Contrasting Contrastive Self-Supervised Representation Learning Models

Klemen Kotar[1], Gabriel Ilharco[2], Ludwig Schmidt[2], Kiana Ehsani[1], Roozbeh Mottaghi[1,2]
[1]PRIOR @ Allen Institute for AI, [2] University of Washington

Abstract

In the past few years, we have witnessed remarkable breakthroughs in self-supervised representation learning. Despite the success and adoption of representations learned through this paradigm, much is yet to be understood about how different training methods and datasets influence performance on downstream tasks. In this paper, we analyze contrastive approaches as one of the most successful and popular variants of self-supervised representation learning. We perform this analysis from the perspective of the training algorithms, pre-training datasets and end tasks. We examine over 700 training experiments including 30 encoders, 4 pre-training datasets and 20 diverse downstream tasks. Our experiments address various questions regarding the performance of self-supervised models compared to their supervised counterparts, current benchmarks used for evaluation, and the effect of the pre-training data on end task performance. We hope the insights and empirical evidence provided by this work will help future research in learning better visual representations.

1. Introduction

Learning compact and general representations that can be used in a wide range of downstream tasks is one of the holy grails of computer vision. In the past decade, we have witnessed remarkable progress in learning representations from massive amounts of *labeled* data [30, 48, 22]. More recently, *self-supervised* representation learning methods that do not rely on any explicit external annotation have also achieved impressive performance [21, 34, 7, 20, 5]. Among the most successful approaches are *contrastive* self-supervised learning methods, which even surpass their supervised counterparts in certain settings. These methods typically learn by contrasting latent representations of different augmentations, transformations or cluster assignments of images. With a sufficient amount of transformations and images to contrast, the model is driven to learn powerful representations.

The most common protocol for comparing representations learned by self-supervised methods is to pre-train models on a large dataset such as ImageNet [13] without using class labels and then use the learned representations for training end tasks such as image classification, object detection or segmentation. Although this protocol has been widely adopted, it provides an incomplete picture of progress, since the noticeable similarities between common pre-training and end tasks might lead to biased and optimistic estimates of performance.

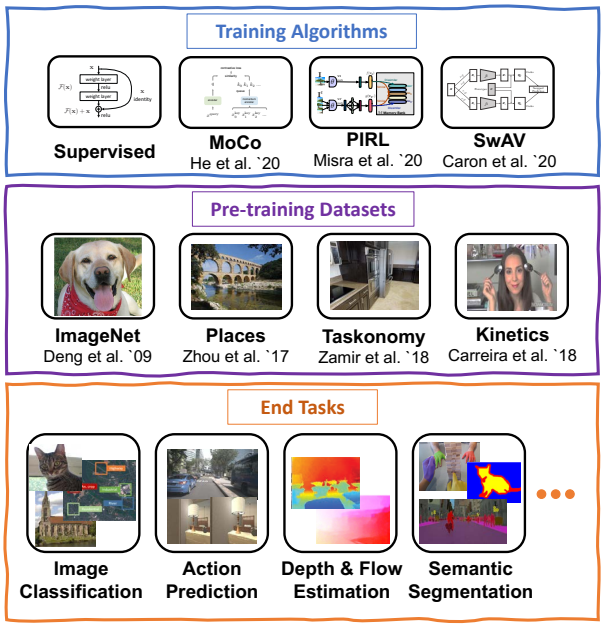

Figure 1. Our goal is to study recently proposed contrastive self-supervised representation learning methods. We examine three main variables in these pipelines: training algorithms, pre-training datasets and end tasks. We consider 4 training algorithms, 4 pre-training datasets and 20 diverse end tasks for this study.

In this work, we provide a comprehensive study of representations learned by contrastive self-supervised methods. We explore various alternatives for algorithms, pre-training datasets and end tasks (Figure 1), covering a total of 735 experiments, using 4 algorithms, 4 pre-training datasets and 20 diverse end tasks. Our goal is to provide answers to the following open questions: (1) Does supervised learning on ImageNet provide a good default encoder choice? (2) Is ImageNet accuracy a good metric for measuring the progress

on self-supervised representation learning? (3) How do different training algorithms compare for different end tasks? (4) Does self-supervision provide better encoders for certain types of end tasks? (5) Do the appearance statistics of pre-training data affect the end task performance? (6) Do we learn poor representations when using highly unbalanced datasets?

We perform an extensive set of experiments to systematically analyze contrastive self-supervision and provide answers to the above questions. We observe a mixture of unintuitive and intuitive results, which better demonstrate the characteristics of contrastive self-supervised models. We hope these analyses better enlighten the path towards learning and evaluation of self-supervised representations.

2. Related Work

We provide an overview of self-supervised representation learning methods, with an emphasis on contrastive representation learning and previous analyses of learned visual representations.

Self-supervised representation learning. To circumvent the need for explicit supervision, various self-supervised approaches have been proposed in previous works. A number of different "pretext" tasks have been proposed with the goal of training visual encoders, for instance: predicting the spatial configuration of images [15], colorizing grayscale images [57], finding the correct ordering of jigsaw puzzles [37], backprojecting to the latent space of GANs [16], counting primitives [38], cross-channel image prediction [58], generating image regions conditioned on their surroundings [42] and predicting the orientation of an image [18]. Previous work also explored learning from videos by using ego-motion as supervisory signal [1, 23], tracking similar patches [53], predicting future frames [52] and segmentation based on motion cues [41]. The recent contrastive methods, which are the focus of this study, outperform these approaches and are described next.

Contrastive representation learning. Here, we discuss a selection of related contrastive learning methods. Contrastive Predictive Coding (CPC) [51] learns a representation by predicting future latent representations using an autoregressive model and a contrastive loss, DIM [14] maximizes the mutual information between a region of the input to the encoder and its output, MoCo [21, 9] maintains a large memory bank of samples for computing the contrastive loss, SimCLR [7, 8] does not use a memory bank and introduces a non-linear transformation between the representation and the loss function, PIRL [34] learns similar representations for different transformations of an image, and SwAV [5] avoids explicit pairwise feature comparisons, contrasting between multiple image views by comparing their cluster assignments. In this paper, we use the most recent methods that provide state-of-the-art results and have public implementations available.

Representation learning analysis. There have been various studies analyzing representations learned via supervised or self-supervised learning. [12] analyze the mismatch between training and deployment domains, [49] analyze the robustness to natural data distribution shifts compared to synthetic distribution shifts, [45] analyze the generalization capabilities of models trained on ImageNet. [55] explore the relationships between visual tasks. In contrast to these approaches, we study self-supervised approaches. [56] provide a standard benchmark for analyzing the learned representations. [2] study representations learned at different layers of networks by self-supervised techniques. [44] study the effect of invariances such as occlusion, viewpoint and category instance invariances on the learned representation. [50] study the effect of training signals (referred to as "views") on the downstream task in the self-supervised contrastive settings. [19] analyze training self-supervised models on uncurated datasets. In contrast, we analyze self-supervised *contrastive* approaches from the perspective of training algorithms, pre-training datasets and end tasks.

3. Self-supervision Variables

Given a set of images $\mathcal{X} = \{x_1, \dots, x_N\}$, the goal of a self-supervised learning algorithm Ψ is to learn parameters θ of a function f_θ that maps images x to representations in a continuous latent space. In other words, given an architecture f, we learn $\theta = \Psi_f(\mathcal{X})$. The learned representations can then be evaluated on various (supervised) end tasks $\mathcal{D} = \{(\bar{x}_1, y_1), \dots, (\bar{x}_M, y_M)\}$ with pairs of inputs and labels.[1] There are various variables involved in this pipeline. We primarily focus on three variables and their relationship: *training algorithms* Ψ, *pre-training datasets* $\mathcal{X}$ and *end tasks* $\mathcal{D}$. Below, we describe each of these variables and the choices for our experiments.

3.1. Training Algorithms

The representation learning algorithms we consider are contrastive self-supervised learning approaches that have recently shown substantial improvements over the previous methods. In this study, we investigate the influence of the training algorithms on the learned representations. We use different algorithms: PIRL [34], MoCov1 [21], MoCov2 [9] and SwAV [5]. The reason for choosing these specific algorithms is that they achieve state-of-the-art results on standard end tasks, have a public implementation available, and do not require heavy GPU memory resources, enabling a large-scale analysis. The list of all 30 encoders is in Appendix H.

[1] The nature of the labels y vary widely depending on the end task, ranging from class labels to depth estimates. More details can be found in Section 3.3.

3.2. Pre-training Datasets

The de facto standard used for pre-training with contrastive methods is the ImageNet [13] dataset [34, 21, 7, 5]. ImageNet is an object-centric dataset with a balanced number of images for each category. Some works [21, 5] have also used less-curated datasets such as Instagram-1B [33]. In this paper, we perform a systematic analysis of the datasets in two dimensions. First, we use datasets with different appearance statistics. We use Places365 [59], Kinetics400 [25] and Taskonomy [55] in addition to ImageNet for pre-training. Places is a dataset that is scene-centric and includes images of various scene categories (e.g., stadium and cafeteria). Kinects is an action-centric dataset and involves videos of activities (e.g., brushing hair and dancing). Taskonomy is a dataset of indoor scene images. Examples from each dataset are provided in Figure 1.

These datasets are larger than ImageNet. To eliminate the effects of training data size, we subsample these datasets to make them the same size as ImageNet (1.3M images). We uniformly sample from each category of the Places dataset. For Kinetics, we sample at a constant frame rate across all videos. For Taskonomy, we uniformly sample across the different building scenes. Note that the labels are discarded after subsampling since we train the models by self-supervision. Moreover, to explore the effect of using a pre-training dataset with a mixed distribution of appearance, we randomly select a quarter of each of the aforementioned datasets and combine them to form a dataset with non-uniform appearance statistics. We refer to this dataset as 'Combination'.

The self-supervised methods are typically pre-trained on ImageNet, which is a category-balanced dataset. We also investigate the representations learned on a set of unbalanced datasets. We create two unbalanced variations of ImageNet. First, we sample images from each category by linearly increasing the number of samples i.e., we sample one image from category 1, two images from category 2, etc. We refer to this dataset as 'ImageNet-½-Lin' and it consists of $500.5K$ images. In the second variation, the number of samples increases according to an exponential distribution.[2] We refer to this unbalanced variation as 'ImageNet-¼-Log' and it consists of $250K$ images. To have comparable size datasets, we create smaller balanced variations of the ImageNet dataset by uniformly sampling a quarter and half of the images in each category. We refer to these as 'ImageNet-¼' and 'ImageNet-½'.

3.3. End Tasks

Representations learned from self-supervised methods can be used for various end tasks, such as image classification, object detection and semantic segmentation. Image classification has been considered as the primary end task for benchmarking contrastive self-supervised techniques [19]. Although this task is a reasonable choice for measuring progress, it might not be an ideal representative for various computer vision tasks that are different in nature. In this study, we consider a wide range of end tasks. To ensure diversity, we study 20 tasks grouped into four categories based both on the structure of the output and the nature of the task (Figure 2). The output type of each end task can be classified into two broad categories: *image-level* and *pixel-wise*. The former involves reasoning about a region in the image or the entire image, while the latter reasons about each pixel.[3] Within each category, we consider two categories of tasks based on their nature: *semantic* and *structural*. Semantic tasks are the ones that associate semantic information such as category labels to image regions (e.g., semantic segmentation or image classification). Structural tasks, on the other hand, provide information about some structure in the image (e.g., depth estimation). We note that the boundary between these two types of tasks can become blurry and some tasks can be considered both structural and semantic (e.g., walkable surface estimation). We put these tasks in the closest category. Hence, we have four types of tasks in total:

- **Semantic Image-level.** In these tasks, we provide semantic labels for a region or the entire image. Examples include image classification (e.g., ImageNet classification) and scene classification (SUN397 [54] classification). This is the most populated category since most common vision tasks fall into this category.

- **Structural Image-level.** These tasks reason about some structural, global information in images. Example tasks in this category are counting (CLEVR-Count [56]) and egomotion estimation (estimating car movements in nuScenes [4], an autonomous driving dataset).

- **Semantic Pixelwise.** In contrast to the two previous categories, the output is pixelwise. The goal is typically to assign a semantic label to each pixel in an image. Semantic segmentation of images in Cityscapes dataset [11] and hand segmentation in EgoHands [3] dataset are example tasks in this category.

- **Structural Pixelwise.** The fourth category involves providing pixelwise predictions for structural properties in a scene. Examples include estimating pixelwise depth in the AI2-THOR [26] framework and walkable surface estimation in the NYU Depth V2 [36] dataset.

[2] More specifically, we sample λe^{an+b} data points for the n-th class, a,b and λ are chosen so that a single image is sampled from the first class and 1000 images are sampled from the last.

[3] While not the focus of our work, some tasks do not fit into these two categories, e.g. generating future human poses.

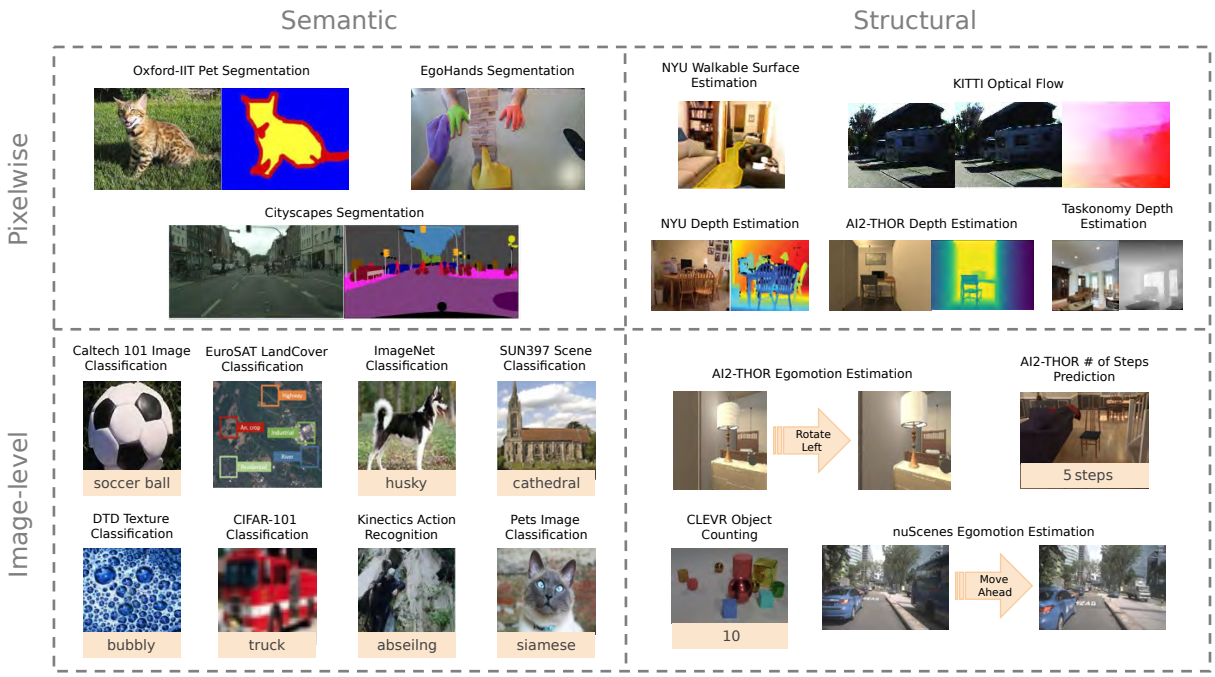

Figure 2. **End tasks.** We study a diverse set of end tasks. We categorize these tasks according to two characteristics: *semantic* vs. *structural* and *pixelwise* vs. *image-level*. We illustrate an image from each task to show the diversity of visual appearances we consider.

Figure 2 illustrates all tasks and their corresponding categories. More details on the task formulations and their datasets are in Appendix A.

4. Architecture Details

With the goal of conducting a controlled study, we fix as many variables as possible, and use the standard PyTorch [40] ResNet50 architecture for every encoder studied. Due to the diverse nature of our tasks and their outputs we have to use several different end task network architectures, but we keep them as small and standard as possible. As a result, we might not achieve state-of-the-art results on every end task. However we ensure that our results are good enough to adequately compare the performance of different learned features. In this section, we describe the architectures used for the backbone encoder and each end task in this study.

4.1. Encoders

We remove the final (classification) layer from each trained backbone model and use it as the encoder for all of our end task experiments. Our goal is to investigate the learned representation as opposed to evaluating whether it is an effective initialization. Therefore, we keep the backbone frozen and do not fine-tune the encoders for any task.

4.2. End Task Networks

The end task network is the section of the model that converts the embedding produced by the encoder into the desired task output. For each end task we have a train and test set. We train the end task network on the train set using a random initialization and then evaluate it on the test set. We use the same set of hyperparameters for each task in all settings. For further details please see Appendix B. We have 5 different architectures to suit the wide variety of our end task types.

Single Layer Classifier. This network contains a single fully connected layer. It takes as input the final ResNet embedding and outputs a vector of size n, where n is the number of classes for the task. This network is used for all the image-level classification tasks (e.g., scene classification).

Multi Input Fusion Classifier. This network contains several "single linear layer modules", each of which processes one image in a sequence. The outputs of these modules get concatenated and passed through a fusion layer. The network takes as input a series of final ResNet embeddings and outputs a vector of size n, where n is the number of classes for the task. This network is used for all the image-level classification tasks that take a sequence of images (e.g., egomotion estimation).

U-Net. This network is a decoder based on the U-Net [46] architecture—a series of consecutive convolutions followed by upsampling and pixel shuffle [47] layers. After every upsample operation, the output of an intermediary representation from the ResNet encoder of matching height and width is added via a residual connection. The final output is a tensor of size $h \times w$, where h and w are the height and width of the input image. This network is used for pixelwise depth prediction.

Siamese U-Net. This network is a modification of the U-Net network which can support two images as input. It takes the final embeddings and intermediary ResNet representations from the two images as input, then fuses them together layer by layer with a point convolution and adds them to the decoder after every convolution via a residual connection. This network is used for flow prediction.

DeepLabv3+. This network is based on the DeepLabv3+ [6] architecture. It takes as input the output of the 5th block of the ResNet and uses dilated convolutions and a pyramidal pooling design to extract information from the representations at different scales. The output is then upsampled and is added to the representation from the 2nd block of the ResNet to recover image structure information. The final output is of size $n \times h \times w$, where n is the number of output channels, h and w are the height and width of the input image. This network is used for pixelwise semantic classification tasks (e.g., semantic segmentation).

For further architectural details, please see Appendix B.

5. Analysis

In this section, we pose several questions on the relationships across pre-training algorithms, pre-training datasets and the end tasks. We discuss our experiments' design and analyze the results to provide answers to each of these questions. We perform an extensive analysis of the contrastive self-supervised models and discuss the performance trends in different settings. We also investigate which common intuition used in supervised training transfers over to the self-supervised domain. Unless noted otherwise all training algorithms have been used for the experiments. The implementation and training details are provided in Appendix C.

(1) Does supervised learning on ImageNet provide a good default encoder choice? An encoder trained with supervised learning on the ImageNet dataset has become the default backbone for many computer vision models. With the recent rise of self-supervised training algorithms we re-evaluate this assumption. For each of the 20 end tasks, we compare the best performing self-supervised encoder with the encoder trained on ImageNet in a supervised fashion. The performance improvements of self-supervised methods are shown in Figure 3, along with the dataset used for pre-training. For the ImageNet v1 and v2 classification as well as Pets classification (which is very close to the ImageNet task), the supervised model performs the best, but for all other tasks some self-supervised encoder achieves a higher performance. This indicates that a self-supervised model (e.g., an encoder trained with SwAV on ImageNet) might be a better default option in many scenarios.

Figure 3 also shows that most of the best performing models are pre-trained on ImageNet or Places. Both of these datasets are curated and structured datasets (as opposed to Kinetics and Taskonomy which are unstructured). This might suggest that self-supervised encoders might also benefit more from well-organized training data.

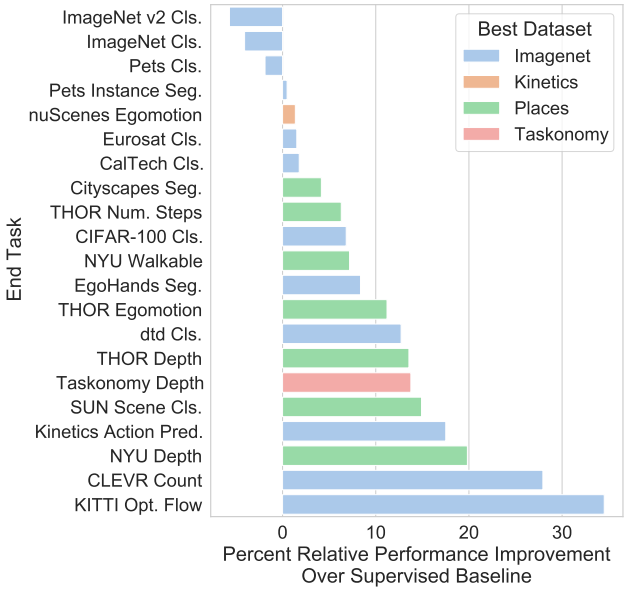

Figure 3. **Comparison of self-supervised and supervised encoders.** The percentage performance improvement of the self-supervised encoders for each end task is shown. The colors of the bars represent the dataset used for pre-training the best performing self-supervised encoder. The plot shows that the self-supervised encoders are better than an encoder trained on ImageNet in a supervised way except for the three end tasks shown on top, which are ImageNet classification and Pets classification (which is quite similar to ImageNet classification).

(2) Is ImageNet accuracy a good metric for measuring progress on self-supervised representation learning? Most recent works in self-supervised representation learning report the performance of their encoders on different tasks, but the common denominator between them is mostly the ImageNet classification task. We test a variety of encoders on our diverse set of 20 end tasks to observe how well the performance on those tasks correlates with ImageNet classification performance.

Figure 4 contrasts the performance of an encoder on ImageNet versus all other end tasks. The x axis denotes the performance of the learned representation on ImageNet classification and the y axis denotes the performance of the end tasks using the self-supervised encoder. Each point in the plot represents a different encoder obtained by different training algorithms, datasets, etc.

While we generally observe a strong correlation between the performance on ImageNet classification and other tasks in the same category (semantic image-level), there is a weaker (and sometimes even negative) correlation with

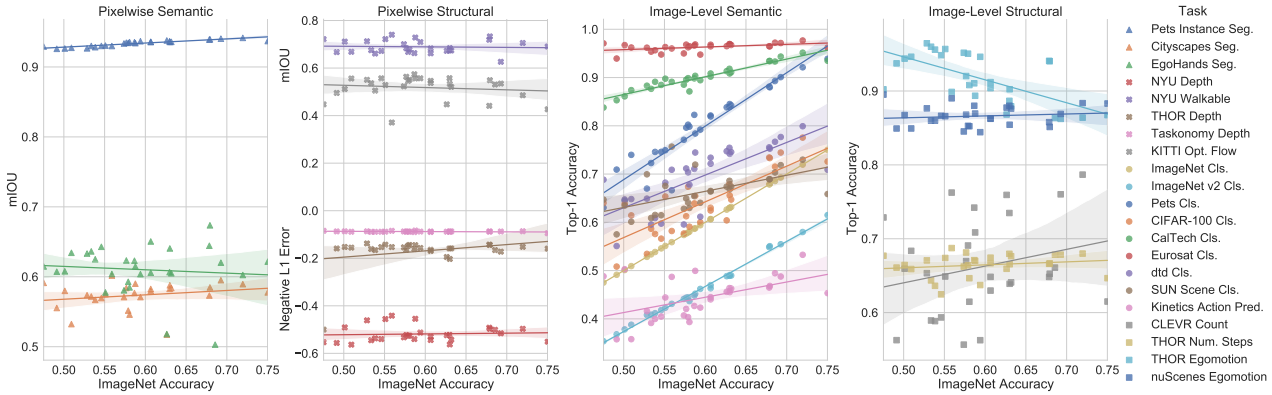

Figure 4. **Correlation of end task performances with ImageNet classification accuracy.** The plots show the end task performance against the ImageNet top-1 accuracy for all end tasks and encoders. Each point represents a different encoder trained with different algorithms and datasets. This reveals the lack of a strong correlation between the performance on ImageNet classification and tasks from other categories.

tasks in other categories (such as THOR egomotion estimation) – refer to Appendix D for Spearman and Pearson correlation analysis. This indicates that the representations that are suitable for ImageNet classification do not always transfer well to other vision tasks. The results for semantic image-level tasks are in line with the findings of [28]. However, we observe a different trend for the other task types. Note that for some end tasks the performance ceiling might have been reached. Hence, we might not observe a significant difference between different encoders for them.

The fact that we find several tasks that appear to be negatively correlated with ImageNet performance suggests that the encoders that perform quite well on ImageNet might be overfitting to a particular task type and output modality. Interestingly, the category that is most negatively correlated with ImageNet performance is image-level structural tasks, which shares relatively similar network architecture and loss function with ImageNet classification. This provides more evidence that the architecture and the loss function are not the variables that determine the correlations.

Considering these analyses, ImageNet classification does not appear to be a strong indicator of self-supervised encoder performance for various computer vision tasks.

(3) How do different pre-training algorithms compare for different end tasks? Two recent strong self-supervised algorithms are MoCov2 [9] and SwAV [5]. We train several encoders using both algorithms to determine if the trends we observe extend beyond a single algorithm. In addition, this allows us to contrast the MoCov2 and SwAV algorithms to determine if either one is a better fit for certain end tasks.

For answering this question, we consider encoders trained for 200 epochs on our pre-training datasets. Therefore, we train 10 encoders in total, using our five datasets (ImageNet, Places, Kinetics, Taskonomy, and Combination) by SwAV and MoCov2 methods. In Figure 5, for each end task, we plot the percentage difference between the average performance of MoCov2 encoders and the average performance of SwAV encoders. MoCov2 encoders tend to do better at tasks where the output is pixelwise (a notable exception is Cityscapes Segmentation). SwAV models are better at classification tasks, especially semantic classification tasks (here the notable exception is THOR egomotion estimation which is also inversely correlated with ImageNet classification).

Under typical evaluation procedures, SwAV might be considered an absolute improvement over MoCov2, since SwAV outperforms MoCov2 on ImageNet classification. However, our results suggest that this is not a universal fact. This underscores the importance of reporting performance on a diverse and standardized battery of end tasks to show a more comprehensive overview of a model's performance.

To investigate if there is some fundamental difference in the representations produced by different encoders, which explains this trend, we compute the *linear Centered Kernel Alignment* (CKA) [27] between the outputs of each ResNet block of the MoCov2 and SwAV models. We use a 10,000 image, balanced subset of ImageNet at half resolution for this evaluation. See Appendix E for details. We observe a stronger agreement between the representations in the earlier blocks and later blocks with MoCov2 models, than we do with SwAV models. These trends may suggest that MoCov2 representations are better at capturing low-level information from an image (making it more suitable for *pixelwise* tasks), while SwAV representations are better at capturing higher-level semantic information.

(4) Does self-supervision provide better encoders for certain types of end tasks? Pre-trained encoders are used for a variety of applications in computer vision, yet most reported results focus on improvements obtained on semantic tasks such as image classification, object detection and instance segmentation [5, 19]. We would like to obtain a gen-

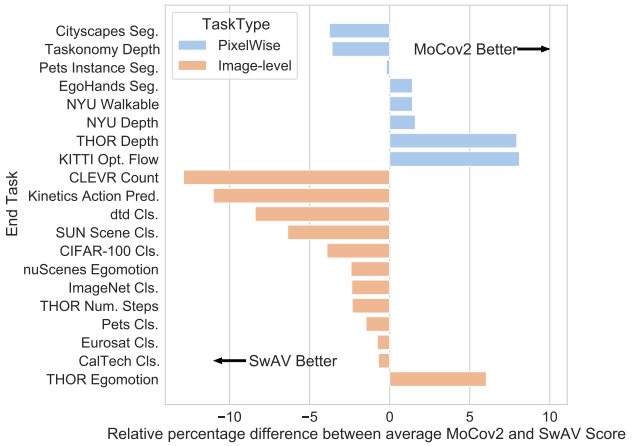

Figure 5. **Training algorithms and tasks.** For each end task, the difference between the average score of all encoders trained with MoCov2 and the average score of all encoders trained with SwAV is shown. Therefore a negative score indicates that SwAV outperforms MoCov2 on average for a given task and a positive score means the opposite. The scores are unscaled evaluation metrics (accuracy, mIOU or negative L1 error depending on the task). With some exceptions, the plot shows SwAV is generally better at *image-level* tasks, while MoCov2 is better at *pixelwise* tasks.

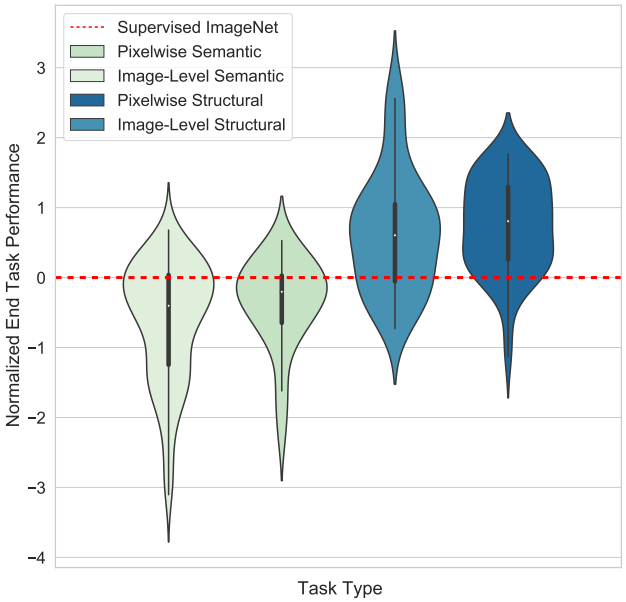

Figure 6. **Distribution of normalized performances for each category of end tasks.** The performances are normalized by first subtracting the performance of the supervised ImageNet encoder and then dividing by the std. deviation of all the performances for the task. Positive values show superior performance to the supervised ImageNet, and the negative values show otherwise. A larger width means more performance values fall in that range. The plot shows *structural* tasks benefit more from self-supervision.

eral picture of how well self-supervised encoders perform across each individual task category. Since end tasks use different success metrics, we use a normalization scheme to effectively compare them. In Figure 6 we take every performance metric obtained by a self-supervised encoder on an end task and subtract the score obtained by the supervised representation trained on ImageNet. Note that this indicates that the points with positive values outperform the supervised baseline. We then further normalize these values by dividing them by their standard deviation.

Figure 6 indicates that *structural* tasks receive a greater benefit from using a self-supervised encoder. Note that the relatively large standard deviation in this plot is due to including self-supervised encoders trained on datasets and algorithms that might not be the best match for the given task type. Note that this plot does not conflict with our observation in Figure 3 on the good performance of self-supervised encoders on *semantic* tasks. As shown in Figure 3, a self-supervised model outperforms the supervised baseline on all but three semantic image-level tasks.

(5) Do the appearance statistics of pre-training data affect the end task performance? We hypothesize using a pre-training dataset similar to the end task's will produce a better encoder. We choose 4 datasets to test this hypothesis: two structured (ImageNet and Places365) and two unstructured (Taskonomy and Kinetics400). We train two encoders on each of them (MoCov2 and SwAV, the best performing algorithms) and pair each pre-training dataset with an end task using either a dataset in the similar domain as the pre-training data (SUN397 [54] classification for Places265 [59] and Caltech101 [31] classification for ImageNet [13]) or using a subset of the same dataset (action prediction for Kinetics400 and depth estimation for Taskonomy).

In Figure 7 we plot the end task performance of MoCov2 and SwAV models trained for 200 epochs on the pre-training datasets mentioned above. The green bars indicate the encoders trained on a dataset that is similar to the end task data, while the gray bars indicate encoders trained on other datasets. The purple bars indicate the encoders trained on the 'Combination' dataset (referred to in Section 3.2).

We find that for every task, the best performing encoder is the one trained on a dataset that includes similar data. However, as Figure 7 shows, the training dataset alone is not enough to determine which encoder will perform the best, as the algorithms we use (SwAV or MoCov2) also impact the performance.

We observe that training on the 'Combination' dataset does not provide us with a model that excels at every task, therefore, simply combining different datasets with different appearance distributions might not be a good strategy for self-supervised training. Note that the combination dataset still benefits from including images similar to the end task images.

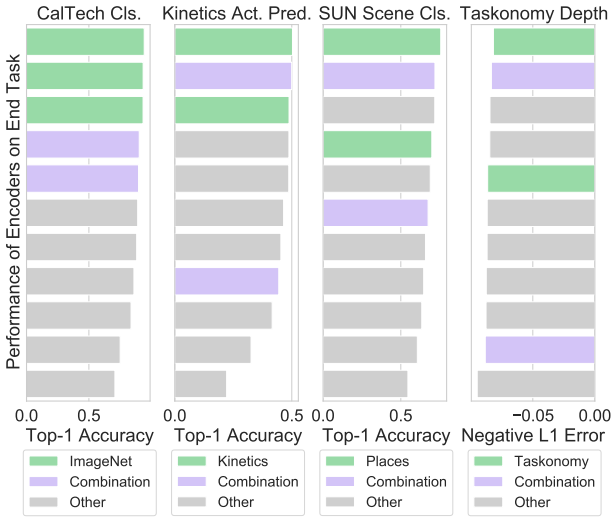

Figure 7. **Similarity of the pre-training datasets and end tasks.** Performance of all the encoders on selected end tasks is shown. Each bar represents a different encoder. The green bars represent encoders that are pre-trained on a dataset similar to/same as the end task dataset. The purple bars represent the encoders pre-trained on the 'Combination' dataset. The plot shows that encoders pre-trained on similar/same datasets have the highest score. Moreover, those encoders are superior to the encoders trained on Combination, which includes not only a subset of that dataset, but also images from other datasets.

(6) Do we learn poor representations if we use unbalanced ImageNet? So far, we have mostly addressed datasets that are curated and well-balanced. Here, we evaluate the learned representations in scenarios where we use unbalanced data for pre-training the encoders. Using unbalanced data better mimics real-world data distributions which are typically long-tailed [32].

We consider two unbalanced subsets of ImageNet (ImageNet-½-Lin and ImageNet-¼-Log) described in Section 3.2, and two correspondingly sized balanced subsets (ImageNet-½ and ImageNet-¼). Encoders are trained on each of the four ImageNet subsets using SwAV and MoCov2 for 200 epochs each, to produce 8 encoders, which are tested on the 20 end tasks. We fit a factorial ANOVA model to the end task results and find no evidence that pre-training on a balanced datasets produces a better encoder. We find that a model being pre-trained on ImageNet-½-Lin is not a statistically significant predictor of model performance (p-value = 0.0777), while a model being trained on ImageNet-¼-Log is (p-value = 0.0101) with an average end task score improvement of 1.53%. This presents weak evidence that pre-training on a heavily unbalanced dataset with contrastive learning might even produce an encoder better suited for the end tasks studied in this work. For further details see Appendix F.

6. Discussion

Here we provide a summary of the analysis. First, we showed that a backbone trained in a supervised fashion on ImageNet is not the best encoder for end tasks other than ImageNet classification and Pets classification (which is a similar end task). Second, we showed that in many cases there is little to no correlation between ImageNet accuracy and the performance of end tasks that are not *semantic image-level*. Third, we showed different training algorithms provide better encoders for certain classes of end tasks. More specifically, MoCov2 proved better for *pixel-wise* tasks and SwAV showed better performance on *image-level* tasks. Fourth, we showed that *structural* end tasks benefit more from self-supervision compared to *semantic* tasks. Fifth, we showed pre-training the encoder on the same or similar dataset to that of the end task provides higher performance. This is a well-known fact for supervised representation learning, but it was not evident for self-supervised methods that do not use any labels. Sixth, we showed that representations learned on unbalanced ImageNet is as good or even slightly better than representations learned from balanced data. The current study has some shortcomings that are noted below:

Empirical study. Our conclusions are based on empirical results. This has two major implications. First, there is no theoretical justification for the results. Second, due to computation limits and the wide range of parameters and variables involved in these types of approaches, our study does not cover all aspects related to contrastive self-supervised representation learning.

Task dichotomy. The task categorization that we studied is based on the type of output and information they capture. There are several other ways of grouping these tasks that are not studied here and are left for future work.

Variables. We focused only on three variables in the representation learning pipeline, namely, training algorithms, pre-training datasets and end tasks. There are various other factors involved in the representation learning pipeline such as network architectures and computational efficiency that are not addressed in this study.

Frozen backbone. We did not fine-tune the encoders during training for end tasks. A future direction can be exploring the trends when the encoder is fine-tuned as well.

7. Conclusion

We studied contrative representation learning as one of the most successful approaches proposed for self-supervision. Our focus was mainly on three variables in representation learning pipelines, namely, training algorithm, pre-training dataset and end task. Our rigorous analysis resulted in interesting findings about the interplay of these variables. We hope our study provides better insights for future research in this vibrant and impactful domain.

Acknowledgments

We would like to thank Luca Weihs for discussions about the statistical analyses.

References

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, 2015. 2

[2] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *ICLR*, 2020. 2

[3] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *ICCV*, 2015. 3, 11

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 3, 11

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 1, 2, 3, 6, 12

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 5

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3

[8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 2

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv*, 2020. 2, 6, 12

[10] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 11

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 3, 11

[12] Alexander D'Amour, Katherine A. Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yi-An Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *arXiv*, 2020. 2

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 3, 7, 11

[14] R Hjelm Devon, Fedorov Alex, Lavoie-Marchildon Samuel, Grewal Karan, Bachman Phil, Trischler Adam, and Bengio Yoshua. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 2

[15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

[16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017. 2

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 12

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[19] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised pretraining of visual features in the wild. *arXiv*, 2021. 2, 3, 6

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020. 1

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[23] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 2

[24] 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. 11

[25] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, 2017. 3, 11

[26] 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*, 2017. 3, 11

[27] Simon Kornblith, Mohammad Norouzi, H. Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *ICML*, 2019. 6, 12

[28] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *CVPR*, 2019. 6

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 11

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[31] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshop*, 2004. 7, 11

[32] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 8

[33] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 3

[34] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 1, 2, 3

[35] Roozbeh Mottaghi, Hannaneh Hajishirzi, and Ali Farhadi. A task-oriented approach for cost-sensitive recognition. In *CVPR*, 2016. 12

[36] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3, 11

[37] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

[38] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *ICCV*, 2017. 2

[39] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 11

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 4

[41] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 2

[42] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[43] Helber Patrick, Bischke Benjamin, Dengel Andreas, and Borth Damian. 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*, 2019. 11

[44] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv*, 2020. 2

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *ICML*, 2019. 2, 11

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4

[47] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 4

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[49] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *NeurIPS*, 2020. 2

[50] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020. 2

[51] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, 2018. 2

[52] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016. 2

[53] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2

[54] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 3, 7, 11

[55] Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 2, 3, 11

[56] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, André Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark. *arXiv*, 2019. 2, 3, 12

[57] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 2

[58] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 2

[59] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 3, 7

Appendix

A. End Tasks

The descriptions of all end tasks are provided below. Semantic Image-level, Structural Image-level, Semantic Pixelwise, and Structural Pixelwise tasks are shown with different colors.

• **ImageNet Cls.** [13] - This is a 1000 class natural image classification task. The dataset includes a variety of categories, such as coffee mugs, drum, and fire engine. The images are of varying resolution but they are all resized to 224×224.

• **ImageNet v2 Cls.** [45] - This is a natural image classification task with the same set of categories as ImageNet. This task has the same train set as ImageNet, but has a re-collected test set with the same distribution and categories.

• **Pets Cls.** [39] - This is a natural image classification task with images of cats and dogs. There are a total of 37 classes corresponding to breeds of cats and dogs including Persian, Chihuahua and Bengal. The images are of varying resolution but they are all resized to 224×224.

• **CalTech Cls.** [31] - This is a 101 class natural image classification task with pictures of objects such as planes, chairs, and animals. The images are of varying resolution but they are all resized to 224×224.

• **CIFAR-100 Cls.** [29] - This is a 100 class natural image classification task. The classes include apples, bottles and bicycles. The Images are of size 32×32 but they are all resized to 224×224.

• **SUN Scene Cls.** [54] - This is a 397 class scenery image classification task. The classes include scene categories such as cathedral, river, or archipelago.

• **EuroSAT Cls.** [43] - This is a 20 class dataset of satellite imagery classification. The spatial resolution corresponds to 10 meters per pixel and includes categories of different land use, such as Residential, Industrial and Highway. All images are resized to 224×224.

• **dtd Cls.** [10] - This is a 47 class dataset of textural imagery classification. Some textures include bubbly, lined and porous. All images are resized to 224×224.

• **Kinetics Action Pred.** [25] - This task consists of predicting the action that a person is performing from 6 ordered video frames. The dataset contains 400 classes including bowling and dining. The dataset for this task is 50,000 image frames captured from the Kinetics400 dataset videos at 6 frames per video. All images are resized to 224×224.

• **CLEVR Count** [24] - This is a synthetic visual question answering dataset designed to evaluate algorithmic visual reasoning. The task consists of classifying the number of objects in the image. All images are resized to 224×224.

• **THOR Num. Steps** - This is a task where the maximum number of forward steps (of 0.25 meters) that a robot in AI2-THOR [26] can take is predicted from a frame of the robot's point of view. This task is structured as classification, rather than regression, of the images from the simulation and the correct answer will always be between 0 and 4 steps inclusive (thus this task is a 5-way classification). This task is first proposed in this paper.

• **nuScenes Egomotion** - This is an egomotion prediction task from two consecutive frames of the nuScenes self driving car dataset [4]. The types of motion include forward, forward-left and forward-right motion as well as a no motion action. Both frames are resized to 224×224. This task is first proposed in this paper.

• **THOR Egomotion** - This is an egomotion prediction task from two consecutive frames in the AI2-THOR [26] simulator. The types of motion include moving forward, left and right rotation, and looking up and down. Frames are resized to 224×224. This task is first proposed in this paper.

• **Cityscapes Seg.** [11] - This is a semantic segmentation task where every pixel is labeled as one of 20 categories. The images consist of dashboard camera views of cities and roads. The task contains categories such as person, traffic light and sky (there is also a background class for pixels that do not fit into any other category). Crops of size 513×513 sampled from the full image are used during training, and evaluation is done at full resolution.

• **Pets Instance Seg.** - This is an instance segmentation task on the Pets dataset [39], where each image contains exactly one cat or dog. Each image (and its ground truth instance label) is resized to 224×224.

• **EgoHands Seg.** [3] - This is an instance segmentation task on a dataset of video frames of human hands performing various tasks. The videos are captured using a Google glass camera and are from the egocentric view of one person performing a task with another person. Each frame has at most 4 hands (the left and right hand of the person wearing the Google glass and the right and left hand of their partner) and each of these has its own associated class (there is also a background class). Crops of size 513×513 sampled from the full image are used during training, and evaluation is done at full resolution.

• **NYU Depth** [36] - This is a pixelwise depth prediction task on a dataset of natural images of building interiors obtained from videos. The images are resized to 224×224 and the output is predicted in meters.

• **THOR Depth** - This is a pixelwise depth prediction task on a dataset of synthetic images of building interiors produced by the AI2-THOR [26] simulator. The images are resized to 224×224 and the output is predicted in meters. This task is first proposed in this paper.

• **Taskonomy Depth** [55] - This is a pixelwise depth prediction task on a dataset of natural images of building interiors from a variety of building types. The images are resized to 224×224 and the output is predicted in meters. This is a

common task but the dataset split is first proposed in this paper.

• **NYU Walkable** [35] - This is a pixelwise detection task. Each pixel is labeled as walkable (floor, carpet, etc.) or non-walkable (wall, window, ceiling, etc). The dataset consists of images of interior rooms. All images are resized to 224×224.

• **KITTI Opt. Flow** [17] - This is an optical flow prediction task from two consecutive frames. The data comes from a self driving dataset. Crops of size 513×513 sampled from the full image are used during training, and evaluation is done at full resolution.

The following tasks have been adopted from VTAB [56]: Caltech Cls., CIFAR-100 Cls., dtd Cls., Pets Cls., SUN Scene Cls., EuroSAT Cls., and CLEVR Count.

B. End Task Networks

The architecture and the loss functions used for each end task have been shown in Table 1. Figures 8–12 show the details of each network. The orange box in the figures shows the frozen encoder. The output dimensions for each block are also shown. The variables h and w represent the height and width of the input image, respectively, while n represents the batch size and s represents the sequence length.

C. Training Details

In this work encoders and end task networks are trained separately. Below we describe the training procedure for each.

We train the encoders by MoCov2 [9] and SwAV [5] algorithms. For the rest of the training algorithms, we use the publicly released weights for the trained models. We train every model using code publicly released by the authors and the same hyperparameters as the original implementation.

We train the end task networks by freezing the encoders and training just the end task network layers. For each task, we perform a grid search of 4 sets of optimizers and learning rates using the encoder trained with SwAV on ImageNet for 200 epochs. We then select the best performing set of hyperparameters and use them for all other runs. We also use the grid search training runs to determine the number of epochs necessary for each task to converge. In Table 2 we report the specific hyperparameters used for each end task.

D. Correlation Analysis of the End Tasks

To better understand the relationships between the end tasks chosen for this paper, we analyze the correlation between their performances using different encoders. Specifically, for every task A and every task B we compute the correlation between the performance of task A and B of all of the encoders we analyze. This shows whether good performance on one task is indicative of good performance on another.

Figures 13 and 14 show the Pearson and Spearman (rank) correlations between the end task performance of the encoders. One clear trend is that we see pockets of strong correlation within each task category. Sometimes they are well defined (Semantic Image-level or Structural Pixelwise tasks represented by red and yellow boxes in Figure 14) and sometimes they are more subtle (Semantic Pixelwise represented by the green box in Figure 14). Another trend that these figures show is that ImageNet classification performance is not a good universal metric for encoder performance (especially for pixelwise output tasks, where there is a low correlation).

E. CKA Analysis Details

Centered Kernel Alignment [27] is a method of quantifying the similarity of representations between images as they are processed through an encoder. For this study we compare how the relationship between the representations of two images change across the different blocks of the ResNet encoder. We select a balanced subset of 10,000 images from the ImageNet dataset to measure the similarity of representations, and downscale the images to 112×112 before processing them through the encoder. We then compute the CKA between the representations of every pair of images in our subset for every block of the ResNet encoder (this similarity metric has a range of 0 to 1). We find that all encoders trained with the MoCov2 algorithm have an average increase of 0.18 of the average correlation between the layers versus the encoders trained with the SwAV algorithm. This indicates that the MoCov2 encoders retain more spatial information about the images in the later layers and offers a potential hypothesis as to why MoCov2 encoders tend to outperform SwAV encoders at pixelwise output tasks.

It is important to note that this analysis was performed using only a subsample of ImageNet data. ImageNet was chosen for this analysis as it is amongst the most diverse datasets utilized in this paper, but it makes this analysis far from entirely comprehensive. The reason for running this analysis on just this subsample was computational complexity, as evaluating the CKA on all the data available to us is computationally impractical.

F. ANOVA Tests

For this test, we consider encoders trained with the MoCov2 and SwAV algorithms on subsets of ImageNet (as discussed in the main text). We examine the relationship between encoders trained on class unbalanced versions of ImageNet and their balanced counterparts with an equivalent number of samples. We use the end task results of the following encoders in our analysis: SwAV Half ImageNet

200, SwAV Linear Unbalanced ImageNet 200, SwAV Quarter ImageNet 200, SwAV Log Unbalanced ImageNet 200, MoCov2 Half ImageNet 200, MoCov2 Linear Unbalanced ImageNet 200, MoCov2 Quarter ImageNet 200, MoCov2 Log Unbalanced ImageNet 200.

Our analysis found evidence that an encoder trained on a Log Unbalanced subset of ImageNet outperforms an encoder trained on a balanced subset of ImageNet with an equivalent number of samples. To further validate this conclusion we trained 2 additional encoders using SwAV on 2 different logarithmically unbalanced subsets of ImageNet and included them in the following test.

We fit an ANOVA model to all of the results we obtain, treating the task, training algorithm, dataset balance, dataset size and number of training steps as variables. We find that (unsurprisingly) the task, dataset size and number of training steps are statistically significant indicators of end task performance. We also find that the algorithm used to train the encoder (MoCov2 vs SwAV) is a statistically significant indicator of end task performance, with SwAV models performing better (this does not contradict our claim that SwAV is not universally better than MoCov2, as we simply have more tasks that SwAV is good at in our test battery). Finally, we do not find any statistically significant evidence that an encoder trained with the balanced ImageNet is better than the encoders trained on the discussed unbalanced variations. We do however find evidence that an encoder trained on a Log unbalanced subset of ImageNet tends to perform better than one trained on a balanced subset. Perhaps the (comparatively) larger number of samples of the same few categories is a good match for the contrastive learning algorithm, but further experiments are needed to determine the exact cause and extent of this phenomenon.

G. Variance of the Results

The main source of variance in our results is the self-supervised training of the encoder. Since each encoder requires over 500 GPU hours to be trained for 200 epochs with the MoCov2 training algorithm, and over 1000 GPU hours to be trained for 200 epochs with the SwAV training algorithm, it is impractical for us to test multiple training runs of every encoder configuration that we study in this work.

To provide some context regarding the magnitude of variations across runs, we train three encoders using SwAV on ImageNet for 200 epoch with different random seeds. All training parameters are exactly the same as those used by the SwAV authors to obtain their SwAV 200 model.

Our results show that, on average, the variation in the performance of the end tasks is less than 0.85% (relative difference with the average performance), which can be negligible.

H. Encoders

Table 3 provides a complete list of all 30 encoders that are used for our analysis.

Task	Category	End Task Network	Loss	Success Metric
• ImageNet Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• ImageNet v2 Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• Pets Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• CalTech Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• CIFAR-100 Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• SUN Scene Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• Eurosat Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• dtd Cls.	Semantic Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• Kinetics Action Pred.	Semantic Image-level	Multi Input Fusion Classifier	Cross Entropy	Top-1 Accuracy
• CLEVR Count	Structural Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• THOR Num. Steps	Structural Image-level	Single Layer Classifier	Cross Entropy	Top-1 Accuracy
• THOR Egomotion	Structural Image-level	Multi Input Fusion Classifier	Cross Entropy	Top-1 Accuracy
• nuScenes Egomotion	Structural Image-level	Multi Input Fusion Classifier	Cross Entropy	Top-1 Accuracy
• Cityscapes Seg.	Semantic Pixelwise	DeepLabv3+	Pixelwise Cross Entropy	mIOU
• Pets Instance Seg.	Semantic Pixelwise	DeepLabv3+	Pixelwise Cross Entropy	mIOU
• EgoHands Seg.	Semantic Pixelwise	DeepLabv3+	Pixelwise Cross Entropy	mIOU
• THOR Depth	Structural Pixelwise	U-Net	L1 Error	Negative L1 Error
• Taskonomy Depth	Structural Pixelwise	U-Net	L1 Error	Negative L1 Error
• NYU Depth	Structural Pixelwise	U-Net	L1 Error	Negative L1 Error
• NYU Walkable	Structural Pixelwise	DeepLabv3+	Pixelwise Cross Entropy	mIOU
• KITTI Opt. Flow	Structural Pixelwise	Siamese U-Net.	L1 Error	1 - 'all'

Table 1. The network architecture, the loss and the success metric for each end task.

Task	Train Set Size	Number of Train Epochs	Optimizer	Learning Rate
• ImageNet Cls.	1,281,167	100	Adam	0.0003
• Pets Cls.	3,680	500	Adam	0.0003
• CalTech Cls.	3,060	5000	Adam	0.0003
• CIFAR-100 Cls.	50,000	100	Adam	0.0003
• SUN Scene Cls.	87,003	250	Adam	0.0003
• Eurosat Cls.	21,600	200	Adam	0.0003
• dtd Cls.	3,760	100	Adam	0.0003
• Kinetics Action Pred.	50,000	100	Adam	0.0003
• CLEVR Count	70,000	100	Adam	0.0003
• THOR Num. Steps	60,000	100	Adam	0.0003
• THOR Egomotion	60,000	100	Adam	0.0003
• nuScenes Egomotion	28,000	100	Adam	0.0003
• Cityscapes Seg.	3,475	100	Adam	0.0003
• Pets Instance Seg.	3,680	100	Adam	0.0003
• EgoHands Seg.	4,800	25	Adam	0.0003
• THOR Depth	60,000	50	Adam	0.0003
• Taskonomy Depth	39,995	50	Adam	0.0003
• NYU Depth	1,159	250	Adam	0.0003
• NYU Walkable	1,159	100	Adam	0.0003
• KITTI Opt. Flow	200	250	Adam	0.0003

Table 2. Training details for each end task.

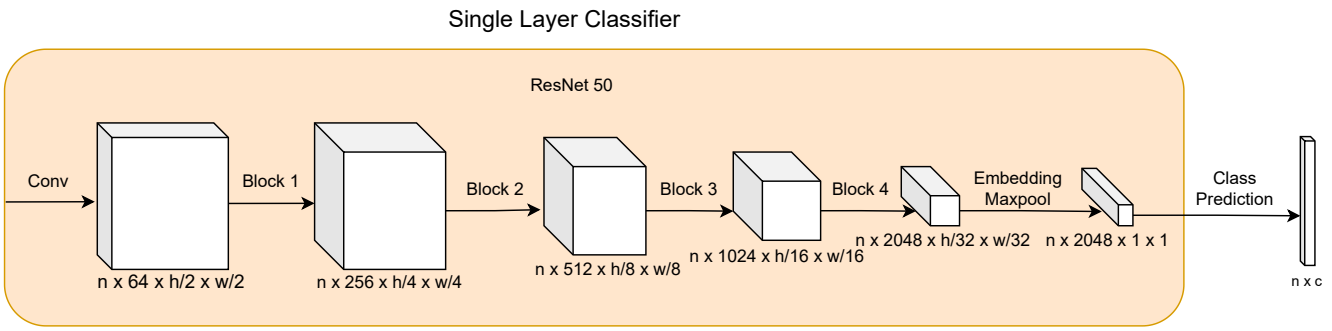

Figure 8. The Single Layer Classifier end task architecture. The orange box shows the frozen encoder.

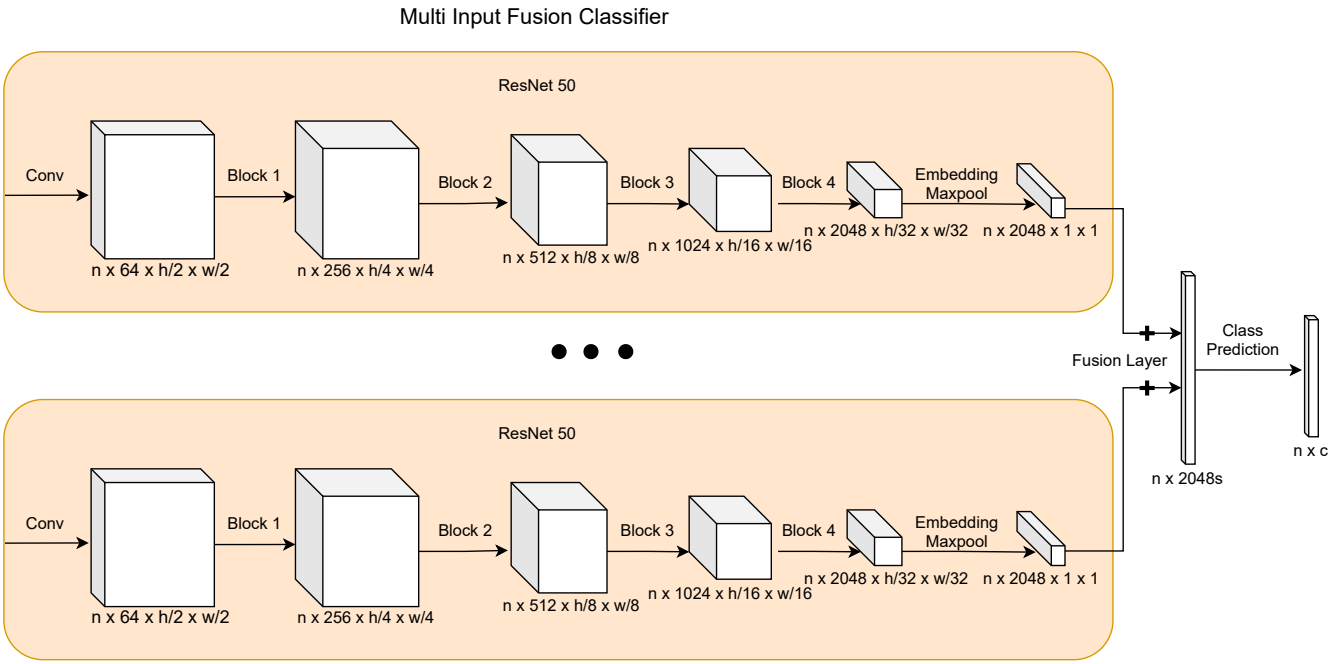

Figure 9. The Multi Input Fusion Classifier end task architecture. The orange box shows the frozen encoder.

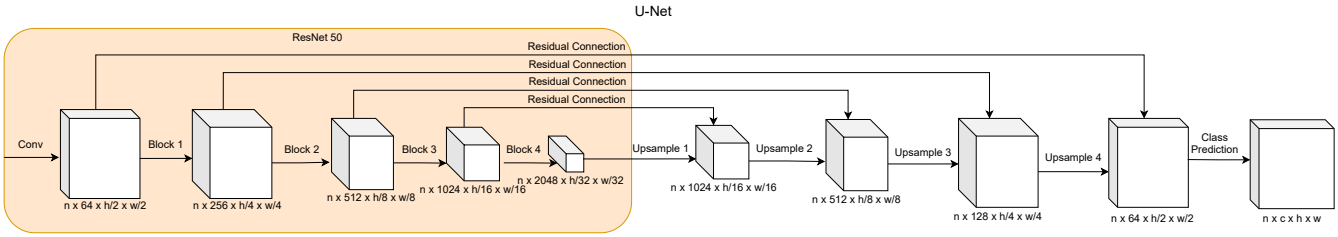

Figure 10. The U-Net end task architecture. The orange box shows the frozen encoder.

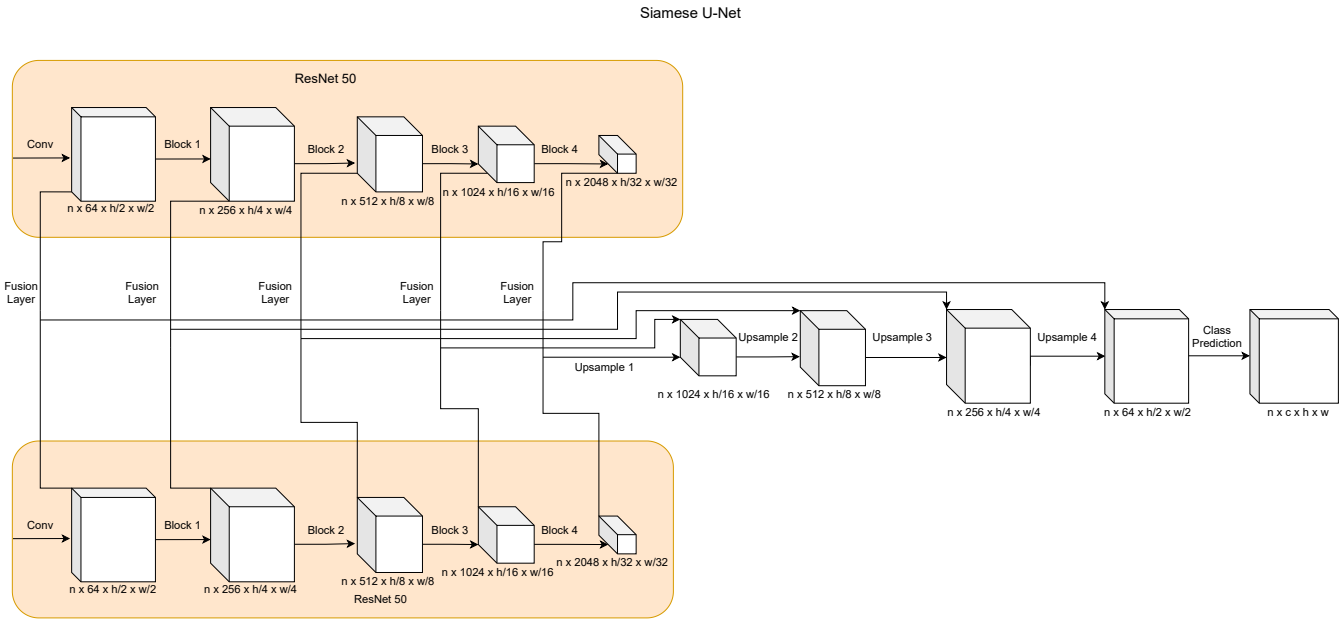

Figure 11. The Siamese U-Net end task architecture. The orange box shows the frozen encoder.

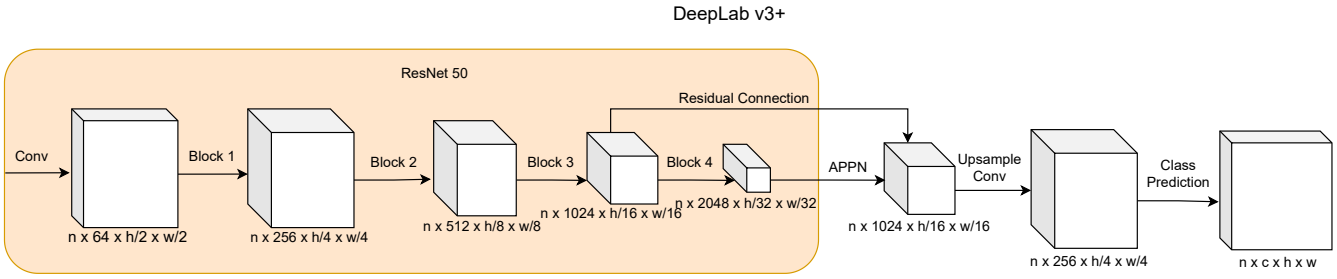

Figure 12. The DeepLabV3+ end task architecture. The orange box shows the frozen encoder.

Encoder Name	Method	Dataset	Dataset Size	Number of Epochs	Trained by us
SwAV ImageNet 800	SwAV	ImageNet	1.3M	800	No
SwAV ImageNet 200	SwAV	ImageNet	1.3M	200	No
SwAV ImageNet 100	SwAV	ImageNet	1.3M	100	Yes
SwAV ImageNet 50	SwAV	ImageNet	1.3M	50	Yes
SwAV Half ImageNet 200	SwAV	ImageNet-½	0.5M	200	Yes
SwAV Half ImageNet 100	SwAV	ImageNet-½	0.5M	100	Yes
SwAV Quarter ImageNet 200	SwAV	ImageNet-¼	0.25M	200	Yes
SwAV Linear Unbalanced ImageNet 200	SwAV	ImageNet-½-Lin	0.5M	200	Yes
SwAV Linear Unbalanced ImageNet 100	SwAV	ImageNet-½-Lin	0.5M	100	Yes
SwAV Log Unbalanced ImageNet 200	SwAV	ImageNet-¼-Log	0.25M	200	Yes
SwAV Places 200	SwAV	Places	1.3M	200	Yes
SwAV Kinetics 200	SwAV	Kinetics	1.3M	200	Yes
SwAV Taskonomy 200	SwAV	Taskonomy	1.3M	200	Yes
SwAV Combination 200	SwAV	Combination	1.3M	200	Yes
MoCov2 ImageNet 800	MoCov2	ImageNet	1.3M	800	No
MoCov2 ImageNet 200	MoCov2	ImageNet	1.3M	200	No
MoCov2 ImageNet 100	MoCov2	ImageNet	1.3M	100	Yes
MoCov2 ImageNet 50	MoCov2	ImageNet	1.3M	50	Yes
MoCov2 Half ImageNet 200	MoCov2	ImageNet-½	0.5M	200	Yes
MoCov2 Half ImageNet 100	MoCov2	ImageNet-½	0.5M	100	Yes
MoCov2 Quarter ImageNet 200	MoCov2	ImageNet-¼	0.25M	200	Yes
MoCov2 Linear Unbalanced ImageNet 200	MoCov2	ImageNet-½-Lin	0.5M	200	Yes
MoCov2 Linear Unbalanced ImageNet 100	MoCov2	ImageNet-½-Lin	0.5M	100	Yes
MoCov2 Log Unbalanced ImageNet 200	MoCov2	ImageNet-¼-Log	0.25M	200	Yes
MoCov2 Places 200	MoCov2	Places	1.3M	200	Yes
MoCov2 Kinetics 200	MoCov2	Kinetics	1.3M	200	Yes
MoCov2 Taskonomy 200	MoCov2	Taskonomy	1.3M	200	Yes
MoCov2 Combination 200	MoCov2	Combination	1.3M	200	Yes
MoCov1 ImageNet 200	MoCov1	ImageNet	1.3M	200	No
PIRL ImageNet 800	PIRL	ImageNet	1.3M	800	No

Table 3. The complete list of all 30 encoders used for the study.

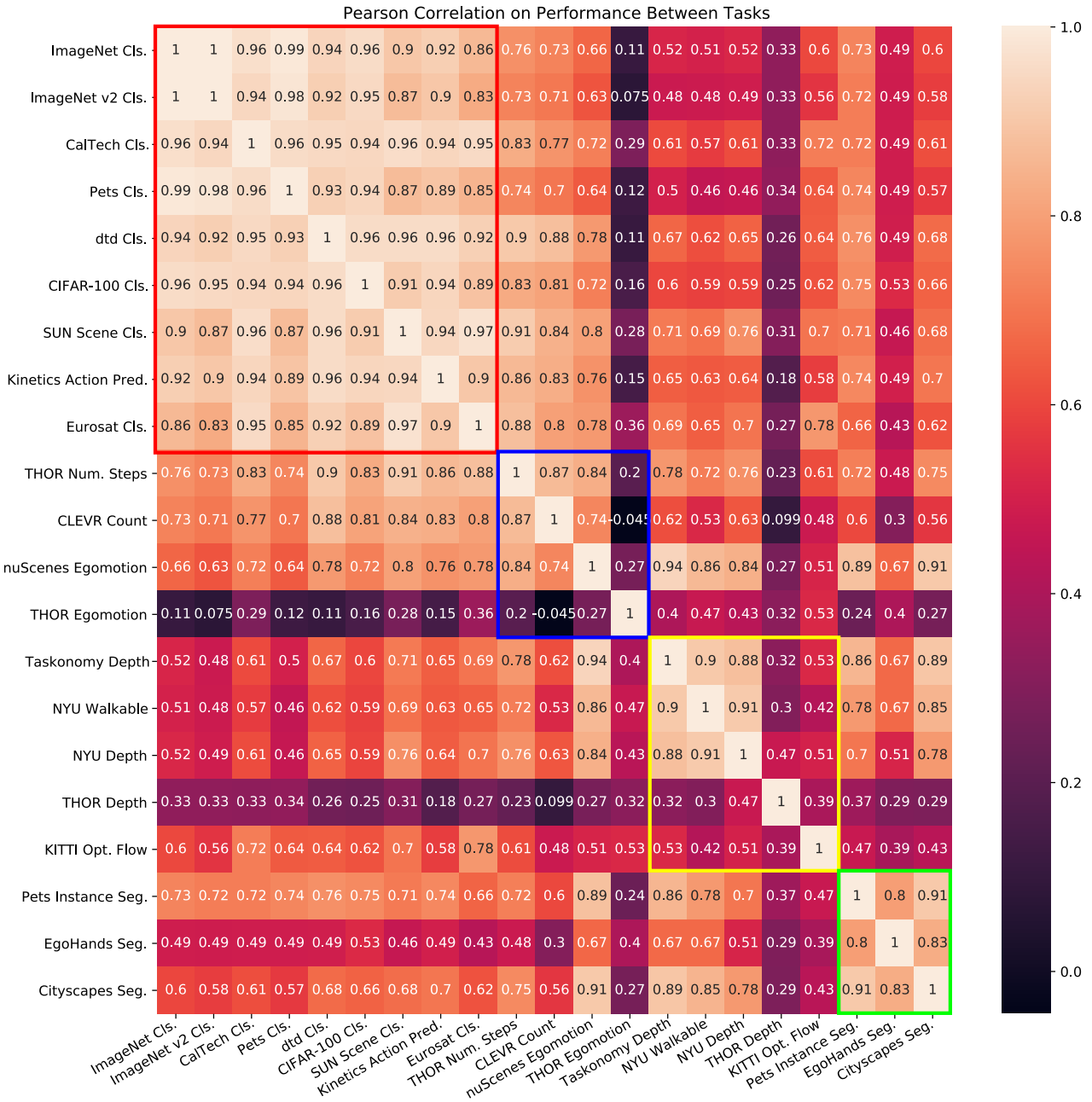

Figure 13. **Pearson Correlation** between the scores of all end tasks obtained using every encoder we study in this paper. Within category correlations for Semantic Image-level, Structural Image-level, Structural Pixelwise, and Semantic Pixelwise tasks are highlighted by red, blue, yellow and green boxes, respectively.

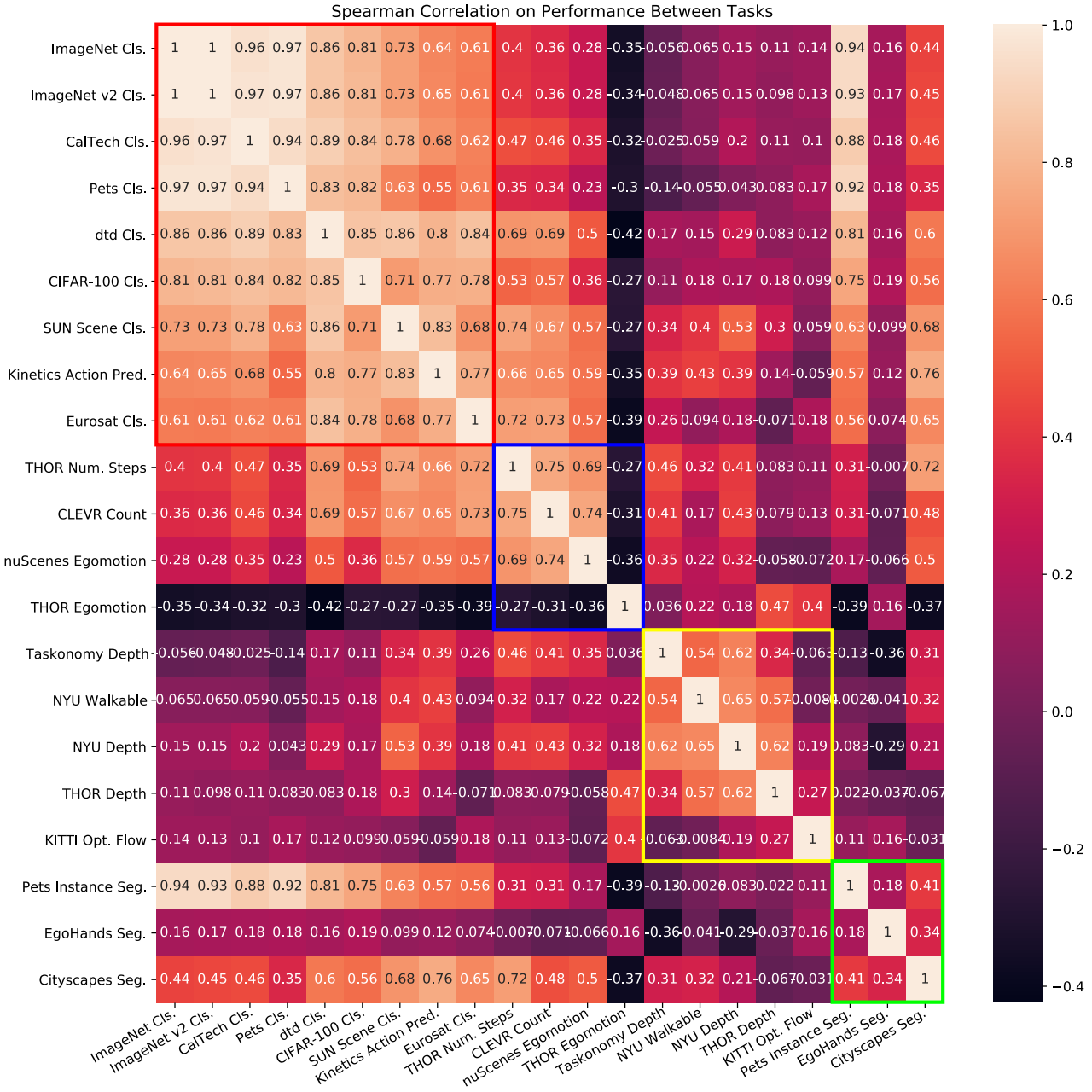

Figure 14. **Spearman Correlation** between the scores of all end tasks obtained using every encoder we study in this paper. Within category correlations for Semantic Image-level, Structural Image-level, Structural Pixelwise, and Semantic Pixelwise tasks are highlighted by red, blue, yellow and green boxes, respectively.