The Evaluation of Unsupervised Representation Learning Methods

Rodrigo Morales Flores

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2021

Abstract

We develop a broad task-based benchmark to evaluate unsupervised representation learning methods. Our benchmark allows a thorough evaluation of a method's capacity along different tasks categories ranging from high-level semantic classification to low-level visual features prediction. We enforce a single evaluation pipeline across the different tasks of the benchmark where we hold fixed the constituent elements, such as the image distribution (ImageNet [48]), encoder architecture (ResNet-50 [28]), downstream model (logistic regression), and optimization algorithm (L-BFGS [39]). In particular, to guide the choice of optimizer, we perform a study to examine the optimization performed by Stochastc Gradient Descent (SGD) [47] compared to L-BFGS in linear evaluation of representations. In this context, we find that depending on the degree of linear separability of the data, the solution found by SGD varies: in a more (linearly) non-separable regime, it can reproduce the results of a robust optimizer (L-BFGS), whereas in a more (linearly) separable case it performs implicit regularized optimization. Moreover, this is highly dependent on the values of the initial learning rate and mini-batch size. Finally, we showcase our benchmark by evaluating thirteen state-of-the-art unsupervised representation learning methods revolving around the contrastive learning framework. Our results suggest three main conclusions: the use of protoypes (clusters centers) for the contrastive task enhances the representation's capacity in high-level semantic tasks, but hinders it in lower-level tasks; the use of a non-linear projection head in combination with strong data augmentation can improve the performance in *all* the tasks categories; while these elements allow to learn representations that are robust to noise, they do not appear to improve their data efficiency, which remains an important area of opportunity for future research.

Acknowledgements

I would like to thank my supervisor, Amos Storkey, for introducing me to this topic, for the guidance throughout the project, and, especially, for providing insight and clarity when the abundance of potential approaches to inspect the results felt overwhelming. I would also like to thank Luke Darlow, whose advice was also crucial for the completion of the project. In particular, the foundational idea behind the project and a draft of the benchmark's broad structure was fully theirs.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Rodrigo Morales Flores*)

Table of Contents

Chapter 1

Introduction

Representation learning is concerned with finding a representation of the data which is useful to solve a task of interest. Deep learning has achieved outstanding results in this domain by automatically learning hierarchical representations from the data that allow to solve complex perceptual tasks [5]. Until recently, most of this success for visual representation learning has occurred in the supervised domain, where labels providing the target values for the task of interest are known and can be used by the method to learn a representation. However, in many settings the acquisition of large amounts of labelled data may prove costly or even unfeasible. Motivated by this issue, the development of methods for learning visual representations without access to such labels has received increased attention [32].

A crucial missing component to foster progress in this field is an adequate benchmark to evaluate the representations learned by the methods. The main difficulty in the development of such benchmark is the choice of evaluation task. While in the supervised domain the objective task (associated with the given labels) is clear, this is not the case in the unsupervised setting where learning more general-purpose features is desirable. Previous work ([22], [33], [2], [60], [17]) have borrowed common evaluation tasks from the supervised domain for evaluating unsupervised methods. While this line of work has proved useful to track progress in this particular kind of tasks, we argue that a more comprehensive tasks set specifically designed to evaluate the content of diverse features in the representations is necessary to encompass the full objective of unsupervised representation learning. The main contribution of this work is the development of this type of benchmark.

We develop a representation learning benchmark with 29 tasks grouped into 7 different categories. Its distinctive feature is the broadness of tasks, specifically designed

to assess the presence of diverse features in the representation learned by a method. In particular, we do not only consider the commonly used semantic classification tasks, but also robustness, data efficiency, lower-level visual tasks and tasks with random target labels. These allow to have a more comprehensive evaluation of a representation's capacity.

Additionally, we enforce a single linear evaluation protocol for all the tasks in the benchmark which addresses common inconsistencies in the evaluation performed by different methods. In particular, we hold fixed constituent elements that can affect the results of the evaluation, like the image distribution [17], encoder architecture [33], [22], and optimizer. The optimization algorithm used in linear evaluation is an understudied design choice. To address this, we perform an optimization study of Stochastic Gradient Descent (SGD) [47] for linear evaluation of representations. Our results indicate that the convergence properties of SGD depend crucially on the degree of linear separability of the data and the hyperparameters values. In particular, for linearly non-separable data and specific values of the initial learning rate and mini-batch size, SGD can reproduce the solution of L-BFGS, a quasi-Newton algorithm with strong convergence guarantees in convex problems [39]; however, when the data is highly linearly separable, for some hyperparameter values, SGD performs implicit regularization.

Finally, we showcase our benchmark by evaluating thirteen state-of-the art methods for unsupervised representation learning revolving around the contrastive learning framework. Our results suggest that contrasting protoypes (cluster centers) and instances can enhance the performance of the representations in high-level tasks, but hinders it in low-level ones. This is not the case for the use of a non-linear projection head and strong data augmentation which can improve the performance in all tasks categories. Finally, we note that while these enhancements have allowed unsupervised methods to learn representations which are robust to noise in the visual data, there is an important room for improvement in terms of data efficiency. Our hope is that our work enables further experimentation in these topics to foster the progress of the field.

The thesis is structured as follows: Chapter 2 provides the technical background for the rest of the chapters and reviews related work, Chapter 3, describes the optimization study of SGD for linear evaluation of representations, Chapter 4 explains the design of the benchmark we developed, Chapter 5 presents the results of evaluating the state-of-the-art unsupervised representation learning methods with our benchmark, and Chapter 6 concludes.

Chapter 2

Background and related work

In this chapter we provide the technical background for the rest of the thesis. In particular, we provide an overview of the optimization algorithms used for the experiments of Chapter 3; we review the fundamental elements of some state-of-the-art unsupervised representation learning methods, with special emphasis on the ones evaluated in Chapter 5; finally, we outline the previous efforts done to evaluate representations, and the difficulties which our benchmark, described in Chapter 4, addresses.

2.1 Optimization in Machine Learning

Optimization plays a central role in Machine Learning. Numerous learning problems can be formulated as optimization problems. More formally, given a training dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ consisting of N training examples composed of an input vector $\mathbf{x}_i$ and a target value y_i, the goal of a machine learning algorithm is to leverage $\mathcal{D}$ to find a predictive function h_θ, parametrised by θ, which can effectively predict a target value, y_*, given a test input, $\mathbf{x}_*$, not contained in $\mathcal{D}$.

A common approach to find effective parameters for this task is to define a loss function $\ell(\hat{y}, y)$ which measures the loss incurred by predicting $\hat{y} = h_\theta(\mathbf{x})$ for an input $\mathbf{x}$ when its target value is y. Then, to find adequate parameters θ one can minimise this loss averaged over all the training examples. This is, one can minimise the function: $J(\theta) = \frac{1}{N}\sum_{i=1}^N \ell(h_\theta(\mathbf{x}_i), y_i) = \sum_{i=1}^N \ell_i$, with respect to θ.

Gradient-based algorithms [7] are a common choice to solve such optimization problem, provided that ℓ is chosen to be differentiable. These methods iteratively update the parameters in a direction that reduces the value of $J(\theta)$ using the slope information provided by the gradient.

One way to categorise gradient-based method is according to the order of the derivatives used to perform the parameters update [54]. First-order methods utilize exclusively information of the first derivatives, while higher-order methods also involve the use of higher-order derivatives. Higher-order methods are typically second-order methods utilizing the Hessian matrix of second derivatives (e.g. Newton's method) or an approximation of it (e.g. quasi-Newton methods like the BFGS algorithm [8], [18], [20], [49]). A different way of categorizing gradient-based methods is according to the amount of data used to compute the update. Batch methods utilize the full training dataset to compute the gradient of the objective function with respect to the parameters, whereas stochastic methods use only a sub-sample of the data to approximate the gradient, and use this approximation to update the parameters.

In this work, we focus on Stochastic Gradient Descent (SGD) [47], a stochastic, first-order method, which is the cornerstone of optimization in deep learning [7], [54]. SGD approximates the gradient $\Delta_\theta^J(\theta)$ using only the gradient of the loss on a (random) example $\Delta_\theta^{\ell_i}(\theta)$. Using this direction to update the parameters, on expectation, decreases the value of the objective function $J(\theta)$. In this way, SGD can update the parameters without the need to make a full pass on the training set. Crucially, this property allows SGD to scale to large datsets, which is one of the main reasons of its widespread use in deep learning [7]. However, using only one sample to approximate the gradient can induce a large variance in the direction of the updates [54]. In practice, it is more common to use a batch (also called mini-batch) of B examples for the approximation. In the limit where $B = N$, mini-batch SGD becomes (full-batch) Gradient Descent (GD).

SGD, and first-order methods in general, do not leverage information of the curvature of the loss surface; this omission can lead to ineffective search directions [7]. Moreover, their updates are not scale invariant which means that scaling all the parameters by a constant factor will scale the size of the update [7]. The learning rate effectively re-scales the steps to an adequate size making it a critical parameter for the convergence of the algorithm. In particular, an adequate schedule for its trajectory along the optimization process is crucial for convergence [54].

L-BFGS [39] is a method which approximates the Hessian matrix (a so-called quasi-Newton method) by using the values of the gradients and parameters from past iterations. The approximations of the Hessian often result in dense matrices which impose a large storage overhead limiting the scale of the methods. This issue is addressed by L-BFGS with a limited memory scheme that avoids the computation and storage of

explicit matrices sequences, but only requires the recent history of gradients and parameters values. In convex problems, it offers strong convergence guarantees [39] [7]. This obervation is crucial for the experimental design of Chapter 3. L-BFGS is a batch method and thus does not scale well to very large datasets, although some stochastic variants have been recently proposed [42], [6].

A distinctive feature of learning problems is that the ultimate objective is not to minimise $J(\theta)$, but rather to correctly predict the target value of unseen inputs not contained in the training set. In some instances, the parameters minimising $J(\theta)$ can lead to poor predictive performance on data not contained in the training set (a problem known as overfitting). Thus, the lower convergence guarantees of SGD compared to higher-order methods may not necessarily be a disadvantage in machine learning problems. In fact, previous work has observed implicit regularization performed by SGD in certain settings [50], [4], [51] [31] [26]. The most relevant for our purpose is the theoretical work from [53], which finds that in the logistic regression problem without a bias vector and with linearly separable data, for a small enough learning rate, (full-batch) GD provides an implicit regularization equivalent to L2 weight decay. Our empirical work from Chapter 3 extends this result to SGD and a non-homogeneous problem (with a bias vector) in the context of linear evaluation of representations.

2.2 Unsupervised representation learning

Representation learning is concerned with finding features in the data that are useful to solve some arbitrary learning task. In a supervised context, the task to solve is given by the target values, y_i. However, in an unsupervised setting we assume that such target values are not accessible for the method to learn a representation.

We can describe an (unsupervised) visual representation learning task by assuming that we are given a dataset of inputs images $\mathcal{D} = \{X_i\}_{i=1}^N$. The goal is to learn a mapping, ϕ, (typically a convolutional network [37]) from the input images into feature vectors, $\mathbf{z}_i = \phi(X_i)$, that allow to predict arbitrary target values $\mathbf{y}_t$ associated with a task t. It is important to stress that this target values are unknown to the method and cannot be used to learn the mapping ϕ. Thus, a desirable property of unsupervised methods is that they learn representations which are useful for a variety of tasks.

2.2.1 Current approaches

Pretext tasks. A recent literature called self-supervised learning has attracted increasing attention due to its capacity to learn useful high-level features from visual data without labels [32]. These methods construct surrogate tasks (often called pretext tasks) for which labels can be automatically derived from the input images. Then, they train a convolutional network to solve this pretext task. For instance, [43] learns representations by solving Jigsaw puzzles. [19] consists in rotating images by a random factor of 90 degrees and then training a network to predict this factor. The pretext task must be designed in a way that the features learned in the process prove useful to solve a downstream task of interest (e.g. image classification); this often requires domain knowledge. For a comprehensive survey of this literature see [32].

Clustering. Another approach consists in the iterative application of two steps: a clustering step in which the representations encoded by ϕ are clustered and a learning step in which the network parameters are updated to predict the cluster assignments from the clustering step. This is precisely the approach followed by DeepCluster-V1 [10], which uses a standard clustering algorithm (K-means) and then updates the parameters of the network using a cross-entropy loss on the cluster assignments. A more principled method called the SeLA algorithm [2] integrates the two steps (clustering and learning) into a single framework in which both steps contribute to minimise a single loss function. This is, they learn both the network parameters and the labels assignment from the same loss signal. The labels assignment (clustering) is done with the Sinkhorn-Knopp algorithm [15].

Contrastive learning. Most of the recent methods for unsupervised representation learning revolve around the contrastive learning framework. It can be seen as a continuation of the pretext tasks literature where the task to solve is an instance discrimination task [58]. This task consist in training a network to discriminate between instances (examples in the dataset). To achieve this in a tractable way, the task is formulated as Noise Contrastive Estimation (NCE) [24]. The methods must generate different views of the images, and then train a network to distinguish among pairs of views taken from the same image (positive pairs) and pairs taken from distinct images (negative pairs). The representation learned by this network is then used to solve a downstream task of interest.

For training the network, the methods use a contrastive loss [25]. A common choice

is the InfoNCE loss [44] which for a positive pair of encoded views, $(\mathbf{v}_i, \mathbf{v}_j)$, is given by:

$$\ell_{i,j} = -\log\left(\frac{\exp(sim(\mathbf{v}_i, \mathbf{v}_j))/\tau}{\sum_{k\neq i}\exp(sim(\mathbf{v}_i, \mathbf{v}_k))/\tau}\right) \tag{2.1}$$

Where i, j is a positive pair, $\mathbf{v}_k$ is the encoded vector of view k, τ is the temperature hyperparameter, and $sim(\mathbf{v}_i, \mathbf{v}_k)$ is the cosine similarity between $\mathbf{v}_i$ and $\mathbf{v}_k$, given by: $sim(\mathbf{v}_i, \mathbf{v}_k) = \frac{\mathbf{v}_i \cdot \mathbf{v}_k}{||\mathbf{v}_i||_2 ||\mathbf{v}_k||_2}$.

The minimisation of this loss encourages representation vectors of positive pairs to be close to each other when projected into a high-dimensional unit sphere and those of negative pairs to be far apart from each other.

Although most contrastive learning methods share this general framework, they are differences in some of their constituent elements. In particular, we highlight two:

1. **Projection head.** Representation vectors from the views, $\mathbf{z}_i$, are projected by a head, h, into a lower-dimensional space to compute the contrastive loss. This is, $\mathbf{v}_i = h(\mathbf{z}_i)$. Some contrastive methods, like InstaDis [58], MoCo-V1 [27], and PIRL [41], use a linear projection head (a fully-connected layer). More recent methods, like InfoMin [56], SimCLR-V1 [12], and MoCo-V2 [14], use a non-linear one: a multi-layer perceptron (MLP) with 1 hidden layer. In Table 5.1 we specify the projection head used by the different methods evaluated in Chapter 5.

2. **Image augmentations**. To produce the views for the contrastive task, the methods rely on a set of image augmentations , $\mathcal{T}$. There are different choices of transformation groups ,$\mathcal{T}$, in the literature, but most share some common elements. For instance, most methods use random cropping and colour distortions. We distinguish between two broad augmentation schemes: one that we refer to as "Strong" which follows the one proposed by the SimCLR-V1 method and is used by many others (e.g. BYOL [23], MoCo-V2 [14], SimCLR-V2 [14]); it includes Gaussian blur in $\mathcal{T}$ along with stronger parameters for the other transformations (e.g. stronger colour distortions); the second one, which we term "Weak", is the one followed by methods like MoCo-V1 [27] and InsDis [58] which use weaker transformation parameters and do not include blurring. In Table 5.1 we categorise into these schemes the augmentations followed by the methods evaluated in Chapter 5.

There are more elements in which methods may differ (e.g. the use of a momentum encoder [27] or a memory bank [58]) but in this work we only focus on the ones outlined above, since previous work has found them to be determinant for performance [12], [14], [45] [56]. In Section 5.3.3 we show that our results strongly support this claim.

Combining contrastive learning and clustering. Recent work has shown strong performance on semantic classification tasks by combining the contrastive learning and clustering frameworks. The motivation for this approach is that the contrastive loss from Equation 2.1 encourages the separation of views from different instances irrespective of their semantic similarity [38]. However, if the contrastive task is done between instances and protoypes of groups of instances (i.e. clusters centers), rather than between individual instances, the methods may learn more general features shared by similar instances clustered together.

PCL-V1 and PCL-V2 [38] leverage this idea by enhancing the contrastive framework from MoCo [27] with a simple algorithm (K-means) for clustering the representations. However, unlike DeepCluster-V1, for the learning step, the task is not to predict the cluster assignments, but a contrastive task defined by the ProtoNCE loss [38], which is composed of two terms: the InfoNCE loss to contrast between encoded instance views (Equation 2.1) and a very similar term which contrast between encoded instance views and cluster centers. In this way, the loss not only encourages the representation of views from the same images to be similar between them, but also to the protoype from their cluster. The only difference between PCL-V1 and PCL-V2 is that the enhanced version (V2) follows SimCLR [12], and uses the Strong augmentation scheme to generate the views along with a non-linear projection head.

DeepCluster-V2 [11] follows a similar approach but removes altoghether the instance-level InfoNCE term and performs the contrastive task exclusively between instance views and clusters centers. SeLA-V2 [11] follows the same approach, but within the SeLA framework [2] for labels assignment. SwAV [11], uses the SeLA framework but it keeps the soft-assignments produced by the Sinkhorn-Knopp algorithm instead of discretizing them into cluster assignments. Additionally, for their learning step they use a swapped contrastive task where the goal is to predict the soft assignment (called code) of a view given a different view of the same image.

2.2.2 Evaluation of unsupervised representation learning

A comprehensive benchmark for evaluating unsupervised representation learning methods remains an open problem. The main difficulty is that it is not clear which is the correct task or set of tasks to evaluate the methods. The currently most used task is the ImageNet-1k classification task from ILSVRC 2012 [35]. This has been the task of choice in studies attempting to benchmark representation learning methods [33], [3], and of individual methods to showcase their capacity ([12], [27], [11], [38], [23], etc.). Evaluation on this task is commonly done by following a linear evaluation protocol [33], [61] where a linear model is trained using as input the frozen representations learned by the methods.

Some previous studies have extended the evaluation to include other related tasks. For instance, [22] evaluates methods using 5 common visual tasks (e.g. image classification, object detection). In the evaluation, they fine-tune the encoder network, ϕ, pre-trained by the representation learning method appending to it a specialised architecture for the task being considered. [17] performs a transfer study where they test the capacity of a representation learned on a specific image distribution (ImageNet) to generalise to different distributions and tasks. For some of their tasks, they also append a specialised architectures to the pre-trained encoder and fine-tune it for the respective task.

Additionally to the tasks used to benchmark representations, an important question is the protocol used. In this regard, there is not consensus in the literature. For instance, some evaluations use a linear protocol, whereas other fine-tune with specialised architectures. Moreover, even within the linear protocol there are many different design choices. Some linear evaluations use logistic regression [33], [23] [12], but others use a linear SVM [22], [38]); some use L-BFGS [33], while most use SGD [12], [27], [58]). Even within those using SGD, there is a very diverse choice of hyperparameters settings. For instance, some use a relatively large learning rate (e.g. 30 [27], [55]), while others use smaller ones (e.g. 1.6 [12], 0.3 [2]).

Our contributions to the evaluation of representations are three: We show empirically, within the linear protocol, some of the different optimization behaviours of SGD depending on the linear separability of the data; we develop a broad task-based benchmark with a unified evaluation protocol where the variation comes exclusively from the tasks; using such benchmark, we evaluate some of the methods that have achieved state-of-the-art performance on the ImageNet-1k classification task.

Chapter 3

SGD optimization for linear evaluation

The most common approach to evaluate unsupervised representation learning methods consists in using the learned representation as fixed input data to train a linear model for a given downstream task (e.g. image classification). However, the implications of the choice of optimization algorithm to fit such linear model has received little attention. Stochastic Gradient Descent (SGD) [47] has been the most widely used optimizer for this kind of evaluation. In this Chapter, we show empirically that this choice has different implications depending on the degree of linear separability of the evaluation task and the hyperparameters configuration. Specifically, in the context of linear evaluation of representations, we address the following questions:

1. Can the choice of optimizer and hyperparameters have a significant effect on the outcome of the evaluation?

2. Can we find some configurations of SGD which are able to reproduce the solution found by an optimizer with strong convergence guarantees (L-BFGS [39])?

3. In the cases where SGD cannot reproduce such solution, can its solution be viewed as the solution of a regularized optimization problem?

The answer to these questions provides guidance on when one can expect SGD to fully optimize the specified loss function and when it should be treated as an implicit regularizer.

3.1 Experimental setup

Dataset. As mentioned in Section 2.2.2, the ImageNet dataset [48] is the most common choice to train and evaluate visual representation learning methods. Therefore, we

use it for our optimization experiments. However, fitting a model with L-BFGS for the full ImageNet dataset is computationally intractable; thus, we sample a training subset consisting of 64000 samples. This subset is class-balanced with respect to the image classification task from ILSVRC 2012 [35] (ImageNet-1k). Specifically, we sample 64 images from each class of this task to construct the training set. We denote this image dataset as $\mathcal{D} = \{(X_i)\}_{i=1}^{N}$, where each X_i is a 224×224 RGB image[1]. We encode these images into representation vectors: $\mathbf{z}_i = \phi(X_i)$, where $\phi : \mathbb{R}^{3\times224\times224} \mapsto \mathbb{R}^{2048}$ is the ResNet-50 (truncated after the last convolutional block) mapping the $i-$th RGB image into a representation vector $\mathbf{z}_i$ of size 2048. Specifically, we consider the ResNet-50 pre-trained with the ImageNet-1k classification task. We use this particular encoder because it is the supervised baseline used in most unsupervised representation learning evaluations. We then use the dataset composed of the representation vectors, $\mathcal{D}^{'} = \{\mathbf{z}_i\}_{i=1}^{N}$, as input data for a softmax logistic regression model (also called multinomial logistic regression).

Tasks. We train a separate instance of this model for two different tasks chosen for the different degrees of linear separability that they impose on the feature vectors. The first task is the ImageNet-1k classification task. When labelled with the classes from this task, The representation vectors from distinct classes can be separated very well by linear decision boundaries. This is revealed by the train accuracy obtained by the logistic model fitted with L-BFGS (shown in Table 3.1). We refer to this as the linearly separable task. We construct a second task from the image data, $\mathcal{D}$, which consists in predicting the image region (among 5 disjoint patches) with the highest luminosity value (See section 4.2 for details). Applying this class labels to the corresponding feature vectors from $\mathcal{D}^{'}$ induces a low degree of linear separability among the different classes, as revealed by the low accuracy in the training set (Table 3.1). Thus, we refer to this as the linearly non-separable task.

Loss function For each task, t, with C_t classes, we denote the (one-hot encoded) target vector of example i from class k as $\mathbf{y}_i^t$. Then, the the cross-entropy loss for this example is given by: $\ell_i = -y_{i,k} \log(Softmax_{i,k})$, where $y_{i,k} = 1$ is the $k-$th element of $\mathbf{y}_i^t$, $Softmax_{i,k}$ is the $k-$th element of the softmax output: $Softmax(W\mathbf{z}_i + \mathbf{b})$. W is a $C_t \times 2048$ weights matrix and $\mathbf{b}$ is a bias vector of size C_t. These are the parameters to be learned, which is done by minimising the loss function averaged over all the training

[1] We center crop the original images to a 224×224 size

Task	Train Accuracy
Image classification	99.97%
Brightest Region	44.26%

Table 3.1: **Accuracy in the training set obtained by the logistic model fitted with L-BFGS**. The classes from the Image classification tasks are well separated by linear decision boundaries. This is not the case for the task consisting in predicting the brightest region of the image.

examples:

$$J(W, \mathbf{b}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C_t} y_{i,c} \log(Softmax_{i,c}) \tag{3.1}$$

This loss function turns out to be convex. In the linearly non-separable case, there is a unique minimum yielding a finite minimum value. In the perfectly linearly separable case, the objective function has an infimum at zero which is not attainable by finite parameter values [53]. The brightness prediction task corresponds to the former case, whereas image classification is closer to the latter.

Optimization settings. For each task, we train separate logistic regression models. One by minimising Equation 3.1 with the L-BFGS [39] algorithm and others by doing the minimisation with different hyperparameters configurations of SGD [47]. As noted in Section 2.1, L-BFGS provides strong convergence guarantees for convex problems; hence, it will serve as the optimization baseline to compare with SGD.

For L-BFGS, we setup the algorithm with the default settings recommended by the implementation used[2]. We allow the algorithm to run for a maximum of 5000 iterations to ensure convergence. In practice, this restriction was largely non-binding as in every case convergence was achieved in a much smaller number of iterations (195 for the linearly separable case and 267 for the non-separable one).

For SGD, we use the simplest version (described in Section 2.1) without momentum and train the model for 5000 epochs. The only enhancement to the SGD algorithm that we apply is a multi-step schedule for the learning rate, since theoretical analysis

[2]For L-BFGS, we used the implementation from scikit-learn. For SGD, We implemented the logistic regression model as a single-linear layer network with cross-entropy loss in PyTorch.

has shown that it is crucial for the convergence of the algorithm [54] [7]. For scheduling, we set three milestones at the 1500-th (30% of training), 3000-th (60% of training), and 4000-th epochs(80% of training). At each milestone, the learning rate is decayed by a factor of 0.1. In this case, there are two hyperparameters to tune: the initial learning rate and the mini-batch size. Different linear evaluations of representations using SGD for the ImageNet-1k task have used a very wide range of optimal initial learning rates and mini-batch sizes obtained by a grid-search (see Section 2.2.2). Therefore, it is difficult to know in advance on which range sensible values for these hyperparameters are. Then, we test various values covering a wide range. For the initial learning rate, we consider the values: 0.01, 0.1, 1, 10, 100, 1000 and for the batch-size: 6400, 12800, 19200, 25600, 32000, 64000. We use larger batch-sizes (e.g. 64000) than those reported by previous work, since we are also interested in the behaviour of the algorithm in the limit of full-batch GD to assess if removing the stochastic element (see Section 2.1) will enhance the convergence of the algorithm.

3.2 Solutions comparison.

Assessing whether both optimizers find the same solution is not straightforward. A direct element-wise comparison of the parameters values learned by the two optimizers is not meaningful, since we are not interested in having identical parameters, but equivalent models; this is, models which make the same probabilistic predictions. In particular, the probabilities predicted by the logistic model are scale-invariant in the parameters: if we scale all the parameters values by a constant factor K, then the predictions remain identical. A direct comparison of the training loss from Equation 3.1 can also be misleading, especially in the more linearly separable case, since there could be different solutions with a very small training loss. Moreover, we are especially interested in the generalisation performance of the models, this is, how they predict for data outside the training set.

A good measure to indicate whether the models are equivalent in terms of their probabilistic predictions is the test loss. This is, the loss from Equation 3.1, but averaged across the test set rather than the training set. This metric takes into consideration the probabilistic predictions of the models in unseen data which is our main interest. Moreover, the difference between this loss computed with two different parameters sets (one trained with SGD and on with L-BFGS) can be interpreted as the average log odds ratio of the two models probability prediction for the correct classes. Specifically, if

we consider the parameters learned with SGD, (W_{sgd}, b_{sgd}), and the ones learned with L-BFGS, (W_{lbfgs}, b_{lbfgs}), then the difference in the test loss is given by:

$$J^{test}(W_{sgd}, b_{sgd}) - J^{test}(W_{lbfgs}, b_{lbfgs}) = -\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \sum_{c=1}^{C_t} y_{i,c} \log\left(\frac{Softmax_{i,c}^{sgd}}{Softmax_{i,c}^{lbfgs}}\right) \quad (3.2)$$

Thus, it is clear that if the models make the same probabilistic prediction on every test example, the difference will be zero. However, even small differences are significant, for instance a difference of 0.1 implies an average odds ratio of 1.1, this is, the probability assigned by the SGD-fitted model to the correct class is on average 10% larger than the one assigned by the L-BFGS-fitted model.

We must note however, that this constitutes only an approximation of our metric of interest, since we are interested in assessing whether the two models will make the same probabilistic prediction for *any* possible input, but our test set is finite. Additionally, the difference from Equation 3.2 only compares the probabilistic prediction of the two models for the correct class of each example. We address these concerns by considering a class-balanced (see Appendix B) and relatively large (50000 examples) test set for each task coming from the ImageNet validation set which we held out from the training sampling. Additionally, in Appendix A.1 we consider alternative similarity measures which support our main conclusion.

3.3 SGD optimizes in the linearly non-separable case

The first question we address is whether we can found some hyperparameters configuration for which SGD can converge to the same solution as L-BFGS in the two settings we consider regarding linear separability. For this, we focus the analysis on the case with the multi-step schedule for the learning rate, but the results without schedule are similar and reported in Appendix A.3.

As shown in Figure 3.1, SGD cannot reproduce the L-BFGS solution in the linearly separable case with any of the tested hyperparameter configurations. In all cases, the magnitude of the difference is above 6.5, indicating an average odd ratio of more than 665. For the non-linearly separable task, with a learning rate of 0.1 and a mini-batch of 6400, SGD arrives at the same solution as L-BFGS as revealed by a small difference in test losses (below 0.01 indicating an average odds ratio very close to 1). In both cases when the learning rate is large (100, 1000), for all the mini-batch sizes, we

observe large differences in test loss. In the non-linearly separable case, for all learning rates, the smallest mini-batch size (6400) is the one with the lowest difference, and in particular lower than 64000 corresponding to full-batch GD. Thus, in this case the stochasticity of SGD does not hinder its convergence.

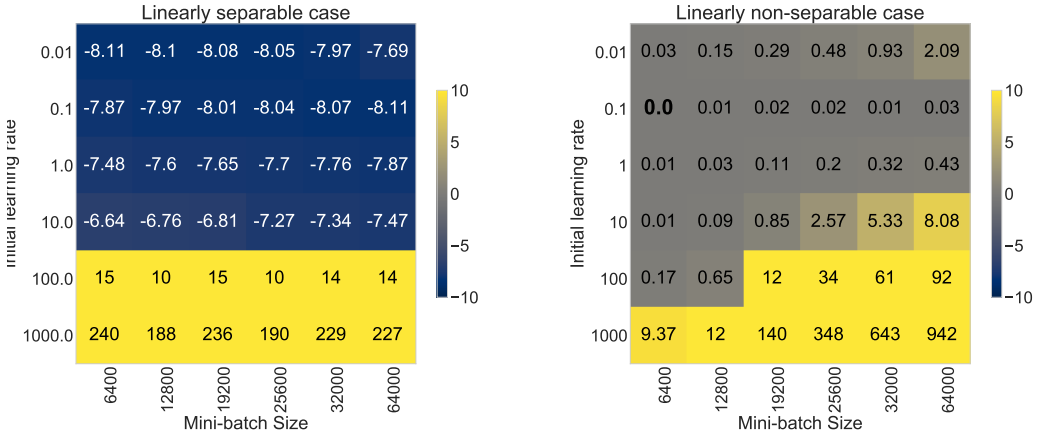

Figure 3.1: **Difference between the L-BFGS and SGD test loss for various hyper-parameters configurations.** Each value* is the difference: $J_{sgd} - J_{lbfgs}$ where the loss is averaged over the test set (See Equation 3.2). In the linearly separable case, SGD cannot reproduce the solution from L-BFGS for any of the configurations tested. With an initial learning rate of 0.1 and a mini-batch size of 6400, SGD reproduces the solution of L-BFGS in the linearly non-separable case.

* Values were rounded to three significant figures

Alternative similarity metrics support our main conclusion (As shown in Appendix A.1).

3.4 SGD regularizes in the linearly separable case

Figure 3.1 shows that, in the linearly separable case, for initial learning rates below 100, the test losses from SGD are consistently lower than the one of L-BFGS. This suggests that SGD yields a model with better generalization performance. We now investigate whether this implicit regularization done by SGD is consistent with other forms of explicit regularization on the linearly separable task (the linearly non-separable case is treated in Appendix A.2).

Weight decay. We consider a common form of regularization referred to as weight decay [36]. Specifically, we consider a modified loss function that penalises the squared L2 norm (also called Frobenius norm) of the weight matrix, W. In the specific implementation we use, the new objective of L-BFGS is to minimise the following loss:

$$\hat{J}(W, \mathbf{b}; \lambda) = \frac{1}{2}||W||_2^2 - \lambda J(W, \mathbf{b}) \tag{3.3}$$

Where $J(W, \mathbf{b})$ is the loss from Equation 3.1, $||W||_2$ is the L2-norm of the weights matrix and λ is a coefficient that regulates the relative importance of the loss and the norm terms. Note that a higher value of λ implies a lower relative importance of the penalty term, and, hence, a weaker regularization. For this, we also test a wide range of values of λ (0.001, 0.01, 0.1, 1, 10, 100).

To assess the degree to which SGD emulates the solution of this regularized problem, we optimize the regularized loss with L-BFGS and compare the solution with the unregularized results from SGD with the same settings described in Section 3.1. Here we focus on SGD with a learning rate schedule but the results without schedule are similar and can be seen in Appendix A.3.

In Figure 3.2, we compare the test loss from SGD (without regularization) with the one of L-BFGS with different values of λ. An important precision is that this test loss is computed with the unregularized loss function of Equation 3.1 averaged over the test set, since we are exclusively interested in comparing the models probabilistic predictions for which the weights magnitude and, hence, the norm term from Equation 3.3 is irrelevant.

For reference, the test loss obtained by L-BFGS without regularization is 9.18, and the losses with initial learning rates of 100 and 1000 are, for every mini-batch size, above 19, so we focus the analysis on the smaller ones (0.01, 0.1, 1, 10). We note that indeed for those initial learning rates and some mini-batch size, there is a value of λ for which the test losses match (difference of less than 0.01). There does not appear to be a direct correspondance between the different values of λ and either the initial learning rate or the mini-batch size. Thus, it appears that both of the SGD hyperparameters modulate the implicit regularization in a non-trivial way.

Early stopping. We perform a similar comparison to another form of regularization called early stopping [26]. For this, we restricted the iterations performed by L-BFGS in the minimisation of Equation 3.1 to a fraction of the iterations it took to converge

(when the limit of 5000 iterations was non-binding). Specifically, we consider 1%, 3%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 80% and 90% of the required iterations. We then compare the test loss of L-BFGS at these milestones with the final solution (after the 5000 epochs) of SGD. Under specific hyperparameters values (initial learning rate of 0.01 and batch size of 25600) SGD arrives at the same solution than a very early (3%) stage of L-BFGS. Thus, for some hyperparameters regimes, the implicit regularization of SGD resembles L2 weight decay and for others, an early milestone of L-BFGS.

3.5 Conclusions

In the context of linear evaluation of representations, we find that in a linearly non-separable problem, for a specific initial learning rate (0.1) and mini-batch size (6400), SGD with a multi-step schedule for the learning rate can match the solution obtained by a robust optimizer (L-BFGS). This is not the case in a highly linearly separable problem, where the solution found by SGD is more akin to a regularized solution. The implicit regularization from SGD leads to solutions equivalent to explicit L2 weight decay with some hyperparameters values and to early stopping of L-BFGS with others.

Our results show that the choice of optimizer for linear evaluation of representations can have relevant implications, since depending on the degree of linear separability and the hyperparameters values SGD can learn distinct models. Additionally, the implicit regularization which SGD can induce may be confounded with the capacity of the representation to enhance the generalisation performance. We leave for future work extending these experiments for the full ImageNet dataset. However, as the dataset increases we would expect that the linear separability of the problem decreases; thus, SGD could be better-suited as an unregularized optimizer for large scale problems. A difficulty for verifying this is to find a good optimization baseline, since it is intractable to naively scale L-BFGS for a dataset of that size. Promising approaches for this are the recently-proposed stochastic versions of L-BFGS [42], [6].

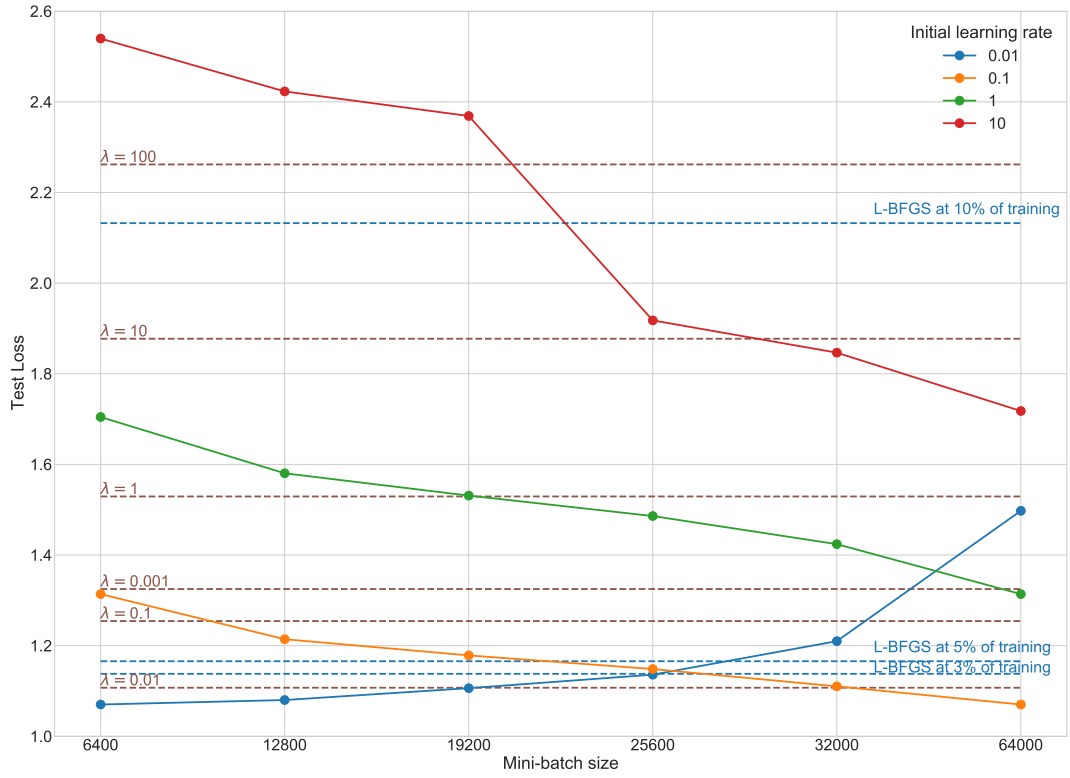

Figure 3.2: **Comparison of the (unregularized) loss in the test set between SGD and regularized solutions of L-BFGS for the linearly separable case.** For regularizing the objective loss of L-BFGS we use L2 weight decay with different values of λ. A lower value of λ implies stronger norm penalty. We also consider early stopping of L-BFGS at different milestones of training *. All the values are computed with the unregularized loss (Equation 3.1) on the test set **. For some hyperparameters values, SGD, without explicit regularization, finds solutions which are very similar to the regularized solutions of L-BFGS.

* For clarity in the chart, we ommit the case $\lambda = 0.0001$, a very early milestone (1%) and later ones (20% and above) since these configurations have a test loss above 3.

** For reference the test loss obtained with L-BFGS without regularization is 9.18.

Chapter 4

Representation learning benchmark

In this chapter, we describe the design of our proposed benchmark to evaluate representation learning methods. We outline the principles that guided the design choices, the tasks categories included, and the evaluation protocol followed. Finally, we explain how all the elements are integrated into an evaluation pipeline that can be used to evaluate any representation learning method.

4.1 Design principles

The core principle for our benchmark is that a good representation should encode useful features to solve a wide variety of tasks. This is specially relevant in an unsupervised context where the labels associated with the tasks of interest are unknown to the method. Thus, the objective is to learn general-purpose representations that are well-suited for a variety of tasks. Following this principle, broadness of tasks is the distinctive property of our benchmark.

Additionally, we design tasks that are complementary to each other, this is, that allow to evaluate distinct aspects of a representation's capacity. For instance, some tasks should require that the representation encodes high-level visual features, while others should require lower-level features; we also consider tasks that test the flexibility of the representation to enable the prediction of arbitrary (random) targets, its robustness to noise in the visual data, and its data efficiency.

Finally, we enforce a single evaluation protocol for the different tasks, thus holding constant the core elements, like the encoder, optimizer, image distribution, and downstream model. In this way, all the variability in the benchmark comes from the tasks themselves characterised by their sets of target labels. We design the full evaluation

pipeline in a way that any method learning a representation from visual data can be evaluated with our benchmark. None of the design choices were made for a particular method or even group of methods.

4.2 Tasks categories

The benchmark consist of 29 different tasks grouped into 7 categories. All tasks are formulated as classification tasks in order to use a single evaluation protocol (see Section 4.4). For generating these tasks, we used a fixed training set of images sampled from ImageNet [48]. Specifically, we sample 64000 training examples, where 64 examples were taken from each class of the ILSVRC 2012 image classification task (ImageNet-1k) [35]. We do not consider a larger subset due to computational constraints which are further described in Section 4.4. We then leverage these sampled images to create a variety of tasks to evaluate the representations learned by the different methods. Specifically, we consider the following tasks categories:

1. **Semantic tasks**. These are image classification tasks with semantic class labels. In this category we include the ImageNet-1k classification task, which is the most commonly used in the literature to evaluate representation learning methods (see Section 2.2.2). In this work, we refer to it as the fine-grained classification task. Additionally, we leverage the hierarchical structure of the WordNet database [40] to perform relabellings of the synsets from the original 1000 classes. Using hyponymy relations, we create a coarse-grained classification task with superclasses of the classes from the fine-grained classification task. With meronymy relations, we also create a parts classification task consisting in classifying whether the object of a given image has a certain part. To successfully discriminate between these semantic classes, the representation should encode relevant high-level features with various levels of granularity.

2. **Colour tasks**. These are 3-way classification tasks consisting in identifying the dominant colour (red, green, or blue). We consider a task at the image level (identifying the dominant colour of the image) and local tasks consisting in identifying the dominant colour at particular pixel locations (e.g. the central pixel, a corner pixel, a random pixel). These tasks require the representation to encode colour information.

3. **Luminosity tasks**. These tasks consist in detecting luminosity as measured by the perceived luminosity equation [46]: $L = 0.299R + 0.587G + 0.114B$, where R, G, B are the pixel values at the red, green, and blue channels respectively. Here we also consider a task at the image level consisting in predicting the brightest region of the image among five equally-sized, disjoint crops (central and corners) (this is the linearly non-separable task from Chapter 3). We also consider tasks at the pixel level consisting in predicting the (integer) luminosity value of a pixel at a specific location (e.g. central, corner, random). These tasks test the degree to which the luminosity values are preserved in the representation learned by a given method.

4. **Low-level visual features tasks**. For this tasks, we rely on the detection of low-level visual features (edges and corners). To generate them, we take five equally-sized, disjoint regions of the image (centre and corners) and use conventional computer vision algorithms to detect the presence of low-level features in them (we use the Canny algorithm [9] for edge detection and the FAST algorithm [57] for corner detection). The classification tasks are then based on predicting the presence of these features in the distinct image patches. These tasks could be considered an intermediate point between the image statistics tasks (colour and luminosity categories) and the high-level semantic tasks.

5. **Random Neural Networks tasks.** To generate the targets for these tasks we use Multilayer perceptrons (MLP) (with one hidden layer and Batch Normalization [30]) with random weights. Specifically, the class label of a given example image flattened into a vector $\mathbf{x}_i$ is given by:

$$\arg\max MLP(\mathbf{x}_i)$$

Where *MLP* is the multilayer perceptron and the maximization is over the output vector. We sample the weights of the networks from Gaussian distributions with mean 0 and different variances. We also vary the number of classes (output units of the MLP). These tasks test the degree to which the representation can target arbitrary class labels produced by random non-linear mappings. We choose this specific architecture for simplicity. However, in future work, an interesting extension of this category could be to use specialised architectures, e.g. convolutional or recurrent networks to assess the degree to which the representations can target the specific inductive bias of each architecture [21].

6. **Scarcity tasks**. These are N-way K-shot tasks. We sample K random examples from N random classes from the full ImageNet dataset to construct them. Specifically, we consider all the combinations of $N = \{10, 20, 50\}$ and $K = \{1, 5, 10\}$, since these are some of the common values used in the few-shot literature [52]. These are roughly sub-tasks of the fine-grained classification task, since their classes are a subset, but the training samples are not. The purpose of these tasks is to test the data efficiency of the representations, this is, the degree to which they allow to solve a standard image classification task with few training samples.

7. **Robustness tasks**. For these tasks, we follow the noise generation processes from [29]. The training set is the same as the one of the fine-grained classification, but the images from the test set are corrupted with different types of noise (Gaussian, shot, and speckle). We then evaluate the models trained on the fine-grained classification task on these corrupted test sets. It is important to precise that the noise is applied directly to the image (before being encoded into a representation vector), since the objective is to evaluate the degree to which representations are robust to the introduction of nuisances in the visual data [1].

Finally, we remark that only the fine-grained classification task and those from the robustness and scarcity categories are perfectly class-balanced by construction. All the others have various class distributions. These distributions along with additional implementation details of the tasks construction can be found in Appendix B.

4.3 Representations

Using the tasks described in Section 4.2, we can evaluate the representation learned by a given method. Specifically, we can test the degree to which such representation contains the necessary features to solve the different tasks in the benchmark. For this purpose we need to first encode the visual data into representation vectors with a convolutional network pre-trained with the specific method we desire to evaluate. These representation vectors are then the input data used to solve a given task (as further explained in Section 4.4).

Previous work ([33], [22]) has found that that the choice of training data, encoder architecture, representation size, and the layer at which the encoder is truncated can

have a significant effect on the evaluation results. For this reason, we hold these elements fixed. Specifically, as the backbone encoder we choose the ResNet-50 [28] truncated after the last convolutional block which renders a representation vector of size 2048. This architecture should be pre-trained with the method to evaluate using the ImageNet dataset. We select these particular settings (architecture and dataset) because they are the most common choice in the literature (e.g. [33], [17], [12], [23], [27], [58], [2], etc.).

More formally, if we desire to evaluate method, m, on task t, we require the ResNet-50 encoder pre-trained with m on the ImageNet dataset: $\phi_m : \mathbb{R}^{3\times224\times224} \mapsto \mathbb{R}^{2048}$. Then, we use it to generate the representations dataset: $\mathcal{D}_m^{'} = \{z_{m,i} = \phi_m(X_i)\}_{i=1}^N$, where $X_i, i = 1, ..., N$, are the images sampled for the training set. Then, $\mathcal{D}_m^{'}$ in combination with the set of target vectors, $\{\mathbf{y}_{i,t}\}_{i=1}^N$, constitute the training set for task t

4.4 Evaluation protocol

For evaluating the representations from the different methods, we follow a linear evaluation protocol [33] [61]. Specifically, using $\mathcal{D}_m^{'}$ and $\{\mathbf{y}_i\}$, we train a softmax logistic regression model (also called multinomial logistic regression) for each task.

We optimize the respective unregularized cross-entropy loss function (Equation 3.1) using L-BFGS [39] as the optimization algorithm. We select this optimizer for two reasons: it offers strong convergence guarantees to a unique solution in a convex problem like the logistic regression problem (see Section 2.1), independently of hyperparameter choices and initial conditions; this avoids the need to tune any hyperparameter which would be impractical given the number of tasks in the benchmark; moreover, this provides determinism to our results, since they are independent of minor settings (e.g. the random seed used). Additionally, as noted in Chapter 3, this optimizer does not provide implicit regularization as SGD, and then any enhancement in generalization performance can be attributed exclusively to the representation, instead of the optimizer. However, a downside of this choice is that it is computationally expensive to scale to large datasets. Therefore, we consider a 64000-samples, class-balanced (with respect to the ImageNet-1k task) subset from ImageNet and leave for future work scaling the protocol the full dataset.

After training each logistic regression model, we evaluate it using as test set the full ImageNet validation set of 50000 samples (using the same preprocessing as for the training data). We use the accuracy in this test set as the score of the represen-

tation in each task. Finally, contrary to previous work doing multi-task evaluation of representations [22], [17], we hold this protocol constant across tasks.

4.5 Outcome

The benchmark is designed in a way that any representation learning method can be evaluated with it. Following the guidelines established above and summarised in Figure 4.1, the outcome of the evaluation allows a comprehensive assessment of the method along different tasks categories. We encourage the users of the benchmark to consider the results of each category independently instead of summarising them, since this allows a more comprehensive assessment of the representation's capabilities. We showcase this in our evaluation exercise from Chapter 5

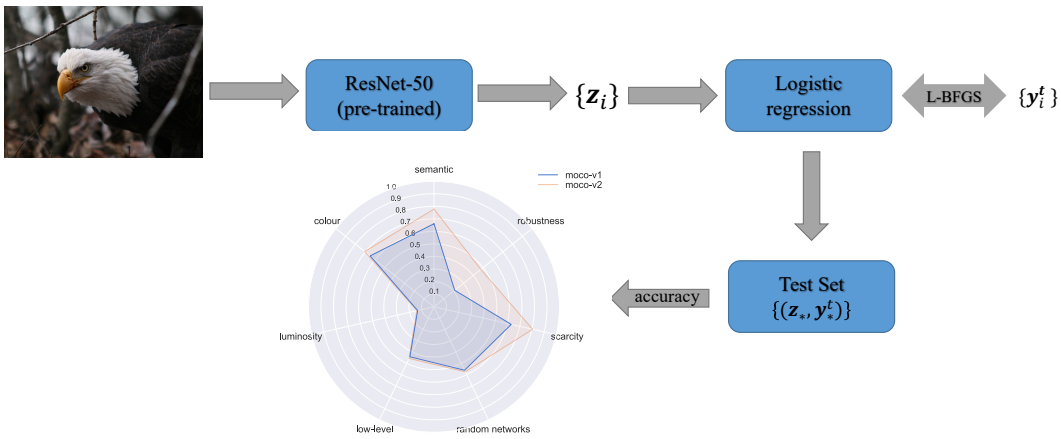

Figure 4.1: **Overview of the evaluation.** To evaluate a given method (characterised by its pre-trained ResNet-50 encoder) we encode the images of the training set into representation vectors, $\mathbf{z}_i$, we then use these as input data to train a logistic regression model to solve each task t using the corresponding target labels $\mathbf{y}_t$. We fit the models using L-BFGS as optimizer. Finally, after training a logistic model for the given task, we evaluate it on the held-out test set and use the test accuracy as the score. The outcome of this is an assessment of the representation's performance along the different axes (categories) of evaluation.

Chapter 5

Evaluation results

In this chapter, we showcase the benchmark we developed, as described in Chapter 4, by evaluating thirteen of the current state-of-the art unsupervised representation learning methods. With the results from this evaluation, we examine some properties of the benchmark itself. In particular, we assess how informative the different tasks are, in the sense of how well they differentiate between the different methods' performance. We also inspect groups of correlated tasks for which the ranking of the methods is similar. Finally, we examine the performance of the methods and analyse particular components which appear to contribute to the observed differences. The main objective of the chapter is to demonstrate how our benchmark can be used to provide a thorough evaluation of representation learning methods along different categories of evaluation, and how this, in turn, can reveal particular strengths and areas of opportunity for future research in the field.

5.1 Methods evaluated

A fully comprehensive evaluation covering all the methods from the literature is not feasible. Therefore, we focus on the contrastive learning framework [12], which is the currently dominant one around which most current methods revolve (as discussed in Section 2.2.1). We are also interested in the very recent line of research incorporating clustering and protoypes into this framework [38], [11] (as explained in Section 2.2.1).

Our selection also has practical considerations, since we require methods pretrained on the ImageNet dataset [48]. The recent study in transfer learning from [17] has followed a similar reasoning for their selection; thus, we evaluate the thirteen state-

of-the-art unsupervised methods considered in it[1]. The summary of these is presented in Table 5.1, but a more thorough description is provided in Section 2.2.1.

Model	Contrastive Learning	Clustering	Augmentation	Projection head
InsDis [58]	Instance	None	Weak	Linear
SimCLR-V1 [12]	Instance	None	Strong	MLP (1 hidden layer)
SimCLR-V2 [13]	Instance	None	Strong	MLP (2 hidden layers)
MoCo-V1 [27]	Instance	None	Weak	Linear
MoCo-V2 [14]	Instance	None	Strong	MLP (1 hidden layer)
BYOL [23]	Instance	None	Strong	MLP (1 hidden layer)
InfoMin [56]	Instance	None	Strong	MLP (1 hidden layer)
PIRL [41]	Instance	None	Weak.	Linear
SeLA-V2 [11]	Protoype	Sinkhorn-Knopp	Strong	MLP (1 hidden layer)
PCL-V1 [38].	Instance and Protoype	K-means	Weak	Linear
PCL-V2 [38].	Instance and Protoype	K-means	Strong	MLP (1 hidden layer)
DeepCluster-V2 [11]	Protoype	K-means	Strong	MLP (1 hidden layer)
SwAV [11]	Protoype	Sinkhorn-Knopp	Strong	MLP (1 hidden layer)

Table 5.1: **Unsupervised models evaluated**. Here we focus only on specific elements that differentiate the methods and that previous work ([12], [14], [38] [11]) has shown to be relevant for their performance: data augmentation, projection head, and the use of prototypes (clusters centers) in the contrastive task. Details of these elements are provided in Section 2.2.1.

For comparison, we consider two baselines: the supervised ResNet-50 [28] pre-trained with the fine-grained classification from the benchmark on the full ImageNet dataset. This baseline has proven strong generalization capabilities in different semantic tasks [34] thus, it provides a good reference for these tasks categories. Additionally, it is the most commonly used baseline for measuring the progress of unsupervised representation learning. However, this representation discards irrelevant features for the fine-grained classification task [16] which are important for performance in other categories in the benchmark. Thus, as a second baseline, we consider: an untrained ResNet-50 (with random parameters). The spatial bias inherent in this architecture, allows a strong performance in some visual tasks even without training [10] [21]. In particular, this network can encode low-level features which may be discarded in the process of training.

[1]To obtain the pre-trained networks we borrow (with permission) the implementation from [17], publicly available at: https://github.com/linusericsson/ssl-transfer/blob/main/download_and_prepare_models.py

All the methods and the baselines are evaluated following the protocol described in Chapter 4 and summarised in Figure 4.1.

5.2 Informative and correlated tasks

To assess the degree to which the different tasks are informative of differences in representations' performance, we inspect how varied is the test accuracy obtained by the different models in them. We acknowledge that this measure has its caveats. For instance, if all the models perform equally well in a task is not that the tasks is not revealing anything about the methods (it reveals that all the representations are well-suited for solving such task), but at a more global level it makes it difficult to discern the specific elements which account for the good performance. On the other side, a task with large variability allows to inspect which components may be contributing to the differences in performance (as shown in our analyses from Section 5.3.2 and Section 5.3.3).

Figure 5.1 shows boxplots of the test accuracy obtained by the different representations in each task. In most of the tasks there is an outlier which typically corresponds to the untrained baseline. For this reason, throughout the analysis, we use the median and the inter-quartile range (IQR) as measures of central tendency and variability of the different models performance in a given task.

Informative and uninformative tasks. The robustness tasks and the scarcity tasks are the most informative of differences in performance. In particular, the three 1-shot tasks exhibit the highest IQRs (19, 22, and 29 percentage points). However, as the data availability increases, the IQR decreases (as revealed by the 5 and 10-shot tasks). This suggest that data efficiency is an important domain of differentiation of the representations capacity. This is analysed in Section 5.3.4

The semantic tasks also have a relatively large accuracy variability. In this case, the fine-grained task is not only a harder task (as revealed by the median accuracy), but also induces greater differences in performance compared to the coarse-grained and parts classification tasks. This might indicate that some models are particularly well-suited (relative to others) at capturing finer high-level features, but most have a similar capacity to encode coarser features.

On the other side, the colour and, especially, the luminosity tasks[2] at the pixel level

[2]The test accuracy for all methods in this task was poor because of strong overfitting.

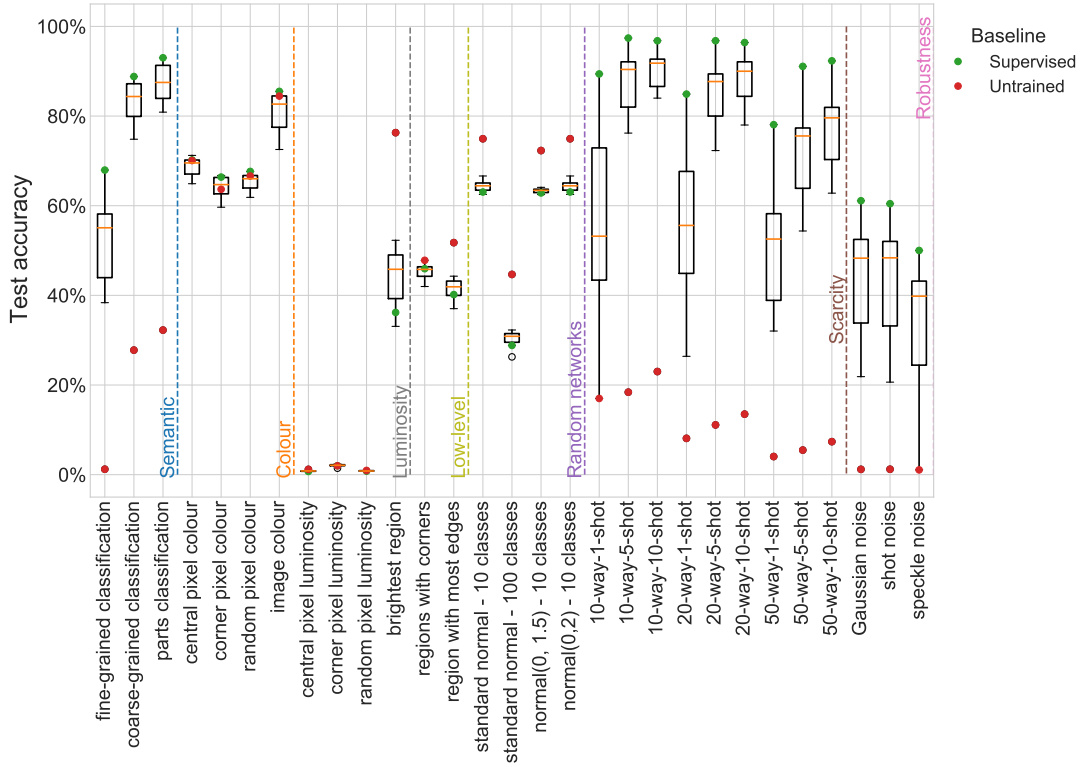

Figure 5.1: **Boxplots of the test accuracy obtained by the different representations in each task.** The tasks from the semantic, robustness, and scarcity categories are the most informative of the different methods' performance. Tasks at the pixel level (in the colour and luminosity categories), the random networks tasks and the low-level tasks do not exhibit large differences in performance.

are mostly uninformative with all methods having similar performance (IQRs below 3 percentage points). However, The tasks from these categories at the image level, i.e. the image colour and brightest region tasks, exhibit a larger variability in performance with IQRs of 7 and 10 percentage points respectively. This seems to indicate that all the models are (roughly) equally capable of retaining basic visual statistics at the pixel level, but some are better than others at retaining information at the image level.

Finally, the low-level and random networks tasks also exhibit low variablity in performance (IQRs below 3 and 2 percentage points repsectively). For the random networks tasks, neither the number of classes nor the standard deviation of the Gaussian distribution used to initialise the weights of the MLP used to generate the labels has a substantial effect on performance variability.

The correlation of tasks. The benchmark was designed in a way intended to evaluate complementary aspects of a representation's capacity. Measuring this in a direct way is not trivial. Therefore, as a proxy, we inspect the correlation of the ranking of the methods in the different tasks. In particular, if two tasks have a high ranking correlation it indicates the the same methods perform well (relative to the others) in both tasks. Then, it can be argued that both tasks are evaluating similar aspects of the representations capacity and, therefore, are not complementary to each other. We consider the ranking correlation instead of the accuracy correlation between tasks, because we are particularly interested in assessing if the same models are the best-performing ones in the different tasks, independently of the accuracy obtained.

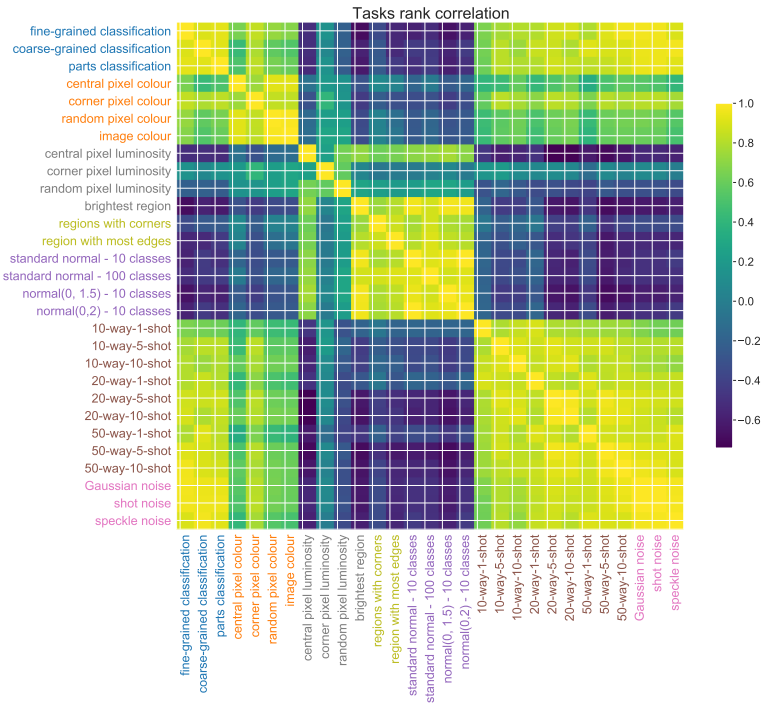

Figure 5.2: **Rank correlation between the different tasks.** We consider a ranking matrix where each entry is the ranking of a method (row) on a given task (column). The plot shows the correlations between the columns of such matrix. The names of the tasks from the same category are shown in the same colour. Roughly, The representations that perform better than the others in the high-level and colour tasks perform worse in the low-level, luminosity, and random networks tasks and viceversa.

As shown in Figure 5.2, there is complementarity in the benchmark as revealed by

the negative correlations across some tasks rankings. This is, representations that are among the best-performing in some tasks, are among the worst-performing in other ones. Morever, the rank correlations reveal two main tasks groups: the first group is composed of the high-level tasks (semantic, robustness, and scarcity categories) and the colour tasks, which surprisingly are more correlated with them, than with the other low-level tasks; the second group is composed of the low-level, luminosity, and random networks tasks which are correlated among them, but decorrelated from the others.

5.3 Representations performance

We assess the performance of the different representations at the category level by taking the average test accuracy across the individual tasks of a given category. Here, we take the mean instead of the median, since we desire that the performance in every task contributes to the score in the category. Some granularity is lost while summarising at the category level, although Figure 5.2 suggests that tasks within the same category are correlated among them, hence not much information is lost in the summary. However, we display the full results at the task level in Appendix C. The average test accuracy in the different categories along with the ranking (according to this average) are displayed in Table 5.2.

	Semantic		**Colour**		**Luminosity**		**Low-level**		**Random networks**		**Scarcity**		**Robustness**	
Model	Accuracy	Ranking	Accuracy	Ranking	Accuracy	Ranking	Accuracy	Ranking	Accuracy	Ranking	Accuracy	Ranking	Accuracy	Ranking
Supervised	83.3%	1	72.4%	1	9.9%	14	43.1%	10	54.4%	14	91.5%	1	57.2%	1
SeLA-V2	79.4%	2	71.5%	5	10.4%	13	41.4%	14	54.5%	12	74.6	8	50.7%	2
DeepCluster-V2	79.3%	3	72.3%	3	11.2%	11	41.9%	12	55%	11	82.3%	2	49.9%	3
SwAV	79.2%	4	72.4%	2	10.4%	12	41.9%	13	54.5%	13	81.3%	3	49%	5
InfoMin	78.6%	5	69.8%	10	12.8%	6	43.7%	9	56.3%	4	80%	6	49.8%	4
MoCo-V2	77.8%	6	70.5%	9	14%	3	45.7%	2	57.4%	2	80.6%	4	47.7%	6
BYOL	76.2%	7	71%	7	11.8%	10	44.1%	7	55.5%	10	77.8%	7	46.4%	7
SimCLR-V2	75.8%	8	72.2%	4	12.8%	7	44.5%	5	56.3%	5	72.8%	9	45.4%	9
SimCLR-V1	75%	9	70.7%	8	12.3%	8	45.2%	3	55.8%	8	69.8%	10	42.2%	10
PCL-V2	74.9%	10	69.7%	11	12.3%	9	42.6%	11	55.6%	9	80.3%	5	46.0%	8
PCL-V1	71.3%	11	65.0%	14	9%	15	39.5%	15	53.4%	15	64.1%	12	40%	11
PIRL	67.2%	12	66.1%	13	13.1%	5	43.9%	8	56.2%	6	64.5%	11	19.6%	13
MoCo-V1	66.1%	13	65%	15	13.4%	4	44.2%	6	56.1%	7	63.1%	13	21.1%	12
InsDis	64.7%	14	66.2%	12	14.1%	2	45%	4	56.8%	3	62.7%	14	18%	14
Untrained	20.4%	15	71.2%	6	20.1%	1	49.7%	1	66.7%	1	12%	15	1.2%	15

Table 5.2: **Methods performance in the different categories**. The accuracy in this table is the average test accuracy across the tasks from the category*.
* We display the values rounded to three significant figures, but for breaking ties in the rankings we consider additional precision.

We make four **overall observations** which are dissected in the following subsections:

1. In every category, one of the baselines, either the supervised or untrained one, outperforms all the unsupervised methods (see Section 5.3.1).

2. A method doing contrastive learning with prototypes (e.g. DeepCluster-V2, SeLA-V2) is the best-performing in every high-level category (semantic, scarcity, and robustness), but have a degraded performance in some lower-level ones (low-level, luminosity, and random networks); the opposite is true for some methods doing contrastive learning only between instances (e.g. MoCo-V2, InsDis) (see Section 5.3.2).

3. There is not a single representation that outperforms all others in every category, but there are some strictly dominated models; for instance, PCL-V2 outperforms PCL-V1 in *all* categories; similarly, MoCo-V2 outperforms MoCo-V1 and PIRL (see Section 5.3.3).

4. Data scarcity and noise affect unsupervised models to very different degrees; for instance, SeLA-V2 is the best unsupervised model in the semantic and robustness categories, but ranks 7th in scarcity, whereas DeepClustering-V2 is among the best three methods in these three categories (Section 5.3.4).

5.3.1 Baselines

We are interested in evaluating how the unsupervised methods perform compared with the two baselines: the supervised and the untrained networks. To make this comparison, we take all the unsupervised models as a group, and for each task, we consider their highest test accuracy and average these across the task of each category. We refer to these averages of best scores as the best unsupervised model, although we stress that it does not corresponds to an actual single model. We compare these with the baselines in Figure 5.3.

The unsupervised models performance lie between the two baselines in every category. In the high-level categories (semantic, robustness, and scarcity), all the unsupervised models are outperformed by the supervised baseline. The difference in the colour category is negligible (0.1 percentage points). On the other side, the best unsupervised model outperforms the supervised baseline in all the other categories, but in turn is outperformed by the untrained network. Overall, the best unsupervised scores are closer to the supervised baseline (as revealed by the shapes in Figure 5.3)

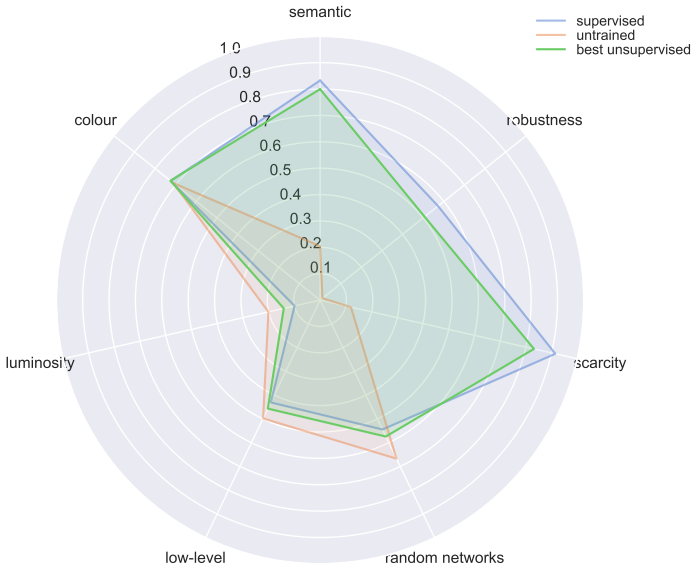

Figure 5.3: **Comparison of the best unsupervised performance with the baselines.** The radar chart measures the average test accuracy in the tasks from each category. The plot of the best unsupervised model is constructed by taking the best test accuracy achieved by an unsupervised model in each task and averaging these across the tasks from each category. The supervised baseline outperforms the unsupervised models in the high-level categories. The untrained baseline outperform them in the luminosity, low-level, and random networks categories. The differences in the colour category are negligible

5.3.2 Instance contrast vs protoype contrast

All the methods we evaluate revolve around the contrastive learning framework (See Section 2.2.1); however, an important difference is that some construct the contrastive task between instances, while others do it between instances and prototypes i.e. clusters centers (see Table 5.1). We investigate the different performance profiles of these types of models. For this, we group the representations according to whether they were learned using protoype-based or instance-based contrastive learning [3]. On each task, we take the median test accuracy of each group and average these medians across the tasks within a category. The results of this are compared in the left panel of Figure 5.4.

[3]For this comparison we ommit the PCL-V1 and PCL-V2 models which use both types of contrastive learning, but including them in either group does not affect the conclusions.

The MoCo-V1 and PCL-V1 methods are identical in many design aspects (augmentations, momentum encoders, projection head), but PCL-V1 enhances the MoCo framework with the addition of protoype-based contrastive learning [38]. Thus, a comparison between these two methods can provide additional evidence for the difference in performance from using protoypes for contrastive learning, as shown in the right panel of Figure 5.4.

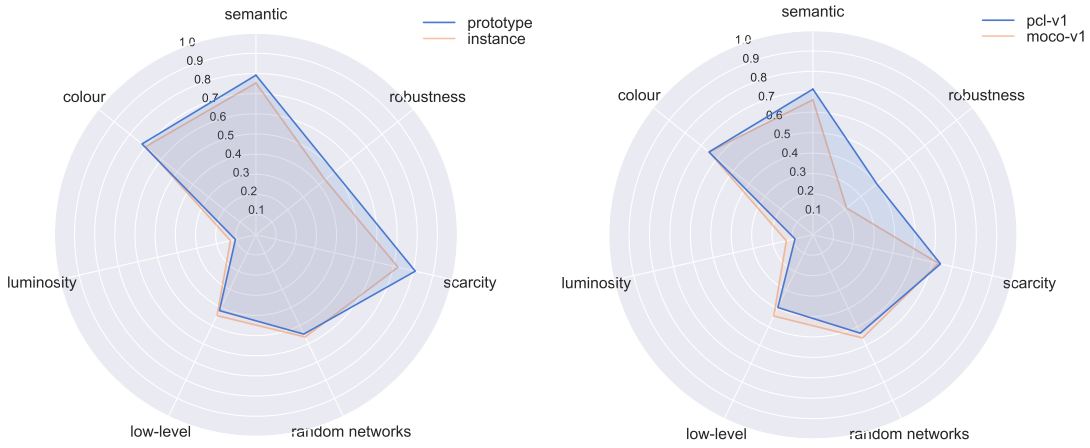

Figure 5.4: **Instance-based and prototype-based contrastive learning compared**. **Left:** Comparison of the group of methods using a contrastive task between protoypes and instances with the group using a constastive task between instances only. For each group, we take the median test accuracy in each task and average these across the tasks within a category. **Right:** Comparison of MoCo-V1 and PCL-V1. PCL-V1 enhances the MoCo-V1 framework by incorporating contrastive learning between clusters centers and instances (See Section 2.2.1). Protoype-based contrastive learning enhances performance in the high-level categories (especially in robustness), but hinders it in the luminosity, low-level, and random networks ones.

Overall, both comparison allow a common conclusion: performing protoype-based contrastive learning allows better performance on the high-level categories (semantic, robustness, and scarcity), but hinder performance in most of the others (luminosity, low-level, and random networks). The evidence is less clear regarding the colour category.

5.3.3 Projection heads and augmentation schemes

As noted in previous work (e.g. [12], [14]), using a non-linear projection head and stronger augmentation in the contrastive task can boost the performance of the methods when evaluated in the fine-grained classification task. This has encouraged various frameworks to incorporate these elements. We investigate if this claim holds in the more comprehensive evaluation of our benchmark.

In Figure 5.5, we perform four comparisons: we compare two methods enhanced exclusively with these elements (MoCo-V2 and PCL-V2) with their V1 counterpart. Additionally, we compare SimCLR-V1 with SimCLR-V2 which both use the same augmentations set and a non-linear projection head, but the V2 model has it one layer deeper, and finally, we compare the median test accuracy averaged across tasks within a category (similar to the procedure done in Section 5.3.2) between the group of models using stronger augmentation and a non-linear projection head with the group using weaker augmentation and linear projection (see Table 5.1).

Our results provide further evidence to the claim that stronger augmentation in combination with a non-linear projection head greatly enhances the performance of the methods, since we note an improved accuracy in *all* the categories for the models having these enhancements. In particular, with these enhancements, we observe significantly stronger performance in the high-level and colour categories, without a degrade in the other ones (and in some cases like PCL also improvement in them). However, an additional layer to an already non-linear head appears to make only a marginal difference as revealed by the comparison between SimCLR-V1 and V2. Our results do not allow to determine to which of the two enhancements (or both) can the performance improvement be attributed. More controlled experiments can leverage our benchmark for further investigation of this.

5.3.4 Scarcity and robustness

The tasks from the scarcity and robustness categories provide the greatest variability in performance as shown in Figure 5.1. We inspect this in more detail. In particular, we are interested in measuring the decrease in performance coming from the introduction of noise and data scarcity.

The robustness tasks are a direct adaptation of the fine-grained classification task (see Section 4.2). Thus, we use this as a baseline of performance and compute the accuracy reduction that the noise impose on each representation. For each method, we

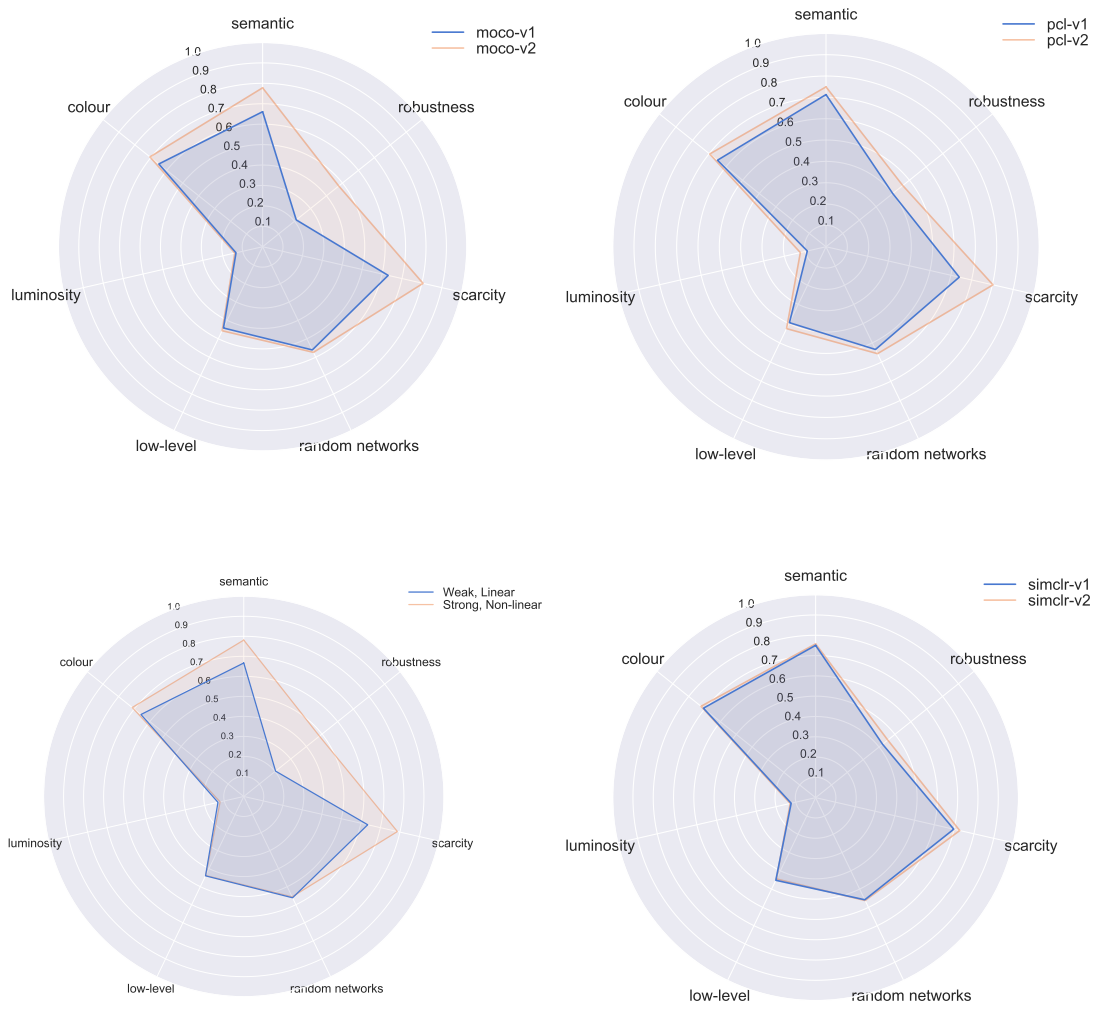

Figure 5.5: **Strong augmentation and a non-linear projection head enhance performance. Top**: the two plots on the top row show methods enhanced exclusively with a non-linear projection head and strong data augmentation. **Bottom-left**: we group the models into those using non-linear projection head and strong augmentation and those with a linear head and weak augmentation (see Table 5.1), for each group, we take the median test accuracy in each task and average them within categories. **Bottom-right**: SimCLR-V2 adds an additional hidden layer to the non-linear projection head of SimCLR-V1.

compute the difference in test accuracy between the fine-grained classification task and each of the different noise tasks, we then average these differences and consider this as a measure of accuracy reduction induced by the presence of noise in the visual data (shown in the left panel of Figure 5.6).

We emphasise that this only measures the accuracy *reduction* due to the noise. For instance, it could be that a representation has a poor performance in the fine-grained classification task and it remain equally poor (but not worse) in the robustness tasks. Then, this representation suffers from a low accuracy reduction. In general, the unsupervised methods have a smaller accuracy reduction than the supervised baseline. However, the models without a non-linear projection, with weak augmentation, and performing only instance-based contrastive learning (see Table 5.1), are more strongly affected by the presence of noise.

For assessing the accuracy reduction due to data scarcity we follow a similar approach. However, here we do not use the performance in the fine-grained classification task as the baseline, since although similar, the scarcity tasks are not exact variations of this one (they have a different number of classes and different training examples). Rather we take the N-way-10-shot tasks as the reference for each $N = 10, 20, 50$. Then, for each value of N, we take the difference between the 10-shot task and the 5-shot and 1-shot tasks respectively. We then average these two sets of differences across the values of N. We can interpret this measure as the accuracy reduction from losing 5 and 9 training samples per class respectively. The results from this are shown in the right panel of Figure 5.6.

Contrary to the robustness case, here the performance of the supervised model is much less affected by the scarcity of data. On the other side, all the unsupervised models suffer a greater accuracy reduction in both cases (reduction by 5 and 9 samples per class). For many models the accuracy reduction is quite significant in the 9 samples case, reducing their accuracy by more than 30 percentage points. This indicates that there is great room for improvement in terms of the data efficiency of the unsupervised methods representations that the common enhancements used so far (non-linear projection head, stronger augmentation, protoype-based contrastive learning) have not provided.

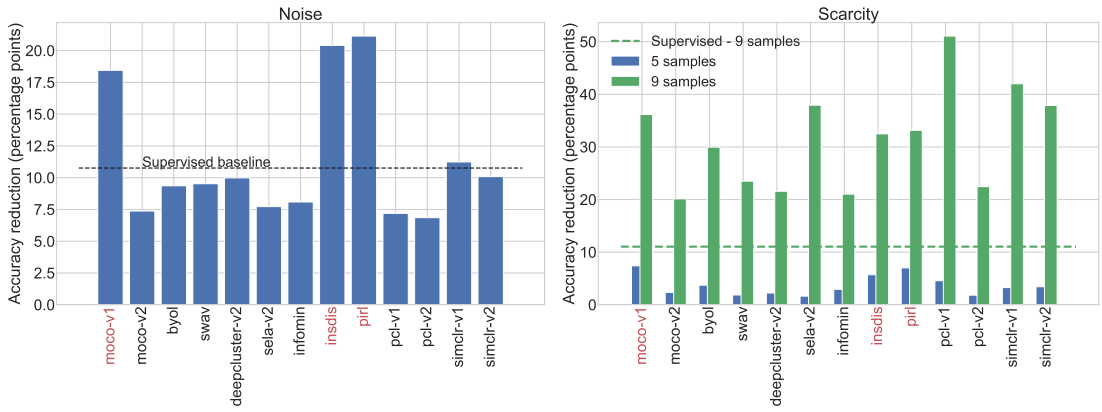

Figure 5.6: **Accuracy reduction by the introduction of noise (left) and data scarcity (right)***. A higher value implies a greater reduction in accuracy. Methods with weak augmentation, linear projection, and only instance-based contrast are shown in red (see Table 5.1). Unsupervised methods are robust to noise, but not data-efficient. Strong data augmentation, a non-linear projection head, and prototype-based contrastive learning appear to provide enhanced robustness to noise, but not data efficiency.

* The reduction in accuracy of the supervised baseline from losing 5 training samples per class is negligible and is not shown.

5.4 Conclusions

Our evaluation results highlight some relevant aspects of the benchmark we developed. Some tasks are extremely informative of differences in performance and allow to examine particular elements contributing to such differences, while others were uninformative in the sense that the performance of the different methods was uniform. Moreover, our benchmark has decorrelated tasks in terms of how they rank the methods. In this sense, there appear to be two broad groups of tasks: the higher-level group (semantic, robustness, and scarcity categories) and the lower-level group (low-level, luminosity, and random networks categories). Interestingly, the colour tasks are more correlated with the tasks from the high-level group. This is a counter-intuitive result, since colour information at arbitrary locations is usually lost while learning high-level features for semantic tasks [12], [45]. We leave for future work to discern the reasons behind this correlation.

In terms of the models' performance, we note that the unsupervised methods lie between the two baselines (supervised and untrained), although they resemble more closely the supervised one. Additionally, our results suggest that protoype-based contrastive learning enhances the representations capacity in the higher-level tasks, par-

ticularly in the robustness ones, but hinders it in the lower-level ones. This appears to provide evidence to support the claim from [38] that clustering allows to capture common features shared between instances with semantic similarity which enhances the performance in semantic clssification tasks. Our results also provide further evidence of the importance of including a non-linear projection head and strong data augmentation in contrastive learning frameworks. In particular, we find that these elements combined can enhance the performance of the methods in every category. Finally, our results indicate that these elements (protoype-based contrast, non-linear head, and strong data augmentation) enhance the representations' robustness to noise, but they do not make them more data-efficient. This is the domain in which unsupervised methods appear to be most strongly lagging behind the supervised baseline, as also suggested by the results from [17].

The results obtained serve to highlight the type of comprehensive evaluation that our benchmark enables and the insights it can reveal. We acknowledge that more controlled experiments should follow up to strengthen the evidence of our claims. We hope that future work will leverage our benchmark for this purpose.

Chapter 6

Discussion

Our work focuses on the evaluation of unsupervised representation learning methods. We perform a study on the optimization behaviour of SGD in the context of linear evaluation of representations. For the non-linearly separable case, the solution found under some hyperparameter configurations is equivalent to the one found by L-BFGS, a quasi-Newton method with strong convergence guarantees. However, for a highly linearly separable case we find that this is not the case; in this case, SGD performs regularized optimization that for some hyperparameter regimes is akin to L2 weight penalty and for others to early stopping of L-BFGS. The implications of these findings are three-fold: the choice of optimization algorithm can have a significant effect on the outcome of the evaluation; SGD can provide implicit regularization which may be confounded with the representation's capacity to enhance generalization performance; when using SGD for linear evaluation, the degree of linear separability of the evaluation task and the hyperparameters chosen must be taken into consideration. We leave for future work the scaling of our analysis to larger datasets; the challenge in this is to find a strong optimization baseline that can scale to them.

We develop a broad task-based benchmark to evaluate representation learning methods. Our benchmark allows a multi-category evaluation of the representations learned by any arbitrary method. Ours is, arguably, the most comprehensive tasks set designed for the evaluation of representations. However, it can still be extended in meaningful ways. For instance, a relevant addition could be semantic tasks with labels unrelated to the standard classes from ImageNet-1k (e.g. elements like clouds or buildings in the background); these kind of tasks could evaluate the presence of distinct high-level visual features not evaluated by the original labels and the related relabellings that we constructed. The development of this kind of tasks could borrow ideas from the

multi-labelling done by [59] or the multi-factor tasks from [56]. The benchmark could also be extended within the existing categories by producing additional similar tasks; however, one must be careful with the tradeoffs between complexity and additional information imposed by additional tasks. In particular, the benchmark should not become prohibitive to use for future work with limited computational resources.

By evaluating thirteen of the current state-of-the-art contrastive learning methods with our benchmark we found that some of its tasks are highly informative of differences in performance. On the other side, in some tasks all the methods have a very homogeneous performance. Future enhancements to the benchmark could consider replacing these tasks for more informative ones. Additionally, we observe complementarity among the tasks in the sense that they highlight different strengths and weaknesses of the methods.

The supervised and untrained baselines seem to be an adequate reference point for the tasks selected. Of particular interest is to track the gap between the unsupervised models and the supervised baseline in the high-level tasks, especially in the scarcity ones, and the degree to which the reduction of this gap increases the one with the untrained baseline in the lower-level tasks.

Finally, the results from our evaluation suggest that prototype-based contrastive learning enhances the performance in high-level tasks, especially the robustness to noise of the representations. However, this come at the cost of reducing performance in the low-level tasks. A non-linear projection head and strong data augmentation can improve the performance of the methods in *all* the tasks categories. Unsupervised methods have achieved strong robustness to noise; however, there is still much room for improvement regarding data efficiency. We acknowledge that our findings require the support of additional evidence from more controlled experiments designed specifically to discern these claims, but we showcase how future work could leverage our benchmark for this purpose.

Bibliography

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2019.

[3] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *International Conference on Learning Representations*, 2020.

[4] David Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2020.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[6] Albert S Berahas, Jorge Nocedal, and Martin Takáč. A multi-batch L-BFGS method for machine learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1063–1071, 2016.

[7] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[8] Charles G Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA journal of applied mathematics*, 6(3):222–231, 1970.

[9] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[10] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[11] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[13] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020.

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

[15] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[16] Luke Nicholas Darlow and Amos Storkey. What information does a resnet compress? *arXiv preprint arXiv:2003.06254*, 2020.

[17] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[18] Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

[19] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[20] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

[21] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020.

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

[23] 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 *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284, 2020.

[24] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[25] 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.

[26] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[28] 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.

[29] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[31] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798. PMLR, 2019.

[32] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[33] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019.

[34] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[36] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2020.

[39] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[40] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[41] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[42] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

[43] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[45] Massimiliano Patacchiola and Amos Storkey. Self-supervised relational reasoning for representation learning. *Advances in Neural Information Processing Systems*, 2020.

[46] C Ridpath and W Chisholm. Techniques for accessibility evaluation and repair tools, w3c working draft, 26 april 2000, 2009.

[47] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[49] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[50] Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *International Conference on Machine Learning*, pages 9058–9067. PMLR, 2020.

[51] Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2020.

[52] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[53] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[54] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681, 2019.

[55] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[56] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 2020.

[57] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and vision computing*, 16(2):75–87, 1998.

[58] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[59] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2340–2350, 2021.

[60] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

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

Appendix A

Additional results from the optimization study

In this Appendix we provide additional results of the optimization study for linear evaluation of representations from Chapter 3.

A.1 Similarity metrics to compare optimizers' solutions

Training loss. In Section 3.2, we argue that the difference in test losses from Equation 3.2 is a sensible metric to determine the equivalence of the models learned with the different optimizers. Here we report additional similarity metrics as a robustness check for our results.

In particular, we compare the solutions from SGD with the different hyperparameters configurations and the one of L-BFGS in the tasks described in Section 3.1.

In Figure A.1, we show the difference in training loss (Equation 3.1) between the parameters learned with different configurations of SGD and L-BFGS. In both cases, we observe that for some hyperparameter values, the difference is negligible. However, as argued before, this is not sufficient to conclude that the models learned are equivalent, especially in the linearly separable case, since there may be distinct parameters values yielding an arbitrary small loss in the training set.

Cosine similarity. Of particular interest is to assess the equivalence of the parameters learned; however, as argued in Chapter 3 an element-wise comparison is not meaningful, and in particular, the magnitude of the parameters is not determinant in the logistic regression model. For this reason, we leverage the cosine similarity from Equation 2.1

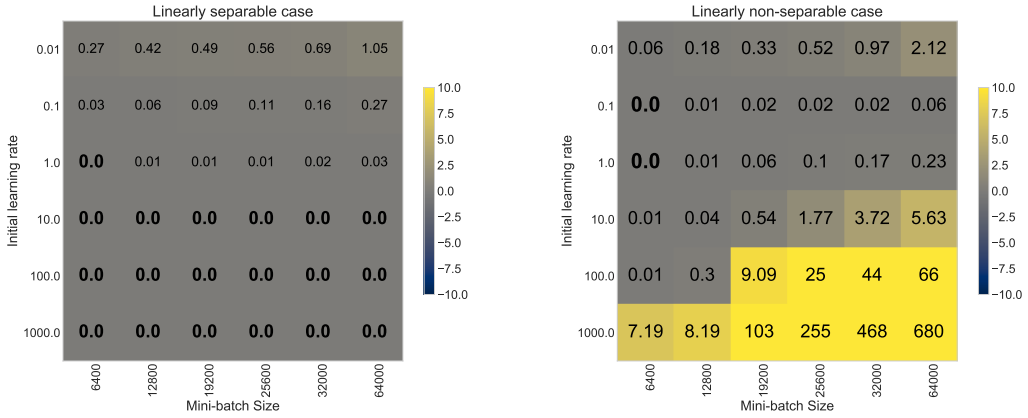

Figure A.1: **Difference between the L-BFGS and SGD train loss (Equation 3.1) for various hyperparameters configurations.** Each value* is the difference: $J_{sgd} - J_{lbfgs}$ where the loss is averaged over the training set.

* Values were rounded to three significant figures

which only considers similarity in direction to compare the weight matrices learned with each optimizer. In particular, we consider two weight matrices, one learned with SGD, W_{sgd}, and one with L-BFGS, W_{lbfgs}. Each row of these matrices serves to discriminate a given class from the others in the model learned by each optimizer (see Section 3.1). Thus, as a way to compare the predictive equivalence between the two models we take the cosine similarity between each of the corresponding rows of W_{sgd} and W_{lbfgs}. This yields C_t similarity values (one for each class), we average this and report them in Figure A.2.

This metric supports the conclusions from Chapter 3: SGD arrives at more similar solutions to L-BFGS in the linearly non-separable case. In particular, we note that the SGD configuration that matched the test loss of L-BFGS (initial learning rate of 0.1 and mini-batch size of 6400) also has a very high average cosine similarity with respect to the L-BFGS weights. The fact that this similarity is 0.99 instead of exactly 1 can be attributed to the fact that we are reporting a summary statistic (the average). We note that the distribution of the differences is skewed in the sense that most of the weight vectors have a similarity of 1 and a few individual weights with lower similarity skew the average. However, this high average allows to conclude that the parameters indeed are equivalent.

Figure A.2: **Average cosine similarity between the weight vectors of each class (rows of W) learned with SGD and L-BFGS** A higher value indicates higher similarity. The highest possible value is 1.

Prediction similarity. A more direct metric to assess whether the models are equivalent in terms of their predictions is to simply compare their predictions in the test set. We inspect the proportion of those predictions coincide between the models learned with the different optimizers. This is shown in Figure A.3. These also support the conclusions from Chapter 3, but we stress that the test loss difference from Equation 3.2 is a more comprehensive measure since not only considers the predicted class, but also the probability associated.

Mistakes similarity. An issue with only comparing the predictions is that in general models that fit the data well (e.g. in the linearly separable case) can have a high accuracy also in the test set. Thus, they will make the same correct predictions. However, an alternative is to consider the mistakes made, since the two models making the same mistakes can provide stronger evidence of equivalence. We assess this using a metric which we refer to as mistakes similarity, for shortness. This metric indicates the number of matching incorrect predictions between the models learned with the different optimizers as a proportion of the number of mistakes made by the model fitted with L-BFGS. The results of this comparison are shown in Figure **??**. These also support the main conclusion from Chapter 3

Overall the alternative similarity metrics considered support the main conclusion from Chapter 3: in the linearly non-separable case, SGD with an initial learning rate of

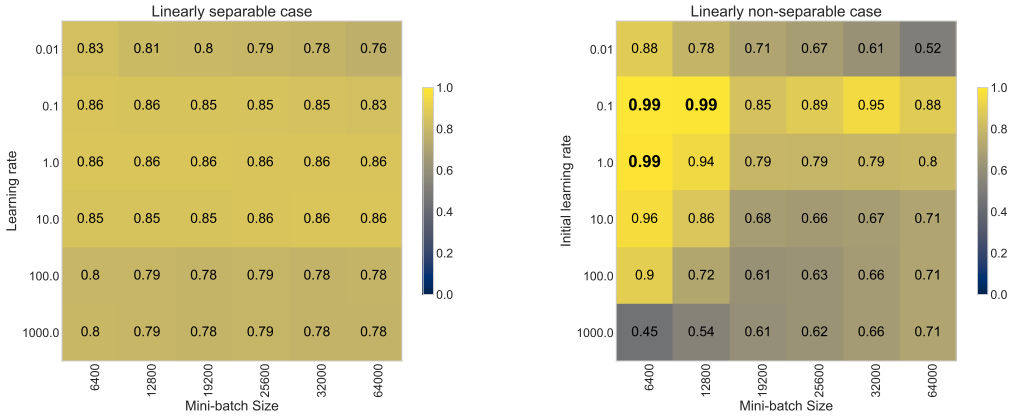

Figure A.3: **Prediction similarity: the proportion of coinciding predictions in the test set between the model learned with SGD and L-BFGS** A higher value indicates higher prediction coincidence. The highest possible value is 1.

0.1 and a mini-batch size of 6400 can reproduce the solution of L-BFGS. In the linearly separable case, this does not happen for any of the hyperparameters configurations considered.

A.2 SGD and implicit regularization in the non-linearly separable case

In Section 3.4, we analysed the implicit regularization of SGD in the linearly separable case. In this Appendix, we report the results of the same comparison in the non-linearly separable case, as shown in Figure A.5.

In this case, for specific hyperparameter values (e.g. initial learning rate of 10 with a mini-batch size of 6400 and initial learning rate of 0.1 with a mini-batch size of 32000) have a very similar loss than weight decay with $\lambda = 0.0001$. There are not close matches with the early milestones of L-BFGS that we considered. However, in this case one must be cautious with the conclusions drawn, since in this case the effect of the explicit regularization are not very significant, since there is not overfitting in the data given that the linear model cannot fit it very well in the first place (the data is linearly non-separable). Thus, it is not meaningful to say that SGD provides implicit regularization in this case, since the explicitly regularized solution is not very

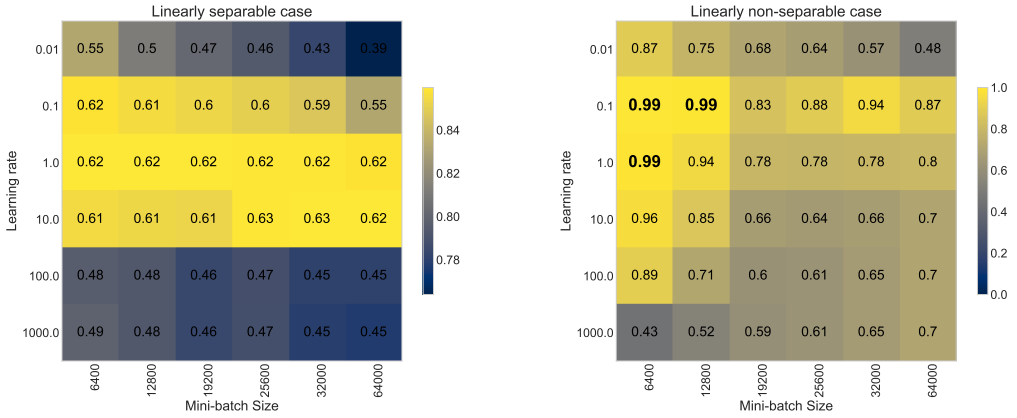

Figure A.4: **Mistakes similarity: the number of matching mistaken predictions in the test set between the model learned with SGD and L-BFGS as a proportion of the number of mistakes made by L-BFGS** A higher value indicates higher prediction coincidence. The highest possible value is 1.

distinct than the unregularized one. For reference, the test loss of L-BFGS without regularization in this case is close to 1.44.

A.3 SGD without learning rate schedule in the linearly separable case

All the results reported in Chapter 3 considered a multi-step schedule for the learning rate. However, our results from the experiments without using a schedule in the linearly separable case allow for similar conclusions. In particular, as shown in Figure A.6, without a learning rate schedule, in the linearly separable case, SGD cannot reproduce the L-BFGS solution for any hyperparameter configurations among the ones we tested.

Additionally, as shown in Figure A.7, even without a learning rate schedule, SGD still provides implicit regularization similar to L2 weight decay for some hyperparameters and to early stopping for others.

Overall, even without a learning rate schedule, the conclusions for the linearly separable case in Chapter 3 hold.

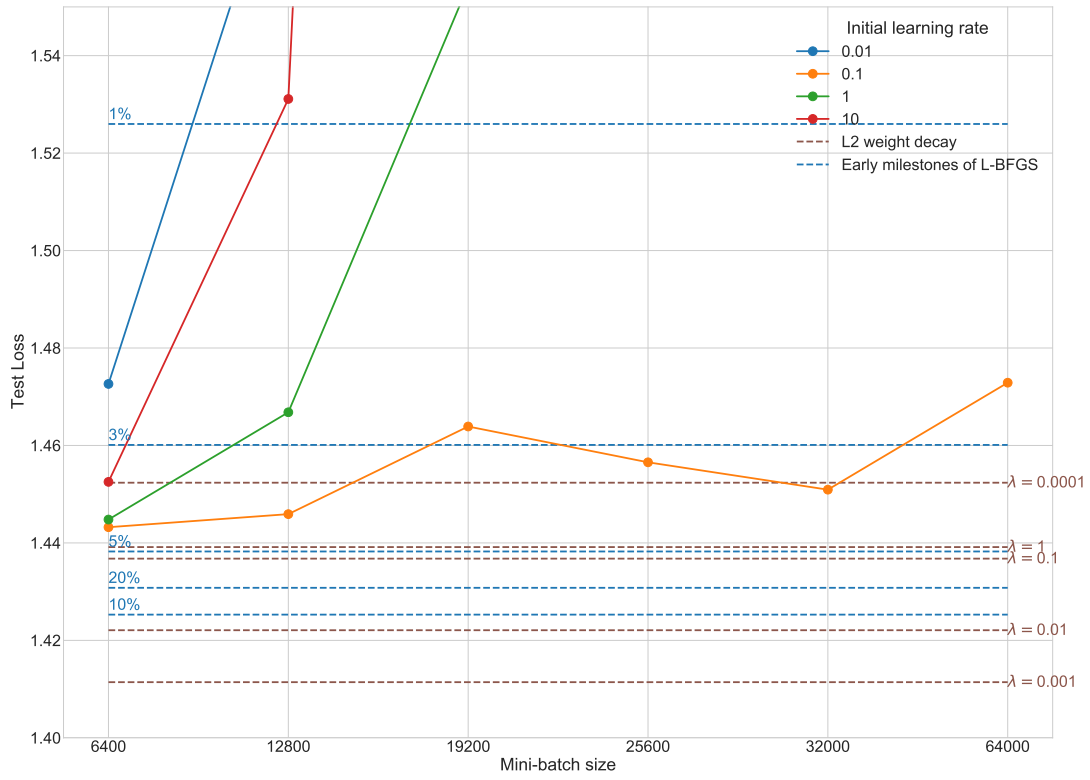

Figure A.5: **Comparison of the (unregularized) loss in the test set between SGD and regularized solutions of L-BFGS for the linearly non-separable case.*** All the specifications are identical as in Figure 3.2. For reference the test loss obtained with L-BFGS without regularization is close to 1.44.

***** For clarity in the chart, we ommit the values of λ and the early L-BFGS milestones which are not relevant for comparison (have significantly different test loss than any SGD configuration).

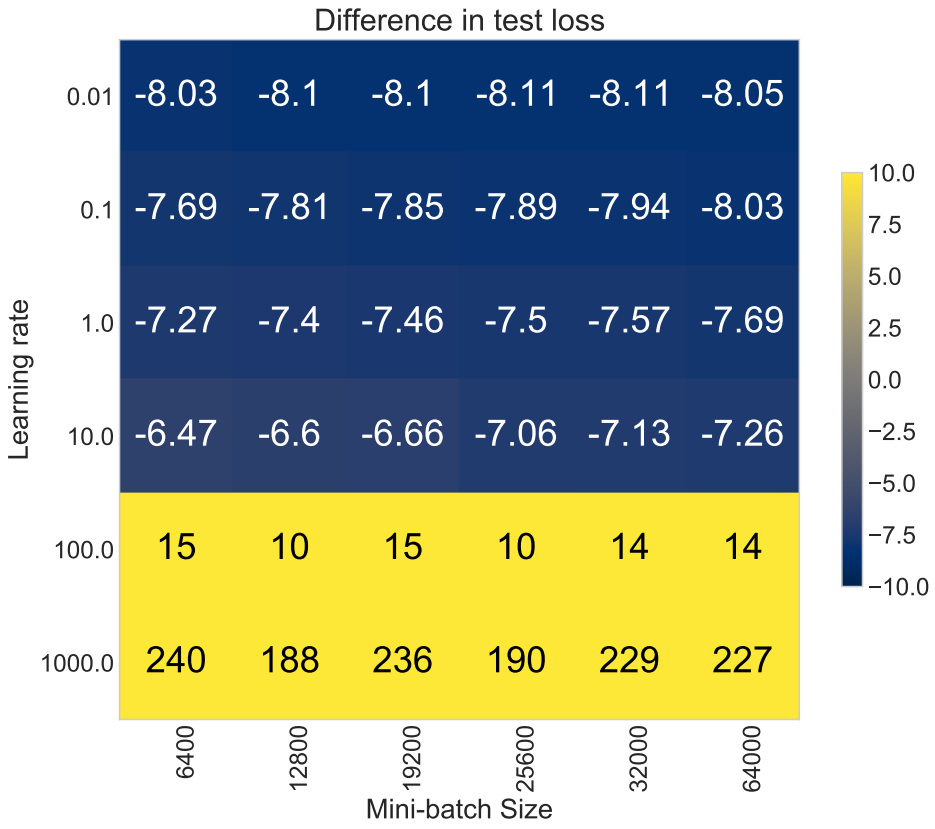

Figure A.6: **Difference in the test loss between L-BFGS and SGD, without a learning rate schedule, for various hyperparameters configurations in the linearly separable case.** Each value* is the difference: $J_{sgd} - J_{lbfgs}$ where the loss is averaged over the test set (See Equation 3.2). The specifications are the same as in the left panels of Figure 3.1, except that here we do not use a schedule for the learning rate.

* Values were rounded to three significant figures

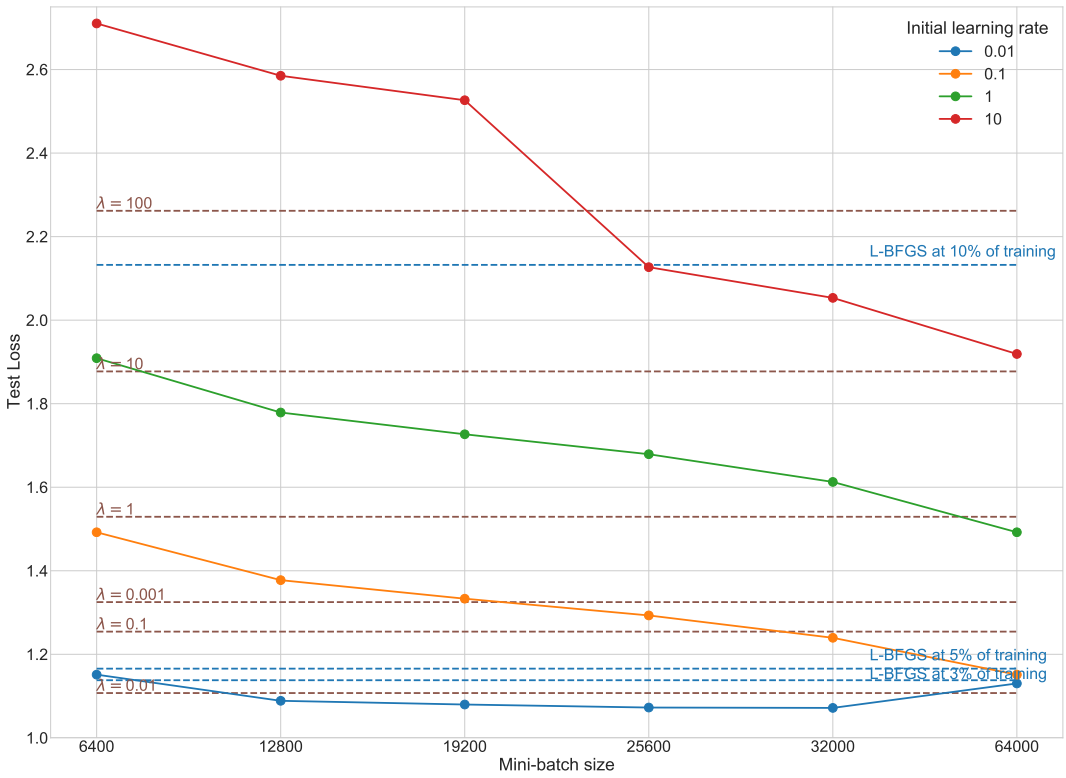

Figure A.7: **Comparison of the (unregularized) loss in the test set between SGD without a learning rate schedule and regularized solutions of L-BFGS for the linearly separable case.*** All the specifications are identical as in Figure 3.2. For reference the test loss obtained with L-BFGS without regularization is 9.18

***** For clarity in the chart, we ommit the values of λ and the early L-BFGS milestones which are not relevant for comparison (have significantly different test loss than any SGD configuration).

Appendix B

Implementation details and class distribution of tasks

In this Appendix, we provide further implementation details of the different tasks composing the benchmark. Additionally, we present the class distribution of each task.

B.1 Tasks details

Supervised Tasks:

1. Fine-grained Classification. The standard image classification task of the the ILSVRC 2012[35].

2. Coarse-grained Classification. An image classification tasks for which the classes were relabelled into one of 25 super classes (hypernyms) of the original classes in the fine-grained classification task. If some original class was not a subclass of the 25 superclasses considered all of its samples were discarded for this task. Some examples of superclasses are: dog, musical instrument, furniture, fruit, car.

3. Parts Classification. An image classification task for which the classes were relabelled into one of six new classes constituting a part (meronym) of the original object. The six distinct parts were chosen so that they are mutually exclusive and objects which do not have them were discarded. Some examples of the new parts classes are: handle, piano keyboard, shelf.

Colour Tasks:

1. Image colour. Predicting the dominant colour of the image defined as the channel (R,G, or B) for which the average pixel intensity of the whole image is highest.

2. central/corner/random pixel colour. 3-way classification tasks consisting in predicting the dominant colour of the centre, one of the four corner pixels (chosen at random), and a random pixel respectively. The dominant colour is the channel (R,G, or B) for which the pixel intensity at the given location is highest.

Luminosity Tasks:

1. central/corner/random pixel luminosity. 256-way classification consisting in predicting the (integer) luminosity value of the centre, one of the four corner pixels (chosen at random), and a random pixel respectively. The luminosity was computed with the perceived luminosity equation: $L = 0.299R + 0.587G + 0.114B$ [46].

2. Brightest region. Predict the region of an image with the greatest average luminosity (computed as above). In particular, five equally-sized disjoint regions were considered: the central crop and the four corners.

Low-level Tasks

1. Regions with corners. Predict which of the five regions of the image mentioned above (the centre and corner crops) has corner peaks (detected with the FAST algorithm [57]). In this way, there are $2^5 = 32$ classes corresponding to all the combinations of regions with corners.

2. Region with most edges. Predict which of the five regions (centre and corners) has the greatest number of edges (detected with the Canny algorithm [9]).

Random Neural Networks tasks.

1. Gaussian networks. The targets for these tasks are generated by a MLP one hidden layer of 100 hidden units, a batch normalization layer, and 10 output units. The target label for each example is simply the index of the output unit with the maximal value for that input. The weights were initialized from a Gaussian distribution with zero mean and standard deviation of 1, 1.5, and 2 respectively for each task. For the standard normal initialisation we considered also a task with 100 classes (output units).

Scarcity Tasks:

1. N-way-K-shot tasks. Image classification tasks with N classes with K samples chosen at random from the ImageNet dataset [48]. We considered the nine combinations of $N = 10, 20, 50$ and $K = 1, 5, 10$.

Robustness Tasks:

1. Noise tasks. Variations of the fine-grained classification task but with the input images from the test set corrupted with noise for evaluation. In particular, we used three types of noise: Gaussian noise with a standard deviation of 0.04, shot/Poission noise with a rate of 250, and speckle noise with a standard deviation of 0.15. These values were taken from the least severe noise category from [29]

B.2 Class distributions

Figures B.1, B.2, B.3 show the class distributions of the training and test sets for the different tasks of the benchmark.

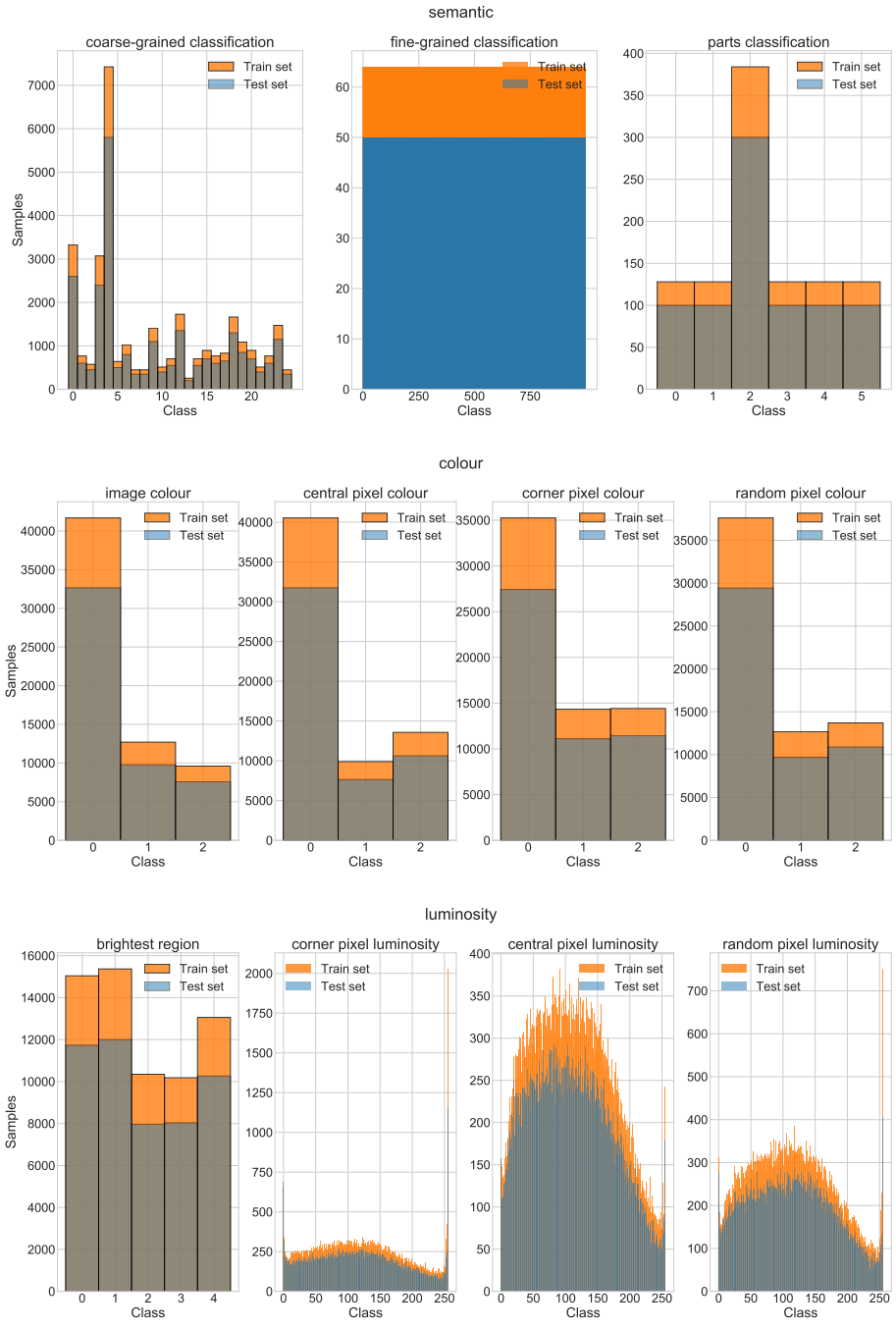

Figure B.1: **Class distributions of the training and test sets for the tasks of the semantic, colour, and luminosity categories.**

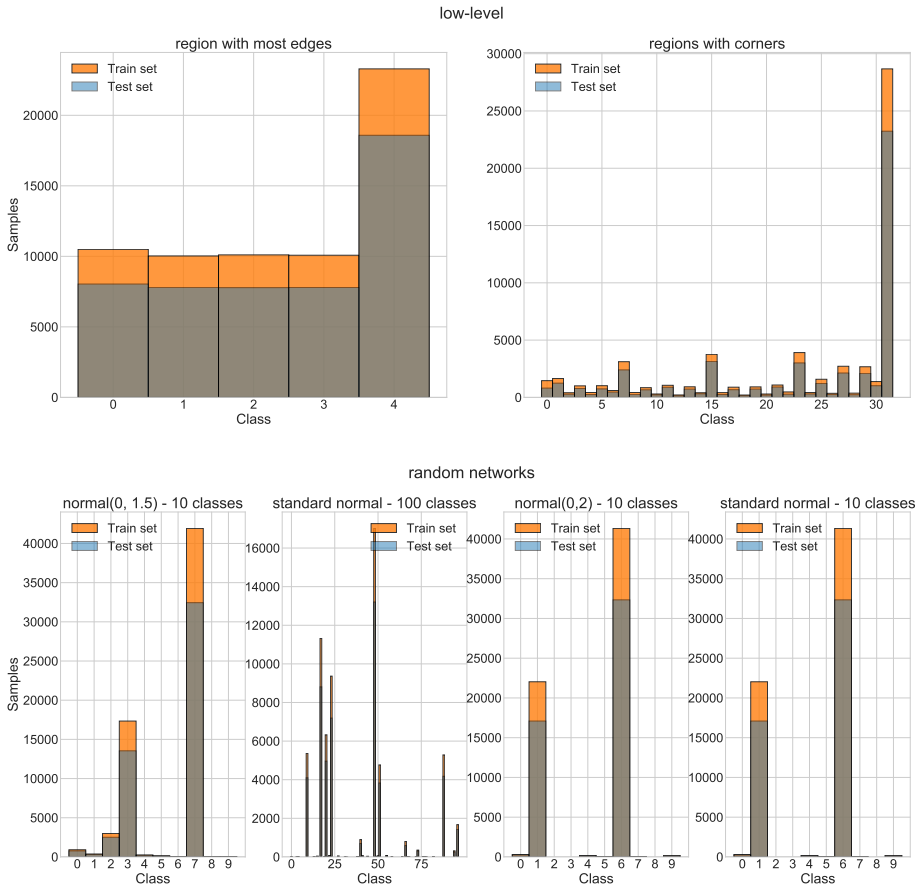

Figure B.2: **Class distributions of the training and test sets for the tasks of the low-level and random networks.**

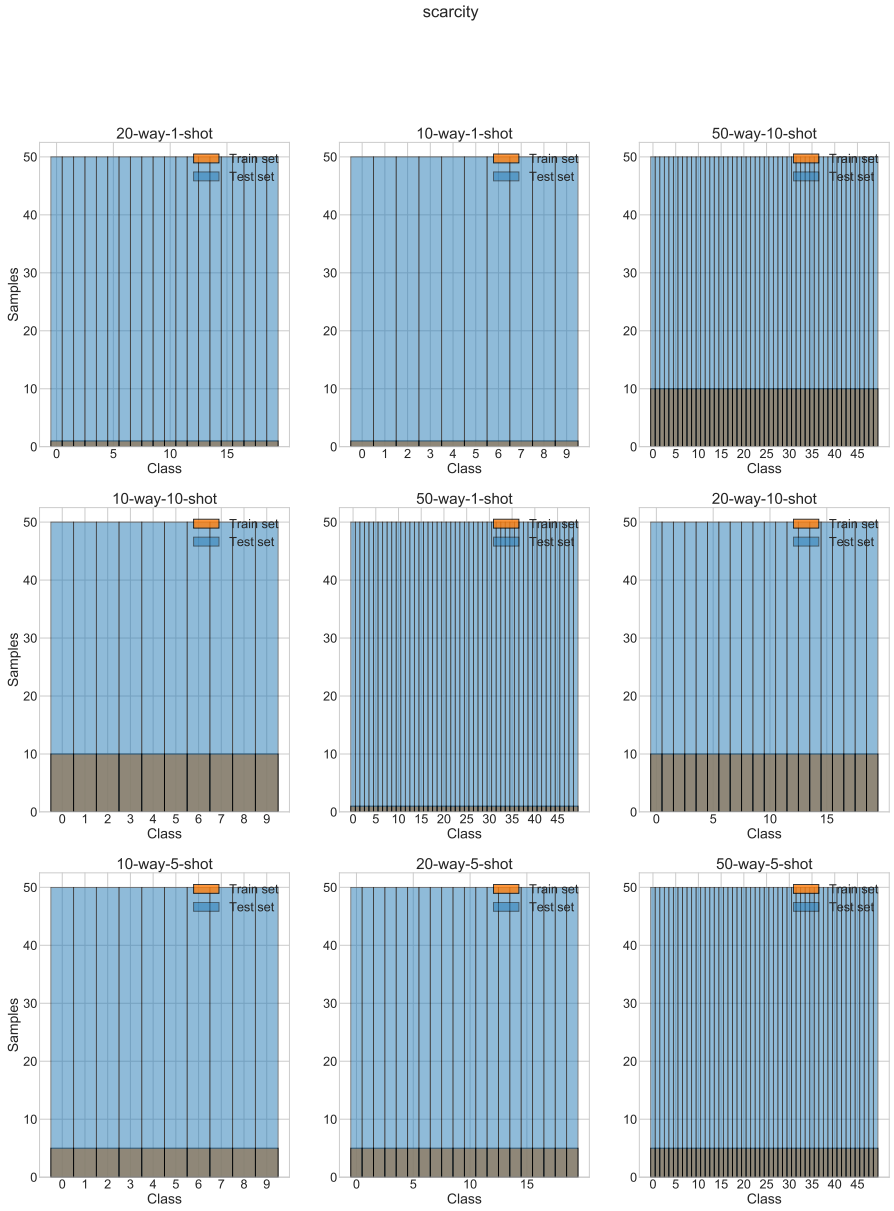

Figure B.3: **Class distributions of the training and test sets for the tasks of the scarcity.** The class distributions of the robustness tasks are not displayed, since these are identical to the fine-grained classification task.

Appendix C

Evaluation results at the task level

Tables C.1, C.2, C.3, C.4, C.5, C.6, and C.7 show the evaluation results for each task of the different categories.

Model	**fine-grained classification**	**coarse-grained classification**	**parts classification**
Supervised	67.95%	88.82%	93.00%
Untrained	1.22%	27.78%	32.25%
MoCo-V1	39.51%	76.90%	81.88%
MoCo-V2	55.07%	86.72%	91.62%
BYOL	55.72%	84.16%	88.62%
SwAV	58.50%	87.37%	91.62%
DeepCluster-V2	59.86%	87.04%	91.00%
SeLA-V2	58.40%	87.54%	92.38%
InfoMin	57.91%	87.63%	90.25%
InsDis	38.37%	74.83%	80.88%
PIRL	40.71%	77.60%	83.38%
PCL-V1	47.18%	82.23%	84.50%
PCL-V2	52.86%	84.66%	87.25%
SimCLR-V1	53.43%	84.17%	87.50%
SimCLR-V2	55.58%	84.36%	87.38%

Table C.1: **Test accuracy in the semantic tasks**

Model	central pixel colour	corner pixel colour	random pixel colour	image colour
Supervised	70.21%	66.39%	67.65%	85.51%
Untrained	70.08%	63.66%	66.63%	84.46%
Moco-V1	64.91%	60.34%	61.86%	72.78%
Moco-V2	69.54%	64.90%	66.02%	81.68%
BYOL	70.19%	64.70%	66.23%	82.98%
SwAV	70.47%	66.75%	67.36%	84.99%
DeepCluster-V2	70.57%	66.56%	67.10%	84.86%
SeLA-V2	69.50%	66.74%	66.55%	83.08%
InfoMin	68.86%	64.35%	65.66%	80.18%
InsDis	65.87%	61.40%	62.52%	74.82%
PIRL	65.65%	61.63%	62.60%	74.68%
PCL-V1	65.74%	59.66%	62.21%	72.54%
PCL-V2	68.26%	64.82%	65.30%	80.45%
SimCLR-V1	69.93%	64.17%	65.97%	82.67%
SimCLR-V2	71.22%	66.22%	66.85%	84.52%

Table C.2: **Test accuracy in the colour tasks**

Model	central pixel luminosity	corner pixel luminosity	random pixel luminosity	brightest region
Supervised	0.71%	2.00%	0.78%	36.19%
Untrained	1.18%	1.98%	0.94%	76.29%
MoCo-V1	0.85%	2.10%	0.81%	49.65%
MoCo-V2	0.81%	2.10%	0.78%	52.19%
BYOL	0.78%	1.89%	0.76%	43.82%
SwAV	0.72%	2.37%	0.92%	37.75%
DeepCluster-V2	0.81%	2.29%	0.81%	40.81%
SeLA-V2	0.83%	2.24%	0.89%	37.64%
InfoMin	0.85%	1.86%	0.80%	47.88%
InsDis	0.86%	2.29%	0.86%	52.29%
PIRL	0.89%	2.21%	0.91%	48.38%
PCL-V1	0.70%	1.45%	0.73%	33.08%
PCL-V2	0.72%	2.01%	0.81%	45.82%
SimCLR-V1	0.87%	2.17%	0.90%	45.45%
SimCLR-V2	0.87%	2.13%	0.88%	47.47%

Table C.3: **Test accuracy in the luminosity tasks**

Model	**regions with corners**	**region with most edges**
Supervised	45.94%	40.17%
Untrained	47.82%	51.76%
MoCo-V1	45.76%	42.56%
MoCo-V2	47.03%	44.28%
BYOL	45.42%	42.79%
SwAV	44.40%	39.42%
DeepCluster-V2	44.05%	39.80%
SeLA-V2	43.80%	38.91%
InfoMin	46.28%	41.08%
InsDis	46.47%	43.46%
PIRL	45.80%	41.91%
PCL-V1	41.97%	37.03%
PCL-V2	44.11%	41.10%
SimCLR-V1	46.54%	43.82%
SimCLR-V2	46.03%	42.92%

Table C.4: **Test accuracy in the low-level tasks**

Model	**standard normal - 10 classes**	**standard normal - 100 classes**	**normal(0, 1.5) - 10 classes**	**normal(0,2) - 10 classes**
Supervised	63.06%	28.83%	62.85%	63.06%
Untrained	74.94%	44.67%	72.30%	74.94%
MoCo-V1	65.02%	30.64%	63.61%	65.02%
MoCo-V2	66.63%	32.27%	64.03%	66.63%
BYOL	63.96%	30.87%	63.03%	63.96%
SwAV	63.08%	29.40%	62.60%	63.08%
DeepCluster-V2	63.61%	29.66%	63.11%	63.61%
SeLA-V2	63.31%	29.07%	62.62%	63.31%
InfoMin	65.10%	31.43%	63.59%	65.10%
InsDis	65.62%	31.76%	64.10%	65.62%
PIRL	64.96%	31.33%	63.74%	64.96%
PCL-V1	62.51%	26.28%	62.49%	62.51%
PCL-V2	64.44%	30.42%	63.21%	64.44%
SimCLR-V1	64.21%	31.52%	63.42%	64.21%
SimCLR-V2	65.04%	31.36%	63.60%	65.04%

Table C.5: **Test accuracy in the random networks tasks**

Model	10-way-1-shot	10-way-5-shot	10-way-10-shot	20-way-1-shot	20-way-5-shot	20-way-10-shot	50-way-1-shot	50-way-5-shot	50-way-10-shot
Supervised	89.40%	97.40%	96.80%	84.90%	96.80%	96.40%	78.08%	91.08%	92.32%
Untrained	17.00%	18.40%	23.00%	8.10%	11.10%	13.50%	4.04%	5.48%	7.36%
MoCo-V1	48.20%	79.40%	86.00%	41.60%	74.80%	81.60%	34.44%	56.36%	65.16%
MoCo-V2	76.40%	92.20%	92.80%	68.40%	89.40%	91.80%	59.00%	75.56%	79.60%
BYOL	63.80%	90.60%	93.00%	60.90%	88.90%	92.40%	52.56%	76.48%	81.76%
SwAV	75.40%	92.60%	93.40%	67.60%	92.40%	93.60%	55.72%	78.64%	82.24%
DeepCluster-V2	80.40%	90.80%	92.40%	67.70%	91.90%	93.60%	57.44%	80.88%	84.28%
SeLA-V2	38.20%	92.00%	91.20%	55.60%	89.20%	91.40%	55.72%	77.24%	80.76%
InfoMin	70.00%	90.40%	91.80%	69.30%	87.70%	89.90%	61.44%	76.92%	82.12%
InsDis	50.40%	82.40%	85.40%	46.30%	72.30%	78.00%	32.04%	54.36%	62.80%
PIRL	53.20%	81.60%	87.20%	45.90%	74.50%	81.30%	34.96%	56.44%	65.08%
PCL-V1	25.40%	76.20%	84.00%	26.40%	86.00%	88.40%	42.80%	71.92%	75.44%
PCL-V2	70.40%	92.80%	92.60%	67.20%	89.40%	91.80%	60.16%	77.44%	80.72%
SimCLR-V1	38.60%	88.40%	90.40%	43.90%	85.20%	87.20%	46.20%	71.32%	77.12%
SimCLR-V2	53.00%	90.20%	92.00%	48.40%	86.80%	90.00%	44.80%	72.56%	77.84%

Table C.6: **Test accuracy in the scarcity tasks**

Model	Gaussian noise	shot noise	speckle noise
Supervised	61.11%	60.45%	50.01%
Untrained	1.19%	1.21%	1.07%
MoCo-V1	24.72%	24.14%	14.33%
MoCo-V2	50.23%	49.87%	42.98%
BYOL	49.97%	49.40%	39.71%
SwAV	52.30%	51.74%	42.90%
DeepCluster-V2	53.40%	52.85%	43.38%
SeLA-V2	53.40%	53.10%	45.51%
InfoMin	52.67%	52.33%	44.48%
InsDis	21.87%	20.64%	11.36%
PIRL	24.06%	22.68%	11.96%
PCL-V1	42.94%	42.22%	34.81%
PCL-V2	47.74%	48.18%	42.10%
SimCLR-V1	46.40%	45.66%	34.54%
SimCLR-V2	48.30%	48.37%	39.82%

Table C.7: **Test accuracy in the scarcity tasks**