TOWARDS A LABEL-FREE AND REPRESENTATION-BASED METRIC FOR EVALUATING MACHINE LEARNING MODELS

by

Isaac Xu

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2022

Dedicated to the memory of my great-grandmother Tian Yong Di.

Table of Contents

List of Tables

List of Figures

Abstract

In this work, we explore the viability of proposed label-free metrics to evaluate models. We begin by examining the effect on linear probe accuracy which different viable label schemes on an identical dataset may cause. We show that in a toy setting, a notion of "complexity" for distinguishing classes can have predictive capabilities for anticipating relative "difficulty" a population of models may encounter for a comparison between classification tasks. In establishing these arbitrary relative differences in valid formulations for an evaluation task, we justify the search for a label scheme independent means to evaluate learning. To this end, we examine label-free clustering-based metrics and entropy on representational spaces at progressive milestones during self-supervised learning and on pre-trained representational spaces. While clustering-based metrics show mixed success, entropy may be viable for monitoring learning and cross-architectural comparisons, despite displaying instability in early training and showing differing trends for certain learning methodologies.

Acknowledgements

I wish to acknowledge my supervisor Dr. Thomas Trappenberg and Dr. Scott Lowe for the valuable insights and advice given to me which helped guide me in my studies as well as the writing of this thesis. I also wish to thank my friend Victor Kwon for the many discussions on machine learning, which helped shape my ideas and understanding of the subject.

Chapter 1

Introduction

In recent years, while training data for many computer vision and classification tasks have become more abundant, labelled data remains expensive to produce. This constraint is particularly true for specialized fields which require expertise and time to generate labels for. To take advantage of the plethora of unlabelled imagery data available, self-supervised learning (SSL) methods have become a popular area of study.

In general, SSL employs pretext tasks in order to train models in the absence of labels. Pretext tasks usually take the form of a manipulation or application of input data. Some examples in computer vision have included training the model to solve jigsaw puzzles [23] or predict image rotations [12]. Others may be generative in nature, trained to reconstruct a given image [1]. The successful completion of such pretext tasks necessitates the model encoder to learn a mapping from raw data into a representational space which carries semantic meaning. As an ideal, the dimensions of this learned representational space should represent meaningful high level features that capture the training data distribution.

Recently, instance learning has been shown to be especially effective at image classification tasks, achieving competitive performance on common benchmark datasets [6–8, 13]. Instance learning involves a comparison between two separately augmented "views" of a single source sample. The principal idea is to minimize a distance metric between the encoded representations of the same source views (positive pairs). This approach may also take into account encoded representations from differing sources, learning to increase the distance metric between representations of differing samples (negative pairs). Using negative pairs, the approach would be considered contrastive [6, 7]. While originally, contrastive learning was believed to be important for avoiding fully collapsed solutions, contemporary research in instance learning have produced methods which were able

to attain state of the art results without contrastive samples [8, 13]. Nonetheless, instance learning in general may be susceptible to a more limited form of solution collapse, known as dimensional collapse, in which models fail to take advantage of the representational dimension space [15, 17]. Despite such potential limitations, instance learning remains popular for academic research.

Typically, research on the efficacy of SSL methods is conducted on benchmark datasets with labels readily available. The most common evaluation metric to assess model quality is performed by training a single linear classifier layer on top of the SSL trained encoder, in a process known as linear probing. The accuracy obtained from this linear probe (LP) is then taken to be a measure for how successful the learning was. Another popular methods is using k-nearest-neighbour accuracy [9, 33], predicting classes for samples using the ground truth of its neighbours. In any case, all such methods are reliant upon a label scheme intended for testing. For application towards real world datasets, a label scheme can bring with it additional challenges such as the resource cost of generating labels and the biases inherent in an arbitrary label scheme. Additionally, while in experimental scenarios with tailored datasets, a test or validation dataset can be made to be of similar domain to a hypothetical downstream task, this is often not the case for many real world scenarios, where test datasets can be severely limited in domain with respect to the intended task for a model. Alternatively, in many fields, a universally agreed upon classification scheme may not exist. Instead, multiple contending and mutually incompatible classification schemes may exist, each returning different scores for a pre-trained model, rendering model evaluation contentious and obtuse. Therefore, it may be worthwhile to examine the progression of the learned representational space to quantify learning, in a manner that does not depend upon the label scheme of a validation or test task.

In this work, we began by examining the impact differing labels schemes may have on evaluating the performance of a pre-trained network, prepared through both supervised and self-supervised means. As we might imagine, the difference in difficulty for each of these testing classification tasks poses challenges for determining the general capabilities of a pre-trained model. We then attempt to derive properties pertaining to the complexity of each label scheme to explain our

observed differences in performance difficulty. Unfortunately, issues pertaining to complexity and difficulty are non-trivial and poorly defined at the moment. Here, we imagine that complexity is tied to the notion of fine-grained classification. In other words, it is a less complex task to separate and distinguish between two dissimilar objects than to distinguish between two similar objects. Difficulty can then be imagined to be a function of complexity, properties of the classifying agent and perhaps other latent variables.

Upon establishing an intuition for how such differences in label schemes can affect evaluation, we looked at methods to measure learning independent of labels. Working with the understanding that distance in representational space carries semantic meaning, with similar samples (such as those from the same class) grouping together as learning progresses, we considered a variety of clustering-based metrics. In order to conduct clustering, we passed a test dataset through the model's encoder to generate a set of embeddings. We expected that over learning, clustering techniques such as k-means will increasingly pick out clusters in these embeddings, representative of ground truth. Furthermore, these clusters should also become more distinct over training. Consequently, familiar metrics which evaluate clustering quality should produce increasingly better results.

Independent of clustering, we also measured the entropy of the embedding distribution over training. We expected that as learning progresses from a randomly initialized state, the embedding distribution will change over time from low modal and high dispersion to high modal and low dispersion. In other words, the number of modes for this representational space distribution should increase to be reflective of the number of ground truth classes (or even sub-classes) in our dataset. Similarly, the intra-cluster distance or dispersion, should decrease as the model becomes more certain of how each class is defined, drawing similar samples together. Therefore, entropy in this case, should begin at a high initial value and decrease over time.

To establish the capability of these proposed metrics, we compared the progression of each metric with respect to LP accuracy on familiar benchmark datasets. The datasets we used for representational evaluations are natural image datasets CIFAR-10 and CIFAR-100 [19]. A strong and significant correlation would indicate greater viability for the metric. We performed these measurements on

different instance learning methodologies to evaluate their effectiveness across SSL techniques. We also applied these measures to different pre-trained vision models as provided by torchvision [24], in order to evaluate their effectiveness across architectures.

In general, we found that entropy was our most successful metric, providing correlation with LP accuracy after a period of early training instability. Additionally, entropy appeared to be capable of making cross-architectural comparisons, while clustering-based approaches appear to fail in this regard. However, before it can be a viable metric, the early training instability for entropy needs to be better understood. The metric also has a positive correlation with LP accuracy for non-contrastive learning which was unexpected and contrary to its behaviour for contrastive learning methods. This phenomenon would also require additional research.

1.1 Problem definition

In consideration of our approach to establishing a potential label-free metric, we must first consider how label schemes can affect model performance in seemingly arbitrary manners. From here we progress to proposing and evaluating potential label-free metrics. The goal of this work is to provide answers for the following six questions:

1. What are potential issues a label-dependent model evaluation metric may encounter and how do different downstream label schemes affect the measured quality of a pre-trained model?

2. What insights into the complexity and difficulty of a classification task can we draw from considering the impact different label schemes have on a downstream measurement?

3. How does a representational space, as mapped by a computer vision classification encoder, change during SSL?

4. Can the change be incrementally quantified independently of the label scheme in a coherent and consistent manner, indicative of learning?

5. What measures can be used to quantify the change?

6. How universal are these measures across different learning methods and network architectures?

Chapter 2

Background

2.1 Instance learning

In this work, we examined three different instance learning methodologies: SimSiam, SimCLR, and MoCo-v2 [6–8]. These methods use strong image augmentations in order to create a pretext task. Each sample in the dataset is augmented twice independently, creating two positive views or positive pairs. The loss function is constructed to incentive the model encoder to learn a mapping which is robust to the image augmentation applied, resulting in similar views to have similar embeddings.

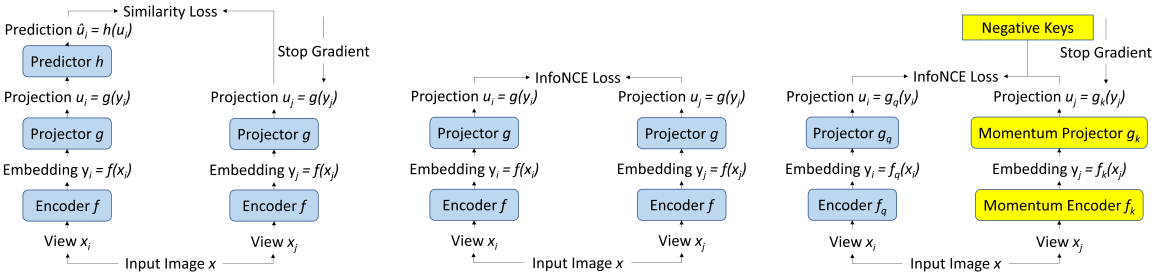
2.1.1 SimSiam

As visualized in Figure 2.1a, SimSiam is a non-contrastive instance learning method employing asymmetric Siamese networks. Given a sample x and the resulting positive pair x_i and x_j, the method feeds the views through the encoder to produce embeddings y_i and y_j. These embeddings are subsequently passed through a multi-perceptron (MLP) section referred to as a projector, generating two projections: $\mathbf{u}_i$ and $\mathbf{u}_j$. Lastly, along the i-branch, the projection is passed through another MLP section called the predictor, generating the prediction $\hat{\mathbf{u}}_i$. The objective is then to minimize a distance metric for $\hat{\mathbf{u}}_i$ and $\mathbf{u}_j$. This can be achieved by minimizing the loss function:

$$L_{i,j} = -\sqrt{\mathrm{d}_{\cos}(\mathbf{u}_i, \hat{\mathbf{u}}_j) + \mathrm{d}_{\cos}(\mathbf{u}_j, \hat{\mathbf{u}}_i)}, \tag{2.1}$$

where $\mathrm{d}_{\cos}(\cdot, \cdot)$ is the cosine similarity distance measure [8].

It is also worth noting that this process is symmetrical for i-branch and j-branch in the sense that both x_i and x_j take turns being the predictor branch, creating the two distance terms seen in the loss. This loss is only back-propagated through the predictor branch, with a stop gradient operation set for the other. The



(a) SimSiam. (b) SimCLR. (c) MoCo-v2.

Figure 2.1: Overview of SSL methodologies. For a given image, x, two views of the image (x_i and x_j) are created with randomly sampled augmentations. In SimSiam (a) and SimCLR (b), the two views are passed through the same encoder, f, and projector, g, layers; in MoCo-v2 (c), view x_j passes through a moving average model instead. For SimSiam, the loss is the similarity between the output of a predictor head on the x_i branch and the projection of x_j; whereas for SimCLR and MoCo-v2, InfoNCE is used with negative samples drawn from the batch (SimCLR) or the memory queue (MoCo). In SimSiam and MoCo-v2, a stop-gradient is applied to prevent the loss returning down the x_j branch of the network.

weights for these model components are shared across the two branches, with their asymmetrical properties intended to help mitigate solution collapse. Intuitively, we can interpret SimSiam as a pretext task which attempts to have a model predict its own projection, given perturbation to its input.

2.1.2 SimCLR

Unlike SimSiam, SimCLR, seen in Figure 2.1b, uses contrastive learning. In addition to the positive pairs x_i and x_j obtained from the same source sample x, the method takes views from other same-batch samples as negative views to be contrasted with views of x. As can be expected, the efficacy of this method is dependent on batch size, with larger batch sizes generally leading to increased performance. As before, x_i and x_j are fed through the encoder to produce their corresponding embeddings, which are once again forwarded through the projector. With the projections $\mathbf{u}_i$ and $\mathbf{u}_j$ and the number of samples in the batch as N, the loss is calculated as:

$$L_{i,j} = -\log \frac{\exp\left(\mathrm{d}_{\cos}(\mathbf{u}_i, \mathbf{u}_j)/\tau\right)}{\sum_{k=1}^{2N}(1-\delta_{ik})\exp\left(\mathrm{d}_{\cos}(\mathbf{u}_i, \mathbf{u}_k)/\tau\right)}, \tag{2.2}$$

where δ is the Kronecker delta function and τ is a temperature scaling factor [6]. This loss is also referred to as the normalized temperature-scaled cross entropy

loss [6] or as InfoNCE [30].

With the consideration of contrastive views, SimCLR is generally believed to be more resistant to solution collapse and therefore does not require the asymmetry introduced by SimSiam.

2.1.3 MoCo-v2

The last SSL method examined, MoCo-v2, seen in Figure 2.1c, uses the same InfoNCE loss as SimCLR, seen in Equation 2.2. The main differences between the methods reside in the use of a momentum encoder (also known as a key encoder f_k) for MoCo-v2 and a memory queue for negative pairs. In conjunction with this momentum encoder is the main encoder (referred to as a query encoder f_q). This query branch is the only path the loss backpropagates along. The key encoder and projector parameters are instead updated as the exponential moving average (EMA) of their query branch counterparts.

As views are passed through the key branch, they are added to a memory queue as negative pairs for future positive views. The objective is then to use InfoNCE loss to compare these positive pairs with each other and against the queue of negative views [7]. In this manner, the method employs contrastive comparisons without the need for large batches as in SimCLR.

The motivation and namesake for its branches originates from database retrieval problems, where a provided query is compared against keys to identify which key, most similar to the query, is to be returned. With this understanding, we can obtain another intuition for the MoCo-v2 task. We are essentially training a query and key mapping mechanism, where positive pairs are taught to the neural network as "correct" keys to be returned and negative pairs are the "incorrect" keys.

2.2 Dimensionality reduction

The representational space for encoders are typically high dimensional, consisting of hundreds if not thousands of learned features. Employing familiar clustering algorithms such as k-means in this space can be challenging as an increase in dimensions causes the volume encompassed by the data to grow quickly, causing the

distribution to become sparse, having consequences on nearest neighbour graphs and clustering. This phenomenon is known as the "curse of dimensionality" [2].

To accommodate clustering and other similar methods, we use a dimensionality reduction technique on the representational space embeddings. The primary technique we use is uniform manifold approximation and projection (UMAP) [20] which is believed to hold advantages in both representing global structure as well as time complexity when compared to older methods such as t-distributed stochastic neighbour embedding (t-SNE) [31]. We will attempt explain how UMAP functions qualitatively at a high level, as the precise mathematics for the technique are beyond the scope of this work.

Firstly, to form a basis for comparison, we can begin with a brief explanation of t-SNE. In essence, t-SNE works by characterizing each point in a high dimensional distribution with every other point. This characterization is in the form of a similarity score, with similarity determined by how close each compared point is from the original, distributed on a Gaussian and then normalized with respect to the scores of all other points. We end up with a matrix of $n \times n$ size with n corresponding to the number of data points, with each cell containing the similarity score between the point of the specified row with the point of the specified column. The diagonal of this matrix are the similarity scores of each point with itself, which is defined by t-SNE to be zero. This matrix gained from high-dimensional information may then be interpreted as a "ground truth" we are trying to imitate. Then, we create a randomized plot of the data points in the required lower dimensions and create another similarity matrix. The main difference here is that instead of using a Gaussian as the means to distribute our compared points for determining similarity scores, we use a t-distribution. The reason we use a t-distribution instead is that the longer tails for this kind of distribution prevents the curse of dimensionality from grouping the majority of our points together in this lower dimensional space. The learning objective is then to move the points on this randomly initialized lower dimensional plot in a fashion such that the lower dimensional similarity matrix most closely resembles that of the original ground truth similarity matrix. Each point is moved in this fashion one at a time for a single iteration, given a preset number of iterations.

The intuition and formulation for UMAP is identical to t-SNE. The main differences between the two techniques reside in how similarity scores for data points are calculated, how the lower dimensional graph is initialized and what each iteration step entails. Unlike in t-SNE, which employs a Gaussian distribution to determine the similarity scores of other data points for each data point, UMAP creates a curve such that the sum of similarity scores for the k neighbouring points comprise the majority of the value $\log_2(k)$; where k is determined as a hyperparameter for the number of nearest neighbours. Roughly speaking, we can imagine k as a tuning mechanism for determining to what degree we wish to preserve local or global structure for our data, with smaller values focusing on local structures and larger values focusing on global structures. For initializing the lower dimensional graph, t-SNE uses random initialization, whereas UMAP uses a deterministic technique known as spectral embedding [22,27], the details of which we will not explore here. Essentially, spectral embedding ensures a more advantageous starting point than random initialization by establishing a notion of which points belong to which cluster. Lastly, while t-SNE moves all points for each learning iteration, UMAP moves only a single point or a few, chosen at a random basis, at a time for each cluster, resulting in better performance for large datasets. In other words, UMAP employs stochastic gradient descent for learning the high-dimensional similarity matrix.

2.3 Entropy

Entropy in terms of information theory, can be intuitively understood as the average amount of information for a system, or more specifically in our case, a distribution. Firstly, if we imagine a density distribution referred to as X, with different partitions x comprising X, we can imagine that each partition has with it a probability $P(x)$ associated with it, that determines the likelihood of a point falling within x. The information content for this "event" is then $\log(1/P(x))$ or $-\log(P(x))$. The entropy H for X is then:

$$\mathrm{H}(X) = -\sum_{x \in X} P(x) \log(P(x)). \tag{2.3}$$

The base for the logarithm determines the units for entropy, where a base of two, results in "bits" and a base of e results in "nats".

2.4 Extrinsic measures

Extrinsic measures as used in clustering analysis refer to means of evaluating clustering quality using an external reference. Typically, such external references take the form of ground truths. Using an extrinsic measure in this case is essentially just a quantitative comparison of cluster labels obtained from a clustering algorithm with ground truth labels. However, this notion of external reference need not be ground truth and could easily be the cluster labels generated from a separate clustering attempt. In this case, we can imagine the extrinsic measure to be an quantitative evaluation of how much one clustering attempt agrees with another, or more succinctly, we can call this clustering agreement.

In this work, we use the extrinsic metric mutual information [1] for k-means generated cluster labels and ground truth to provide an expected upper bound on the effectiveness our clustering metrics may have in capturing learning. We also use mutual information between two clustering attempts as a clustering agreement metric, which we expect to improve over learning. For discrete random variables X and Y, their mutual information is defined as

$$\mathrm{I}(X;Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y) \log \left(\frac{P(x,y)}{P(x)P(y)} \right), \tag{2.4}$$

where $P(x,y)$ denotes the joint probability distribution for X and Y, while $P(x)$ and $P(y)$ denotes their respective marginal distributions.

An intuitive way of interpreting mutual information is that the metric provides a measure of how much information we can obtain about Y given X and vice versa.

For implementation, we use the *adjusted* mutual information (AMI) score provided in the scikit-learn library [25]. The AMI corrects for the chance level of mutual information which would be measured given a certain finite number of

[1] In addition to adjusted mutual information, we also looked at normalized mutual information and Rand indices. However, these measures did not differ in notable ways and as such we only present adjusted mutual information in this work.

samples. The AMI between two discrete random variables X and Y is defined as

$$\mathrm{AMI}(X;Y) = \frac{\mathrm{I}(X;Y) - \mathbb{E}[\mathrm{I}(X^*;Y^*)]}{(\mathrm{H}(X) + \mathrm{H}(Y))/2 - \mathbb{E}[\mathrm{I}(X^*;Y^*)]}, \tag{2.5}$$

where H is entropy and the expected information is taken over a hypergeometric model of X and Y.

2.5 Intrinsic measures

An intrinsic measure uses properties inherent to the clustering in order to evaluate the clustering quality. An example of such a metric which we use, is the silhouette score [26] [2].

The silhouette score considers inter-cluster and intra-cluster distances and ranges from -1 to 1, with negative scores indicating potentially incorrectly assigned samples, near zero values indicating overlapping clusters and high scores suggesting distinct and compact clusters. The silhouette score for a given sample i is defined as:

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \tag{2.6}$$

where b_i is the average inter-cluster distance between sample i and the nearest neighbouring cluster's samples, while a_i is the average intra-cluster distance from sample i to other same cluster samples. An overall silhouette score is obtained by averaging over all samples.

2.6 Related works

Deep clustering [21] is a field of study that incorporates representational clustering into the machine learning process. Generally, deep clustering combines a standard network loss with a clustering loss in order to train the network to learn a task while also learning cluster-friendly features. Such a loss typically takes the form of:

$$L = \lambda L_R + (1 - \lambda) L_C, \tag{2.7}$$

[2] As with extrinsic measures, we also looked at a variety of intrinsic measures, such as the Davies-Bouldin score and the Calinski-Harabasz score. However, once again these metrics did not differ in terms of our results in notable ways to warrant their inclusion

where the total loss is L, with L_R referring to the standard network loss (eg. reconstruction loss in the case of autoencoders), L_C being the clustering loss and λ being a weighting parameter.

The clustering loss typically is an extrinsic measure, incorporating ground truth. While we use such ground truth based measures in this work, it is primarily intended as a benchmark to compare our label-free metrics with, rather than a metric with which to aid learning. Furthermore, while a method incorporating clustering loss can be expected to produce increasingly cluster-friendly representations over learning, it is less certain that a SSL task unrelated to clustering well also achieve similar results.

Chapter 3

Label schemes and transfer learning

In this chapter, we look to justify the need for a label-free metric by examining the effects differing label schemes would have on assessing the quality of learned representations for a trained model. As such, we demonstrate the reliance of LP evaluation on seemingly arbitrary label schemes and provide insight into a core aspect of what constitutes "complexity" and consequently "difficulty" in classification tasks. We imagine "complexity" of a task to be tied to the degree of fine-tuned classification required to distinguish between two objects. Another similar potential definition is related to information theory, where "complexity" may be interpreted as the minimum number of bits of information required to perform classification [5]. One can imagine that very similar objects would require additional information to distinguish whereas semantically distinct objects would require less information. Difficulty then, would be a function of complexity and the particularities of the agent among perhaps other latent variables. Therefore, difficulty only has meaning when discussing the agent(s) for which the difficulty applies. While we would require more than just the difficulty results from a single agent to know complexity, we expect to be able to at least provide basic predictions for the complexity of a problem when examining universal or near universal performance across a variety of independent agents.

In order to begin our investigation into label schemes and their effects on evaluation, we create a toy dataset, consisting of four shapes: rectangles, squares, ellipses and circles. It is important to note that since squares and circles are regular subsets of rectangles and ellipses respectively, we consider rectangles and ellipses to be only their irregular forms. From here, we can define two classes labelled zero and one. An image of such a dataset can be seen in Figure 3.1.

A "natural" way to assign shapes to two classes may be to place rectangles and squares into class zero, while ellipses and circles are placed into class one. In this

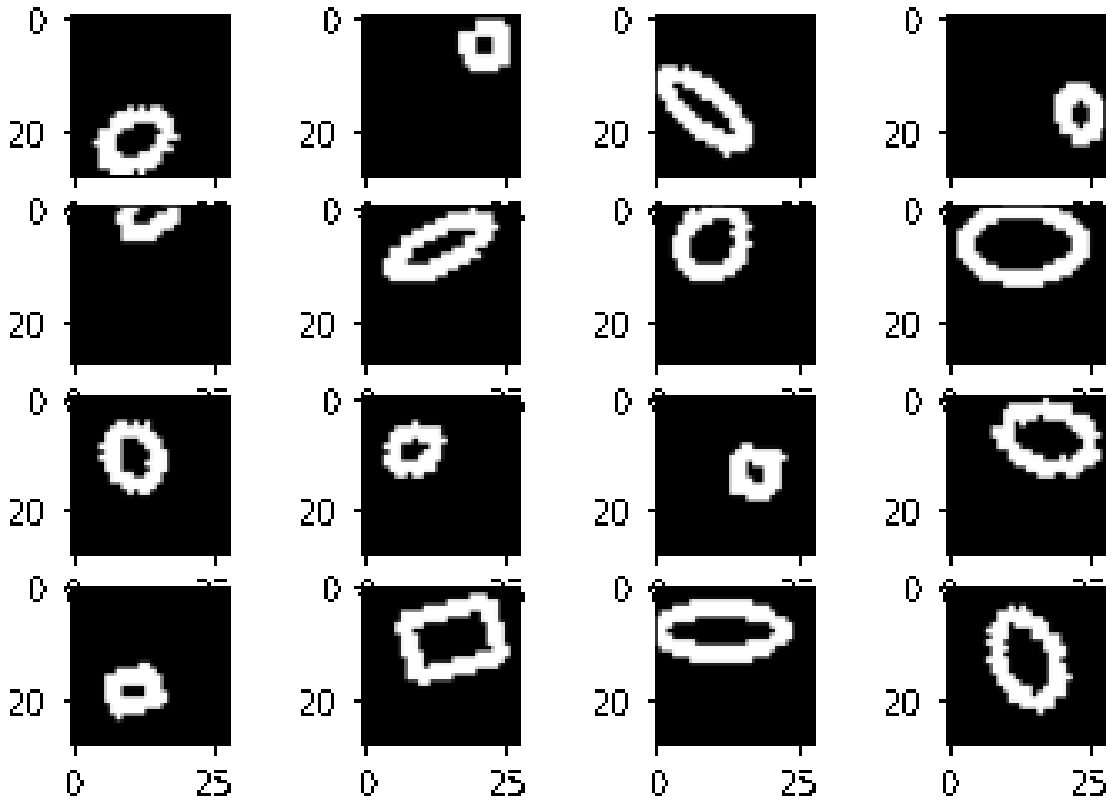


Figure 3.1: Sample images of toy dataset consisting of rectangles, squares, ellipses and circles.

scenario, we can imagine the classification task to be distinguishing between shapes with "edges" and shapes that are "round" (since the images are pixelated, circles and ellipses inevitably still have edges). Another possible classification task would be to group squares, circles and ellipses into one group and rectangles alone in the other. We can consider this task to be separating or distinguishing rectangles from the other shapes. If applying the previously established notion of distinguishing between round and edge classes, one may be tempted to label this second proposal as an "unreasonable" classification scheme. However, since the category of rectangles is not inclusive of squares, the problem may be interpreted as "distinguishing irregular rectangles from other shapes", rather than "forcing a mathematical subset of shapes to belong to a different category". In the event that the task is truly unreasonable, in that rectangles also include samples of squares, then the result would be trivial with the model performing poorly as it would be truly impossible to distinguish between certain samples for not only machines but also humans or any other potential agent. Similar to this proposed label scheme, we can imagine other tasks which distinguish squares, circles or ellipses individually. We can also imagine other classifications distinguishing between regular and irregular shapes or "inversed" scenarios where rectangles and circles belong to one class with ellipses and squares belonging to the other. The experimental process is then to test how pre-trained encoders, trained using different methods will perform with LPs under these different classification schemes. The expectation is that depending on the label scheme, the learning task will differ in complexity and difficulty for the LPs.

Although we do not yet have a reliable quantitative means to evaluate the difficulty or even complexity of a learning task, we do oftentimes have some intuition that can help guide us. Typically, when we consider a simple dataset, we assume that a valid classification task cannot be overly complex and therefore, we assume that the dataset is relatively "simple". However, one can imagine the even in our relatively trivial dataset consisting of rectangles and ellipses, we can create an arbitrary complex task. For example, while distinguishing edge and edge-less shapes may be relatively simple, distinguishing squares from irregular rectangles is more complicated since they are more semantically similar. Even more complex would be distinguishing between rectangles whose length-width ratios match a series of

provided arbitrary values from those that do not.

In general, while the observed data itself can play a role in determining task complexity, it is also crucial to consider what one may wish to do with the observed data. Furthermore, even with data of the same domain, an assessment of a trained model is inevitably tied to the biases inherent in the label scheme. Thus, when a model is assessed for quality, care must be taken not only to select a dataset of similar domain, but also a scheme consistent with the intended goal. These are the constraints and limitations we hope to demonstrate in this experiment concerning different label schemes and their effect on LPs as an evaluation metric.

3.1 Methods

In implementing this experiment, we use a modified ResNet-18 [14] backbone for our encoder architecture. Modifications are such that the first convolutional layer is reduced from a 7×7 input size to 3×3. Additionally, the first max pooling layer is removed. This setup is similar to the one used in SimCLR for smaller (low resolution) images and ResNet-50 [6]. For convenience, we will refer to this version of ResNet-18 as ResNet-18L, with "L" indicating light.

3.1.1 Dataset

To generate our dataset of 28×28 images of edged and edge-less shapes, we use the OpenCV Python library [3]. The generated edged shapes have a minimum size of five pixels, while the edge-less shapes have a minimum size of three. All shapes have a thickness of two pixels to overcome the difference in OpenCV implementation[1] of circle and ellipse. The maximum is mathematically determined such that the whole shape is visible within the image. In total, we generate 10 000 images for our training dataset with 1 000 for our test dataset. For the training images, we use random rotations as an augmentation for supervised learning, while the test images remain unaugmented. After rotation, some images may fall outside of the image border as can be seen in the first image on the second row in Figure 3.1. For SSL,

[1] When using OpenCV's circle function, the result is different from OpenCV's ellipse of equal semi-major and minor axes. However, increasing border thickness to greater than two overcomes this difference entirely.

we use the same 10 000 image training dataset. However, in addition to the rotational augmentation, we also add colour distortion and random resized cropping as suggested in SimCLR for the CIFAR-10 dataset [6]. Although these images are generated black and white, to accommodate ResNet architecture, the same pixel intensity value was broadcasted across all three RGB channels and normalized with an empirically obtained mean of 0.87 and a standard deviation of 0.33. When we generate our dataset and divide it into the number of classes, we ensure that each class has an equal proportion of the samples and each subclass also has an equal proportion of the samples within each class. This approach ensures that all tasks employ balanced datasets.

3.1.2 Training process

For pre-trained models, we look at six different ways of preparing the encoder:

1. Random initialization;
2. Self-supervised learning with SimCLR methodology;
3. Supervised learning on edge and edge-less shape classification;
4. Supervised learning on classifying all four shapes independently (irregular rectangles, squares, irregular ellipses and circles);
5. Transfer learning from pre-trained ImageNet [11] weights on ResNet-18 architecture; and
6. Transfer learning from pre-trained ImageNet weights on ResNet-18L architecture.

In the second scenario, when we were working with SimCLR, we used stochastic gradient descent (SGD) with a one-cycle learning rate scheduler [28] with cyclic momentum from 0.85 to 0.9. The peak learning rate (η) was set to 0.5, weight decay (λ) to 1×10^{-4} and InfoNCE loss temperature (τ) to 0.5. The batch size we use throughout all SSL and supervised learning is 512, as this was determined to be an effective size for SimCLR [6] and for consistency throughout trials. The SSL preparation had a total training period of 200 epochs.

Following encoder preparation, we add a single linear layer that maps from the 512-d representational space of ResNet-18 to the number of classes as required in the label scheme. We then train this LP with an Adam [18] optimizer and a one-cycle scheduler. Here, $\eta = 0.003$ and $\lambda = 1 \times 10^{-4}$. All supervised training, including LPs, have training periods of 100 epochs.

A final point to consider is that since the model pre-trained on ImageNet uses the full encoder architecture, when we switch out the initial convolutional layer in the case of constructing ResNet-18L, we are getting rid of the trained parameters in this 7×7 layer.

3.2 Results and discussion

Our results for this experiment are available in Table 3.1.

Table 3.1: Linear probe accuracy and loss measured for encoders prepared with different learning methods across different label schemes. We display the label scheme in the top section of the table indicating which class each shape was assigned to. In the bottom sections, the linear probe results are displayed with red values indicating the most difficult label scheme for each particular pre-training method.

	Label Schemes (Class Assignment)							
Shape	Edge	Rectangle	Square	Ellipse	Circle	Regular	Inverse	All
Rectangle	0	0	1	0	0	0	0	0
Square	0	1	0	0	0	1	1	1
Ellipse	1	1	1	1	0	0	1	2
Circle	1	1	1	0	1	1	0	3
Encoder Preparation	**LP Accuracy**							
Random Init.	0.6699	0.6852	0.6091	0.6297	0.6625	0.6441	0.5299	0.4213
SSL (SimCLR)	1.0000	0.9750	0.9648	0.9889	0.9949	0.9679	0.9430	0.9692
Supervised (Edge)	1.0000	0.8361	0.8207	0.8347	0.8828	0.7008	0.7078	0.7120
Supervised (All)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
ImageNet (ResNet-18)	0.9116	0.8044	0.7805	0.7985	0.8256	0.7117	0.6550	0.6455
ImageNet (ResNet-18L)	0.9821	0.8344	0.8864	0.8926	0.9278	0.8120	0.7313	0.8328
	LP Loss							
Random Init.	0.6267	0.6089	0.6523	0.6525	0.6143	0.6248	0.6904	1.2585
SSL (SimCLR)	0.0019	0.1077	0.0885	0.0417	0.0168	0.0868	0.1518	0.0876
Supervised (Edge)	0.0001	0.3465	0.3618	0.2924	0.2853	0.5981	0.6158	0.5952
Supervised (All)	0.0001	0.0004	0.0003	0.0007	0.0020	0.0001	0.0001	0.0002
ImageNet (ResNet-18)	0.2491	0.4369	0.4841	0.4223	0.4280	0.5603	0.6128	0.8027
ImageNet (ResNet-18L)	0.0673	0.3447	0.3109	0.2716	0.2143	0.4171	0.5120	0.4028

It is readily apparent that the label schemes, even though they are reasonable for the dataset in question, play a significant role in impacting LP measurements intended to evaluate model quality. However, there are many other interesting observations to be made as well.

We begin by discussing the randomly initialized encoder. Even with random initialization, there exists weak semantic signals from the sample that can propagate through the network [4]. A LP accuracy study for random initialization can then be thought of as a look into what kinds of signals are preferred by a convolutional model as a result of of its structural properties rather than any learned parameters. In our dataset, these weak signals appears sufficient to achieve LP results significantly greater than random chance.

Also of note is the case where we conduct supervised learning for the encoder on the "all" label scheme, which each shape belongs to its own class. It is perhaps unsurprising from an information theory perspective, that when trained to distinguish each shape individually, the model obtains perfect scores across all label schemes as it would be the most complex task. What is interesting is that this model appears to score the highest loss for the circle label scheme, which we do not anticipate to be particularly complex. Certainly, none of the other encoders appear to have the same relative difficulty with it.

3.2.1 Analysis of label scheme complexity and difficulty

From this section on, we note that since we are looking at a population of agents and their performance on a task, it is more valid to make predictions of difficulty based on complexities of the task. This is especially true since our toy dataset contains few latent variables which may impact difficulty. As such, although care is taken to use precise diction in appropriate settings, generally for analysis, when difficulty is mentioned, it is implied to be a consequence of largely complexity.

An interesting observation we can make is that most learning methods we employ (with the exceptions of random initialization and supervised on "all" classification) appear to have surprisingly large difficulty with the "inverse" label scheme, at times even exceeding the difficulty for the "all" label scheme — which was believed to be the most involved task. It is not entirely clear why this seemingly

counter-intuitive observation about difficulty is the case. Perhaps the issue is a result of a single linear layer to simultaneously observe edge-related features and length-related features prior to making a classification. This is in contrast to, for example, earlier label schemes where in the case of isolating for circle or ellipse, the model can assign a class after simply detecting an edge. However, this explanation alone does not explain why so many encoders perform *worse* here than on the all-shapes classification task. If we consider instance learning, since the nature of the task is to create similar embeddings for similar images, we can imagine that the "inverse" classification task is therefore especially adversarial to its structured method. Another more "philosophical" proposal for this phenomenon that may be difficult to prove is that due to the mathematical and geometric relations between these shapes, the "inverse" classification scheme is not one that is typically important in the natural world. Consequently, SSL methods which are strong performers on natural images datasets and supervised methods which train on natural images perform poorly on this label scheme in comparison to more "naturally relevant" schemes.

More generally, it comes at no surprise that the "edge" label scheme is relatively easy for the networks. One can imagine that only a single mechanism for sharp corner detection needs to be learned in the network in order to differentiate regular and irregular rectangles from their ellipse counterparts. What is perhaps a bit surprising is that for the randomly initialized encoder, irregular rectangle detection is almost just as easy if not easier. This result appears to run counter to our prediction for difficulty given complexity. Distinguishing irregular rectangles from squares should be a more involved task as they are more closely semantically related and the task would require a greater degree of fine-grained classification. Given that this observation does not appear for other methods of encoder preparation, we suggest that perhaps in most learning tasks, features that enable corner detection are emphasized to a greater degree than features enabling length detection. In other words, length detection capabilities of the model match or exceed corner detection only in the randomly initialized states.

In order to support the argument that the result is not due to our seed, we conduct an investigation across five different seeds. In four of these five trials, we

observe that performance on isolating rectangle exceeds that of edge and edge-less detection. The average for the edge detection task is 0.607 with a standard deviation of 0.017. The average for isolating rectangles is 0.612 with a standard deviation of 0.014. Although we cannot say that random initialization *always* favours the "rectangle" scheme over the "edge" scheme, they are nonetheless quite close in value, contrary to the often significant advantage learning seems to provide performance on the "edge" scheme.

Another interesting result is the apparent increased difficulty in isolating for squares over isolating for rectangles. We would expect that isolating for squares or irregular rectangles would be a symmetrical problem and detecting squares and rectangles from edgeless shapes be largely equivalent between the two shapes. This appears to not be the case, with square detection seeming to be more difficult throughout all examined encoder preparation methods, with the exception of transfer learning from an ImageNet pre-trained model onto ResNet-18L architecture. One possible reason for this is due to the models being unable to entirely separate squares from circles. In addition to their close semantic relation to other rectangles, squares are perhaps more closely related to circles than ellipses given their regular shapes. This observation may be especially relevant for smaller squares and circles generated by our dataset in a pixelated image. Inevitably, circles must be represented with edges, becoming more similar to squares the smaller they are, despite our efforts to limit the smallest shapes generated. This compounding factor is likely what makes square recognition more complex and hence difficult than rectangle recognition. It is also worth noting that this relationship between squares and circles does not appear to exist to the same degree for rectangles and ellipses. Rectangles and ellipses would require both length and width to be small in order to resemble each other, a much less likely scenario than for square and circles.

We observe a similar kind of asymmetry with circles and ellipses. However, the situation is now reversed, where isolating circles is easier across the learning methods when compared to isolating for ellipses. The cause for this is likely to be the fact that rectangles of similar length and width more closely resemble squares in terms of distinguishing features in comparison to ellipses of similar semi-major and minor axes with circles. An visual comparison of these shapes is available in Figure 3.2.

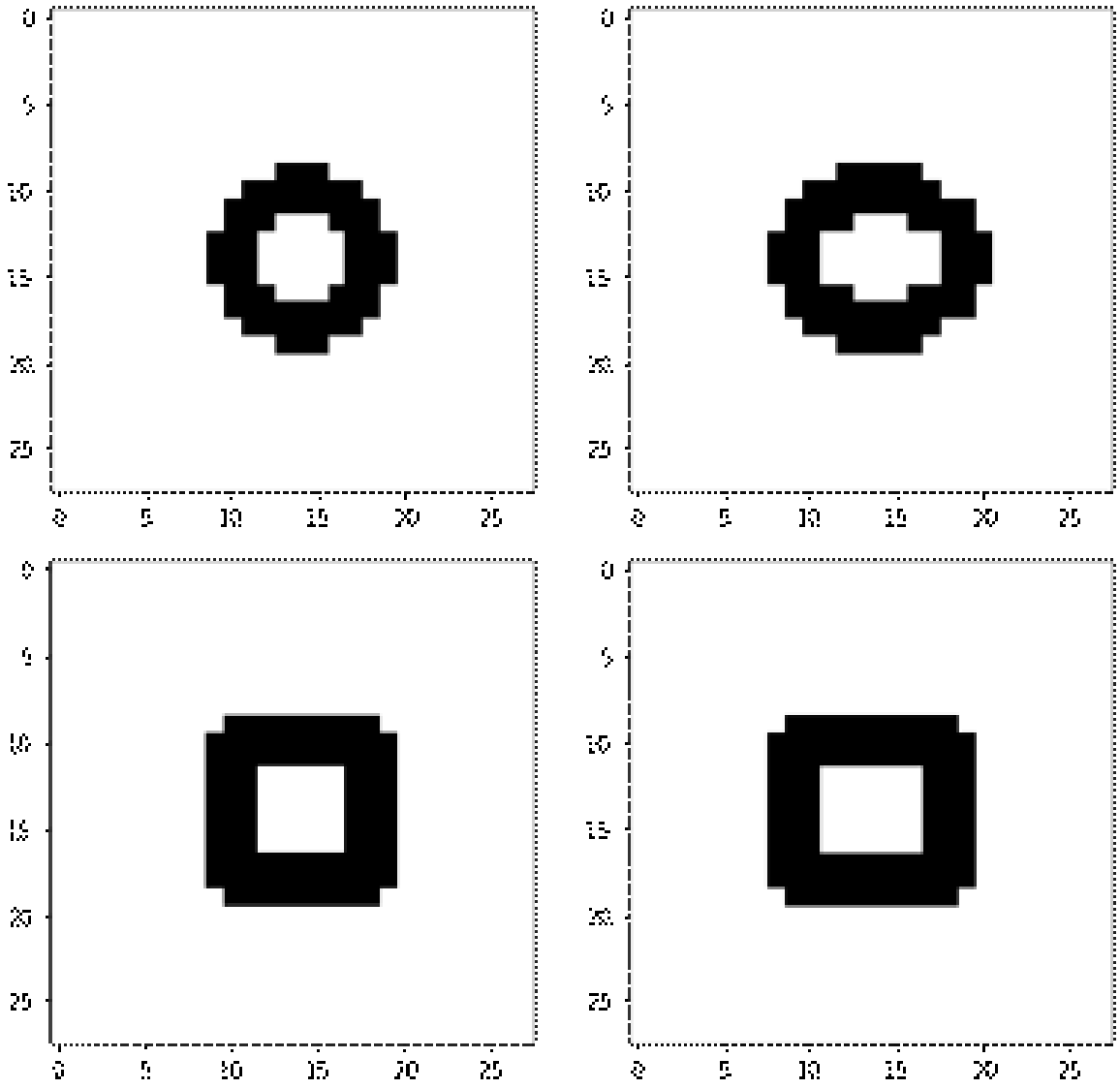


Figure 3.2: Visual example of near-regular rectangle and ellipse in comparison to their regular counterparts. Both the edge pixels in the ellipse and the internal shape appear to more prominently distinguish the shape from a circle. In comparison, the square and rectangle can only be distinguished by comparing the length and width.

Therefore, the difference in prominent and indicative features may be the reason why isolating for ellipses and circles is generally observed to be an easier task than isolating for squares or rectangles. However, we quickly run into a problem here because we are unable to explain why the ellipse label scheme is harder than the circle simply by proposing that it is due to an additional complexity in distinguishing ellipses from rectangles, since we have already said that it should be easier to distinguish rectangles from ellipses than squares from circles. Instead, our only recourse is to suggest that "it is harder to distinguish ellipses from squares than rectangles from circles". This claim may appear strange, however we can derive this result from observing a chart of how each shape relates to one another in terms of distinguishing complexity and through transitive relations.

3.2.2 Exercise in deriving properties pertaining to classification task complexity in toy setting

Given our set of four shapes, we can imagine that each pair of shapes share a symmetrical relationship representing a level of "complexity" associated with distinguishing between the pair, as portrayed in Figure 3.3. We make the following assumptions:

1. The level of complexity can be quantified in a manner where there is meaning in claiming one complexity is larger than, equal to or less than another — in other words, transitive relations hold for evaluating complexity;

2. When considering such complexities together to express a combined complexity associated with a classification task, the operation combining them is equally respective of all component complexities and is therefore additive or multiplicative in nature — they can also always be converted between the two using logarithms or exponentials; and

3. If the operation is multiplicative, then complexity must expressible in a form greater than or equal to zero[2].

[2]If we consider complexity as describable with information theory, it isn't typically reasonable to discuss negative information

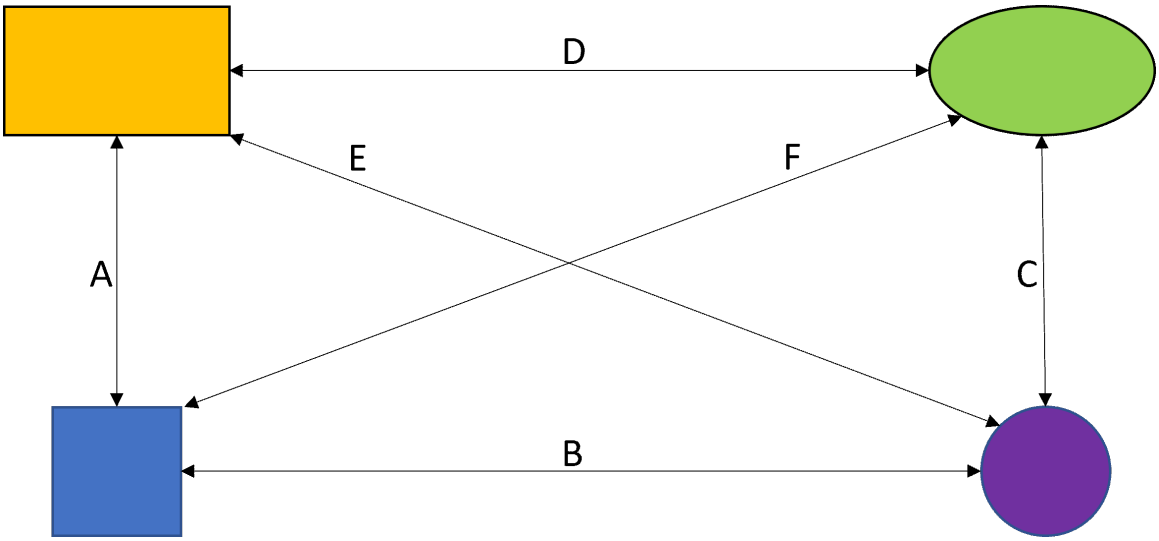


Figure 3.3: Graphical representation of complexities distinguishing each shape in our toy dataset from the others.

Using the naming convention for each difficulty relationship as shown in Figure 3.3, we note that up until this point, we have been attempting to explain our observed difficulties using solely the complexities A, B, C and D. We therefore establish the following *a priori* conditions taken from our earlier analysis in subsection 3.2.1, with justification for each condition presented in round brackets:

1. $A > B$ (Squares and rectangles are semantically more similar than squares and circles);
2. $A > D$ (Semantic similarity, as in first condition);
3. $A > C$ (Due to the greater similarity between squares and near-regular rectangles versus circles and near-regular ellipses);
4. $C > B$ (Semantic similarity);
5. $C > D$ (Semantic similarity); and
6. $B > D$ (As a result of pixelation, small circles are visually similar to small squares).

Combining these six conditions, we obtain the following master condition $A > C > B > D$. However, as stated earlier and as we'll soon discover, this master

condition alone is insufficient to explain our observations and we are required to consider complexities E and F as well. Unfortunately, we lack a theoretical framework to accurately understand their relative standing in comparison to each other and to other complexity relationships. We are therefore motivated to apply our current knowledge basis to our observations. Before that however, we need to define our single shape detection classification schemes in terms of the provided individual complexity relationships:

1. ADE (Rectangle);
2. ABF (Square);
3. CDF (Ellipse); and
4. BCE (Circle).

For the sake of convenience, we represent these combined complexities in a multiplicative fashion, although as stated earlier, they can be easily transformed from one means of combination to another.

In applying these condition to our observed relative difficulty results comparing the ellipse and circle label schemes, we have:

$$CDF > BCE = DF > BE, \tag{3.1}$$

where models universally find the ellipse scheme to be more challenging than the circle scheme. However, since $B > D$, this result is only possible if $F > E$. In other words, it must more complex to distinguish between rectangles and circles than ellipses and squares. Furthermore, we note that $F > E$ must take precedence over $B > D$. In the case of addition, this simply implies the difference between F and E is greater than the difference between B and D. However, in the multiplicative case, it is not as succinctly expressed. As such, we simply leave the conclusion as a notion of precedence or priority.

We can now refer to another observation comparing rectangle with ellipse label schemes:

$$ADE > CDF = AE > CF. \tag{3.2}$$

Since we believe that $F > E$ and $A > C$, we therefore must conclude that the latter condition must take precedence over the former. Armed with these conditions and knowledge of their order of priority, we can apply our theoretical framework for complexity to predict results for difficulty and compare them with our remaining observations.

1. For square and rectangle, we have ABF and ADE respectively, which reduce down to BF and DE. Since $B > D$ and $F > E$, we have $BF > DE$. With the exception of the encoder prepared with ImageNet pre-trained weights transferred onto a ResNet-18L architecture, we observe that it is indeed the case where the square label scheme appears to be more challenging than the rectangle.

2. Moving on to square and circle, we have ABF and BCE, becoming AF and CE. Since $A > C$ and $F > E$, we have $AF > CE$, suggesting that square is harder than circle. Our observations match this result throughout all encoder preparation methods.

3. For rectangle and circle, we have ADE and BCE, reducing to AD and BC. Since $A > C$ takes precedence over $B > D$, we have $AD > BC$, meaning rectangle should be more challenging than circle, which is what we observe, with the exception of random initialization.

4. In our final single shape detecting label scheme, we compare square and ellipse with ABF and CDF, reducing to AB and CD. Given $A > C$ and $B > D$, we have $AB > CD$, meaning square is expected to be harder than ellipse. This matches our observations exactly.

With the relatively successful application of our theoretical framework for complexity towards single shape detection label schemes difficulties, we may be tempted to apply this framework to our remaining label schemes. However, we lack one key detail relating $F > E$ to our master condition $A > C > B > D$. We argue that since rectangles and ellipses are both irregular, and squares and circles are both regular, it is easier to tell a square from an ellipse or a rectangle from a circle than a square from a circle or a rectangle from an ellipse. In other words, once again due to

relative semantic similarity, B and D are both more complex than E and F. Therefore, we can update our conditions to the following:

1. $A > C > B > D > F > E$;
2. $A > C$ takes precedence over $B > D$ and $F > E$; and
3. $F > E$ takes precedence over $B > D$.

We can use these tools to investigate the remaining label schemes. Firstly we consider the edge and edge-less label scheme. We expect such a task to consider the complexity relationship $BDEF$. Our observations for this task being the relative simplest requires: $(A > DE) \wedge (A > BF) \wedge (C > DF) \wedge (C > BE)$. While this condition is neither a fact that we've encountered previously nor something we can easily derive intuitively, it nonetheless remains compatible with our current framework. We note one exception in difficulty here with the randomly initialized encoder, where the edge detection task may be more challenging than the rectangle label scheme.

Next, we consider the regular and irregular shapes classification task, where the complexity relationship $ACEF$ is now relevant. Since this task contains the two most complex relationships A and C, we expect that this should be our most difficult task so far. In most of our observations, this is indeed the case, with an exception for the randomly initialized encoder and a relatively smaller exception for the SSL encoder.

Lastly, we consider the complexity cases for the inverse and the all-shapes classification tasks. We observe that the inverse scheme can be interpreted as $ABCD$, consisting of the top four most complex relationships. Naturally, such a scheme would be more difficult than all previously considered schemes which is what we universally observe. The all-shapes task would consist of $ABCDEF$ since we are distinguishing each shape from every other. We would expect that this final task is the most challenging, a conclusion supported by the randomly initialized encoder and the ImageNet pre-trained encoder on ResNet-18. For the observed counterexamples, we refer to the discussion and proposed explanations for this counter-intuitive phenomenon in subsection 3.2.1.

3.3 Summary

In this chapter, we looked to investigate the effects differing label schemes may have on the evaluation scores of prepared encoders. We also looked to gain insight into the nature of what constitutes complexity and difficulty in classification tasks and to examine how difficulty for a particular task may be analyzed based on complexity in a toy setting.

Our results and analysis in this section point towards two meaningful conclusions. The first being that as expected, different reasonably defined downstream tasks inevitably introduce biases resulting in different measurements pertaining to model quality. In a situation where we use SSL to pre-train an encoder for a general set of unknown tasks, it becomes difficult to ascertain which measurements can be trusted to provide a more reasonable estimate for model quality. While we can mitigate this issue by examining a set of benchmark datasets encompassing a variety of label schemes, such approaches are still inevitably subject to arbitrary assumptions and caveats, not to mention difficult to combine into a single estimation. In any regard, these approaches lack the objectivity and simplicity a single label-free measure conducted for the model on a relevant test dataset can potentially bring.

Our second conclusion revolves around insights gained into assessing the difficulty of a classification task. Although we do not yet have a means of quantifying "difficulty" or even "complexity", we proposed a theoretical framework (in the context of a toy setting) derived from hypothesized conditions, exploiting mathematical transitive properties assumed to exist for complexity. This framework appeared to have useful and predictive capabilities in characterizing task difficulty, at least in simple settings with few latent variables, largely matching our observations. It is worth noting that even if difficulty relationships do not always consistently relate to each other as a result of dependence on latent variables (as one can imagine is the case for many natural images), so long as they are understood, we can express still these difficulty relationships as functions of these latent variables and thereby produce a consistent theoretical framework for comparing difficulty.

Lastly, to reiterate the subjectivity associated with the difficulty of a particular task for a specific model, in our experiment, although we did not have such a model,

one can imagine that a model pre-trained on the "inverse" label scheme would perform strongly on that particularly downstream task perhaps at the cost of doing relatively worse on the others. There is some evidence for this hypothesis from the encoder pre-trained on the "all" label scheme. As noted in section 3.2, the loss for the circle scheme is the largest, while the loss for the tasks such as "inverse" and "all", although believed to be more complex and hence challenging, are small. Arguably, loss may be an even better indicator than accuracy for how difficult a model finds a particular task since it also encompasses a notion of "certainty" in its softmax classifications. However, this subjectivity need not be insurmountable. Instead of merely looking at a single pre-training method, we can look at multiple independent methods as we do here and obtain an understanding of which tasks tend to be more difficult for machine learning models in general. As an analogy, although humans due to their subjective experience and knowledge can differ in what they perceive to be a difficult task, there are nonetheless tasks, which appear to be universally accepted (by all humans) as being easier than others. One can hypothesize then that the cause for this widely accepted notion of difficulty is due to the underlying relative complexity of the task as we have defined here. This is also why we were able to derive properties of complexity by examining our results (based on difficulty) for a population of models. In any case, with these insights into the difficulty and complexities of classification tasks and their subsequent impact on model evaluation metrics, we can move on to considering how a label-free metric for model quality may be constructed.

Chapter 4

Label-free metrics across instance learning methods

When we consider any set of conceptualized real world objects, we can imagine that they exist in continuous distributions across a manifold in high-dimensional space. For example, given the concept of dog and cat, we can imagine that there exists a theoretical distribution encompassing all the dimensions necessary to describe each animal that is continuous in nature, providing an infinite possible examples of incrementally different dogs and cats. We can also imagine these distributions are connected — that there exists an uninterrupted path traversable on this manifold that extends from some distribution we arbitrarily call "dogs" to the distribution we dub "cats". When it comes to real world images, we observe that these distributions and the connections between them are in fact not of equal density and that the samples we can obtain are finite and discrete. At least in terms of contemporary real world experience, we expect to find significantly more images of cats and dogs than "cat-dogs" or "dog-cats"[1]. This finite observed distribution is what we come to learn in our lives and what we use to train our neural networks. A well trained network should therefore in theory, map a high dimensional image to a lower dimensional representational space, where we should see distributions that are functionally similar in terms of density, to the theoretical finite observed distribution.

Instance learning in particular, has a structured task such that a single sample is transformed by augmentations into two similar samples which are brought together in representational space by a distance based loss metric. If these augmentations are carefully selected, we can imagine that over training not only are same sample views drawn together, but also views from other samples — provided that they differ in a manner similar to the differences generated by the augmentation stack. With this understanding we see that the pretext task for instance learning has a similar goal to our proposed end result of a "well trained" network. It then stands to reason

[1] A tangentially related point is that visual characteristics are often a limited and arbitrary manner in which we can classify objects.

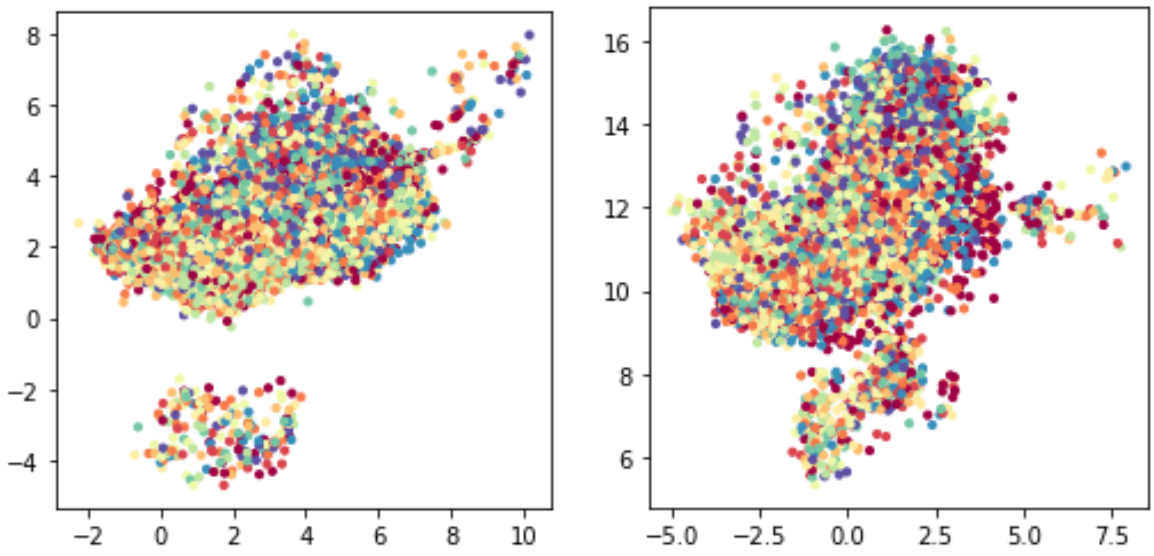


Figure 4.1: Examples of UMAP reduced embeddings for CIFAR-10 test dataset through randomly initialized ResNet-18L encoder using different seeds, with colours depicting ground truth classes.

that given these semantic relations in representational space, we can examine how the distributions for in-domain data embeddings change in this space over learning, in order to observe patterns which correlate with learning and are independent of labels. Two common ways with which we can measure this embedding distribution are clustering-based metrics and entropy.

We expect that as learning draws semantically similar samples together, it would naturally begin to form clusters of ground truth classes or subclasses. Additionally these clusters should become better defined over time, enabling intrinsic scores evaluating clustering to improve. As the spread of samples becomes less random and more coherent, we also expect clustering to become more consistent. Hence two independent clustering attempts should begin to agree more over learning — which is why we also look at clustering agreement.

We also hypothesize (based on initial observations) that at initialization, given the random mapping parameters, we will obtain low modal high dispersion embedding distributions when feeding forward a test dataset, as seen in Figure 4.1.

Over learning, as in the justification for using clustering, we expect the number of modes to increase to match the number of ground truth classes and for the

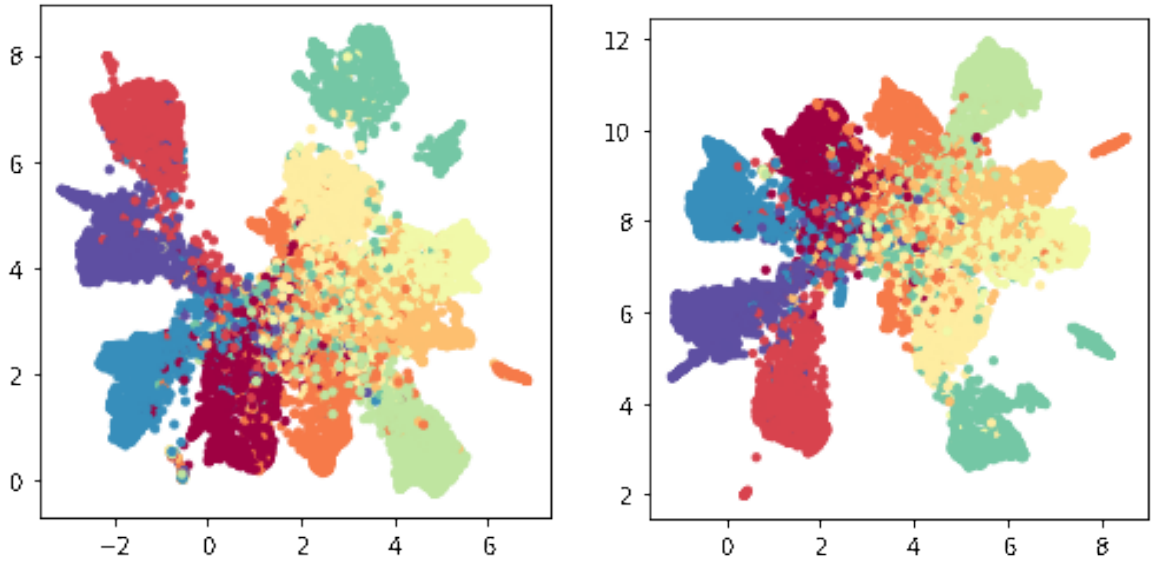


Figure 4.2: Examples of UMAP reduced embeddings for CIFAR-10 test dataset as produced by a fully trained ResNet-18L encoder with SimCLR methodology. Individual colours depict ground truth classes.

dispersion across these modes to become more compact similar to what we see in Figure 4.2. As a result, we can imagine that an information theory metric such as Shannon's entropy will be high at the start of learning before decreasing. In this manner, a lower entropy would correspond to a better trained model. Hence, we used both clustering and entropy in our attempt to measure learning progress without labels.

4.1 Methods

4.1.1 Datasets

The experiments in this chapter are conducted on the CIFAR-10 and CIFAR-100 [19] natural image datasets. These datasets consist of 60 000 32×32 sized images, partitioned into 50 000 training and 10 000 test images. There are 10 and 100 classes respectively for CIFAR-10 and CIFAR-100. All models in this chapter were trained on the training partition, while embeddings and linear probes employ the test partition.

4.1.2 Self-supervised learning

We used the modified version of ResNet-18 for small images referred to in this work as ResNet-18L, as described in chapter 3. The images are augmented into views using the SimCLR augmentation stack for CIFAR-10 images, which employs random horizontal flip, random resized crop and colour jitter, but removes Gaussian blur. [6]

We examine three SSL methods: SimSiam, SimCLR and MoCo-v2. When implementing SimSiam, we added to the ResNet-18L backbone, a three layer projector and a two layer predictor, with a width of 2048 throughout the projector. The predictor on the other hand had an input and output dimension of 2048 (to match the projector) except for the bottleneck layer, which had a width of 512. For SimCLR, we instead added a two-layer projector, with hidden dimension 2048 and output dimension 128. These are also the same dimensions we use for the MoCo-v2 architectures. It is important to note that the fact that the projector or predictor embedding space sizes vary depending on method is not particularly important. Analysis on embeddings are always examined on the ResNet-18L backbone's 512-d space. The projector and and predictor can be largely thought of as temporary tools deployed for the sole purpose of training the encoder and removed from the model afterwards. Their dimensions can therefore be interpreted as hyperparameters which may be permitted to vary depending on the SSL learning task such as a learning rate, rather than an inherent component of the network which should be controlled.

The networks were trained with these SSL methods using stochastic gradient descent (SGD) using a one-cycle learning rate schedule [28] with cyclic momentum from 0.85 to 0.95. For SimSiam, the peak learning rate $\eta = 0.06$, weight decay $\lambda = 5 \times 10^{-4}$, and we trained the network for 800 epochs. For SimCLR, $\eta = 0.5$, $\lambda = 1 \times 10^{-4}$, temperature $\tau = 0.5$, and we trained the network for 1000 epochs. Lastly, for MoCo-v2, $\eta = 0.06$, $\lambda = 5 \times 10^{-4}$, $\tau = 0.1$, queue length was 4096, EMA multiplier $m = 0.99$, and the network was trained for 800 epochs.

4.1.3 Representation evaluation

We captured snapshots of the models under training every 20 epochs, producing a "milestone". For each milestone, we fed the test dataset through the encoder, generating a set of 512-d embedding vectors we refer to as Z_{512}. Since typical

clustering methods do not work well in high dimensions [2], we reduced Z_{512} down to a 3-d space using UMAP, with $n = 50$ neighbours. We refer to this set of 3-d embedding vectors as Z_3 for the sake of convenience. Clustering is then performed on Z_3 using k-means, with the cosine distance metric. We set $k = k_1$, where k_1 for the sake of this experiment corresponds to the number of ground truth classes in the corresponding dataset. The cluster labels for Z_3 generated by this clustering attempt are then referred to as C_1.

The clustering attempt C_1 can then be compared to ground truth, C_{GT}, by measuring the amount of information about ground truth contained in C_1 through the extrinsic mutual information metric, $\text{AMI}(C_1; C_{\text{GT}})$. The adjusted mutual information with ground truth information here can be interpreted as a benchmark we can compare label-free metrics against. Certainly, it would be strange for a clustering-based label-free metric with no information about ground truth to exceed the utility of a clustering-based metric with ground truth information for reasons other than chance. As such, if even $\text{AMI}(C_1; C_{\text{GT}})$ does not obtain a strong correlation with LP accuracy, then we cannot justifiably expect a label-free clustering metric to have strong LP accuracy correlation either.

Next, using C_1, we measure the silhouette score on the original embeddings Z_{512}, a metric we refer to as S_1. As a benchmark, we measure S_{GT}, where we use C_{GT} instead of C_1, measured once again on Z_{512}. As with $\text{AMI}(C_1; C_{\text{GT}})$, the S_{GT} score is intended as a benchmark with which we can use to obtain an intuition for the maximum utility of the silhouette score.

For our final clustering-based metric, k-means is applied again to generate a second set of cluster labels C_2. Unlike C_1, C_2 uses $k = k_2$, where $k_2 = 2k_1$. We then evaluate clustering agreement $\text{AMI}(C_1; C_2)$. The reason for this doubling of clusters is due to preliminary testing, where we observed that although setting $k_2 = k_1$ produced a noticeable trend, the trend was susceptible to perturbations from stochastic processes introduced by dimensionality reduction. This susceptibility is due to clustering agreement values being quite high and occupying a small range when the two attempts share an equal number of clusters. Thus, even small perturbations to these values can bring relatively significant changes.

In addition to these clustering-based means of evaluation, we also employ

entropy. To measure the entropy of the embedding space, we take the Z_3 vectors and bin the values along each dimension to yield a 3-d histogram. The range of the distribution is defined by the difference between the maximum and minimum values for the Z_3 vectors along each dimension. The bin width is set separately for each dimension as $l_i = 0.4\,\sigma_i$, where σ_i is the standard deviation of the Z_3 vectors in dimension i. The 3-d histogram bin counts are divided by the total number of test samples, to yield an empirically observed probability distribution, from which we measure the entropy. As one can imagine, the reason we apply entropy on Z_3 is because the binning operation grows rapidly in terms of time-complexity, with respect to number of dimensions, reflective of volume. Additionally, since we do not desire the measure be affected by scaling factors, we use standard deviations as units for bin dimensions.

4.1.4 Linear probe

For each milestone, the encoder is extracted from the network and its weights are frozen. A linear layer was then added on top, to be trained on the training partition to map from the 512-d representational space to a space with dimensions equal to the number of classes. The linear layer was trained with the Adam optimizer [18], for 20 epochs, using a one-cycle learning rate schedule with peak learning rate η, of 0.08, cyclic momentum from 0.85 to 0.95, and weight decay of $\lambda = 1 \times 10^{-4}$. During LP training, we used the augmentations from the CIFAR-10 policy discovered by AutoAugment [10].

4.2 Results and discussion

We record our observations in Table 4.1. These are the Pearson correlation values for each metric compared to LP accuracy. For each dataset and learning method, we record two columns with and without the random initialization milestone before training begins. We find that this initial milestone inclusion for many of our models, can have a significant impact on the correlation with LP accuracy. As we will see later on, this strong effect induced by early milestones may reflect instabilities with early training where the model learns relatively quickly.

We observe that $\mathrm{AMI}(C_1, C_{\mathrm{GT}})$ correlates strongly with LP accuracy throughout

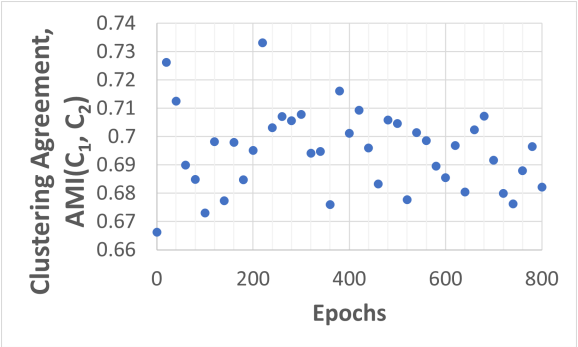
Table 4.1: Pearson correlation between performance metrics and linear probe accuracy. We display correlation scores both with (w/ init) and without (w/o) the network initialization (i.e. before training begins) milestone included in the trend. Grey: no significant correlation ($p < 0.05$). Black: positively correlated. Red: negatively correlated.

		SimSiam				SimCLR				MoCo-v2			
		CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Metric	Label-free	w/ init	w/o	w/ init	w/o	w/ init	w/o	w/ init	w/o	w/ init	w/o	w/ init	w/o
AMI(C_1, C_{GT})	✗	0.97	0.96	0.98	0.98	0.93	0.97	0.93	0.97	0.92	0.95	0.88	0.91
AMI(C_1, C_2)	✓	0.00	−0.21	−0.62	−0.71	0.71	0.91	0.76	0.90	0.64	0.80	0.47	0.61
S_{GT}	✗	0.93	0.97	0.94	0.96	0.89	0.62	0.95	0.91	0.37	−0.29	0.51	−0.11
S_1	✓	0.05	−0.59	0.16	−0.57	0.59	−0.07	0.28	−0.78	−0.09	−0.78	−0.47	−0.85
H(Z_3)	✓	0.82	0.92	0.86	0.87	−0.57	−0.83	0.18	−0.20	−0.26	−0.62	0.60	0.47

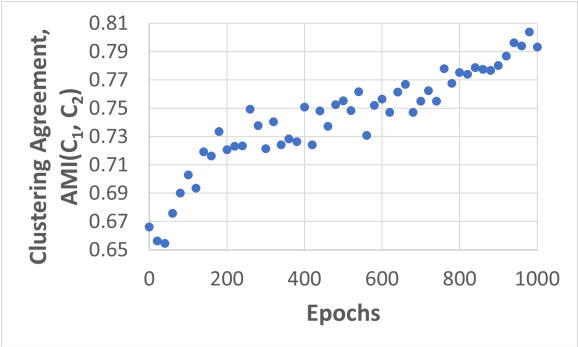
all combinations of methods and datasets. This result suggests that clustering is indeed able to progressively pick out ground truth classes as clusters in a manner reflective of learning progression. Our other ground truth benchmark, S_{GT} performs strongly for SimSiam and SimCLR but the result appears much weaker for MoCo-v2. This suggests that while ground truth based clusters did appear to separate and become more distinct over learning, this was only apparent in the earlier two methods. As such, we cannot expect that the label-free metric S_1 would do particularly well either. Indeed, in our observations, not only does the metric achieve weak correlations across the methods, in cases of strong correlation, the trend is negative, which appears to defy our reasoning and is contradictory to the observed positive correlation for S_{GT}. Therefore, we cannot conclude that S_1 is a particularly effective metric.

When we observe clustering agreement AMI(C_1, C_2), we see that the metric appears reasonably successful for SimCLR and MoCo-v2. However, not only does the metric fail for SimSiam, it also appears negative. In Figure 4.3, we display a comparison between how clustering cluster performs over training epochs for our three SSL methods.

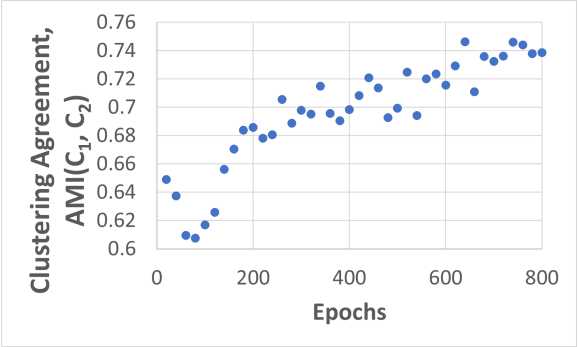
It appears that the general pattern of clustering agreement follows an overall non-linear ascension, where the metric decreases rapidly from initialization before increasing rapidly and following an almost linear trend. The initial decreasing portion of the metric appears to vary between SimCLR and MoCo-v2, being more pronounced in the latter. Our Pearson coefficient is therefore largely affected by this initial non-linear portion of development. In SimSiam's case, clustering agreement



(a) SimSiam



(b) SimCLR

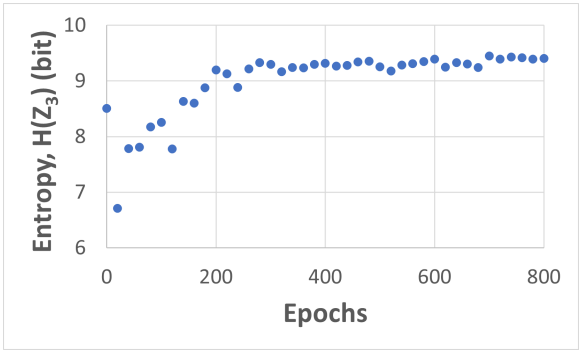


(c) MoCo-v2

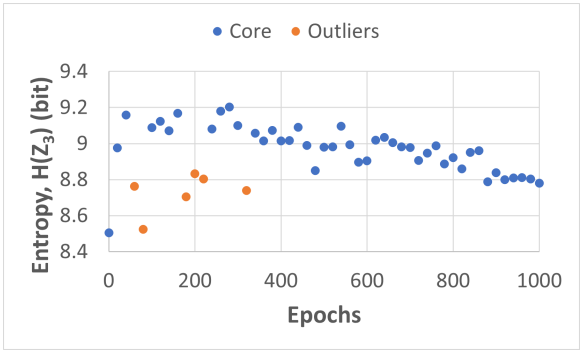
Figure 4.3: Examples of clustering agreement behaviour across different SSL methodologies on CIFAR-10. We observe similar behaviour on CIFAR-100 (not shown).

appears to bear a vague resemblance to this pattern, but is much less distinct. In particular, there appears to be outliers at epochs 20, 40 and 220 which severely impact the correlation with LP accuracy for this metric. It is not clear what may be causing this difference between SimSiam and the contrastive methods. It is known that non-contrastive methods such as SimSiam may be more prone to dimensional collapse [17]. Much like in the case where neural networks can have complete solutional collapse (also known as point collapse) where it ceases to process input and returns only a single output, dimensional collapses reduce the output of a network along a few limited dimensions. In future work, we can confirm this hypothesis by observing the singular value spectra of the projector and predictor embedding spaces.

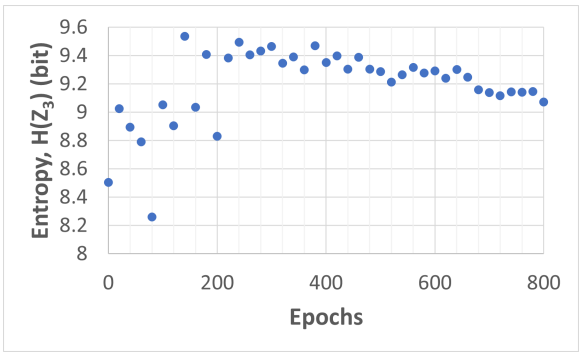
When we examine entropy, we observe another interesting phenomenon. Firstly, although we have reasonable correlation in SimSiam's case, we observe that the trend is positive in contrast to our prediction. For our contrastive methodologies,



(a) SimSiam



(b) SimCLR

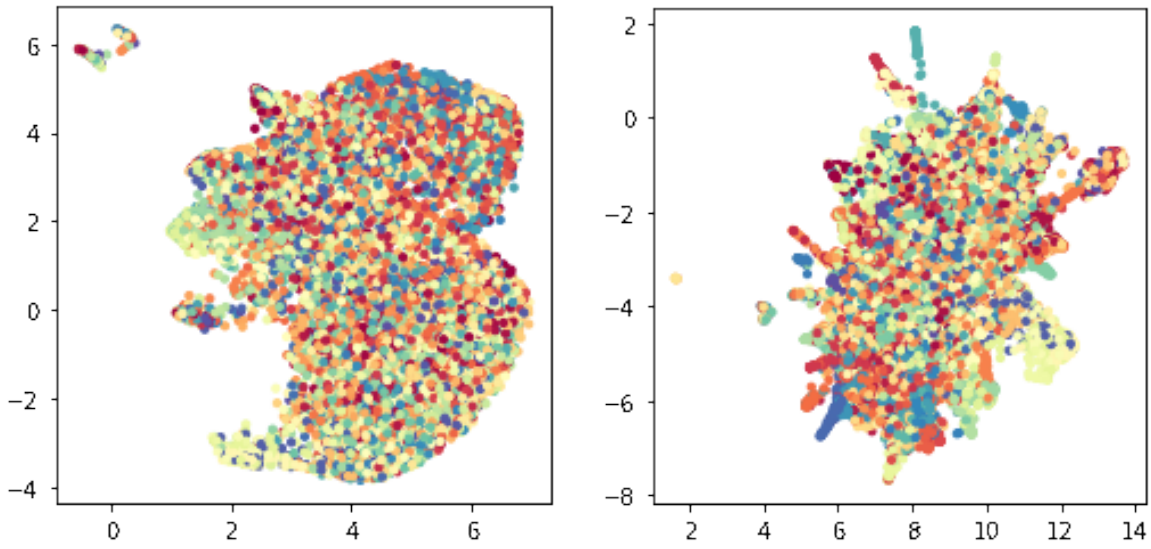


(c) MoCo-v2

Figure 4.4: Graphs for entropy measurements across different SSL methodologies conducted on CIFAR-100. In Figure 4.4b, we distinguish between "core" and "outlier" milestones, where the embeddings for the latter appeared to have lower entropy due to extreme outlier samples.

we observe that the trend for entropy is generally negative, in accordance with expectations, occasionally weak or insignificant. In examining the graphs for entropy, we observe that trend is in fact quite distinct and the weak or insignificant correlation score is a result of instability in the early epochs of training. These graphs are displayed in Figure 4.4.

Most noticeably for early training across methodologies, the entropy for these milestones appear to rapidly jump between low and high values. When we examine the representational space of these milestones, we observe that there are often extreme outliers for embeddings which are likely to reduce entropy, as seen in Figure 4.5. Interestingly, these outliers will be present for one milestone but may not be as severe for the next, leading to the erratic behaviour of the entropy measurements prior to stabilizing.



(a) Epoch 60 (b) Epoch 320

Figure 4.5: UMAP visualizations of outlier milestones observed for SimCLR at outlier milestones on CIFAR-100. Colours correspond to ground truth classes.

4.3 Summary

Relatively speaking, it appears that among our examined metrics, entropy is the strongest candidate for a potential label-free metric. Despite the weak correlation measured, we see that the cause is reasonable, due to the Pearson correlation being strongly affected by early training circumstances. There remains however, important issues which must be addressed prior to serious practical usage of the metric to measure learning. Firstly, it would be important to understand why the trend for entropy progression appears to become positive for non-contrastive learning methods. Additional work on other non-contrastive instance learning SSL techniques may be required. The instability of early learning also necessitates study. One option to gain a better understanding of this effect may be to observe in greater "resolution", the behaviour for the metric beyond our 20 epoch milestones. Early learning changes rapidly in comparison to later learning and since milestone examination can be expensive, in future iterations of this experiment, we may wish to concentrate a greater number of milestones early on rather than rely on equal spacing across learning. It is also worth noting that entropy instability may potentially have value as a label-free indicator for when the model has not

sufficiently learned the dataset.

Chapter 5

Label-free metrics across pre-trained architectures

When considering the usefulness of a label-free metric, while it is important to obtain an understanding how the metric behaves during learning progression across different SSL methodologies, another key area of interest would be to examine how comparable the metric is across different neural net architectures. A metric which is limited to within architecture comparisons would also be limited in its usefulness lacking the utility of a metric capable of comparing model qualities across architectures. To this end, we apply our metrics from chapter 4 on models prepared on the ImageNet-1k dataset implemented by torchvision [24].

5.1 Methods

Our methods are largely identical to those described in chapter 4. Conceptually, the largest differences are that we now conduct linear probes on pre-trained models rather than SSL training milestones. We used models with ResNet [14], DenseNet [16], and EfficientNet [29] architectures of varying sizes: ResNet 18, 34, 50, 101, and 152; DenseNet 121, 161, 169, and 201; EfficientNet (v1) sizes b0 through b7. The same methodology as described above was applied, using the pre-trained model as a (frozen) encoder, with the following changes:

1. CIFAR-10 and CIFAR-100 images were upscaled to the same resolution as that which the model was trained on;

2. Due to the significantly larger image resolutions for the bigger EfficientNet models, we reduced all batch sizes to 48;

3. The maximum learning rate was reduced to 0.003 for LPs; and

4. The entropy bin width was increased to $l_i = 0.8\,\sigma_i$, because the distribution in the representation space was found to be up to twice as large for SL

pre-trained models when compared with SSL.

As implied by upscaling the datasets images, we do not make any modifications to the pre-trained architectures, unlike in previous chapters.

5.2 Results and discussion

We record our results in Table 5.1. Due to the limited number of sample points we had for each architecture, we were unable to obtain significant results for all but the strongest correlations.

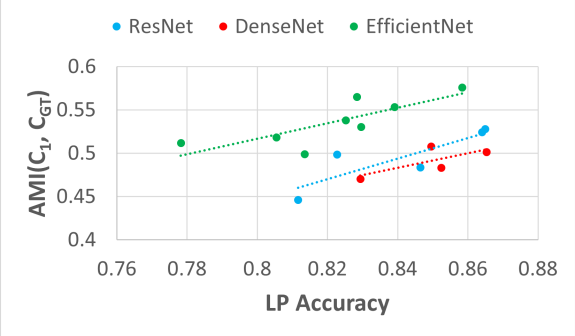
Table 5.1: Correlation between performance metrics and linear probe accuracy when transferring models pre-trained on ImageNet to CIFAR-10 or CIFAR-100. Correlations were measured across pre-trained models of the same architecture (ResNet, EfficientNet, or DenseNet), but different sizes. Grey: no significant correlation ($p < 0.05$). Black: +ve correlation. Red: −ve.

		ResNet		DenseNet		EfficientNet		Overall	
Metric	Label-free	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
AMI(C_1, C_{GT})	✗	0.86	0.92	0.73	0.69	0.80	0.72	0.20	0.38
AMI(C_1, C_2)	✓	0.83	0.96	−0.67	0.32	0.38	0.45	0.03	0.23
S_{GT}	✗	0.95	0.97	−0.15	0.10	0.24	0.83	0.16	0.21
S_1	✓	0.95	0.99	0.44	−0.06	−0.16	−0.12	−0.06	0.13
H(Z_3)	✓	−0.19	−0.87	0.75	−0.89	−0.85	−0.54	−0.34	−0.52

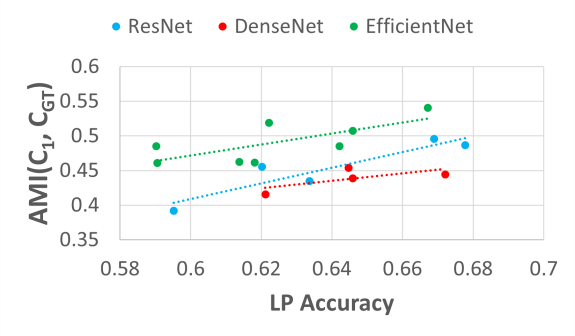
One of the key observations we can make is that while AMI(C_1, C_{GT}) appears to perform strongly across architectures, the results appear much weaker in the "overall" scenario where we combine all our data points across the three different kinds of architecture. We can glean more information on these results by examining the metric in greater detail in Figure 5.1.

We see that AMI(C_1, C_{GT}) has essentially three distinct mostly non overlapping lines for each of the three architectures. In other words, AMI(C_1, C_{GT}), despite correlating well with LP accuracy on models of the same architecture, its effectiveness does not appear to be independent of architecture. As such, there reason to doubt that label-free clustering-based metrics would similarly be effective for cross-architecture model evaluation. Indeed, AMI(C_1, C_2) appears to largely perform weaker than AMI(C_1, C_{GT}) in almost all scenarios.

When we examine S_{GT}, we observe a somewhat similar but weaker general trend as AMI(C_1, C_{GT}). As per our expectations then, S_1 does not perform strongly either.



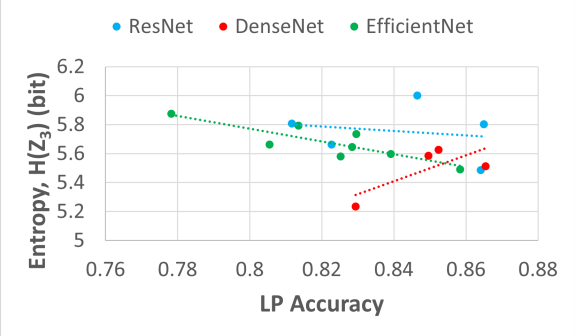
(a) CIFAR-10



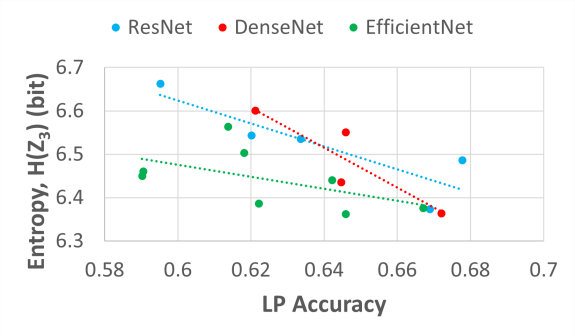
(b) CIFAR-100

Figure 5.1: Behaviour of AMI(C_1, C_{GT}) across different ImageNet-1k pre-trained architectures.

Lastly, we can observe that entropy in fact produces the only significant result in the "overall" scenario, suggesting that there may be viability to the metric for cross-architectural evaluation. However, we also note that entropy produces a positive correlation for DenseNet and CIFAR-10 dataset. We can obtain more clues about this result by examining the entropy graphs in Figure 5.2.



(a) CIFAR-10



(b) CIFAR-100

Figure 5.2: Behaviour of entropy across different ImageNet-1k pre-trained architectures.

There are two notable observations we can make. Firstly, it is now clear why entropy for CIFAR-10 appears to be positively correlated with LP accuracy. As we see in Figure 5.2a, it is primarily a single point which causes the observed positive correlation for DenseNet. Additionally, the data points for different network architectures are intertwined to a greater degree, once again suggesting that entropy may be viable for comparisons across architectures.

5.3 Summary

In order to determine if our metrics were viable across different network architectures, we evaluate them on models pre-trained on ImageNet, using downstream datasets CIFAR-10 and CIFAR-100. Due to the limited quantity of pre-trained models, we were unable to obtain strong results for this experiment. Nonetheless, our observations suggests that while clustering-based metrics may be useful for within-architecture comparisons or to monitor learning, based on the weak overall performance and the seemingly architecturally dependent data points we have for $\mathrm{AMI}(C_1, C_{\mathrm{GT}})$, it appears that such metrics may be fundamentally unable to useful for cross-architectural comparisons.

Our entropy metric however, was the only metric which appeared to produce a significant score when considering the overall data between architectures. Furthermore, it appears that in areas where the metric failed to obtain strong correlations, it was strongly impacted by outlier data points — perhaps due to our limited data. Future experiments may wish to produce and study additional pre-trained networks using entropy to obtain stronger results. Additionally, adjustments to how we bin data to obtain an entropy distribution may be required to refine the metric. One area of improvement may be to bin the data using the convex hull of the embedding distribution rather than considering the entirety of the distribution based on the absolute differences along each dimension.

Chapter 6

Conclusion

In this work, we considered how to approach the possibility of establishing a label-free metric to evaluate the quality of learned representations for a trained model, in particular for SSL. We began by examining the effect different reasonable label schemes (or classification tasks) for a toy dataset may have on model evaluation. We found that due to the fact that the complexity of these tasks can differ dramatically, so too are the results they provide for model evaluation. In circumstances where we are preparing a general model for a variety of unknown downstream tasks, or for tasks where we do not have a sufficient labelled test dataset, it can therefore be quite difficult or impossible to assess the quality of our model using label-based metrics alone. As such, there is value in a potential label-free metric which progresses in a predictable fashion correlating with learning.

In conjunction with understanding the difference in performance caused by different label schemes, we also presented an exercise in determining transitive properties for task complexity that may be useful for predicting the amount of difficulty a model may encounter with each task. We observed that deductions regarding task complexity did indeed correspond to our results for model performance. As expected, a greater complexity appeared to lead to a decrease in performance suggesting the model had greater difficulty with the task.

We then presented results for a number of label-free metrics and how well they correlated with LP accuracy on familiar benchmark datasets and instance learning methods. We used Person correlation in this manner, with high correlation suggesting greater viability for each metric. We found that while label-free intrinsic clustering measures were largely unsuccessful in capturing the learning progression, clustering agreement proved to be a more viable for contrastive learning. However, the same success was not observed for our studied non-contrastive learning method. Instead, the entropy of the embedding distribution appeared to be our most

successful result, producing strong correlations overall despite some initial instability early in training. The issue remains however, that the trend for entropy progression is reversed in the case of non-contrastive learning. In order for entropy to develop into a viable label-free metric, further research is required to establish the cause for the trend reversal observed. Additionally, while instability for the metric may be an indicator that the model has not sufficiently learned, the behaviour would require study as well.

For our final experiment, we presented results for the behaviour of our metrics across models with differing architectures pre-trained on ImageNet-1k. We observe that ground-truth based clustering metrics were only capable of capturing relative model performance for same-architectural cases. As an upper bound for label-free clustering-oriented metrics, we therefore do not expect such metrics to be capable of effectively comparing the performance of models with different architectural bases. Indeed, we observe much weaker results for these metrics. However, with the exceptions of a few outlier data points (which held strong influence due to the limited number of pre-trained models available), entropy appears to largely correlate with LP accuracy and in a manner independent of architecture, producing significant results. Refinements on how we bin data to calculate entropy may also enhance the metric's performance.

Bibliography

[1] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[2] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[5] Aditya Chattopadhyay, Benjamin David Haeffele, Donald Geman, and Rene Vidal. Quantifying task complexity through generalized information measures, 2021.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020.

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

[9] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers, 2021.

[10] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] Tianyu Hua, Wenxiao Wang, Zihui Xue, Yue Wang, Sucheng Ren, and Hang Zhao. On feature decorrelation in self-supervised learning. *CoRR*, abs/2105.00470, 2021.

[16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[17] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *CoRR*, abs/2110.09348, 2021.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[20] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, 2020.

[21] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[22] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[23] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[27] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[28] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018.

[29] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[30] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[32] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *CoRR*, abs/2012.04456, 2020.

[33] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018.

Appendix A

PaCMAP-based results

In this section, we look to use a new dimensionality reduction technique known as pairwise controlled manifold approximation and projection (PaCMAP) to reduce our dataset embeddings from 512-d to 3-d. This newer method appears to hold several advantages over t-SNE and UMAP in terms of preserving global distribution structure and local structures at least for many popular dimensionality reduction benchmark datasets [32]. The main reason we use this method is to examine if the outliers in early training we observed were a result of artifacts generated by UMAP or if they were fundamental to training. In doing so, we look at our CIFAR-10 embeddings for SimCLR. We set the nearest neighbour hyperparameter for PaCMAP to 15 and obtain the relationship seen in Figure A.1.

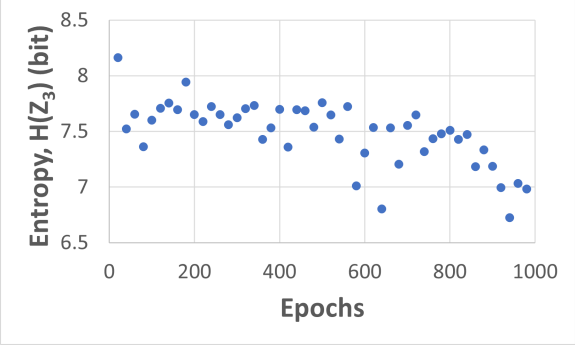
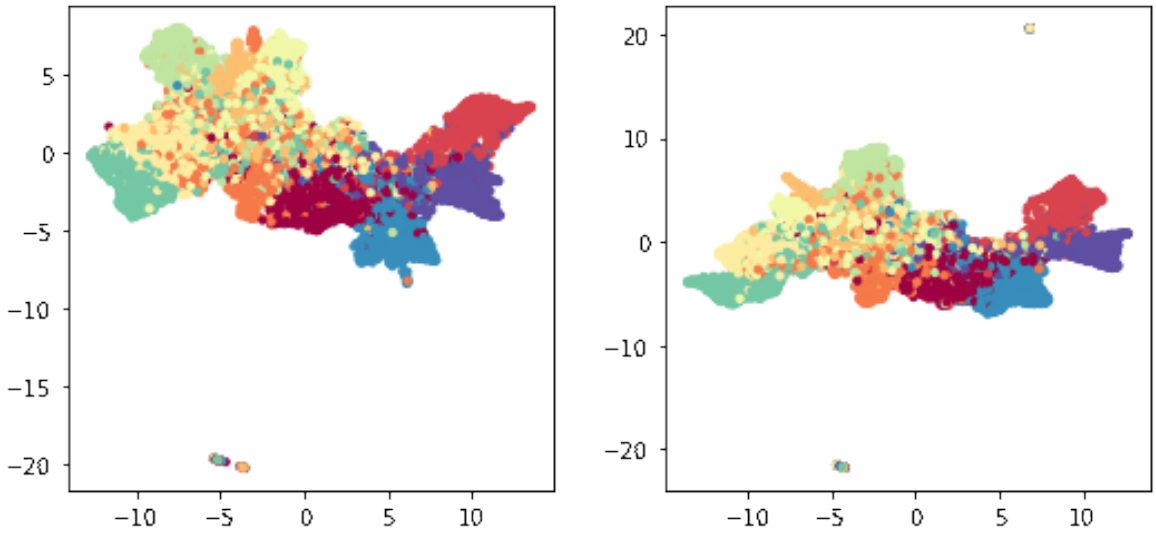


Figure A.1: Entropy measurement with respect to LP accuracy for SimCLR CIFAR-10 milestones examined using PaCMAP.

While we see the same general downward trend over learning as when we used UMAP, the Pearson correlation with LP accuracy is weaker at -0.56 without initialization state, compared to -0.83 for UMAP. Furthermore, the outlier states which appear to generate instability appear to be quite prominent even later in training as seen in Figure A.2. Although not depicted here, we also explore different values for nearest neighbours, including 5 and 50. These values did not eliminate the presence of outlier embeddings.



(a) Epoch 220 (b) Epoch 300

Figure A.2: PaCMAP visualizations of outlier milestones observed for SimCLR at outlier milestones on CIFAR-10. Colours correspond to ground truth classes.

It may be that these dimensionally reduced embeddings are truly reflective of the original high dimensional embeddings. As mentioned in our conclusion, a potential work around to eliminate many of the empty bins in our embedding space could be to change how we perform the binning operation. For this work, we attempted to use the minimum and maximum values for each dimension as the volume upon which we perform binning. However, another potential method would be to use the outline of the embedding shapes, or the convex hull of the distribution as the bounds for which we conduct binning. This approach would likely keep entropy more consistent and stable.

As for an explanation for why this instability occurs, it may be that the phenomenon is unique to instance learning methodologies, which focus on bringing together same sample views. It is possible that some of these outlier samples contain unusual properties that do not easily enable them to be associated with a particular defined class. For future work in confirming this idea, we can extract these samples manually or examine the progression of entropy for non-instance learning methodologies.