Task2Sim: Towards Effective Pre-training and Transfer from Synthetic Data

Samarth Mishra[†1]   Rameswar Panda[2]   Cheng Perng Phoo[†3]   Chun-Fu (Richard) Chen[*2]
Leonid Karlinsky[2]   Kate Saenko[1,2]   Venkatesh Saligrama[1]   Rogerio S. Feris[2]
[1]Boston University   [2]MIT-IBM Watson AI Lab   [3]Cornell University

Abstract

Pre-training models on Imagenet or other massive datasets of real images has led to major advances in computer vision, albeit accompanied with shortcomings related to curation cost, privacy, usage rights, and ethical issues. In this paper, for the first time, we study the transferability of pre-trained models based on synthetic data generated by graphics simulators to downstream tasks from very different domains. In using such synthetic data for pre-training, we find that downstream performance on different tasks are favored by different configurations of simulation parameters (e.g. lighting, object pose, backgrounds, etc.), and that there is no one-size-fits-all solution. It is thus better to tailor synthetic pre-training data to a specific downstream task, for best performance. We introduce Task2Sim, a unified model mapping downstream task representations to optimal simulation parameters to generate synthetic pre-training data for them. Task2Sim learns this mapping by training to find the set of best parameters on a set of "seen" tasks. Once trained, it can then be used to predict best simulation parameters for novel "unseen" tasks in one shot, without requiring additional training. Given a budget in number of images per class, our extensive experiments with 20 diverse downstream tasks show Task2Sim's task-adaptive pre-training data results in significantly better downstream performance than non-adaptively choosing simulation parameters on both seen and unseen tasks. It is even competitive with pre-training on real images from Imagenet.

1. Introduction

Using large-scale labeled (like ImageNet [9]) or weakly-labeled (like JFT-300M [5, 18], Instagram-3.5B [34]) datasets collected from the web has been the go-to approach for pre-training classifiers for *downstream* tasks with a relative scarcity of labeled data. Prior works have demonstrated that as we move to bigger datasets for pre-training, down-
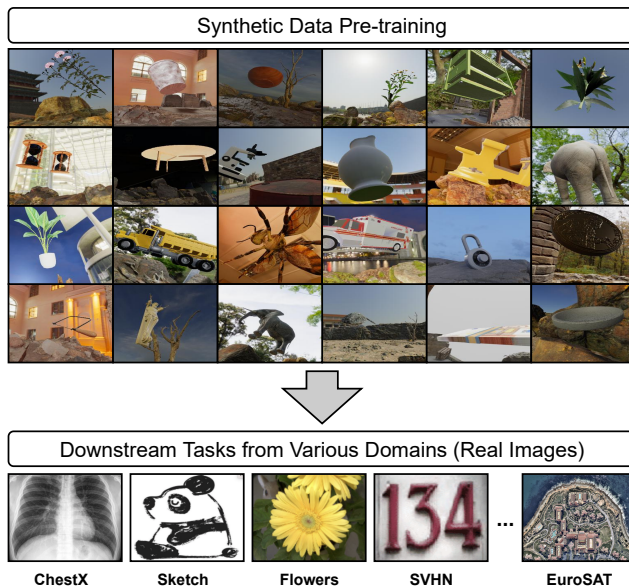
Figure 1. We explore how synthetic data can be effectively used for training models that can transfer to a wide range of downstream tasks from various domains. Is a universal pre-trained model for all downstream tasks, the best approach?

stream accuracy improves on average [34, 56]. However, large-scale real image datasets bear the additional cost of curating labels, in addition to other concerns like privacy or copyright. Furthermore, large datasets like JFT-300M and Instagram-3.5B are not publicly available posing a bottleneck in reproducibility and fair comparison of algorithms.

Synthetic images generated via graphics engines provide an alternative quelling a substantial portion of these concerns. With 3D models and scenes, potentially infinite images can be generated by varying various scene or image-capture parameters. Although synthetic data has been used for transfer learning in various specialized tasks [2, 48, 55, 59], there has not been prior research dedicated to its transferability to a range of different recognition tasks from different domains (see Figure 1). In conducting this first of its kind (to the best of our knowledge) study, we first ask the question : in synthetic pretraining for different downstream

Pretraining Data Variations	Downstream Accuracy			
	EuroSAT	SVHN	Sketch	DTD
Pose	87.01	28.49	37.89	37.39
+Lighting	88.57	32.36	**38.81**	40.32
+Blur	**90.20**	35.58	35.53	37.66
+Materials	84.54	**44.84**	30.81	**38.51**
+Background	80.44	29.93	14.60	32.39

Table 1. Downstream task accuracies using linear probing with a Resnet-50 backbone pretrained on synthetic datasets with different varying parameters (successively added). We see different simulation parameters have different effects on downstream tasks.

classification tasks, does a one-size-fits-all solution (*i.e.*, a universal pre-trained model for all tasks) work well?

With graphics engines, we can control various *simulation* parameters (lighting, pose, materials, etc.). So, in an experiment, we introduced more variations successively from different parameters into a pretraining dataset of 100k synthetic images from 237 different classes (as many categories as are available in Three-D-World [11]). We pre-trained a ResNet-50 [16] on these, and evaluated this backbone with linear probing on different downstream tasks. The results are in Table 1. We see that some parameters like random object materials result in improved performance for some downstream tasks like SVHN and DTD, while hurting performance for other tasks like EuroSAT and Sketch. In general different pre-training data properties seem to favor different downstream tasks.

To maximize the benefit of pre-training, different optimal simulation parameters can be found for each specific downstream task. Because of the combinatorially large set of different simulation parameter configurations, a brute force search is out of the question. However, this might still suggest that some, presumably expensive, learning process is needed for each downstream task for an optimal synthetic image set for pre-training. We show this is not the case.

We introduce Task2Sim, a unified model that maps a downstream task representation to optimal simulation parameters for pre-training data generation to maximize downstream accuracy. Using vector representations for a set of downstream tasks (in the form of Task2Vec [1]), we train Task2Sim to find and thus learn a mapping to optimal parameters for each task from the set. Once trained on this set of "seen" tasks, Task2Sim can also use Task2Vec representations of novel "unseen" tasks to predict simulation parameters that would be best for their pre-training datasets. This efficient one-shot prediction for novel tasks is of significant practical value, if developed as an end-user application that can automatically generate and provide pre-training data, given some downstream examples.

Our extensive experiments using 20 downstream classification datasets show that on seen tasks, given a number of images per category, Task2Sim's output parameters gen-

erate pre-training datasets that are much better for downstream performance than approaches like domain randomization [2, 25, 74] that are not task-adaptive. Moreover, we show Task2Sim also generalizes well to unseen tasks, maintaining an edge over non-adaptive approaches while being competitive with Imagenet pre-training.

In summary, (i) We address a novel, and very practical, problem—how to optimally leverage synthetic data to task-adaptively pre-train deep learning models for transfer to diverse downstream tasks. To the best of our knowledge, this is the first time such a problem is being addressed in transfer learning research. (ii) We propose Task2Sim, a unified parametric model that learns to map Task2Vec representations of downstream tasks to simulation parameters for optimal pre-training. (iii) Task2Sim can generalize to novel "unseen" tasks, not encountered during training, a feature of significant practical value as an application. (iv) We provide a thorough analysis of the behavior of downstream accuracy with different sizes of pre-training data (in number of classes, object-meshes or simply images) and with different downstream evaluation methods.

2. Related Work

Training with Synthetic Data. Methods that learn from synthetic data have been extensively studied since the early days of computer vision [31, 39]. In recent years, many approaches that rely on synthetic data representations have been proposed for image classification [11, 36], object detection [43, 44], semantic segmentation [50, 67], action recognition [49, 61], visual reasoning [22], and embodied perception [27, 53, 71]. Unlike previous work, we focus on a different problem: how to build task-adaptive pre-trained models from synthetic data that can transfer to a wide range of downstream datasets from various domains.

Synthetic to Real Transfer. The majority of methods proposed to bridge the *reality gap* (between simulation and real data) are based on domain adaptation [8]. These include reconstruction-based techniques, using encoder-decoder models or GANs to improve the realism of synthetic data [19, 47, 54], discrepancy-based methods, designed to align features between the two domains [51, 75], and adversarial approaches, which rely on a domain discriminator to encourage domain-independent feature learning [13, 45, 60]. Contrasting from these techniques, our work aims at building pre-trained models from synthetic data and does not assume the same label set for source and target domains. The most prevalent approach in a setting similar to ours, is domain randomization [2, 25, 44, 59, 74], which learns pre-trained models from datasets generated by randomly varying simulator parameters. In contrast, Task2Sim *learns* simulator parameters to generate synthetic datasets that maximize transfer learning performance.
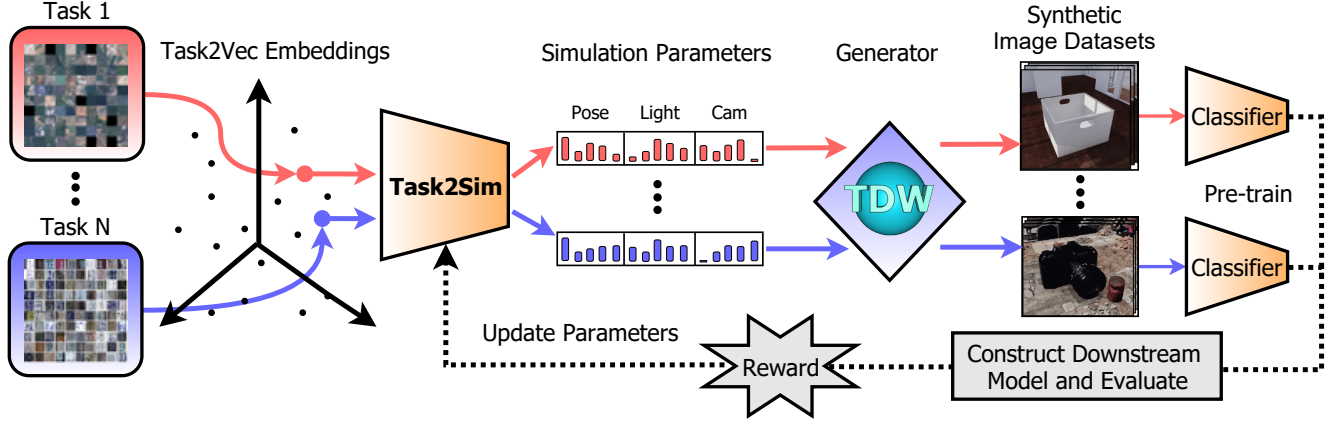
Figure 2. **Illustration of our proposed approach**. Given a batch of tasks represented by Task2Vec representations, our approach (Task2Sim) aims to map these representations to optimal simulation parameters for generating a dataset of synthetic images. The downstream classifier's accuracy for the set of tasks is then used as a reward to update Task2Sim's parameters. Once trained, Task2Sim can be used not only for "seen" tasks but also can be used in one-shot to generate simulation parameters for novel "unseen" tasks.

Optimization of Simulator Parameters. Recently, a few approaches have been proposed to learn synthetic data generation by optimizing simulator parameters [3, 26, 52, 73]. SPIRAL [12], AVO [33] and Attr. Desc. [74] minimize the distance between distributions of simulated data and real data. Learning to Simulate [52] optimizes simulator parameters using policy gradients that maximize validation accuracy for a specific task, while Auto-Sim [3] speeds up the search process using a differentiable approximation of the objective. Meta-Sim [10, 23] learns to modify attributes obtained from probabilistic scene grammars for data generation. These methods are specifically tailored to applications in autonomous driving, whereas our goal is to transfer synthetic data representations to a wide range of downstream tasks. Notably, our proposed approach is significantly different from previous methods, as it maps task representations to simulation parameters through a unified parametric model, enabling one-shot synthetic data generation, even for unseen tasks, without requiring expensive training.

Conditional Computation. Albeit not apparent, our method is also related to dynamic neural network that adaptively change computation depending on input [15]. These methods have been effectively used to skip computation in deep neural networks conditioned on the input [62, 66, 69], perform adaptive fine-tuning [14], and dynamically allocate computation across frames for efficient video analysis [35, 70]. In particular, Adashare [57] learns different computational pathways for each task within a single multi-task network model, with the goal of improving efficiency and minimizing negative interference in multi-task learning. Analogously, our approach learns different *data simulation pathways* (by adaptively deciding which rendering parameters to use) for each task, using a single parametric model, with the goal of generating task-specific pre-training data.

3. Proposed Approach

Our goal is to create a unified model that maps task representations (e.g., obtained using task2vec [1]) to simulation parameters, which are in turn used to render synthetic pre-training datasets for not only tasks that are seen during training, but also novel tasks. This is a challenging problem, as the number of possible simulation parameter configurations is combinatorially large, making a brute-force approach infeasible when the number of parameters grows.

3.1. Overview

Figure 2 shows an overview of our approach. During training, a batch of "seen" tasks is provided as input. Their task2vec vector representations are fed as input to Task2Sim, which is a parametric model (shared across all tasks) mapping these downstream task2vecs to simulation parameters, such as lighting direction, amount of blur, background variability, etc. These parameters are then used by a data generator (in our implementation, built using the Three-D-World platform [11]) to generate a dataset of synthetic images. A classifier model then gets pre-trained on these synthetic images, and the backbone is subsequently used for evaluation on specific downstream task. The classifier's accuracy on this task is used as a reward to update Task2Sim's parameters. Once trained, Task2Sim can also be used to efficiently predict simulation parameters in *one-shot* for "unseen" tasks that it has not encountered during training.

3.2. Task2Sim Model

Let us denote Task2Sim's parameters with θ. Given the task2vec representation of a downstream task $x \in \mathcal{X}$ as input, Task2Sim outputs simulation parameters $a \in \Omega$. The model consists of M output heads, one for each simula-

tion parameter. In the following discussion, just as in our experiments, each simulation parameter is discretized to a few levels to limit the space of possible outputs. Each head outputs a categorical distribution $\pi_i(\boldsymbol{x}, \theta) \in \Delta^{k_i}$, where k_i is the number of discrete values for parameter $i \in [M]$, and Δ^{k_i}, a standard k_i-simplex. The set of argmax outputs $\nu(\boldsymbol{x}, \theta) = \{\nu_i | \nu_i = \arg\max_{j \in [k_i]} \pi_{i,j} \ \forall i \in [M]\}$ is the set of simulation parameter values used for synthetic data generation. Subsequently, we drop annotating the dependence of π and ν on θ and $\boldsymbol{x}$ when clear.

3.3. Task2Sim Training

Since Task2Sim aims to maximize downstream accuracy after pre-training, we use this accuracy as the reward in our training optimization[1]. Note that this downstream accuracy is a non-differentiable function of the output simulation parameters (assuming any simulation engine can be used as a black box) and hence direct gradient-based optimization cannot be used to train Task2Sim. Instead, we use REINFORCE [68], to approximate gradients of downstream task performance with respect to model parameters θ.

Task2Sim's outputs represent a distribution over "actions" corresponding to different values of the set of M simulation parameters. $P(a) = \prod_{i \in [M]} \pi_i(a_i)$ is the probability of picking action $a = [a_i]_{i \in [M]}$, under policy $\pi = [\pi_i]_{i \in [M]}$. Remember that the output π is a function of the parameters θ and the task representation $\boldsymbol{x}$. To train the model, we maximize the expected reward under its policy, defined as

$$R = \underset{a \in \Omega}{\mathbb{E}}[R(a)] = \sum_{a \in \Omega} P(a)R(a) \qquad (1)$$

where Ω is the space of all outputs a and $R(a)$ is the reward when parameter values corresponding to action a are chosen. Since reward is the downstream accuracy, $R(a) \in [0, 100]$. Using the REINFORCE rule, we have

$$\nabla_\theta R = \underset{a \in \Omega}{\mathbb{E}}[(\nabla_\theta \log P(a))R(a)] \qquad (2)$$

$$= \underset{a \in \Omega}{\mathbb{E}}\left[\left(\sum_{i \in [M]} \nabla_\theta \log \pi_i(a_i)\right)R(a)\right] \qquad (3)$$

where the 2nd step comes from linearity of the derivative. In practice, we use a point estimate of the above expectation at a sample $a \sim (\pi + \epsilon)$ (ϵ being some exploration noise added to the Task2Sim output distribution) with a self-critical baseline following [46]:

$$\nabla_\theta R \approx \left(\sum_{i \in [M]} \nabla_\theta \log \pi_i(a_i)\right)(R(a) - R(\nu)) \qquad (4)$$

---

[1] Note that our rewards depend only on the task2vec input and the output action and do not involve any states, and thus our problem can be considered similar to a stateless-RL or contextual bandits problem [29]

where, as a reminder ν is the set of the distribution argmax parameter values from the Task2Sim model heads.

A pseudo-code of our approach is shown in Algorithm 1. Specifically, we update the model parameters θ using minibatches of tasks sampled from a set of "seen" tasks. Similar to [41], we also employ self-imitation learning biased towards actions found to have better rewards. This is done by keeping track of the best action encountered in the learning process and using it for additional updates to the model, besides the ones in Line 12 of Algorithm 1. Furthermore, we use the test accuracy of a 5-nearest neighbors classifier operating on features generated by the pretrained backbone as a proxy for downstream task performance since it is computationally much faster than other common evaluation criteria used in transfer learning, e.g., linear probing or full-network finetuning. Our experiments demonstrate that this proxy evaluation measure indeed correlates with, and thus, helps in final downstream performance with linear probing or full-network finetuning.

Algorithm 1: Training Task2Sim

1 **Input:** Set of N "seen" downstream tasks represented by task2vecs $\mathcal{T} = \{\boldsymbol{x}_i | i \in [N]\}$.
2 Given initial Task2Sim parameters θ_0 and initial noise level ϵ_0
3 Initialize $a_{max}^{(i)} | i \in [N]$ the maximum reward action for each seen task
4 **for** $t \in [T]$ **do**
5   Set noise level $\epsilon = \frac{\epsilon_0}{t}$
6   Sample minibatch τ of size n from $\mathcal{T}$
7   Get Task2Sim output distributions $\pi^{(i)} | i \in [n]$
8   Sample outputs $a^{(i)} \sim \pi^{(i)} + \epsilon$
9   Get Rewards $R(a^{(i)})$ by generating a synthetic dataset with parameters $a^{(i)}$, pre-training a backbone on it, and getting the 5-NN downstream accuracy using this backbone
10   Update $a_{max}^{(i)}$ if $R(a^{(i)}) > R(a_{max}^{(i)})$
11   Get point estimates of reward gradients $dr^{(i)}$ for each task in minibatch using Eq. (4)
12     $\theta_{t,0} \leftarrow \theta_{t-1} + \frac{\sum_{i \in [n]} dr^{(i)}}{n}$
13   **for** $j \in [T_{si}]$ **do**
       // Self Imitation
14   Get reward gradient estimates $dr_{si}^{(i)}$ from Eq. (4) for $a \leftarrow a_{max}^{(i)}$
15        $\theta_{t,j} \leftarrow \theta_{t,j-1} + \frac{\sum_{i \in [n]} dr_{si}^{(i)}}{n}$
16   **end for**
17     $\theta_t \leftarrow \theta_{t,T_{si}}$
18 **end for**
19 **Output**: Trained model with parameters θ_T.

---

4. Experiments

4.1. Details

Downstream Tasks. We used a set of 20 classification tasks for our experiments with Task2Sim. We used the 12 tasks from [21] as the set of "seen" tasks for our model and a separate set of 8 tasks as the "unseen" set. All our tasks can be broadly categorized into the following 6 classes (S:seen, U:unseen):

- Natural Images: CropDisease(S) [37], Flowers102(S) [40], DeepWeeds(S) [42], CUB(U) [63]
- Aerial Images: EuroSAT(S) [17], Resisc45(S) [4], AID(U) [72], CactusAerial(U) [32]
- Symbolic Images: SVHN(S) [38], Omniglot(S) [28], USPS(U) [20]
- Medical Images: ISIC(S) [7], ChestX(S) [65], ChestX-Pneumonia(U) [24]
- Illustrative Images: Kaokore(S) [58], Sketch(S) [64], Pacs-C(U), Pacs-S(U) [30]
- Texture Images: DTD(S) [6], FMD(U) [76]

Task2Sim details. We used a Resnet-18 probe network to generate 9600-dimensional Task2Vec representations of downstream tasks. The Task2Sim model is a multi-layer perceptron with 2 hidden layers, having ReLU activations. The model shares its first two layers for all M heads, and branches after that. It is trained for 1000 epochs on seen tasks, with a batch-size 4 and 5 self-imitation steps (*i.e.* $n = 4, T_{si} = 5$ and $T = 1000$). We used a Resnet-50 model for pre-training and downstream evaluation for Task2Sim's rewards. Complete details are in the supplementary.

Synthetic Data Generation. We used Three-D-World (TDW) [11] for synthetic image generation. The platform provides 2322 different object models from 237 different classes, 57 of which overlap with Imagenet. Using TDW, we generated synthetic images of single objects from the aforementioned set (see Figure 1 for examples).

In this paper, we experimented with a parameterization of the pretraining dataset where $M = 8$ and $k_i = 2 \forall i \in [M]$ (using terminology from Section 3). The 8 parameters are:

- Object Rotation: If 1, multiple poses of an object are shown in the dataset, else, an object appears in a canonical pose in each image.
- Object Distance (from camera): If 1, object distance from the camera is varied randomly within a certain range, else, it is kept fixed.
- Lighting Intensity: If 1, intensity of the main lighting source (sun-like point light source at a distance) is varied, else it is fixed.
- Lighting Color: If 1, RGB color of the main lighting source is varied, else it is fixed.

- Lighting Direction : If 1, the direction of the main light source is varied, else it is a constant.
- Focus Blur: If 1, camera focus point and aperture are randomly perturbed to induce blurriness in the image, else, all image contents are always in focus.
- Background: If 1, background of the object changes in each image, else it is held fixed.
- Materials: If 1, in each image, each component of an object is given a random material out of 140 different materials, else objects have their default materials.

Hence in our experiments, for each of the above 8 parameters, Task2Sim decided whether or not different variations of it, would exhibit in the dataset. For speed of dataset generation while training Task2Sim, we used a subset of 780 objects with simple meshes, from 100 different categories and generated 400 images per category for pre-training.

4.2. Task2Sim Results

Baselines. We compared Task2Sim's downstream performance with the following baselines (pre-training datasets): (1) Random : For each downstream dataset, chooses a random 8-length bit string as the set of simulation parameters. (2) Domain Randomization : Uses a 1 in each simulation parameter, thus using all variations from simulation in each image. (3) Imagenet : Uses a subset of Imagenet with equal number of classes and images as other baselines[2]. (4) Scratch : Does not involve any pre-training of the classifier's feature extractor, training a randomly initialized classifier, with only downstream task data.
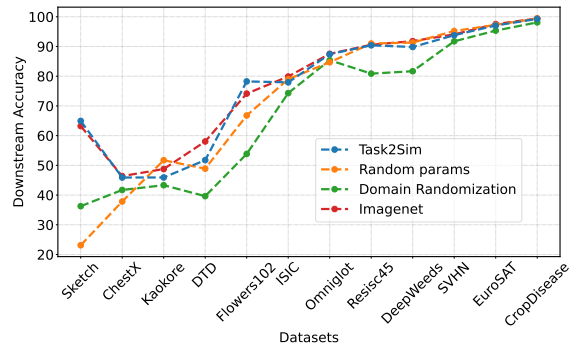

Figure 3. Performance of Task2Sim vs baselines on 12 seen tasks for 237 class / 100k image pre-training datasets evaluated with full-network finetuning. Best viewed in color.

Performance on Seen Tasks. Table 2 shows accuracies averaged over 12 seen downstream tasks for Task2Sim and all baselines using different evaluation methods for a Resnet-50 backbone. For the last two columns, we included all

---

[2]We also compared pre-training using Imagenet with 1K classes and an equal number of images, but this was poorer on average in downstream performance than the subset with fewer classes. Tables 2 and 3 and Figures 3 and 4 do not include it for succinctness

| | Average Downstream Accuracy — `Seen Tasks` | | | | |
| | 100 classes / 40k images | | | 237 classes / 100k images | |
Pretraining Dataset	5NN	Linear Probing	Finetuning	Linear Probing	Finetuning
Scratch	-	-	64.85	-	64.85
Random	25.30	54.06	70.77	55.14	72.18
Domain Randomization	19.42	35.31	62.96	45.31	68.51
Imagenet*	<u>28.91</u>	**63.12**	<u>74.26</u>	**68.44**	**77.61**
Task2Sim	**30.46**	<u>62.70</u>	**75.34**	<u>62.71</u>	<u>76.87</u>

Table 2. Comparing the downstream accuracy on seen tasks for the Task2Sim chosen pretraining dataset and other baselines. Simulation parameters found on seen tasks by Task2Sim generates synthetic pretraining data that is better for downstream tasks than other approaches like using Random simulation parameters or Domain Randomization. Pre-training with Task2Sim's data is also competitive with pre-training on images from Imagenet. *Imagenet has been subsampled to the same number of classes and images as indicated at the top of the column. boldface=highest, underline=2nd highest in column.

| | Average Downstream Accuracy — `Unseen Tasks` | | | | |
| | 100 classes / 40k images | | | 237 classes / 100k images | |
Pretraining Dataset	5NN	Linear Probing	Finetuning	Linear Probing	Finetuning
Scratch	-	-	76.86	-	76.86
Random	51.80	74.68	83.97	74.11	83.49
Domain Randomization	45.06	56.96	72.64	69.12	78.15
Imagenet*	**54.12**	<u>75.47</u>	<u>84.78</u>	<u>81.33</u>	<u>87.84</u>
Task2Sim	<u>53.06</u>	**79.25**	**87.05**	**82.05**	**88.77**

Table 3. Comparing the downstream accuracy on unseen tasks for the Task2Sim chosen pretraining dataset and other baselines. Task2Sim also generalizes well to "unseen" tasks, not encountered during training, maintaining an edge over other synthetic data, while still being competitive with Imagenet. *Imagenet subsampled as in Table 2. boldface=highest, underline=2nd highest in column.
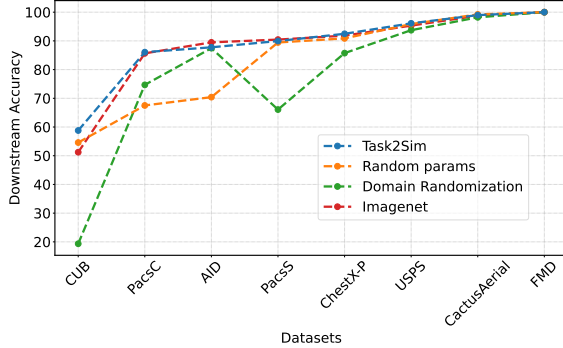

Figure 4. Performance of Task2Sim vs baselines on 8 unseen tasks for 237 class / 100k image pre-training datasets evaluated with full-network finetuning. Best viewed in color.

objects of TDW from 237 categories, and kept the number of images at roughly 400 per class, resulting in about 100k images total, regenerating a new dataset with the simulation parameters corresponding to the different synthetic image generation methods. On average, over the 12 seen tasks, simulation parameters that Task2Sim finds are better than Domain Randomization and Random selection and are competitive with Imagenet pre-training, both for the subset of classes that Task2Sim is trained using, and when a larger set of classes is used. Figure 3 shows accuracies for the 12 seen datasets for different methods, on the 237 category

100k image pre-training set.

Performance on Unseen Tasks. Table 3 shows average downstream accuracy over 8 unseen datasets, of a Resnet-50 pretrained on different datasets. We see that Task2Sim generalizes well, and is still better than Domain Randomization and Random simulation parameter selection. Moreover, it is marginally better on average than Imagenet pretraining for these tasks. Figure 4 shows the accuracies from the last column of Table 3 over the 8 individual unseen tasks.

4.3. Analysis

Task2Sim Outputs. Figure 5 shows the output distribution from the trained Task2Sim model for different seen and unseen tasks. Each output shows the probability assigned by the model to the output 1 in that particular simulation parameter. From the outputs, we see the model determines that in general for the set of tasks considered, it is better to see a single pose of objects rather than multiple poses, and that it is better to have scene lighting intensity variations in different images than have lighting of constant intensity in all images. In general, adding material variability was determined to be worse for most datasets, except for SVHN. Comparing predictions for seen vs unseen tasks, we see that Task2Sim does its best to generalize to unseen tasks by relating them to the seen tasks. For *e.g.*, outputs for ChestXPneumonia are similar to ChestX, while outputs

of CactusAerial are similar to those of EuroSAT, both being aerial/satellite image datasets. A similar trend is also seen in PacsS and Sketch both of which contain hand-sketches, and for CUB and CropDisease, both natural image datasets.

Another inspection shows Task2Sim makes decisions that are quite logical for certain tasks. For instance, Task2Sim turns off the "Light Color" parameter for CUB. Here, color plays a major role in distinguishing different birds, thus needing a classifier representation that should not be invariant to color changes. Indeed, from Figure 9, we see that the neighbors of Task2Sim are of similar colors.

Effect of Number of Pretraining Classes. In Figure 6, we plot the average accuracy with full network finetuning on the 12 seen downstream tasks. On the x-axis, we vary the number of classes used for pre-training, with 1000 images per class on average (200 classes=200k images). We see all pre-training methods improve with more classes (and correspondingly more images) at about similar rates. Task2Sim stays better than Domain Randomization and competitive with (about 2% shy of) pre-training with an equivalent subset (in number of classes and images) of Imagenet.

Effect of Number of Different Objects Per Class. In TDW, we have 2322 object meshes from 237 different categories. In Figure 7, we vary the number of object meshes used per category. The point right-most on the x-axis has 200k images with all objects used, and moving to the left, the number of images reduces proportionately as a fraction of these objects are used (the number of categories being the same). We find that with increasing number of different objects used for each category, Domain Randomization improves downstream performance at a slightly higher rate than our proposed Task2Sim.

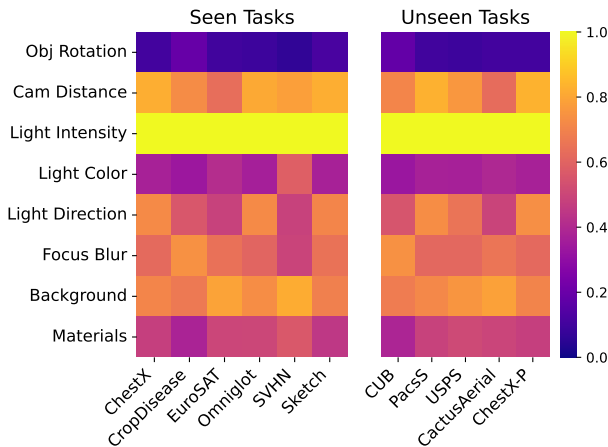**Effect of Number of Pretraining Images.** In Figure 8, we

Figure 5. Task2Sim outputs for different seen and unseen tasks. Values shown are predicted probability of value 1 in the specific simulation parameters. Best viewed in color.
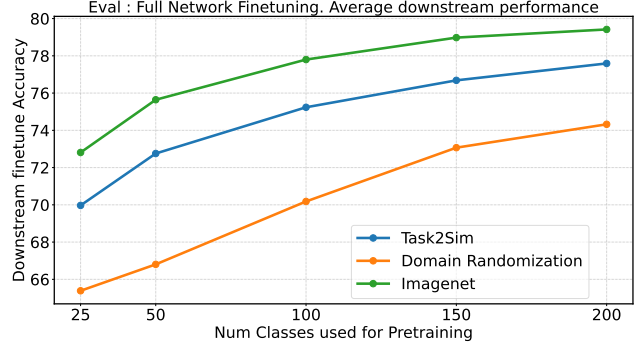

Figure 6. Avg performance over 12 seen tasks at different number of classes for pre-training. All methods improve performance at similar rates with the addition of more classes.
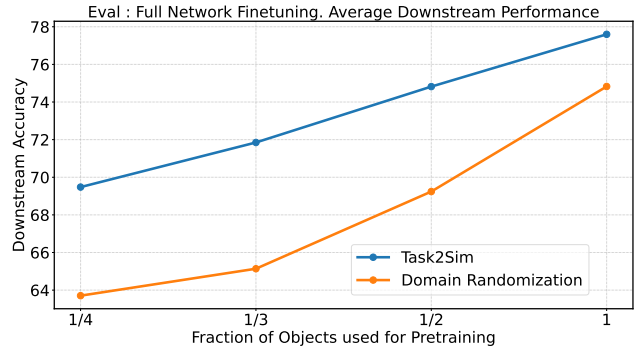

Figure 7. Avg performance over 12 seen tasks at different number of object meshes used per category for generating synthetic pre-training data. Both methods of synthetic data generation improve performance with addition of more objects with Domain Randomization improving at a slightly higher rate.
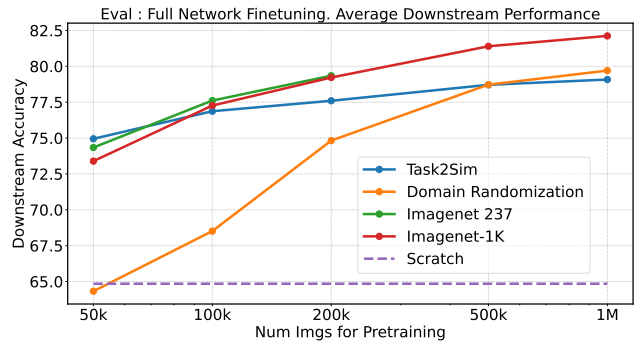

Figure 8. Task2Sim performance (avg over 12 seen tasks) vs other methods at different number of images for pretraining. Task2Sim is highly effective at fewer images. Increasing the number of images improves performance for all methods, reaching a saturation at a high enough number. See Section 4.3 for more discussion.

show average downstream task accuracy, for the 12 seen tasks, with different number of images used for pretraining. All methods, except Imagenet-1K and Scratch, use 237 image categories, with synthetic datasets using all available
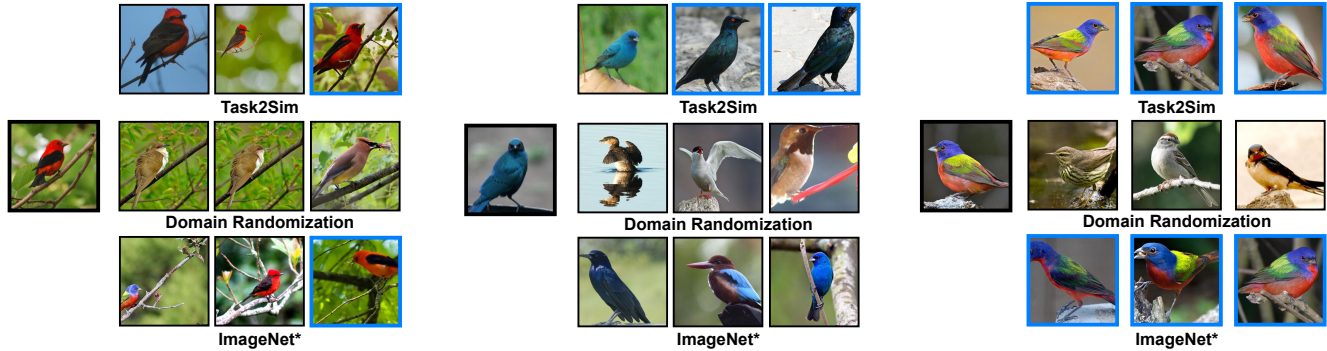
Figure 9. 3 nearest neighbors of three test examples from the CUB dataset based on different pretrained feature representations (top: Task2Sim, middle: domain randomization, bottom: ImageNet*). Neighbors with a blue box share the same class with the anchor image on the left. For Task2Sim, similar to Imagenet, the neighbors are of similar color which suggests that the pretrained representation captures color similarity which can be crucial for identifying different bird species. Best viewed in color.

object models. Imagenet-237 is a subset of Imagenet-1K containing 237 categories that were randomly picked. We see Task2Sim is highly effective in the regime where fewer images are available for pre-training, and is even slightly better than pre-training with Imagenet at 50k images. It maintains its advantage over non-adaptive pretraining up to a significant extrapolation of 500k images, having only trained using smaller datasets (of 100 classes and 40k images). At 1M images, it is still competitive with Imagenet pre-training and is much better than training from scratch.

We also observe that all methods improve when more pre-training images are available, although the rate of improvement decreases as we move along positive X-direction. Initially, Domain Randomization improves at a higher rate than Task2Sim and at 1M pretraining images, matches its performance. This is likely because at a higher number of images, even when there are all variations possible from simulation in each image (corresponding to Domain Randomization), the deep feature extractor grows robust to the variations which may not add any value to the representation for specific downstream tasks.

Our hypothesis is that at a fixed number of categories there may exist some point in number of pre-training images when the above robustness can be good enough to match Task2Sim's downstream performance. With a 237-category limit from TDW and using the set of variations from our 8 chosen parameters, 1M images seems to be this point. However, as the number of classes increases, this point shifts towards higher number of images. As evidence, consider Figure 6, where we see that as more classes of objects are added with more data, different methods improve at similar rates. Moving further along positive X, if this holds with more classes, Task2Sim maintains its edge over Domain Randomization even at higher numbers of images. This suggests a non-adaptive pre-training method like Domain Randomization has potential to be as effective on average as Task2Sim, but only at the cost of more pre-training images.

However, this cost would keep increasing as pre-training data encompasses more object categories, and would be unknown without experimentation.

For additional results and discussion, we refer readers to the supplementary.

5. Conclusion

We saw the approach that is optimal for downstream performance in using synthetic data for pre-training is to specifically adapt the synthetic data to different downstream tasks. In this paper, we parameterized our synthetic data via different simulation parameters from graphics engines, and introduced Task2Sim, which learns to map downstream task representations to optimal simulation parameters for synthetic pretraining data for the task. We showed Task2Sim can be trained on a set of "seen" tasks and can then generalize to novel "unseen" tasks predicting parameters for them in one-shot, making it highly practical for synthetic pre-training data generation. While a large portion of contemporary representation learning research focuses on self-supervision to avoid using labels, we hope our demonstration with Task2Sim motivates further research in using simulated data from graphics engines for this purpose, with focus on adaptive generation for downstream application.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, 2019. 2, 3

[2] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *CoRL*, 2021. 1, 2

[3] Harkirat Singh Behl, Atilim Güneş Baydin, Ran Gal, Philip HS Torr, and Vibhav Vineet. Autosimulate:(quickly) learning synthetic data generation. In *ECCV*, 2020. 3

[4] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. 5

[5] 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

[6] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014. 5

[7] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 5

[8] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017. 2

[9] 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

[10] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *ECCV*, 2020. 3

[11] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, et al. Threed-world: A platform for interactive multi-modal physical simulation. In *NeurIPS, Datasets Track*, 2021. 2, 3, 5

[12] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *ICML*, 2018. 3

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2

[14] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *CVPR*, 2019. 3

[15] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021. 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[17] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. 5

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[19] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 2

[20] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994. 5

[21] Ashraful Islam, Chun-Fu Chen, Rameswar Panda, Leonid Karlinsky, Richard Radke, and Rogerio Feris. A broad study on the transferability of visual representations with contrastive learning. *arXiv preprint arXiv:2103.13517*, 2021. 5

[22] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 2

[23] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, 2019. 3

[24] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018. 5

[25] Rawal Khirodkar, Donghyun Yoo, and Kris Kitani. Domain randomization for scene-specific car detection and pose estimation. In *WACV*, 2019. 2

[26] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5820–5829, 2021. 3

[27] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2

[28] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through proba-

bilistic program induction. *Science*, 350(6266):1332–1338, 2015. 5

[29] John Langford and Tong Zhang. Epoch-greedy algorithm for multi-armed bandits with side information. *Advances in Neural Information Processing Systems (NIPS 2007)*, 20:1, 2007. 4

[30] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 5

[31] James J Little and Alessandro Verri. Analysis of differential and matching methods for optical flow. 1988. 2

[32] Efren López-Jiménez, Juan Irving Vasquez-Gomez, Miguel Angel Sanchez-Acevedo, Juan Carlos Herrera-Lozada, and Abril Valeria Uriarte-Arcia. Columnar cactus recognition in aerial images using a deep learning approach. *Ecological Informatics*, 52:131–138, 2019. 5

[33] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. In *AISTATS*, 2019. 3

[34] 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

[35] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *ECCV*, 2020. 3

[36] Hiroaki Mikami, Kenji Fukumizu, Shogo Murai, Shuji Suzuki, Yuta Kikuchi, Taiji Suzuki, Shin-ichi Maeda, and Kohei Hayashi. A scaling law for synthetic-to-real transfer: How much is your pre-training effective? *arXiv preprint arXiv:2108.11018*, 2021. 2

[37] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016. 5

[38] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, pages 722–729, 2011. 5

[39] Ramakant Nevatia and Thomas O Binford. Description and recognition of curved objects. *Artificial intelligence*, 8(1):77–98, 1977. 2

[40] 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

[41] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018. 4

[42] Alex Olsen, Dmitry A. Konovalov, Bronson Philippa, Peter Ridd, Jake C. Wood, Jamie Johns, Wesley Banks, Benjamin Girgenti, Owen Kenny, James Whinney, Brendan Calvert, Mostafa Rahimi Azghadi, and Ronald D. White. Deep-Weeds: A Multiclass Weed Species Image Dataset for Deep Learning. *Scientific Reports*, 9(2058), 2 2019. 5

[43] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *ICCV*, 2015. 2

[44] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *ICRA*, 2019. 2

[45] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *CVPR*, 2018. 2

[46] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024, 2017. 4

[47] Stephan R Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing photorealism enhancement. *arXiv preprint arXiv:2105.04619*, 2021. 2

[48] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 1

[49] Cesar Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez. Procedural generation of videos to train deep action recognition networks. In *CVPR*, 2017. 2

[50] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[51] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 41(4), 2018. 2

[52] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *ICLR*, 2019. 3

[53] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 2

[54] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 2

[55] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 1

[56] 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. 1

[57] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. In *NeurIPS*, 2020. 3

[58] Yingtao Tian, Chikahiko Suzuki, Tarin Clanuwat, Mikel Bober-Irizar, Alex Lamb, and Asanobu Kitamoto. Kaokore: A pre-modern japanese art facial expression dataset. *arXiv preprint arXiv:2002.08595*, 2020. 5

[59] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 1, 2

[60] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 2

[61] Gül Varol, Ivan Laptev, Cordelia Schmid, and Andrew Zisserman. Synthetic humans for action recognition from unseen viewpoints. *International Journal of Computer Vision*, 129(7):2264–2287, 2021. 2

[62] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 3

[63] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011. 5

[64] Haohan Wang, Songwei Ge, Eric P. Xing, and Zachary C. Lipton. Learning robust global representations by penalizing local predictive power. *arXiv preprint arXiv:1905.13549*, 2019. 5

[65] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2106, 2017. 5

[66] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 3

[67] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-mei Hwu, Thomas S Huang, and Honghui Shi. Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In *CVPR*, 2020. 2

[68] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. 4

[69] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018. 3

[70] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *CVPR*, 2019. 3

[71] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 2

[72] Gui-Song Xia, Jingwen Hu, Fan Hu, Baoguang Shi, Xiang Bai, Yanfei Zhong, Liangpei Zhang, and Xiaoqiang Lu. Aid: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, 2017. 5

[73] Dawei Yang and Jia Deng. Learning to generate 3d training data through hybrid gradient. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 779–789, 2020. 3

[74] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, 2019. 2, 3

[75] Yang Zhang, Philip David, Hassan Foroosh, and Boqing Gong. A curriculum domain adaptation approach to the semantic segmentation of urban scenes. *IEEE transactions on pattern analysis and machine intelligence*, 42(8):1823–1841, 2019. 2

[76] Yide Zhang, Yinhao Zhu, Evan Nichols, Qingfei Wang, Siyuan Zhang, Cody Smith, and Scott Howard. A poisson-gaussian denoising dataset with real fluorescence microscopy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11710–11718, 2019. 5