuable input and resources. This work is supported by Berkeley Artificial Intelligence Research (BAIR) industrial alliance programs, the Berkeley Deep Drive project, and gifts from Meta and Open Philanthropy. This work is also supported by NSF SaTC Award CNS-1804794.

References

[1] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019. 3

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 3

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[5] Tommaso Dreossi, Somesh Jha, and Sanjit A Seshia. Semantic adversarial deep learning. In *International Conference on Computer Aided Verification*, pages 3–26. Springer, 2018. 3

[6] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek

Bhandwaldar, Nick Haber, Megumi Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020. 3

[7] 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. *arXiv preprint arXiv:1811.12231*, 2018. 3

[8] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 3

[9] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups, 2020. 3

[10] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 2

[11] Lakshya Jain, Varun Chandrasekaran, Uyeong Jang, Wilson Wu, Andrew Lee, Andy Yan, Steven Chen, Somesh Jha, and Sanjit A Seshia. Analyzing and improving neural networks by generating semantic counterexamples through differentiable rendering. *arXiv preprint arXiv:1910.00727*, 2019. 3

[12] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020. 5

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[14] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021. 1, 2, 3

[15] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multisensor photo-realistic indoor scenes dataset. *arXiv preprint arXiv:1809.00716*, 2018. 3

[16] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, et al. Openrooms: An open framework for photorealistic indoor scene datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7190–7199, 2021. 3

[17] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018. 1

[18] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2678–2687, 2017. 3

[19] Samarth Mishra, Rameswar Panda, Cheng Perng Phoo, Chun-Fu Chen, Leonid Karlinsky, Kate Saenko, Venkatesh Saligrama, and Rogerio S Feris. Task2sim: Towards effective pre-training and transfer from synthetic data. *arXiv preprint arXiv:2112.00054*, 2021. 1, 2, 3

[20] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 5

[21] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 5(4):e00024–002, 2020. 6, 7

[22] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017. 6

[23] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 5

[24] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 3

[25] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10912–10922, 2021. 3

[26] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 2

[27] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 5

[28] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 3

[29] Yue Yao, Liang Zheng, Xiaodong Yang, Milind Naphade, and Tom Gedeon. Simulating content consistent vehicle datasets with attribute descent. In *European Conference on Computer Vision*, pages 775–791. Springer, 2020. 1, 3

[30] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 2

[31] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. *Advances in Neural Information Processing Systems*, 31, 2018. 1