PROCEDURAL ADVERSARIAL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

For many computer vision tasks, neural networks can be trained to produce a low average error, yet they still fail on seemingly simple scenes. Reducing these failures requires first finding them, but they may not be in a fixed dataset. We propose procedural adversarial training (PAT), a minimax framework in which a learner trains a model while an adversary searches the input space of a procedural generator. The procedural generator is an interpretable program mapping a low-dimensional vector of control parameters (e.g. layout, lighting, pose) to a rendered image and its label. We introduce the Simple-PAT algorithm, which uses zeroth-order optimization in the procedural generator's parameter space to propose adversarial training samples that are generated on demand. We also create PAT-Dojo, a collection of efficient procedural generators for surface normal estimation, chair segmentation, and human pose estimation. Against a cross-model black-box adversary with 50M queries, Simple-PAT reduces worst case error by a geometric mean of $2.09\times$ compared to standard training techniques.

1 INTRODUCTION

Vision models deployed in the wild are often relied on to produce high-quality predictions in all situations. This is particularly important for self-driving cars, surgical robots, and heavy machinery, where failures in perception can lead to dangerous and costly mistakes. However, even well-trained neural networks often fail on simple scenes or natural adversarial examples (Hendrycks et al., 2021; Alcorn et al., 2019). This is not easily solved by scaling data or model size, as foundation VLMs are also susceptible to these natural adversarial examples (Li et al., 2024). These failures motivate training and evaluation that explicitly account for worst-case error.

How can we train a model that has low worst-case error? A natural and existing idea is to define a minimax learning objective: minimizing the worst case error, defined as the maximum error over all possible instances of the task space. While simple to state, this minimax objective is challenging to operationalize: typically, we are given a fixed finite sample of the task instances such as a fixed set of labeled images; this sample does not allow us to reliably estimate the maximum error over all possible images because the sample may be biased and miss important failure cases. If we cannot reliably measure the worst case error, we cannot reliably improve it.

To discover missed failure modes and estimate worst case error, prior work in robust and adversarial machine learning has attempted to perturb existing images (e.g.Goodfellow et al. (2015); Engstrom et al. (2019); Hosseini & Poovendran (2018)). However, such perturbations are usually limited to simple transformations such as color jitter and spatial cropping, and cover only a tiny subset of valid image transformations. This is because automatic generation of physically valid perturbations of existing images covering view point change, dynamic lighting, shape deformation remains active research and is plagued by its own failure modes. On other hand, collecting additional real world data can be slow and expensive.

Thus, a key challenge of improving worst case error lies in scalable discovery and generation of failure cases. In this work, we address this by introducing Procedural Adversarial Training (PAT), a learning framework whose key idea is to adversarially train models under a procedural generator. A procedural generator generates data through interpretable code: an image of a tree can be procedurally generated from scratch by code that simulates tree growth, bark material, light transport, and perspective projection. Procedural generation can create diverse photorealistic images (Raistrick et al., 2023; 2024), visual ground truth, and useful training data (Ma et al., 2026; Yan et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

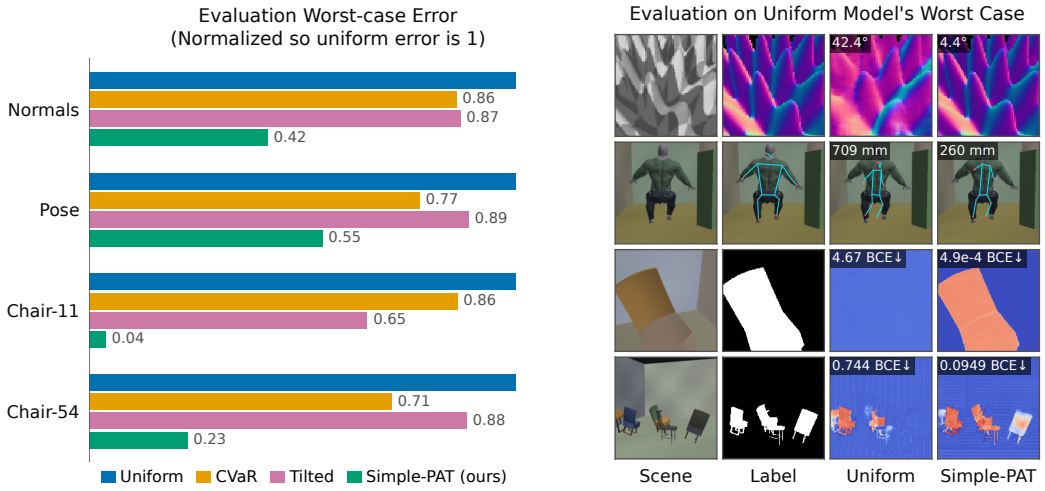

Figure 1: Left: Simple-PAT reduces worst-case error more effectively than uniform training and alternative methods. Right: Simple-PAT models are able to do much better on cases uniformly trained models found challenging. Uniform refers to the "uniform-Muon" setting, which is the stronger of the uniform training methods.

Recent advances in agentic 3D coding capabilities, as demonstrated by GPT-Astra, have made procedural generation even more promising, because the conventionally laborious manual development of procedural rules could be substantially sped up by coding agents (OpenAI, 2026).

Procedural generation enables scalable discovery and generation of failure cases. A well designed procedural generator can offer a combination of properties unavailable from image-space perturbation: (1) wide coverage of task instances, (2) physically valid perturbations, (3) a small number of control parameters that help traverse the task space, (4) on-demand generation of new task instances. For example, for pose estimation of articulated objects, a procedural generator can easily and controllably create valid and varied poses from diverse viewing angles and lighting conditions.

PAT poses new algorithmic design questions. Like conventional adversarial learning, PAT consists of a learner that trains a model to minimize worst case error and an adversary that tries to maximize it. Unlike conventional adversarial learning, the adversary is equipped with a new weapon: a procedural generator that can create, on demand and with precise control, a wide range of new task instances that could trip up the model. This opens up new questions: with procedural generation, how should one train a robust model?

We propose Simple-PAT, a simple algorithm for procedural adversarial training using adversaries that attack via zeroth-order optimization. Simple-PAT consists of a learner that trains the model through minibatch gradient descent and an adversary that uses Covariance Matrix Adaptation Evolution Strategy (CMA-ES) Hansen & Ostermeier (2001) to search on procedural controls to generate the next minibatch based on existing cases the model finds challenging.

To study algorithmic design of PAT, we perform experiments using procedural synthetic data on three vision tasks: surface normal estimation, human pose estimation, and image segmentation. We design these generators to control the scenes' layout, lighting, shapes, and poses using between 11 and 54 parameters, prioritizing simplicity and efficiency of generation, to facilitate algorithm development. We refer to this suite of generators as PAT-Dojo.

PAT also poses interesting questions in evaluation. In general, it is impossible to measure the true worst case error, because the number of possible task instances can be infinite but the adversary has limited compute. In this work we use best effort adversaries that search for worst cases by repeatedly probing models as black boxes subject to a compute budget. We use an adversary allowed to observe only a single model, as well as one allowed to pool worst cases across models.

We first compare Simple-PAT to a naive learner that samples uniformly at each batch ("uniform training"). We evaluate with a single-model adversary that treats each network as a black box and

searches for the worst case with 2M queries. Here, we find that Simple-PAT reduces the discoverable worst case by $1.83\times$–$25.9\times$ across tasks, with a geometric mean of $4.71\times$ (results shown in Figure 1). We also extend the evaluation to a cross-model adversary with a 50M sample budget. In this setting, compared to uniform training, Simple-PAT reduces the searchable worst case by a geometric mean factor of $2.09\times$ across tasks.

In addition, we adversarially evaluate state-of-the-art foundation models on PAT-Dojo. We find that SOTA foundation models have surprising failure modes that can be easily discovered through a procedural generator. Using only a budget of less than 2×10^5 queries, an adversary exposes foundation model failures over $3\times$ worse than found with the 2M sample search on the specialist models trained through Simple-PAT. We note that this is not an apples-to-apples comparison as the specialist models are narrowly trained and lack the general capabilities of the foundation models. But these results show that although PAT-Dojo prioritizes simplicity and efficiency in design, it is a meaningful testbed that is already sufficiently challenging for state-of-art foundation models.

Our contributions are threefold: (1) Procedural Adversarial Training (PAT), a minimax framework that adversarially trains models with a procedural generator; (2) Simple-PAT, a simple yet effective PAT algorithm; (3) PAT-Dojo, a suite of simple, efficient but sufficiently hard procedural generators for vision tasks, designed to facilitate the study of PAT algorithms. Code will be open sourced.

2 Related Work

Natural Adversarial Examples. Image classifiers and vision-language models are known to fail on unmodified natural images that humans label easily (Hendrycks et al., 2021; Li et al., 2024). Search can also reveal these failures systematically, by adjusting the pose as in Alcorn et al. (2019), or other elements including object placement and lighting Zeng et al. (2019); Leclerc et al. (2022).

Adversarial training and robust objectives. Classical adversarial examples were defined as small pixel perturbations (Szegedy et al., 2014; Goodfellow et al., 2015), and adversarial training addresses this by minimizing loss under perturbations within an L^∞ ball around each image (Madry et al., 2017). However, these perturbations are limited and the inner optimization loop during training can be costly. Alternative methods instead emphasize difficult examples through a distributionally robust training objective. CVaR averages the worst α fraction of samples (Rockafellar & Uryasev, 2000; Levy et al., 2020), and tilted ERM weights each example in proportion to $exp(\beta\ell)$ for a loss ℓ and hyperparameter β (Li et al., 2021).

Adversarial scenario and environment generation. Research in reinforcement learning and driving agents has also searched for challenging training instances. AdvSim uses a black-box optimizer to perturb actor trajectories in recorded driving logs to generate safety-critical scenarios. They train on these scenarios to improve the performance of their system (Wang et al., 2021). In reinforcement learning, PAIRED trains a level generator to maximize the agent's regret (Dennis et al., 2020). ACCEL applies random mutations to high-regret levels to generate new difficult instances (Parker-Holder et al., 2022). Importantly, both optimize regret instead of loss since directly optimizing loss would produce impossible tasks in their environments. Additionally, none of these works attempt to evaluate the worst-case performance over the entire parameter space.

Adversarial game formulations have also been used to develop safe and robust learning-based controllers, through minimax policy training against disturbance forces (Pinto et al., 2017) and safety control against worst-case disturbances (Fisac et al., 2018).

3 Procedurally Generated Tasks

A procedural generator is a deterministic map g from a parameter vector $u \in [0, 1]^d$ to a rendered image x_u and its ground-truth label y_u. We can view this as an infinite source of controllable data for both training and evaluation. In contrast, a fixed dataset can be viewed as N queries of the generator g determined by some unknown selection process. Access to the generator has the following consequences:

1. The training data need not be fixed in advance. At each training step we select B parameter vectors to form the next batch, and this selection can depend on the current model. A

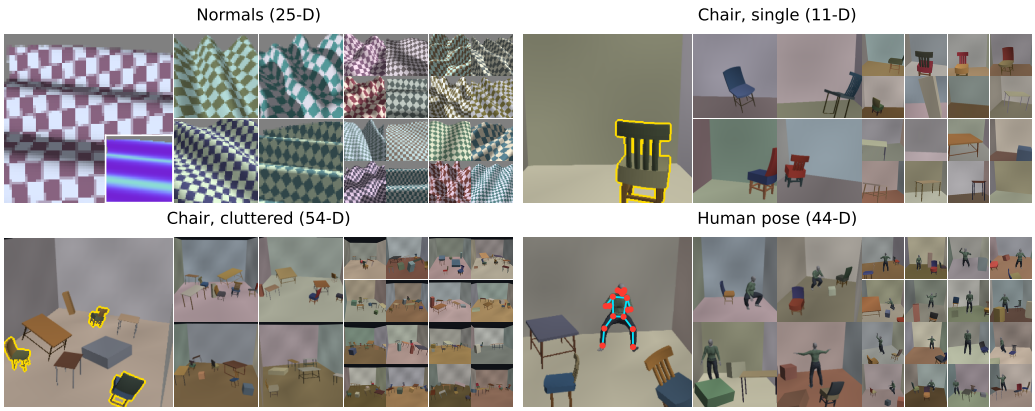

Figure 2: Samples from our procedural generators. The leftmost panel of each task shows a visualization of the ground truth labels

fixed dataset only allows choices among its N members, and must repeat after a number of training steps. If certain regions of the parameter space are particularly valuable, the number of unique useful samples may be quite small. Given access to g, adaptive training can concentrate arbitrarily on these useful regions of the parameter space and present the model with new examples at every step.

2. The worst case can be evaluated over the entire parameter space. We consider the absolute worst-case

$$W(\theta) = \sup_{u \in [0,1]^d} \ell(\theta, u) \qquad \ell(\theta, u) = L\big(f_\theta(x_u), y_u\big).$$

This loss is consistently estimable, since as we densely sample the parameter space our estimate of the worst-case loss will converge to its true value (under mild conditions [1]). However, the structure of the procedural generator enables more effective search. Each parameter controls a perceptual quality of the scene (e.g. lighting, object shape, layout). Since these qualities also relate to the difficulty of the scene, we expect the difficulty to vary reasonably smoothly in the parameter space. This enables search algorithms to identify the challenging cases much more quickly than random sampling.

In order to make $W(\theta)$ meaningful, it must be the case that for every $u \in [0,1]^d$, y_u is solvable given x_u. Otherwise the worst case error will be an ambiguous scene. We build four procedural generators to represent a variety of vision tasks. We design them to be solvable and extremely efficient so that we can generate data before each training step and practically search for the worst cases.

3.1 Normal Estimation

For the normal estimation task, the model must estimate the 3-dimensional surface normal vector at every pixel on a curved surface. This surface is a heightfield determined by a mixture of sinusoidal waves. The material is a checkerboard pattern rendered with Lambertian shading under a directional light with an ambient term. The presence of a repetitive checkerboard pattern makes the normal estimation tractable, since any warping in the pattern is due to surface normal variation. The network must also learn to appropriately handle shadows and lighting cues.

The generator has 25 parameters, with 13 parameters specifying the waves, 2 describing the light azimuth and elevation, and 10 describing the checkerboard texture. This includes 24 continuous parameters, and 1 discrete parameter specifying the number of waves.

We use the metric of average angular error (in degrees) to describe models' performance.

[1] This requires the loss to be continuous in a region of positive measure near its supremum. Each procedural parameter is either continuous or a finite discrete choice, and our networks and task loss functions are continuous, which is sufficient.

3.2 Object Segmentation

Our object segmentation task asks models to classify each pixel as belonging to a chair object or not. The generator places chairs and distractor objects within a room while ensuring the chair is not occluded excessively. We have a simpler 11D variant and a more complex 54D variant.

The procedural objects that can be placed in the room include chairs, tables, desks, and cubes. These objects are adapted from ProcFunc implementations of objects in Infinigen, with a custom backend designed for rapid generation (Raistrick et al., 2026).

In the 11D chair tasks, we have 4 parameters for the room shape and camera placement, and we have 7 parameters for the object type, shape, and position.

For the 54D chair task, we place up to 8 objects on a 16×16 grid of positions on the floor; each object has a parameter d_i that controls its position. Objects are placed sequentially. For object i, we identify valid grid cells, partition $[0, 1]$ into equal intervals for each valid cell, and select the cell whose interval contains d_i. We determine cell validity so objects do not intersect and are placed inside the room. We further restrict cells based on occlusion in the camera frame, so that no object is heavily occluded and unidentifiable (e.g. a chair where you can only see its legs). This arrangement procedure ensures that the problem is tractable, but also introduces discontinuities and sensitive parameters.

We evaluate models based on average pixel binary cross entropy (BCE) loss.

3.3 Human Pose Estimation

The human pose estimation task builds on the scene generator from the object segmentation task. It includes three object slots, and one human slot.

The human figure uses the Anny model with clothing from MakeHuman MPFB2. We include five identity coordinates (gender, age, muscle, weight, proportions), and 8 pose coordinates. For pose, we perform dimensionality reduction on the poses from the CMU mocap database (CMU Graphics Lab). Our goal is to represent plausible poses instead of every possible joint configuration, which would produce self-intersection and other non-physical configurations.

Similar to the object segmentation task, we ensure the human figure is visible and unoccluded by only placing it in valid cells.

For pose estimation, we use root-aligned mean per-joint position error (MPJPE) as our metric: $MPJPE = \frac{1}{|V|} \sum_{j \in V} \|(\hat{p}_j - \hat{p}_{root}) - (p_j - p_{root})\|_2$ where p_{root} is the average of the left and right hip positions. We only calculate the error for joints visible in the scene.

3.4 Performance Characteristics

Task	Dimension	Resolution	Scenes/second
Normals	25	64^2	8700
Chair-11	11	256^2	1000
Chair-54	54	256^2	340
Pose	44	256^2	290

Table 1: Performance of procedural generators. Median over 20 batches measured on a NVIDIA RTX4090 GPU at a batch size of 256.

In Table 1 we show the performance of our procedural generators. Scene generation and rendering are optimized for batched throughput and GPU efficiency, making them ideal for neural network training pipelines.

4 Adversarial Training

We design our adversarial training procedure Simple-PAT around three principles: (1) *coverage:* the model must be presented with examples near the worst-case example (2) *diversity:* each batch the model receives should contain maximally distinct examples (3) *computation reuse:* samples which

we render and evaluate should be used to update the network. Though simple, these principles allow us to design an algorithm whose training throughput is comparable to uniform training, while achieving a large reduction in evaluation worst-case error.

Algorithm 1 describes Simple-PAT. We maintain k CMA-ES instances $\pi_1, \ldots, \pi_k$. At each step we select a batch using n_0 uniform samples and λ samples from each π_i. We run a forward pass on every data point and use the results to update the CMA-ES parameters. We then run a backward and optimization step, as in standard neural network training. Every R steps we reset the weakest CMA-ES algorithms as determined by a fitness function.

CMA-ES Optimization. CMA-ES is a zeroth-order optimizer that uses a multivariate normal distribution to sample candidate solutions to the optimization problem (Hansen & Ostermeier, 2001). This is amenable to our setup due to batched computation and diversity in samples. At each step, λ points are sampled from $N(\mu, \Sigma)$, these points are evaluated by fitness, and μ and Σ are updated. We constrain Σ to be diagonal since λ is relatively small. For fitness, we use the evaluation metric of each task. This may differ from the training loss function due to stability or smoothness reasons.

Algorithm 1 Simple-PAT

Require: generator $g : [0,1]^d \to \mathcal{X} \times \mathcal{Y}$, model f_θ, per-scene loss ℓ, reset period R, sampler parameters n_0, λ

init k CMA-ES distributions $\pi_1, \ldots, \pi_k$ on $[0,1]^d$
for $t = 1, \ldots, T$ **do**
  $U \leftarrow \{u \sim \pi_i\}_{i \le k}^{\times \lambda} \cup \{u \sim \mathcal{U}[0,1]^d\}^{\times n_0}$
  $(x, y) \leftarrow g(U)$
  $\hat{y} \leftarrow f_\theta(x)$ ▷ forward, per scene
  $f \leftarrow \textsc{Fitness}(\hat{y}, y)$
  $\pi_i \leftarrow \textsc{UpdateCMAES}(\pi_i, f)$ for all i
  $\nabla \leftarrow \nabla_\theta \frac{1}{B} \sum \ell(\hat{y}, y)$ ▷ backward
  $\theta \leftarrow \mathrm{OptimStep}(\theta, \nabla)$
  if $t \bmod R = 0$ **then**
    $\{\pi_i\} \leftarrow \textsc{ResetWeakest}(\{\pi_i\})$
  end if
end for

We also change standard CMA-ES by pooling updates over 2 generation steps (i.e. μ and Σ are updated every other training step). This increases the number of samples used to compute the updates and captures some of the non-stationarity in the objective. If worst case is moving in parameter space due to neural network updates, this pooling operation discourages CMA-ES from over-concentrating at an optimum which exists only transiently.

CMA-ES Resets. We find it is helpful to reset the CMA-ES instances periodically. Individual instances may be stuck in a local minimum, or multiple instances could converge to the same point and reduce the batch's diversity. For each π_i we define a fitness function and replace the least fit fraction of π_i instances every R training steps.

We define the fitness by $f = f_{\text{error}} + f_{\text{separation}}$. Both terms are normalized between 0 and 1. For the error term, we use the worst fitness from π_i's last generation, scaled so that the mean error on the uniform samples is 0 and the overall worst found error is 1. For the separation term, we use the distance between μ_i and its nearest neighbor μ_j. We scale this by $\sqrt{d/6}$, the RMS distance between two uniform random points in $[0,1]^d$, and we clamp it between 0 and 1. Thus, the most fit π_i samplers are those in regions with high error and far from others.

5 Experiment Setup

We train models using Simple-PAT on the tasks described. The primary remaining challenge is evaluating the worst-case performance of the trained networks fairly and consistently.

5.1 Evaluation

Our primary evaluation is an adversarial search against the final trained model with a budget of 2 million samples. This represents 39% of the overall training budget of 5.12M samples. We split this between 1 million samples of a DIRECT-L search (Gablonsky & Kelley, 2001) and 1 million samples of a CMA-ES search. This CMA-ES search differs from the search performed during training in several important ways: (1) it uses a population size of 512 instead of 28 (2) it updates a full covariance matrix instead of a diagonal parameterization (3) the objective is stationary, allowing for better convergence (4) IPOP restarts are permitted (Hansen et al., 2019). Combined, these settings give the evaluation a substantially more powerful search than training.

Our secondary evaluation uses a test set of 65,536 scenes. These are selected uniformly at random and we evaluate all models on the same scenes. We report the maximum error and the median error.

We call the maximum over both searches and the test set the *evaluation worst case*. In section 6.4 we consider a stronger search for the worst case.

5.2 Baselines

We compare Simple-PAT against the following baselines:

Uniform Training. At each batch, the model selects B parameter vectors from a uniform distribution. This is identical to training on an infinite fixed dataset with every point weighted equally.

CVaR. CVaR training draws a uniform batch but considers only the worst α fraction of scenes for backpropagation.

Tilted. Tilted empirical risk minimization likewise draws a uniform batch, but weights each scene's gradient by $\mathrm{softmax}(\beta\ell)$ over the batch. As $\beta \to \infty$ this approaches the gradient of the batch's worst case. Thus, tilted ERM can be viewed as a soft approximation of optimizing the worst case within each batch. We must choose some value of β for each task, as it depends on the scale of the loss. To do this, we take a converged model and select β so that the effective sample size ($ESS(w) = (\sum_i w_i)^2 / \sum_i w_i^2$) is close to the ESS of CVaR.

5.3 Hyperparameters

On each environment, we train neural networks of size 5M-7.8M parameters for 20,000 steps at a batch size of 256. We use Muon for all settings unless AdamW is explicitly stated (Jordan et al., 2024; Loshchilov & Hutter, 2019). Details on the hyperparameters for each training configuration can be found in Appendix A.

6 Results

6.1 Main Results

Table 2 shows our primary results. At a budget of 2×10^6 evaluations, Simple-PAT is the most robust on all four tasks. Compared to uniform Muon, it reduces the evaluation worst case error by $2.39\times$ on normals, $25.9\times$ on chair-11, $4.35\times$ on chair-54, and $1.83\times$ on pose. Against the strongest baseline for each tasks, the evaluation worst case error reductions are $2.06\times$, $16.9\times$, $3.08\times$, and $1.42\times$.

Fifteen out of the 16 comparisons achieve a p-value of 0.0079, the smallest possible value given we train only 5 seeds. This indicates that every Simple-PAT seed's evaluation worst case was better than that of every seed of the alternate training method. The one exception is CVaR on the pose estimation task. However, on this particular task Simple-PAT also has a lower test set maximum and median error.

Typical case cost. Our primary goal is to improve worst-case performance, but Simple-PAT also incurs less typical-case degradation than alternative robust training procedures. Moreover, the raw difference in median-case performance is small in magnitude. For chair-11 BCE, for example, the median error of 8.7×10^{-6} is less than the error incurred by being uncertain about a single pixel (i.e. predicting probability 0.5). For the pose task, Simple-PAT is 1.2cm worse on the median case while improving the evaluation worst case by 28 cm.

6.2 Worst Case Evaluation of SOTA Models

We also use our evaluation framework to understand the robustness of SOTA foundation models. Though these models have not been trained on our data, the images are perceptible to humans and we would expect them to generalize here. Moreover, foundation models benefit from diverse training data and may have learned rules for these tasks that generalize to all settings. We evaluate MoGe-2 for normal estimation, SAM 3 for segmentation, and SAM 3D body for pose estimation.

Environment	Method	Search, 2×10^6 queries		Test set, 65,536 scenes	
		Worst case ↓	p vs. Simple-PAT	Max ↓	Median ↓
Normals	Uniform (AdamW)	50.3 ±2.9	0.0079	10.57 ±0.71	1.665 ±0.058
angle (°)	Uniform (Muon)	44.3 ±1.7	0.0079	5.77 ±0.26	0.8922 ±0.0058
	Tilted	38.5 ±1.6	0.0079	8.46 ±0.43	3.756 ±0.018
	CVaR	38.1 ±2.7	0.0079	5.33 ±0.16	1.8114 ±0.0096
	Simple-PAT	**18.5** ±8.4	—	**3.54** ±0.29	1.703 ±0.017
Chair-11	Uniform (AdamW)	4.43 ±0.19	0.0079	0.10 ±0.13	5.20e-6 ±8.4e-7
BCE	Uniform (Muon)	4.28 ±0.39	0.0079	0.0162 ±0.0047	4.65e-6 ±7.3e-7
	Tilted	2.78 ±0.12	0.0079	0.0188 ±0.0077	5.08e-4 ±7.6e-5
	CVaR	3.69 ±0.12	0.0079	**0.0057** ±0.0016	5.9e-5 ±1.7e-5
	Simple-PAT	**0.16** ±0.15	—	0.0087 ±0.0058	8.7e-6 ±1.2e-6
Chair-54	Uniform (AdamW)	0.90 ±0.16	0.0079	0.045 ±0.021	1.08e-4 ±2.3e-5
BCE	Uniform (Muon)	0.56 ±0.18	0.0079	0.028 ±0.017	2.928e-5 ±6.6e-7
	Tilted	0.494 ±0.088	0.0079	0.0197 ±0.0055	2.412e-4 ±5.6e-6
	CVaR	0.40 ±0.15	0.0079	**0.017** ±0.011	7.41e-5 ±1.1e-6
	Simple-PAT	**0.128** ±0.046	—	0.017 ±0.015	2.885e-5 ±7.0e-7
Pose	Uniform (AdamW)	670 ±110	0.0079	196 ±17	30.66 ±0.32
MPJPE (mm)	Uniform (Muon)	610 ±110	0.0079	157 ±15	21.071 ±0.071
	Tilted	546 ±84	0.0079	172.8 ±5.7	42.69 ±0.21
	CVaR	480 ±130	0.056	181 ±15	45.03 ±0.17
	Simple-PAT	**335** ±40	—	**124** ±18	33.4 ±1.3

Table 2: Evaluation worst case for various training methods. Lower is better in every column. Each entry is the mean of 5 seeds with ± 1 standard deviation shown. In the "p vs. Simple-PAT" column we give the two-sided Mann-Whitney p-value comparing our method to baselines.

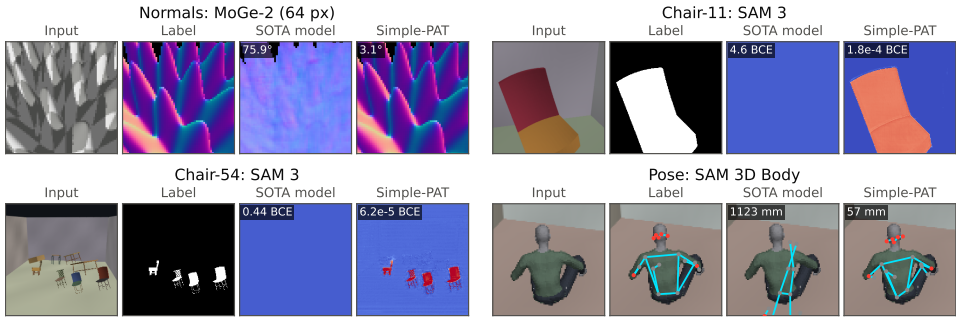

Figure 3: Failure cases of SOTA foundation models. We show the worst case found from a CMA-ES search against the foundation models with budget 10^5

For MoGe-2 and SAM 3, we use the predicted normals and probabilities of the model to compute our loss. We give SAM 3 the prompt "chair." For SAM 3D body, in addition to aligning the root as in MPJPE, we also align the scale of the prediction. This is necessary since the scale of these synthetic scenes is ambiguous without being trained on our specific generator.

In figure 3 we visualize the worst cases found for the SOTA foundation models in our environments. For the normals and chair-11 environments, the search identifies perceptually challenging cases, while for the chair-54 and pose environments the models fail on easier-looking scenes. Table 4 in the appendix presents the numeric error of the SOTA foundation models' worst found cases.

6.3 Training Sampling Concentration

To quantify how strongly Simple-PAT concentrates samples in particular regions of the parameter space, we examine the distribution of samples from a training run for the nor-

mal estimation task. We partition the parameter space into $8^6 = 262{,}144$ bins by selecting six coordinates and dividing each $[0, 1]$ interval into 8 equal-length sections. Figure 4 shows the resulting distribution of counts. Under uniform sampling, the distribution is tightly concentrated around its mode of 19 samples per bin, consistent with a Poisson distribution. Under adversarial sampling, the mode is only 5 samples per bin, but with a long tail extending over several orders of magnitude. This concentration is consistent with these cases being particularly challenging for the network to learn and therefore requiring many more samples.

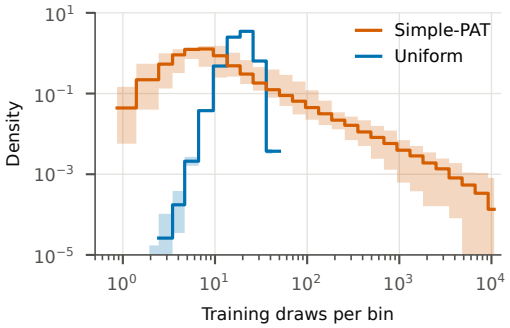

Figure 4: Distribution of training draws per bin. The highlights display the min-max range over 5 seeds and 8 coordinate subsets for each seed.

Compared to uniform sampling, Simple-PAT over-samples the top 1% and 0.1% of bins by factors of $40\times$ and $176\times$, respectively. Therefore uniform training would need to extend $40\times$ or $176\times$ longer to reach equal exposure to these challenging cases.

6.4 Evaluation with a Cross-Model Adversary

For each task, we evaluate every trained model on the worst case found by its own search and the worst cases found by other searches. This is effectively defining a new cross-model adversary with a budget of $25\times$ each evaluation optimizer, for a total of 50 million samples. Under this new adversary, we find that the margins of improvement for Simple-PAT are reduced but Simple-PAT is still significantly better than alternative training methods in 15 out of 16 comparisons.

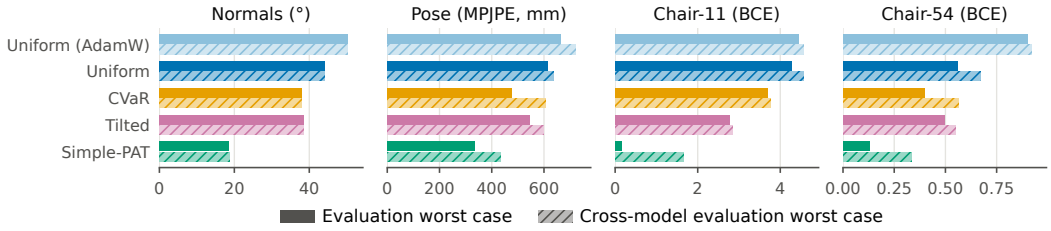

Figure 5: Cross-model evaluation worst case compared to standard evaluation worst case.

Figure 5 shows the worst cases found before and after pooling. The normals task changes minimally. For other tasks, the worst observed case increases for every model, with Simple-PAT experiencing the worst degradation on both chair tasks.

Under the cross-model evaluation, Simple-PAT outperforms uniform training by $2.37\times$ on normals, $1.47\times$ on pose, $2.74\times$ on chair-11, and $2.00\times$ on chair-54 (geometric mean $2.09\times$). The Mann-Whitney p-values comparing Simple-PAT to alternative models are less than 0.02 for every case, except tilted on chair-11 ($p = 0.15$). We present detailed results in Appendix B.2.

7 Conclusion

We propose procedural adversarial training (PAT) as a framework that trains robust neural networks via a minimax game where the adversary can search over perceptual perturbations. We design a Simple-PAT algorithm that realizes this adversarial game with zeroth-order CMA-ES optimization on the parameter space. Across our four PAT-Dojo tasks, Simple-PAT improves the robustness over uniform training, robust reweighting variants, and foundation models evaluated zero-shot. The cross-evaluation results suggest that there is shared structure between various training runs that is sometimes missed by black-box optimizers. Future work might exploit this structure to produce a stronger adversary and more robust models.

AI USE STATEMENT

In this work, we used generative AI tools to "design or provide feedback on research methodology or experiments" and "implement methods." We have not used generative AI tools to "help develop theoretical models or conceptual frameworks," "propose or refine hypotheses," "assist with translation," "clean and reformat dataset," "support qualitative and thematic data analysis," or "interpret results," and the tasks "generate synthetic data sets," "formulate mathematical claims," "provide critical ingredients for proving mathematical claims," and "assist in the writing of proofs," are not applicable to this work. Additionally, we used generative AI tools to "create or modify scientific figures or images." We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. Core experimental code was implemented by a human or checked for correctness by humans multiple times. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken careful steps to ensure the reproducibility of this paper. We include the code in the supplementary material. This contains an experimental file that can be used to run the sweeps for this paper. In addition to the code, the hyperparameters for our training runs are described in the appendix section A. Our experimental results do not rely on any data not included in the code release. We specify our dependencies so that the environment can be reproduced.

REFERENCES

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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4845–4854. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00498. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Alcorn_Strike_With_a_Pose_Neural_Networks_Are_Easily_Fooled_by_CVPR_2019_paper.html.

CMU Graphics Lab. CMU Graphics Lab Motion Capture Database. URL http://mocap.cs.cmu.edu/.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1802–1811. PMLR, 2019. URL http://proceedings.mlr.press/v97/engstrom19a.html.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems, 2018. URL https://arxiv.org/abs/1705.01292.

Joerg M. Gablonsky and Carl T. Kelley. A locally-biased form of the DIRECT algorithm. *J. Glob. Optim.*, 21(1):27–37, 2001. doi: 10.1023/A:1017930332101. URL https://doi.org/10.1023/A:1017930332101.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning*

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6572.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398. URL https://doi.org/10.1162/106365601750190398.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL https://doi.org/10.5281/zenodo.2559634.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 15262–15271. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01501. URL https://openaccess.thecvf.com/content/CVPR2021/html/Hendrycks_Natural_Adversarial_Examples_CVPR_2021_paper.html.

Hossein Hosseini and Radha Poovendran. Semantic adversarial examples, 2018. URL https://arxiv.org/abs/1804.00499.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Yuanqing Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, Ashish Kapoor, and Aleksander Madry. 3db: A framework for debugging computer vision models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/3848bc3112429079af85dedb7d369ef4-Abstract-Conference.html.

Daniel Levy, Yair Carmon, John C. Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/64986d86a17424eeac96b08a6d519059-Abstract.html.

Baiqi Li, Zhiqiu Lin, Wenxuan Peng, Jean de Dieu Nyandwi, Daniel Jiang, Zixian Ma, Simran Khanuja, Ranjay Krishna, Graham Neubig, and Deva Ramanan. Naturalbench: Evaluating vision-language models on natural adversarial samples. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1e69ff56d0ebff0752ff29caaddc25dd-Abstract-Datasets_and_Benchmarks_Track.html.

Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. Tilted empirical risk minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=K5YasWXZT3O.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Zeyu Ma, Alexander Raistrick, and Jia Deng. Simpleproc: Fully procedural synthetic data from simple rules for multi-view stereo, 2026. URL https://arxiv.org/abs/2604.04925.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017. URL http://arxiv.org/abs/1706.06083.

OpenAI. Gpt-6 astra: A new generation of intelligence. OpenAI Blog, 2026. URL https://openai.com/index/gpt-6-astra/.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning, 2017. URL https://arxiv.org/abs/1703.02702.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 12630–12641. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01215. URL https://doi.org/10.1109/CVPR52729.2023.01215.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 21783–21794. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02058. URL https://doi.org/10.1109/CVPR52733.2024.02058.

Alexander Raistrick, Karhan Kayan, Jack Nugent, David Yan, Lingjie Mei, Meenal Parakh, Hongyu Wen, Dylan Li, Yiming Zuo, Erich Liang, and Jia Deng. Procfunc: Function-oriented abstractions for procedural 3d generation in python, 2026. URL https://arxiv.org/abs/2604.26943.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000. doi: 10.21314/JOR.2000.038.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Bhavya Vasudeva, Puneesh Deora, Yize Zhao, Vatsal Sharan, and Christos Thrampoulidis. How muon's spectral design benefits generalization: A study on imbalanced data, 2025. URL https://arxiv.org/abs/2510.22980.

Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 9909–9918. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00978. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_AdvSim_Generating_Safety-Critical_Scenarios_for_Self-Driving_Vehicles_CVPR_2021_paper.html.

Shuche Wang, Fengzhuo Zhang, Jiaxiang Li, Cunxiao Du, Chao Du, Tianyu Pang, Zhuoran Yang, Mingyi Hong, and Vincent Y. F. Tan. Muon outperforms adam in tail-end associative memory learning, 2025. URL https://arxiv.org/abs/2509.26030.

David Yan, Alexander Raistrick, and Jia Deng. What makes good synthetic training data for zero-shot stereo matching?, 2026. URL https://arxiv.org/abs/2504.16930.

Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L. Yuille. Adversarial attacks beyond the image space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4302–4311. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00443. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Zeng_Adversarial_Attacks_Beyond_the_Image_Space_CVPR_2019_paper.html.

A HYPERPARAMETERS

- **Training:** each model is trained for 20,000 steps at a batch size of 256.
- **Models:** for normals and chair tasks we train small unets around 7.5M and 7.8M parameters. They consist of standard blocks with two 3×3 convolutions followed by a GroupNorm and ReLU. We select GroupNorm instead of BatchNorm since our custom sampling procedure may affect the batch statistics. The pose network has a similar structure, but with global average pooling and an output head instead of upsampling layers. Consequently, the pose network has only 5.01M parameters
- **Optimizers:** we use AdamW with a learning rate of 3×10^{-4} or Muon with a learning rate 0.02. Muon is applied to the matrix parameters and AdamW is applied to the remaining parameters. Every training method uses Muon besides "Uniform (AdamW)". We use a cosine learning rate schedule with a warmup of 200 steps.
- **Simple-PAT:** We use $k = 16$, $\lambda = 14$ and $n_0 = 32$. Hence, each batch is composed of 32 uniform samples and 14 samples from 16 CMA-ES instances. Every $R = 40$ training steps (20 CMA-ES generations) we replace the least fit 25% of instances.
- **CVaR Training:** for all tasks, we use $\alpha = 0.05$, leading to 13 scenes contributing to the loss the network trains on.
- **Tilted Training:** for each task we use a converged uniform-Muon model to compute the ESS. We select β to match CVaR's ESS of 13, averaged over 64 batches, finding $\beta = 2550$ (normals), 18.3 (pose), 10400 (chair-54), and 14800 (chair-11).

B ADDITIONAL ANALYSES

B.1 ADAMW VS MUON FOR WORST CASE PERFORMANCE

Previous studies (Wang et al., 2025; Vasudeva et al., 2025) have found that the Muon optimizer learns rarely-seen training data better than AdamW. Consequently, models trained with Muon might exhibit improved robustness. We evaluate this on the normal estimation task using 15 seeds for each combination of optimizer (Muon or AdamW) and training method (uniform or Simple-PAT). In Figure 6 we display these results. On the test-set maximum, Muon reduces the error by 5.1 degrees with uniform training and 2.3 degrees with Simple-PAT. When evaluating the worst case, Muon displays a significant improvement for uniform training but not Simple-PAT. This may be an issue of statistical power. Alternatively, Simple-PAT may concentrate sampling near the worst case. Then, these cases are no longer rare and Muon's advantage over AdamW may be lessened.

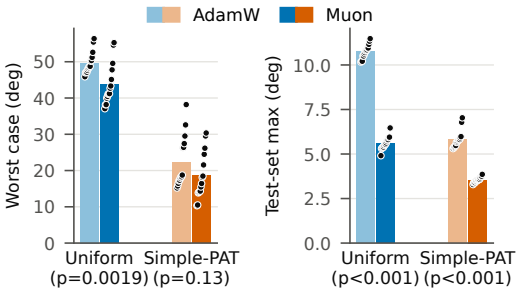

Figure 6: Muon vs. AdamW on the normal estimation task. Scattered points show results of every seed. p-values are calculated by the two-sided Welch's t-test comparing Muon and AdamW

B.2 Evaluation with a Cross-Model Adversary

		Evaluation	Cross-model evaluation		
Environment	Method	Worst case ↓	Worst case ↓	p vs. Simple-PAT	Ratio
Normals	Uniform (AdamW)	50.3 ±2.9	50.3 ±2.9	0.0079	1.00×
angle (°)	Uniform (Muon)	44.3 ±1.7	44.3 ±1.7	0.0079	1.00×
	Tilted	38.5 ±1.6	38.5 ±1.6	0.0079	1.00×
	CVaR	38.1 ±2.7	38.1 ±2.7	0.0079	1.00×
	Simple-PAT	**18.5** ±8.4	**18.6** ±8.3	—	1.01×
Chair-11	Uniform (AdamW)	4.43 ±0.19	4.555 ±0.036	0.0079	1.03×
BCE	Uniform (Muon)	4.28 ±0.39	4.563 ±0.089	0.0079	1.07×
	Tilted	2.78 ±0.12	2.85 ±0.15	0.15	1.02×
	CVaR	3.69 ±0.12	3.75 ±0.15	0.0079	1.02×
	Simple-PAT	**0.16** ±0.15	**1.7** ±1.1	—	14.2×
Chair-54	Uniform (AdamW)	0.90 ±0.16	0.92 ±0.15	0.0079	1.02×
BCE	Uniform (Muon)	0.56 ±0.18	0.673 ±0.050	0.0079	1.31×
	Tilted	0.494 ±0.088	0.550 ±0.062	0.016	1.13×
	CVaR	0.40 ±0.15	0.565 ±0.062	0.016	1.63×
	Simple-PAT	**0.128** ±0.046	**0.34** ±0.10	—	3.16×
Pose	Uniform (AdamW)	670 ±110	721 ±74	0.0079	1.09×
MPJPE (mm)	Uniform (Muon)	610 ±110	638 ±69	0.0079	1.05×
	Tilted	546 ±84	599 ±45	0.0079	1.11×
	CVaR	480 ±130	605 ±52	0.016	1.34×
	Simple-PAT	**335** ±40	**434** ±75	—	1.30×

Table 3: Cross-model evaluation worst case for various training methods. Lower is better in every column. Each entry is the mean of 5 seeds with ± 1 standard deviation shown. In the "p vs. Simple-PAT" column we give the two-sided Mann-Whitney p-value comparing our method to baselines.

In Table 3 we show the complete cross-model evaluation results as described in Section 6.4.

C SOTA Foundation Model Worst-case

		Search		Test set, 65,536 scenes	
Environment	Model	Worst case ↓	× Simple-PAT	Max ↓	Median ↓
Normals	MoGe-2 (64 px)	75.9	4.1	54.5	22.2
angle (°)	MoGe-2 (512 px)	68.5	3.7	52.0	18.9
	Simple-PAT	18.5 ±8.4	—	3.54 ±0.29	1.703 ±0.017
Chair-11	SAM 3	4.60	27.9	3.72	9.5e-7
BCE	**Simple-PAT**	0.16 ±0.15	—	0.0087 ±0.0058	8.7e-6 ±1.2e-6
Chair-54	SAM 3	0.442	3.4	0.442	0.00272
BCE	**Simple-PAT**	0.128 ±0.046	—	0.017 ±0.015	2.885e-5 ±7.0e-7
Pose	SAM 3D Body	1123	3.3	919	65.2
MPJPE (mm)	**Simple-PAT**	335 ±40	—	124 ±18	33.4 ±1.3

Table 4: Worst case evaluation of SOTA foundation models. We include Simple-PAT worst case numbers for reference, though these values are not directly comparable. The worst case for the SOTA models is evaluated as the worst of the test set and a single CMA-ES search with budget 10^5, compared to the Simple-PAT evaluation of CMA-ES with budget 10^6 and DIRECT-L with budget 10^6. We evaluate MoGe-2 at the generator's standard resolution of 64×64, and a version of the generator rendered at 512×512 to more closely match MoGe-2's standard input resolution.

D Evaluation Worst-case Visualizations

Here we show visualizations of each model's evaluation worst case. We select the model with the median loss of the five seeds.

D.1 Normals

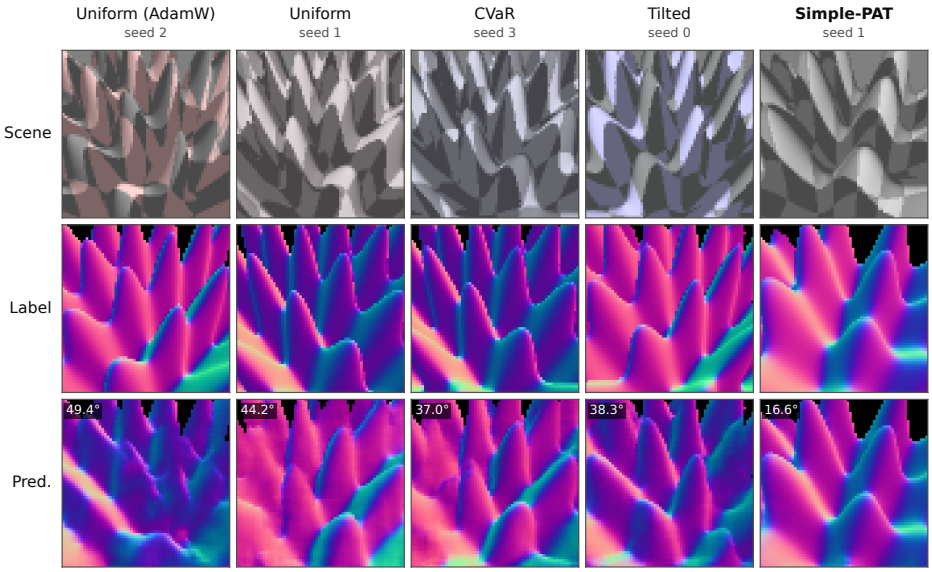

Figure 7: Evaluation worst case samples on the normals task.

D.2 Chair 11 Dimensions

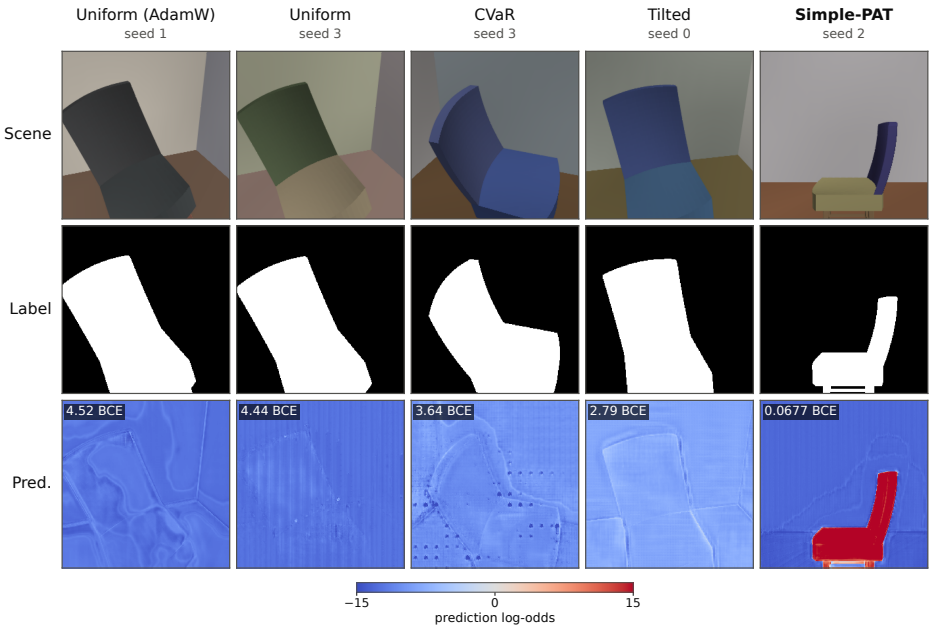

Figure 8: Evaluation worst case samples on the chair-11 task.

D.3 CHAIR 54 DIMENSIONS

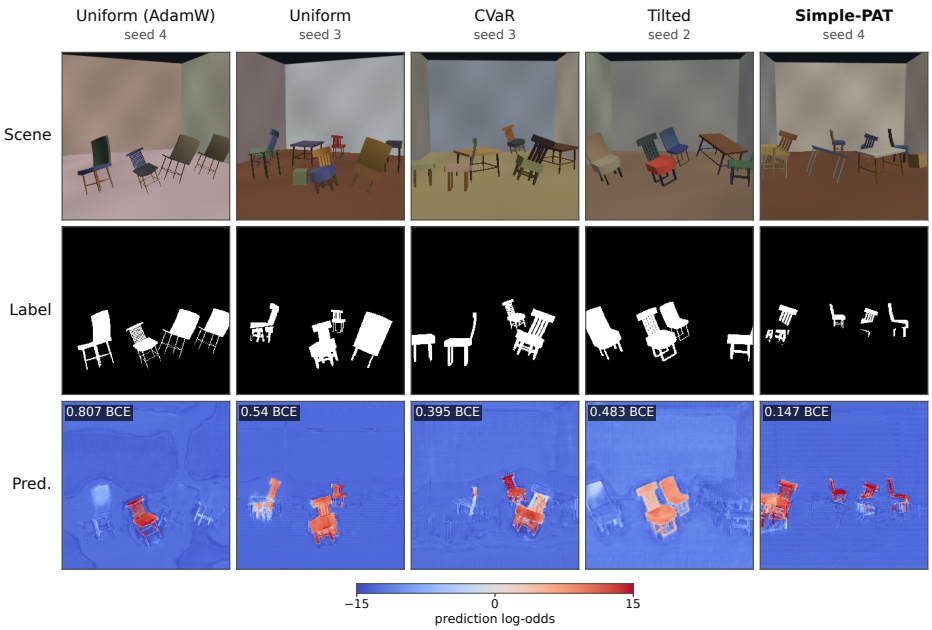

Figure 9: Evaluation worst case samples on the chair-54 task.

D.4 POSE

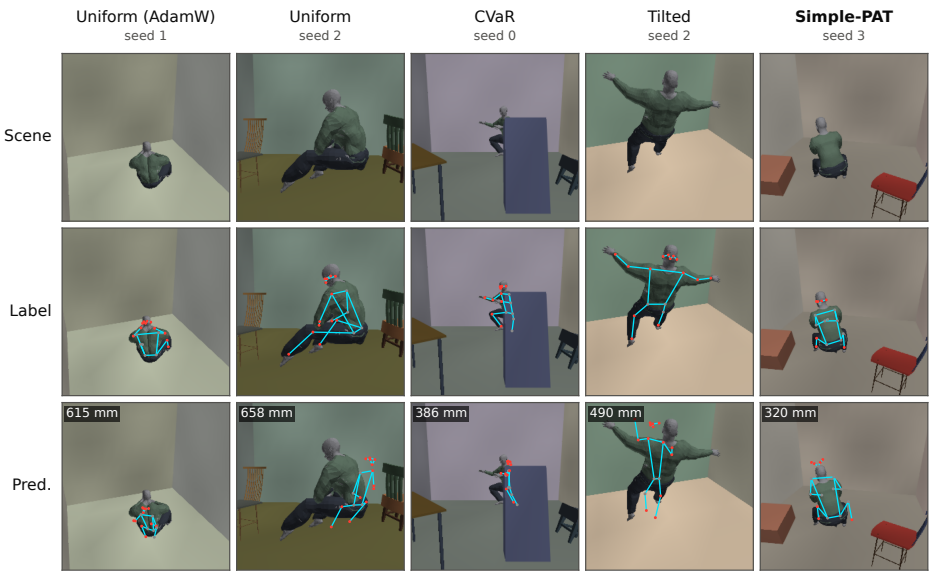

Figure 10: Evaluation worst case samples on the pose task.

E PROCEDURAL GENERATORS

Here we show visualizations of uniform randomly selected points from each procedural generator.

E.1 NORMALS

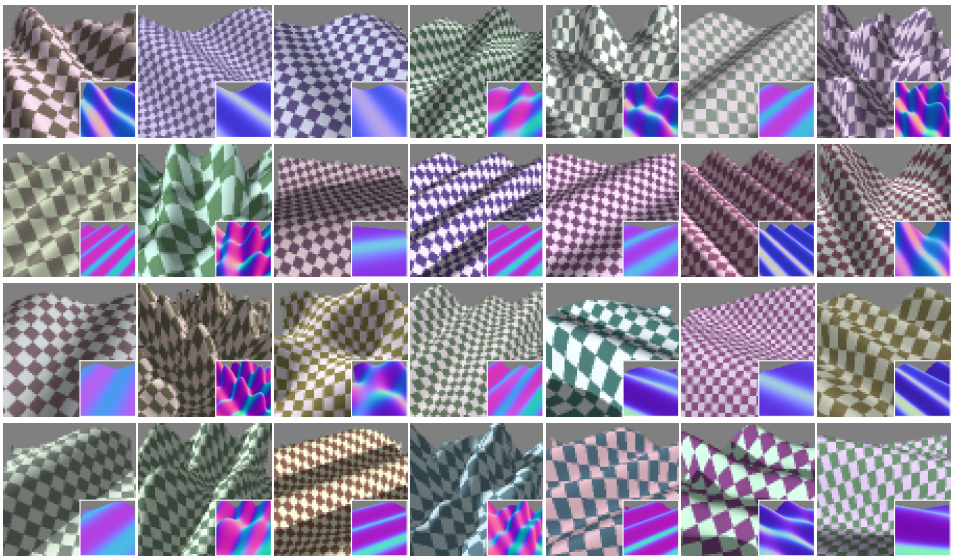

Figure 11: Uniform random samples from the normals procedural generator.

E.2 CHAIR 11 DIMENSIONS



Figure 12: Uniform random samples from the chair-11 procedural generator.

E.3 CHAIR 54 DIMENSIONS

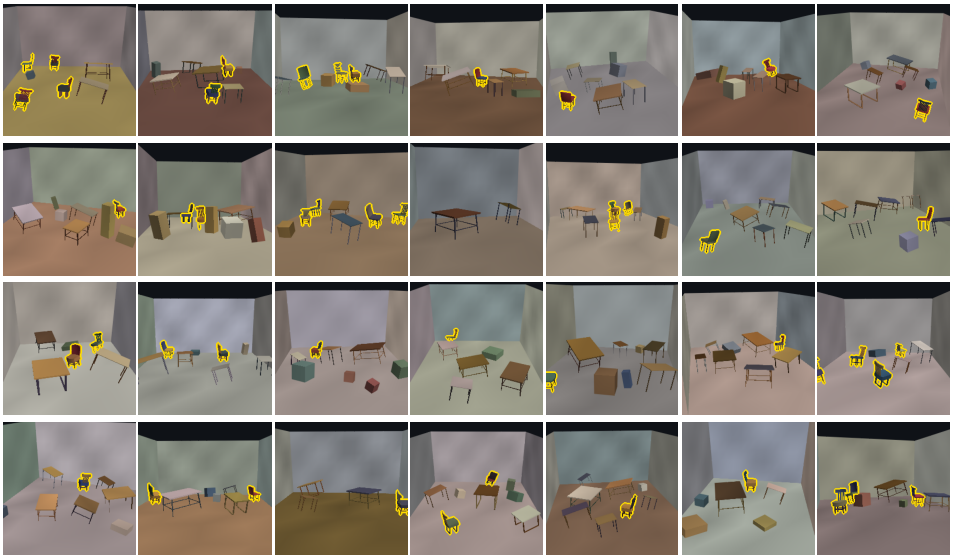

Figure 13: Uniform random samples from the chair-54 procedural generator.

E.4 POSE

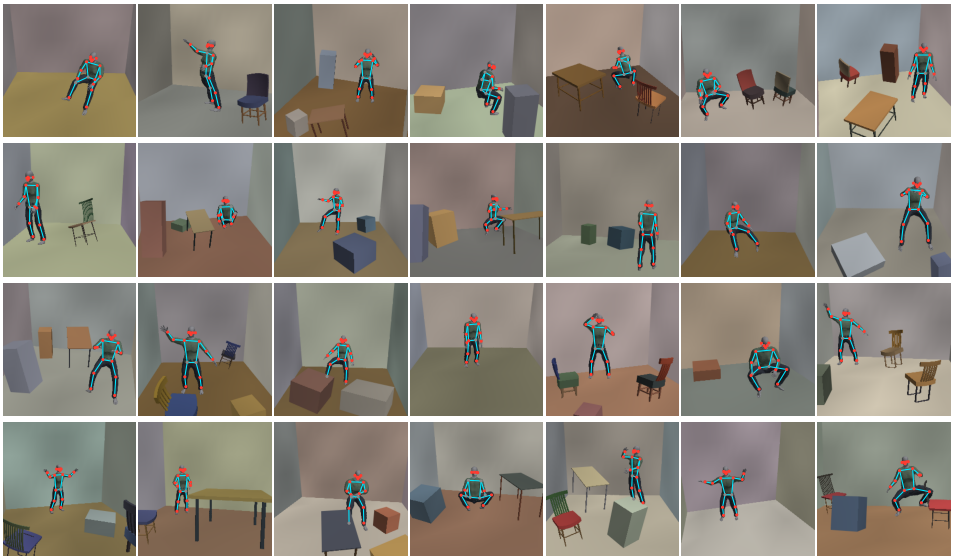

Figure 14: Uniform random samples from the pose procedural generator.