Simple Control Baselines for Evaluating Transfer Learning
Supplementary Material

Abstract

The supplementary material provides further discussions about the evaluation standard, along with additional qualitative visualizations:

1. Visualizations of depth and surface normals predictions for all pre-training methods (Section 1).
2. A discussion on the motivations behind the proposed calibrated cumulative improvement metric (Section 2).
3. Training details and hyper-parameters for all experiments in the main paper (Section 3).
4. A discussion on the variance of empirical results across multiple training runs (Section 4).
5. A note on characterization of downstream tasks using the calibrated risk curves of their scratch control baselines (Section 5).

1. Visualizations of Model Predictions.

We visualize predictions for depth and surface normals estimation (Fig. 1) for the scratch control baseline and different pre-training methods across different data-regimes. As mentioned in the main paper, we see that qualitative results show little difference between pre-training methods and scratch performance at high-data regimes.

2. Motivating the Calibrated Cumulative Improvement Metric

In Section 3.3 of the main paper, we introduce calibrated cumulative improvement as a global metric to evaluate the effectiveness of a pre-training method for a given downstream task. The main goal of this metric is to integrate the relative improvement over scratch for different data-regimes, in order to further compress the information contained in the learning curves into a single comparable number.

For such a metric to be useful when comparing different methods, it should at minimum satisfy the following property: if the calibrated risk curve for one transfer method lies strictly below another, then its metric should be higher. This property holds for the following class of functions:

$$I_w(f) = \int_{n_{low}}^{n_{high}} w(n)(\mathrm{cR}_{\mathrm{scratch}}(n) - \mathrm{cR}_f(n))dn,$$

where $w(n)$ is any strictly positive function weighting the relative improvements over scratch across different data-regimes, and n_{low} and n_{high} denote the lowest and highest dataset sizes used for transfer-learning.

Choosing $w(n) = 1$ corresponds to the original area between curves in the $\mathbf{cR_f\text{-}n}$ plot. This weighting, however, overemphasises the long-tail of the curves at the high data-regimes where little improvement occurs as the dataset size is increased (Fig1 and Fig.2). One, therefore, needs to choose $w(n)$ appropriately in order to balance the contributions from the low- and high-data regimes.

The weighting factor $w(n)$ should also take into account the inherent differences in how the calibrated risk values scale across data regimes for different downstream tasks (i.e., the inherent difficulty of learning a given downstream task, as captured by the $\mathrm{cR}_{\mathrm{scratch}}(n)$ curve of the scratch control-baseline). We, therefore, suggest setting $w(n) = \mathrm{cR}'_{\mathrm{scratch}}(n)$.

This weighting function assigns equal volume to those

	CCI values ↑				
	Normal	Depth	ImageNet	CIFAR-100	EuroSAT
SwAV IN	0.297	0.379	**0.606**	1.027	**0.624**
MOCOv2 IN	0.302	0.375	0.330	0.843	0.492
SimCLR IN	0.367	0.271	0.520	1.064	0.596
SimSiam IN	0.232	0.235	0.185	0.507	0.449
Barlow Twins IN	0.294	0.305	0.580	**1.126**	0.591
PIRL IN	0.230	0.115	0.252	0.828	0.462
Jigsaw IN	0.257	0.304	0.249	0.546	0.473
Colorization IN	0.122	0.115	0.058	0.112	0.308
SwAV Taskonomy	**0.473**	**0.431**	/	/	/
MOCOv2 Taskonomy	0.382	0.358	/	/	/

Table 1. *Do different tasks benefit differently from self-supervised pre-training?* The complete list of CCI values for different pre-training methods on different downstream tasks (corresponding to the plots provided in Fig. 4 of the main paper).

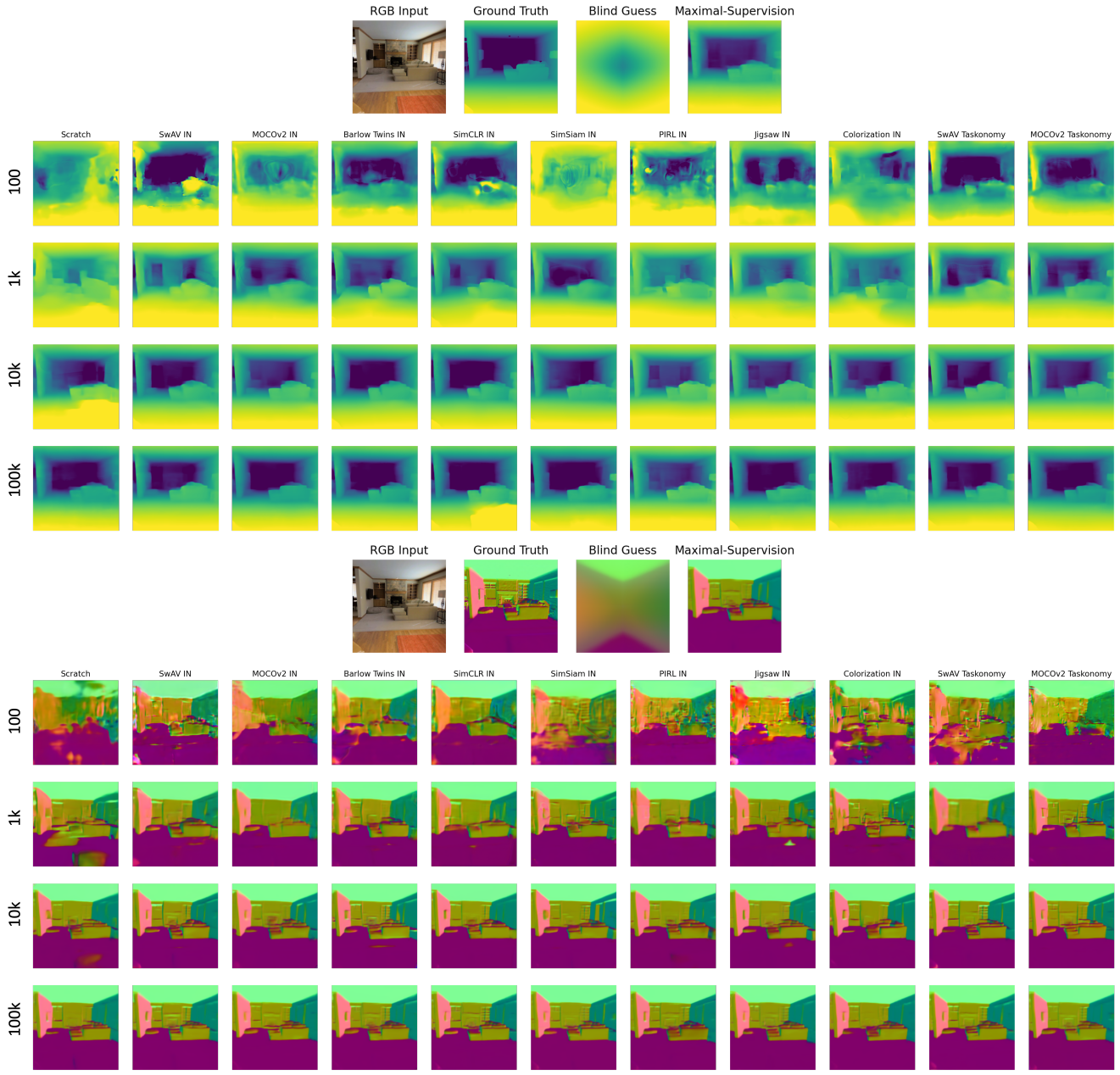

Figure 1. Visualizations of depth and surface normals predictions for different methods across different data-regimes. It can be observed that the differences between methods become insignificant at the high data-regime.

differences Δn in dataset sizes that lead to the same improvement $\Delta \mathrm{cR}_{\mathrm{scratch}}$ in terms of the calibrated risk of the scratch baseline. Using this weighting directly corresponds to calculating the integral in the $\mathbf{cR_f\text{-}cR_{scratch}}$ plot:

$$
\begin{aligned}
I_{\mathrm{cR}'_{\mathrm{scratch}}} &= \int_{n_{low}}^{n_{high}} (\mathrm{cR}_{\mathrm{scratch}}(n) - \mathrm{cR}_f(n))\mathrm{cR}'_{\mathrm{scratch}} dn \\
&= \int_{n_{low}}^{n_{high}} (\mathrm{cR}_{\mathrm{scratch}}(n) - \mathrm{cR}_f(n)) d(\mathrm{cR}_{\mathrm{scratch}}(n)) \\
&= \int_{\mathrm{cR}_{\mathrm{low}}}^{\mathrm{cR}_{\mathrm{high}}} (\mathrm{cR}_{\mathrm{scratch}} - \mathrm{cR}_f(\mathrm{cR}_{\mathrm{scratch}})) d(\mathrm{cR}_{\mathrm{scratch}}),
\end{aligned}
$$

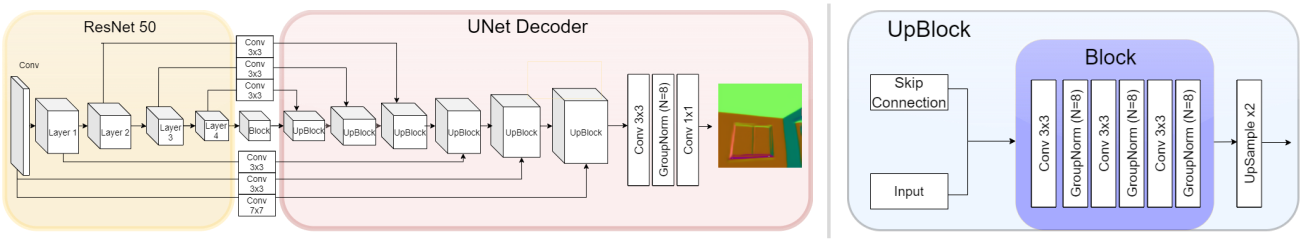

Figure 2. The network architecture used for pixel-wise regression tasks. We use a ResNet50 encoder and a UNet decoder with 6 upsample blocks and skip connections.

where cR_{low} and cR_{high} denote the calibrated risk of the scratch control baseline for dataset sizes n_{low} and n_{high} respectively. Normalizing the last expression using the overall area under the scratch curve corresponds to the proposed calibrated cumulative improvement metric:

$$\text{CCI}_f = \frac{\int_{\text{cR}_{\text{low}}}^{\text{cR}_{\text{high}}} (\text{cR}_{\text{scratch}} - \text{cR}_f) d(\text{cR}_{\text{scratch}})}{\int_{\text{cR}_{\text{low}}}^{\text{cR}_{\text{high}}} \text{cR}_{\text{scratch}} d(\text{cR}_{\text{scratch}})}.$$

Table 1 shows the complete CCI values for Section 5.2 of the main paper.

3. Training Details.

3.1. Pre-trained encoders

We use ImageNet pre-trained ResNet50 encoders from VISSL [1]. All encoders, except for colorization, were used without any modifications to the VISSL implementation. For colorization, we follow the setup of [2] and modify the `conv1` layer to take a single-channel input.

3.2. Pixel-wise Tasks

For all pixel-wise regression tasks we sample random subsets of sizes 100, 1K, 10K and 100K from the `full+` training split of the Taskonomy dataset [3]. We further split them into train and validation sets with a ratio of 80 / 20. For evaluation, we sampled 60K test images from the corresponding test split of the `full+` version.

As a decoder, we use a standard UNet with 6 upsample blocks (Fig. 2). Skip connections are used, where the corresponding features are extracted from `conv1`, `layer1`, `layer2`, `layer3` and `layer4` of the ResNet encoder using a convolution to match the size of the corresponding UNet upsampling block.

We train the model parameters by optimizing the L_1 loss with the AdamW optimizer. We use a weight decay of 1e-3 and a learning rate of 1e-3. When performing transfers, we also warm-up the learning rate for the first 1000 steps. We use a batch size of 64.

3.3. Classification Tasks

ImageNet. We sample random subsets to create different data-regimes: 31.25K, 62.5K, 125K, 250K, 500K and 1M. We use a ratio of 80 / 20 to create the train and validation splits. We use SGD optimizer with momentum 0.9 and weight decay is 1e-4, and train for 100 epochs with a batch size of 256. For transfer models, the initial learning rate is set to 0.1, and warmed-up for the first 5 epochs, after which it is decayed by 0.1 every 30 epochs. We use standard horizontal flip and random resized crop augmentations. The 50K ImageNet validation split is used to evaluate the final performance.

CIFAR100. We sample random subsets to create different data-regimes: 1.5K, 3.125K, 6.25K, 12.5K, 25K, and 50K. We use a ratio of 80 / 20 to create the train and validation splits. Additionally, we create subsets with 3, 5, 7, 10 training images per class and for validation we use one image per class. We set the batch size to 64, and use horizontal flipping and random crop augmentations. We run training for 40k training steps using cross-entropy loss function and apply early stopping (that is, if validation error doesn't decrease by 1e-4 after 10k steps, training is stopped). We use the AdamW optimizer with a weight decay of 1e-4. The learning rate is set to 1e-3 for scratch models, and 1e-4 when fine-tuning pre-trained encoders. Results for all data regimes, except maximum data regime, are averaged across 2 runs.

EuroSAT. EuroSAT dataset doesn't have default train, validation, and test splits; so we randomly split it using a ratio of 8/1/1 respectively, while keeping the distribution of classes the same. Different data-regimes consist of the following dataset sizes: 18, 37, 75, 303, 1.21K, 4.86K, 19.44K. For dataset sizes of 18, 37 and 75 we create subsets using stratification and use 1 image per class for validation. We only use horizontal flip for data augmentation. For training, we use the same hyper-parameters as CIFAR100.

4. Variance Analysis.

To give an idea about the general scale of variances for calibrated risk values across different training runs (i.e., due

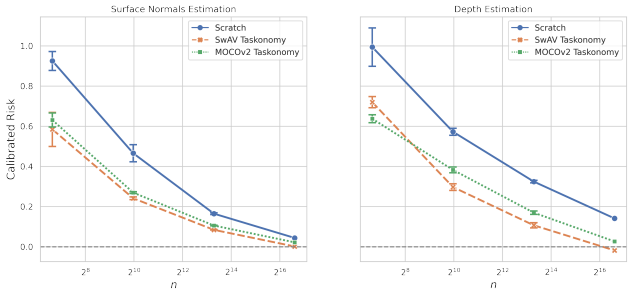

Figure 3. **Variance Analysis.** We show standard deviations over the choice of training samples for the scratch control baseline and two pre-training methods (namely, MOCOv2 and SwAV).

to factors like the random sub-sampling of transfer training data in different data-regimes, and the initialization of network parameters), we compute standard deviations for a limited set of pre-training methods (i.e., the scratch baseline, SwAV Taskonomy, and MOCOv2 Taskonomy) using multiple training runs (namely, 4 runs for 100 and 1000 data-regimes and 3 runs for 10000). The results are shown in Fig.3. We generally observe that the means for different methods do not lie within the standard deviations of other methods, suggesting that valid comparisons can be made. We also observe that standard deviations in general are relatively low.

5. Characterizing the Sample Complexity of Tasks via Scratch Calibrated Risks

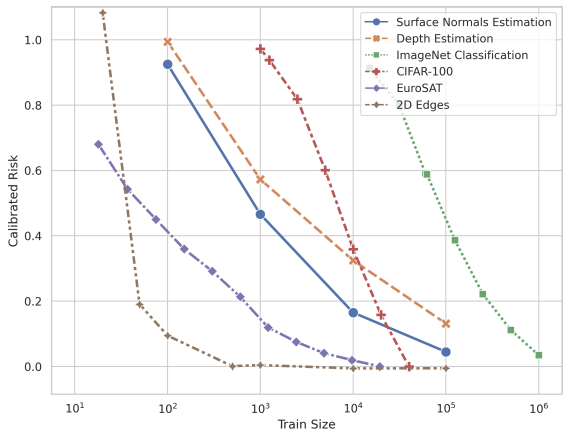

Figure 4. **Calibrated risk curves for the scratch model of different tasks**. It can be observed that the learning speed and the required number of training examples, even for those that used exactly the same dataset (i.e. surface normals, depth, and 2D edges which all used Taskonomy dataset). This is a useful characterization of tasks.

In this section, we discuss the calibrated risk curves of scratch networks can behave differently for different tasks and how that can be an informative signal. Looking at Fig. 4, it can be observed that learning speed, as indicated by the number of training samples, varies between different tasks. This is true for the tasks that used *exactly the same datasets* (i.e. surface normals, depth, and 2D edges) – hence the training/testing dataset is not a confounder, and the observed variance can be attributed to the inherent specifics of each task and their interplay with the learning machinery (the inductive biases in the network, the optimizer, etc). Thus one can use the calibrated risk of scratch as a particular characterization of a task, e.g. in terms of its inherent sample complexity (i.e. how much data learning a certain tasks demands given a certain learning machinery). This is particularly informative in the context of transfer and self-supervised learning since, as evidenced in Fig. 4, certain tasks need massively less data to learn, thus the value of developing a pre-trained feature for them is proportionally less. This view extracted out of scratch calibrated risk curves can be extended to aspects besides learning speed as well, if one plots the curves for other forms of generalization, e.g. across different domains.

We included 2D-edges (i.e., the sobel filter) as an additional pixel-wise regression task from Tasknonomy's dictionary [3] in this plot to make an illustration using a task known to be relatively simpler. However, the variance exists across other tasks too, even those that seem similar to each other (e.g. depth and surface normals). This further supports the potential usefulness of this quantification.

References

[1] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. https://github.com/facebookresearch/vissl, 2021. 3

[2] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6391–6400, 2019. 3

[3] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 3, 4