Transfer Learning or Self-supervised Learning? A Tale of Two Pretraining Paradigms

Xingyi Yang*, **Xuehai He***,**Yuxiao Liang, Yue Yang**
University of California San Diego
{x3yang,x5he,yul154}@eng.ucsd.edu,yuey9923@gmail.com

Shanghang Zhang
University of California Berkeley
shz@eecs.berkeley.edu

Pengtao Xie
University of California San Diego
pengtaoxie2008@gmail.com

Abstract

Pretraining has become a standard technique in computer vision and natural language processing, which usually helps to improve performance substantially. Previously, the most dominant pretraining method is transfer learning (TL), which uses labeled data to learn a good representation network. Recently, a new pretraining approach – self-supervised learning (SSL) – has demonstrated promising results on a wide range of applications. SSL does not require annotated labels. It is purely conducted on input data by solving auxiliary tasks defined on the input data examples. The current reported results show that in certain applications, SSL outperforms TL and the other way around in other applications. There has not been a clear understanding on what properties of data and tasks render one approach outperforms the other. Without an informed guideline, ML researchers have to try both methods to find out which one is better empirically. It is usually time-consuming to do so. In this work, we aim to address this problem. We perform a comprehensive comparative study between SSL and TL regarding which one works better under different properties of data and tasks, including domain difference between source and target tasks, the amount of pretraining data, class imbalance in source data, and usage of target data for additional pretraining, etc. The insights distilled from our comparative studies can help ML researchers decide which method to use based on the properties of their applications.

1 Introduction

Pretraining is a commonly used technique in deep learning to learn more effective representations for alleviating overfitting. Given a target task where the amount of training data is limited, training deep neural networks on this small-sized dataset has high risk of overfitting. To address this problem, one can pretrain the feature extraction layers in the network on large-sized external data from some source tasks, then finetune these layers on the target data. The abundance of source data enables the network to learn powerful representations that are robust to overfitting. And such representation power can be leveraged to assist in the learning of the target task with more resilience to overfitting.

Arguably, the most popular pretraining approach is transfer learning (TL) [1], which learns the weight parameters of a representation network by solving a supervised source task, i.e., correctly mapping input data examples to their labels (e.g., classes, segmentation masks, etc.). While successful, one concern of TL is that it uses labels in the source task to learn network weights, which may be biased

*Equal Contribution

Preprint. Under review.

to the source labels and generalize less well on the target task where the classes in the target labels are different from those in the source task.

This problem can be potentially alleviated by unsupervised pretraining, which trains the network weights purely based on input data examples without using any labels in the source task. Recently, self-supervised learning (SSL) [2, 3], as an unsupervised pretraining approach, has achieved promising success and outperforms transfer learning in a wide range of applications [2]. Similar to TL, SSL also solves predictive tasks. But the output labels in SSL are constructed from the input data, rather than annotated by human as in TL. The auxiliary predictive tasks in SSL could be predicting whether two augmented data examples originate from the same original data example [3], inpainting masked regions in images [4], etc. Since SSL does not leverage labels provided by human, it does not have the risk of being biased to labels in a source task. On the other hand, the potential pitfall of not using human-annotated labels is that the learned representations by SSL may not be as discriminative as those in TL. In sum, conceptually, SSL and TL both have advantages and disadvantages. It is difficult to judge whether one is better than the other. The existing empirical results show that in certain tasks, SSL outperforms TL [2]; in other tasks, TL performs better than SSL [5]. But these studies did not provide a clear guidance on what properties of data and tasks render one approach works better than the other. Consequently, ML researchers are not informed about how to select the right pretraining method and they have to try both empirically to find out, which is time-consuming and resource-intensive.

To address this issue, we perform comprehensive studies to compare SSL and TL, and investigate which method works better under different properties of data and tasks, including domain difference between source and target tasks, the amount of pretraining data, class imbalance in source tasks, and the usage of target data for additional pretraining. The studies are performed on 5 source tasks and 4 target tasks from various domains including daily-life objects, general scenes, natural creatures, and medical imaging. We summarize the insights distilled from the comparative studies to help ML researchers decide which pretraining method to use based on the specific properties of their applications. The major insights obtained from the studies include:

- When the domain difference between the source task and the target task is large, SSL outperforms TL. When domain difference is small, TL outperforms SSL.
- On the same source task, when the amount of pretraining data is small, SSL outperforms TL. When the amount is large, TL outperforms SSL.
- SSL is less sensitive to domain difference than TL: on the same target task, the performance of SSL pretrained on source tasks with varying domain difference with the target task is relatively stable while that of TL varies a lot.
- When domain difference is small, SSL is less sensitive to the amount of pretraining data than TL: given a target task and a source task that have small domain difference, the performance of TL on the target task changes a lot under varying amount of pretraining data in the source task while SSL is relatively stable.
- SSL is more robust to class imbalance compared with TL: pretrained on source datasets with varying ratios of class imbalance, the performance of SSL varies less than TL.
- For SSL, using the training examples in the target task as additional pretraining data works better than using source data only, which is not the case for TL.

2 Transfer Learning and Self-supervised Learning

Figure 1 illustrates transfer learning (TL) and self-supervised learning (SSL). Both TL and SSL consist of two phrases: pretraining on source task and finetuning on target task. The purpose of pretraining is to train the weight parameters of a representation

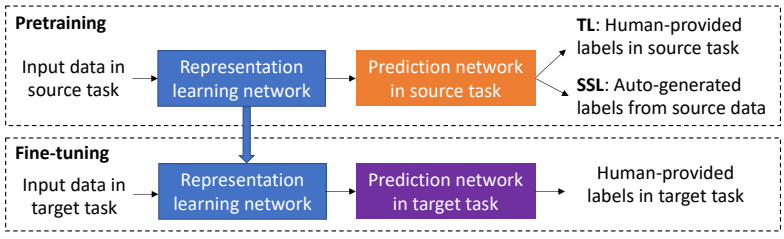

Figure 1: Workflow of transfer learning and self-supervised learning.

network into a good state. The pretrained weights are used to initialize the representation network in the target task (which has the same architecture as that in the source task). The initialized representation network in the target task is further trained on the target data. The difference between TL and SSL is that: TL performs pretraining using labeled source data in a supervised way while SSL performs pretraining using unlabeled source data in an unsupervised way.

Arguably, transfer learning [1] is the most widely used method for pretraining. Given data examples $\mathcal{D}_T$ and their class labels $\mathcal{L}_T$ in a target task T, a neural network $\mathcal{N}_T$ is designed to fulfill this task. $\mathcal{N}_T$ is composed of two parts: a feature extraction sub-network $\mathcal{F}_T$ that learns latent feature representations of $\mathcal{D}_T$ and a prediction sub-network $\mathcal{P}_T$ which is tailored to the predictive task in T. Typically, the majority of weight parameters of $\mathcal{N}_T$ lie in the feature extraction part $\mathcal{F}_T$. Meanwhile, one has access to data examples $\mathcal{D}_S$ and their class labels $\mathcal{L}_S$ in a source task S, where $\mathcal{D}_S$ is in general much larger than $\mathcal{D}_T$. How transfer learning (TL) works is: it creates a network $\mathcal{N}_S$ to solve the predictive task in S where the feature extraction sub-network $\mathcal{F}_S$ has the same architecture as $\mathcal{F}_T$, but the prediction sub-network $\mathcal{P}_S$ is tailored to task S. TL trains $\mathcal{N}_S$ on $\mathcal{D}_S$ and $\mathcal{L}_S$ until convergence. Then it uses the weights of $\mathcal{F}_S$ to initialize $\mathcal{F}_T$ and trains $\mathcal{N}_T$ on $\mathcal{D}_T$ and $\mathcal{L}_T$ until convergence. While TL has demonstrated pervasive effectiveness, there is a major concern about this approach. The weights in $\mathcal{F}_S$ are trained by fitting the labels $\mathcal{L}_S$. Therefore they are naturally biased to the classes in $\mathcal{L}_S$. When applied to predicting a new set of classes in $\mathcal{L}_T$, this bias may render the prediction less accurate.

Recently, in parallel to transfer learning, another pretraining paradigm – self-supervised learning (SSL) [2, 3] – has arisen much research interest. Different from TL which is typically conducted in a supervised manner (by labels in source tasks), SSL is undertaken mostly in an unsupervised way without using any human-provided labels. The basic idea of SSL is to construct some auxiliary tasks solely based on the input data itself without using any human-offered annotations and encourage the network to learn meaningful representations by performing the auxiliary tasks well. Given the source data $\mathcal{D}_S$, SSL designs an auxiliary task A and a network $\mathcal{N}_A$ to perform this task. $\mathcal{N}_A$ has a feature extraction sub-network $\mathcal{F}_A$ that has the same architecture as $\mathcal{F}_T$ and a prediction sub-network that is tailored to the task A. SSL trains $\mathcal{N}_A$ on $\mathcal{D}_S$ until convergence. Then it uses the weights of $\mathcal{F}_A$ to initialize $\mathcal{F}_T$, then trains $\mathcal{N}_T$ on $\mathcal{D}_T$ and $\mathcal{L}_T$ until convergence. Since SSL does not leverage labels in the source task, the learned feature extraction sub-network is not biased to these labels and is presumably more generalizable on $\mathcal{D}_T$. However, a potential downside is: $\mathcal{F}_A$ is learned without human supervision, which renders the network less discriminative.

2.1 Contrastive self-supervised Learning

The auxiliary tasks in SSL can be constructed using many different mechanisms, such as rotation prediction [6], image inpainting [7], automatic colorization [8], context prediction [9], etc. Recently, a contrastive mechanism [10] has gained increasing attention and demonstrated promising results in several studies [2, 3]. The basic idea of contrastive SSL is: generate augmented examples of original data examples, create a predictive task where the goal is to predict whether two augmented examples are from the same original data example, and learn the representation network by solving this task.

Different methods have been proposed to implement contrastive SSL. In SimCLR [3] designed for image data, given the input images, random data augmentation is applied to these images. If two augmented images are created from the same original image, they are labeled as being similar; otherwise, they are labeled as dissimilar. Then SimCLR learns a network to fit these similar/dissimilar binary labels. The network consists of two modules: a feature extraction module $f(\cdot)$ which extracts the latent representation $\mathbf{h} = f(\mathbf{x})$ of an image $\mathbf{x}$ and a multi-layer perceptron $g(\cdot)$ which takes $\mathbf{h}$ as input and generates another latent representation $\mathbf{z} = g(\mathbf{h})$ used for predicting whether two images are similar. Given a similar pair $(\mathbf{x}_i, \mathbf{x}_j)$ and a set of images $\{\mathbf{x}_k\}$ that are dissimilar from $\mathbf{x}_i$, a contrastive loss can be defined as: $-\log(\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)/(\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau) + \sum_k \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)))$, where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity between two vectors and τ is a temperature parameter. SimCLR learns the network weights by minimizing losses of this kind. After training, the feature extraction sub-network is used for downstream tasks and $g(\cdot)$ is discarded.

While SimCLR is easy to implement, it requires a large minibatch size to yield high performance, which is computationally prohibitive. MoCo [2] addresses this problem by using a queue that is independent of minibatch size. This queue contains a dynamic set of augmented data examples (called keys). In each iteration, the latest minibatch of examples are added into the queue; meanwhile, the oldest minibatch is removed from the queue. In this way, the queue is decoupled with minibatch

size. The keys are encoded using a momentum encoder. Given an augmented data example in the current minibatch (called query) and a key in the queue, they are considered as a positive pair if they originate from the same image, and a negative pair if otherwise. A similarity score is calculated between the encoding of the query and the encoding of each key. Contrastive losses are defined on the similarity scores.

3 Design of Studies

In this section, we study several factors that may affect the comparative advantages between SSL and TL, including domain difference between source task and target task, the amount of pretraining data, class imbalance in source task, and usage of target data for additional pretraining.

Study of domain difference We use five source domains and four target domains, from the following areas: (1) objects in daily life, (2) scenes in daily life; (3) nature such as plants and animals, and (4) medical images such as CTs and X-rays. We use five source datasets: ImageNet, SUN, iNaturalist, LUNA, and ChestX-ray8, from objects, scenes, nature, CT, and X-ray domains respectively. The details of these five datasets are:

- The ImageNet [11] dataset contains 1,281,167 training and 50,000 validation images from 1,000 classes. These classes cover common objects in daily life, such as monitor, school bus, sleeping bag, teapot, broccoli, etc.
- The SUN-397 [12] dataset contains 130,519 images from 397 scene categories, such as abbey, balcony, cafeteria, etc. Each category has more than 100 images.
- The iNaturalist 2018 dataset [13] contains 437,513 training and 24,426 validation images from 8,142 fine-grained categories about nature, including plants, insects, birds, etc. The numbers of images in different categories are highly imbalanced.
- The LUNA [14] dataset contains 888 CT scans about lung nodules.
- The ChestX-ray8 [15] dataset contains 112,120 frontal-view X-ray images from 30,805 patients from 15 disease classes, such as emphysema, pneumonia, fibrosis, infiltration, cardiomegaly, etc.

We use four target datasets: Caltech-256, Flowers-102, COVID-CT, and Pneumonia from objects, nature, CT, and X-ray domains respectively. The detailed information of these datasets is as follows.

- The Caltech-256 [16] dataset contains 256 categories of objects in daily life, such as instruments, furniture, animals, food, vehicles, etc. The number of training, validation, and testing images is 7710, 6425, and 6425 respectively.
- The Flowers-102 [17] dataset contains 102 types of flowers. The number of training, validation, and testing images is 6149, 1020, and 1020 respectively.
- The COVID-CT [18] dataset contains 349 COVID-19 CT images from 216 patients and 463 non-COVID-19 CTs. The number of training, validation, and testing images is 425, 118, and 203.
- The Pneumonia [19] dataset contains 5,863 X-ray images which are either positive for pneumonia or negative. The number of training, validation, and testing images are 5216, 16, and 624 respectively.

We perform a quantitative measurement of domain difference of two tasks on their visual contents and class labels. To measure visual distance between domains, we use the method in [20]. We sample 1000 images from a source task and label them as 0, and sample 1000 images from a target task and label them as 1. Then we split the 2000 images into a training set and a test set. We train a classifier on the training set to distinguish whether an image is from source or target, then measure the classification error ϵ on the test set. The visual difference is defined as $d = 2(1 - 2\epsilon)$. Intuitively, if source images and target images are easy to be told apart (i.e., ϵ is small), then their visual difference is large. In addition to visual difference, we also measure domain difference in the label space. Given $\{(c_i, f_i)\}_{i=1}^m$ in the source dataset, where c_i is a class name, f_i is the frequency of this class in this dataset, and m is the number of classes in the source dataset, and similarly $\{(c_j, f_j)\}_{j=1}^n$ in the target dataset, we calculate the class similarity of these two domains using this equation: $(\sum_{i=1}^m \sum_{j=1}^n f_i f_j \sigma(c_i, c_j))/(\sum_{i=1}^m f_i \sum_{j=1}^n f_j)$, where $\sigma(c_i, c_j)$ is the cosine similarity between the GloVe [21] embeddings of c_i and c_j. This equation basically measures the average similarity of the labels of a source image and a target image.

Figure 2 shows the visual difference and class similarity between source and target tasks. If the visual distance is less than 1.6, a source and a target are considered as in the same domain (denoted by "+"); otherwise, in different domains (denoted by "-"). For class similarity, "+" if the similarity is greater than 0.6, and "-" if otherwise. The third panel in Figure 2 shows the results, where the first and second symbol in each cell correspond to visual distance and class similarity respectively. A source and a target with "++" have high domain similarity, and those with "- -" have large domain differences. The results are in accordance with intuitive judgement. For example, ImageNet and Caltech-256 are in the same domain about daily objects; iNaturalist and Flowers-102 are in the same domain about nature.

	Caltech-256	Flowers-102	COVID-CT	Pneumonia
Visual distance				
ImageNet	1.35	1.73	1.69	1.89
SUN	1.09	1.67	1.71	1.84
iNaturalist	1.34	1.32	1.75	1.89
LUNA	1.81	1.90	1.30	1.90
ChestX-ray8	1.99	1.83	1.81	1.54
Class similarity				
ImageNet	0.07	0.06	0.02	0.01
SUN	0.07	0.04	0.04	0.02
iNaturalist	0.03	0.07	-0.11	-0.07
LUNA	0.05	0.01	0.62	0.57
ChestX-ray8	0.01	0.01	0.23	0.25
ImageNet	+ +	- +	- -	- -
SUN	+ +	- -	- -	- -
iNaturalist	+ -	+ +	- -	- -
LUNA	- -	- -	+ +	- +
ChestX-ray8	- -	- -	- +	+ +

Figure 2: Source-target domain difference.

Study of the amount of pretraining data To study how SSL and TL are affected by the amount of pretraining data, we perform the following controlled study: for each source dataset, we use 1%, 10%, and 100% of the dataset for pretraining.

Study of class imbalance in source tasks In a source task with class labels, the frequencies of classes are typically imbalanced. We are interested in studying how class imbalance affects the performance of SSL and TL. From a source dataset, we create imbalanced datasets with different imbalance ratios, pretrain SSL and TL on these datasets, and check how their performance varies with the imbalance ratio. Given a source dataset containing K classes, we rank these classes in ascending order of their frequencies. Let n_j denote the frequency of the j-th ranked class. The class imbalance ratio ρ is defined as n_K/n_1, which is the ratio between the number of training examples of the most and least frequent class. We create datasets with different imbalance ratios by tuning n_K and n_1. For the rest classes, we perform random sampling to ensure the frequencies of these classes form an arithmetic sequence: specifically, the frequency of class j is $(j-1) \times (n_K - n_1)/(K-1) + n_1$, for $j = 2, \cdots, K-1$.

Study of using target data for additional pretraining In existing approaches, pretraining is mostly conducted on source data. We are interested in investigating whether it is helpful to add the training examples in the target task as additional data for pretraining and how this will affect the comparative advantages of SSL and TL. We compare the following pretraining settings: (1) SSL on source data only, on target data only, and on the combined data of source and target; (2) TL on source data only, on target data only, and on the combined data of source and target. For SSL on target data only, we pretrain SSL on training images in the target task, then finetune using both input images and output labels in the target training set. For SSL on the combined data of source and target, we pretrain SSL on source images and target training images. For TL on target data only, it is the same as finetuning on the target task with random initialization of the network. For TL on the combined data, it is a multi-task learning problem which trains classification models for the source task and target task simultaneously.

4 Experiments

4.1 Experimental settings

All source tasks and target tasks are about image classification. We used ResNet-50 [22] for classification. The architecture of ResNet-50 is the same for different tasks, except the number of output units (which is the same as the number of categories in a task). The TL and SSL models

pretrained on the full ImageNet dataset were borrowed from [22] and [2]. The rest of the models were pretrained from scratch.

For TL on source tasks, we applied data augmentation including random resize with a ratio sampled from $[0.08, 1]$, random crop of 224×224, and the ImageNet AutoAugment policy [23]. The weights were optimized using SGD with an initial learning rate of 0.1 and a momentum of 0.9. We pretrained TL for 200 epochs on every source task with a minibatch size of 128. The learning rate was reduced by 0.1 every 50 epochs.

For SSL on source tasks, we used the data augmentation methods in [24] and used SGD as the optimizer with an initial learning rate of 0.015 and a momentum of 0.9. We pretrained SSL for 200 epochs and a batch size of 128. The learning rate was adjusted using cosine learning rate scheduling.

For finetuning on target tasks, we utilized the same data augmentation methods as in TL pretraining. We used Adam [25] as the optimizer with an initial learning rate of 0.001. The finetuning was performed for 200 epochs on Caltech-256 and Flower-101, 100 epochs on COVID-CT, and 30 epochs on Pneumonia. Cosine scheduling with a period of 6 was used to adjust the learning rate. The implementation was based on PyTorch and the experiments were conducted on 8 Tesla V100 GPUs.

4.2 Results on domain difference

We perform controlled studies to investigate how domain difference between source and target tasks affects the performance of SSL and TL. To rule out the influence of the amount of pretraining data, we make the number of pretraining examples in all source tasks the same. This number is chosen to be 90,000. For a source dataset whose size is larger than 90,000, we randomly sample 90,000 data examples from this dataset.

Table 1: Results on domain difference. Rows correspond to source datasets where SSL and TL are pretrained. Columns correspond to target datasets where pretrained models by SSL and TL are finetuned. The numbers are top-1 image classification accuracy achieved on the test data in target tasks. The number of pretraining data examples in different source datasets is the same.

	Caltech-256		Flowers-102		COVID-CT		Pneumonia	
	SSL	TL	SSL	TL	SSL	TL	SSL	TL
ImageNet	57.53	62.14	82.02	86.23	76.01	73.08	94.07	92.47
SUN	56.34	62.21	79.55	63.34	80.33	71.24	93.75	92.41
iNaturalist	56.59	47.53	82.19	86.47	79.59	74.48	94.87	93.59
LUNA	53.43	48.31	76.77	51.40	79.11	74.04	93.43	93.27
ChestX-ray8	53.50	47.31	72.09	55.40	78.63	72.20	93.27	95.19

Table 1 shows the results on domain difference. From this table, we make the following observations. On most source-target pairs with strong domain similarity ("++"), including (ImageNet, Caltech-256), (SUN, Caltech-256), (iNaturalist, Flowers-102), (ChestX-ray8, Pneumonia), TL achieves better accuracy than SSL. This shows that when the domain difference is small, TL is more effective than SSL. The possible reason is: when the domain difference is small, the classes in source labels have a lot of overlap with those in target labels. Due to this overlap, it is beneficial to transfer the semantic information in source labels to the target task. TL utilizes sources labels while SSL does not. Hence TL works better than SSL. The only exception is (LUNA, COVID-CT), where TL performs worse than SSL. Second, on source-target pairs where the domain difference is large ("- -"), including (ImageNet, COVID-CT), (ImageNet, Pneumonia), (SUN, Flowers-102), (SUN, COVID-CT), (SUN, Pneumonia), (iNaturalist, COVID-CT), (iNaturalist, Pneumonia), (LUNA, Caltech-256), (LUNA, Flowers-102), (ChestX-ray8, Caltech-256), (ChestX-ray8, Flowers-102), SSL achieves better accuracy than TL. This shows that when domain difference is large, SSL works better than TL. The possible reason is: when the domain difference is large, the classes in source labels are largely different from those in target labels. When the representations are learned by fitting source labels, they are biased to source classes and generalize less well on the target task which has a different set of classes. SSL avoids using source labels, hence is not prone to such a bias. Third, SSL is less sensitive to domain difference than TL. For example, on Caltech-256, the performance of SSL is relatively stable. The difference between the highest and lowest performance is about 4% (absolute). In contrast, for TL, the difference between the highest and lowest performance is about 15% (absolute). This is because

TL uses source labels for pretraining. When the domain difference varies substantially, the labels change substantially, which makes the learned representations vary a lot. Fourth, on the same target domain, the closer a source domain is to this target, the better the performance is, for both SSL and TL. For example, in terms of domain similarity with Caltech-256, we have the following order: ImageNet, SUN $>$ iNaturalist $>$ LUNA, ChestX-ray8. As can be seen, the accuracy corresponding to these source domains decreases in general. This is because pretraining on a closer source domain can make the learned representations more suitable for the target domain.

4.2.1 Results on the amount of pretraining data

We evaluate the effect of the amount of pretraining data by pretraining SSL and TL on different subsets of each source dataset. Given a source dataset, we create two subsets by randomly sampling 1% and 10% of data examples. The experimental results are shown in Table 2. From this table, we make the following observations. First, when the amount of training data is small, SSL outperforms TL. For example, pretrained on 1% ImageNet, SSL outperforms TL on all target tasks. One possible reason is: when the training data is small, TL has a risk of overfitting to the labels of the small dataset and generalizes less well on target tasks. In contrast, SSL is an unsupervised pretraining method, which does not have the risk of overfitting to labels of small-sized source data. Second, when the amount of pretraining data is large, TL outperforms SSL. For example, pretrained on 100% ImageNet data, TL achieves better performance than SSL on all target tasks. The possible reason is: when the dataset is large, TL is less likely to suffer overfitting. On the contrary, the diverse labels contained in the large dataset enables TL to learn discriminative representations. Third, when domain difference is small, SSL is less sensitive to data amount than TL. For example, when the pretraining ImageNet data increases from 1% to 100%, the relative improvement of SSL on Caltech-256 is about 30% whereas the relative improvement of TL is about 62%. Fourth, for both SSL and TL, increasing the amount of training data leads to better performance. This is not surprising since deep learning methods are data hungry. Fifth, when the pretraining data is small, the performance of SSL and TL may be worse than random initialization. This is because SSL and TL may be overfitted to the small-sized pretraining data and generalize less well on the target tasks.

Table 2: Results on the amount of pretraining data. Rows correspond to datasets for pretraining. Columns correspond to target datasets for finetuning.

			Caltech-256		Flowers-102		COVID-CT		Pneumonia	
		Number	SSL	TL	SSL	TL	SSL	TL	SSL	TL
1%	ImageNet	10,000	52.33	49.87	73.13	63.05	75.26	66.39	93.59	90.54
10%	ImageNet	100,000	57.78	69.09	81.27	87.09	78.85	72.86	94.55	93.43
100%	ImageNet	1,000,000	68.16	80.62	87.52	91.48	80.00	82.03	94.55	95.67
1%	SUN	1,088	26.73	31.74	38.84	55.87	55.11	67.65	87.50	86.21
10%	SUN	10,875	40.29	52.61	56.46	64.88	71.95	68.18	89.74	86.54
100%	SUN	108,754	53.92	60.46	78.12	83.75	79.37	73.61	93.59	91.19
1%	iNaturalist	4,375	52.85	38.97	75.99	64.50	74.56	69.54	90.97	89.10
10%	iNaturalist	40,710	54.18	45.83	76.38	78.46	79.37	71.68	91.19	92.47
100%	iNaturalist	437,513	60.48	60.23	86.33	88.60	80.77	74.30	95.35	93.75
1%	LUNA	895	43.94	46.38	55.50	52.85	65.90	66.17	87.98	86.70
10%	LUNA	8,945	47.25	44.79	64.01	52.90	72.86	71.90	88.62	87.34
100%	LUNA	89,455	53.43	48.31	76.77	54.44	79.11	74.04	93.43	93.27
1%	ChestX-ray8	900	51.29	31.93	45.88	50.86	57.82	64.99	91.99	90.87
10%	ChestX-ray8	9,000	50.64	50.36	56.80	54.77	70.45	67.79	92.79	93.27
100%	ChestX-ray8	89,953	53.50	47.31	72.09	57.81	78.63	72.20	93.27	95.19
Random initialization			50.75		59.26		65.51		88.62	

4.3 Results on class imbalance in source tasks

From ImageNet, we create 3 imbalanced datasets, with imbalance ratio $\rho = 1, 5, 25$, respectively. For $\rho = 1$, we set $n_1 = n_N = 130$; for $\rho = 5$, $n_1 =$

	Caltech-256		Flowers-102		COVID-CT		Pneumonia	
Ratio	SSL	TL	SSL	TL	SSL	TL	SSL	TL
1	59.34	66.62	84.45	88.06	80.29	79.33	95.35	94.43
5	58.72	65.77	83.12	86.72	79.81	74.04	94.87	93.91
25	58.46	64.88	82.68	86.10	75.96	72.86	94.71	93.75

Figure 3: Comparison of results under different imbalance ratio

43, $n_N = 215$; for $\rho = 25$, $n_1 = 10$, $n_N = 250$. Such settings ensure the total number of data examples in each dataset to be approximately the same ($\approx 13,000$), so that the factor of pretraining data size can be ruled out.

Figure 3 shows the results under different class imbalance ratios. From this table, we make the following observations. First, as the imbalance ratio increases, the performance of both SSL and TL decreases. The reason is that class imbalance renders the learned representations biased to images from frequent classes in the source task. The more imbalanced the classes are, the larger the bias is. A larger bias leads to worse generalization to target tasks. Second, SSL is more robust to class imbalance than TL. For example, on the COVID-CT task, the gap between the best and worst performance of SSL under different imbalance ratios is about 4.3% (absolute) while the gap for TL is about 6.5% (absolute). Similar results are observed on other target tasks as well. The reason is: TL is pretrained using class labels and therefore is more sensitive to the distribution of labels, including imbalance ratio. In contrast, the pretraining of SSL is label-free, hence is less affected by label distribution.

4.4 Results on using target data for pretraining

In this study, we compare the following pretraining settings: on source data only, on target data only, and on the combined data of source and target. The source dataset is SUN and target data refers to the data examples in the training set of a target task. Figure 4 shows the results. From this table, we make the following observations. First, for SSL, pretraining on the combined data performs better than on source-only and on target-only. The possible reason is: combined data has more training examples, which helps to learn better representations. Second, TL on the combined data performs better than on target-only. The reason is that performing TL on the combined data effectively leverages the source task to help with the learning of the target task via multi-task learning. Third, TL on the combined data performs worse than on source-only. The reason is: TL on source-only first pretrains on source data, then finetunes on target data. The focus is finetuning on target tasks, where the network is learned to best fit the target data. For TL on the combined data, training on source task and target task is performed simultaneously. The network aims to perform both tasks well instead of focusing on the target task. The reduced focus incurs worse performance on target tasks.

Source	Target	Source only		Target only		Combined	
		SSL	TL	SSL	TL	SSL	TL
SUN	Caltech-256	53.92	60.46	56.59	50.75	57.46	59.57
SUN	Flowers-102	78.12	83.75	57.59	59.26	79.69	78.00
SUN	COVID-CT	79.37	73.61	61.72	65.51	80.77	69.74
SUN	Pneumonia	93.75	91.19	93.43	88.62	94.39	90.04

Figure 4: Results on the usage of target data for additional pretraining.

5 Related Works

Several works conducted preliminary comparisons of SSL and TL. In [2], supervised transfer learning (TL) is compared with an SSL approach – MoCo. It is shown that in certain tasks SSL outperforms TL, but the other way around in other tasks. It is not studied what factors incur such inconsistent results, which renders ML researchers have to try both approaches to find out which one works better empirically without an informed guidance. We aim to bridge this gap in our work, by systematically investigating under what situations SSL performs better than TL and vice versa. Zhai et al. [26] compared a number of representation learning methods, including supervised TL and SSL. They compared SSL and TL in terms of a single factor: image style. In our work, we perform a comprehensive comparison of SSL and TL on a number of factors. Newell and Deng [27] studied what factors affect the performance of SSL. The studies are performed on SSL only, rather than on the comparison with TL. Resnick et al. [28] performed a detailed study of SSL and presented several findings such as linear probes is insufficient to evaluate the representations learned by SSL, the bias towards success stemming from the architecture is high, etc. The study is mostly focused on SSL, instead of comparing with TL. Our work differs from these two in that our goal is to study what factors affect the comparative advantages between TL and SSL.

6 Conclusions

In this work, we make a comprehensive comparative study about SSL and TL, regarding which method works better under different properties of data and tasks. Specifically, we study how the domain difference between source and target tasks, the amount of pretraining data, class imbalance in

source tasks, and the usage of target data for additional pretraining affect the comparative advantages of SSL and TL. On 5 source tasks and 4 target tasks from various domains, on varying amount of pretraining data, we conduct experiments and distill a set of insights. These insights can potentially help ML researchers to decide which pretraining method to use based on the properties of their applications and foster the development of new SSL and TL methods.

References

[1] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[2] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[4] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[5] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[6] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[7] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[8] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[9] T Nathan Mundhenk, Daniel Ho, and Barry Y Chen. Improvements to context based self-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9339–9348, 2018.

[10] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[11] 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.

[12] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[13] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[14] Luna. `https://luna16.grand-challenge.org/data/`.

[15] 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.

[16] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[17] 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.

[18] Jinyu Zhao, Yichen Zhang, Xuehai He, and Pengtao Xie. Covid-ct-dataset: a ct scan dataset about covid-19. *arXiv preprint arXiv:2003.13865*, 2020.

[19] 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.

[20] 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.

[21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] 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.

[23] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[24] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[26] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. The visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[27] Alejandro Newell and Jia Deng. How useful is self-supervised pretraining for visual tasks? *arXiv preprint arXiv:2003.14323*, 2020.

[28] Cinjon Resnick, Zeping Zhan, and Joan Bruna. Probing the state of the art: A critical look at visual representation evaluation. *arXiv preprint arXiv:1912.00215*, 2019.