Exploring time-invariant low-dimensional contrastive representation learning

Ali Yang

Under the direction of

Rumen Dangovski and Peter Lu
Prof. Marin Soljačić
MIT Department of Physics

Research Science Institute
August 2, 2021

Abstract

In computer vision, self-supervised models now outperform supervised models at representation learning tasks. To compare self-supervised and supervised representations, we use contrastive learning to train an encoder to distinguish between simple images representing pendulums of different energy levels. The pendulum dataset is an ideal case study because it is well-controlled, well-understood, and the learned representations should be one-dimensional. We find that the self-supervised model learns a better representation than a supervised baseline that was given the ground truth. The self-supervised model also performs better under moderate levels of noise and dataset limitations. Additionally, the learned representations have interesting properties of their own, such as being denser in regions of higher noise. Overall, our results suggest that as long as the dataset is large and rather complete, self-supervised models are more robust than their supervised counterparts.

Summary

Most machine learning models require large datasets labeled by humans. Self-supervised learning is a recently developed kind of machine learning where models learn without any explicit data labeling, as opposed to supervised learning which uses labels. Recent results suggest self-supervised models may actually be even better than supervised models at a task known as representation learning, where the objective is to learn good representations of data. In other words, in representation learning, it seems that human labels make the models *less* effective. We replicate that result in a simple physics dataset. Our self-supervised model slightly outperforms a near-identical supervised counterpart. However, the supervised counterpart was told exactly what the energy was, unlike in many other tasks where the labels are at best approximate. Thus, self-supervised models do not outperform supervised models because the supervised models have non-optimal labels. Instead, our results suggest that the very act of giving a model labels to learn from may make it worse.

1 Introduction

Self-supervised learning is a subfield of machine learning where models are trained to perform tasks that are machine-generable from data. It is not *un*-supervised, but merely self-supervised in that the machine generates its own input-output pairs without human intervention. In practice, this usually refers to recovering a whole input from either corrupted or incomplete input. For example, in natural language processing, self-supervised models are trained to predict a missing word from the context around it [1]. If the inputs had no interpretable, recognizable structure, this completion task would be impossible as it would require recovery of completely unknown and independent data. In practice, however, the inputs, such as human-generated text or images of the natural world, are highly coherent and structured. Such structure must be learned by the model in order for it to succeed at the completion task. Furthermore, it must learn to ignore meaningless noise.

The goal of most self-supervised models is not to do any specific task, but to learn a meaningful representation of the data. Ideally, representations are easily interpretable and map closely to semantic concepts. Self-supervised models were originally motivated by a lack of human-labeled training data, or the innate incompleteness of human labels for vague semantic concepts. However, they also have other advantages over supervised models. Self-supervised models focus less on spurious correlations and seem to generate richer representations than supervised models [2–7]. They may also be more flexible and generalizable than supervised models. In some domains, self-supervised models now outperform supervised models.

We focus on self-supervised learning within the realm of computer vision. Self-supervised learning methods can generally be categorized as either generative or contrastive. Generative methods try to generate and mimic inputs, whereas contrastive methods attempt to discriminate between inputs in a meaningful way. Recently, contrastive methods have been shown to outperform generative methods and have become more popular. Most contrastive methods

focus on *instance classification*, an extreme form of classification where each input image is treated as its own class, and the task is to distinguish each image in the input from all others. Typically, multiple versions, or *views* of each image are given, and these comprise the class for that image. Views are automatically generated through image augmentations like cropping, rotation, color shift, or even using adversarial methods [4, 7, 8]. The representations learned are invariant to the particular image augmentations used. Good augmentations are absolutely critical for this method to work; without it, the model is free to learn arbitrary and meaningless representations [4].

In instance classification, the number of possible classes grows linearly with input size. This creates a problem where we require an enormous amount of computation for large datasets. It is more feasible to train a model to learn to identify whether two views come from the same image than to explicitly identify which image the input is. This reframing also enables easy zero-shot learning [9]. Such models almost always come in the form of two branches, each a neural network fed with distinct inputs. Each branch begins with an encoder for the image, and then may have either a projector or predictor after the encoder. The model receives, broadly, two classes of samples:

- **Positive sample.** Two views of the same image are given to the two branches. The branches should have similar outputs; the loss function approximates some notion of distance between the outputs.
- **Negative sample.** Two views of two different images are given to the two branches. The branches should have different outputs; the loss function approximates how close the outputs are. This type of input makes the model contrastive as it is contrasting between different images.

Often, the two types of samples are combined into a single task. For example, the model is given a view of one image, and is tasked to predict which of several other views belong to

the same image [4, 10].

Some recent work has removed the need for negative samples, which makes the method non-contrastive. One danger in the non-contrastive paradigm is *collapse*, in which the model simply outputs the same representation for all images. This trivially satisfies the similarity requirement without actually learning anything meaningful about the image. A variety of methods have been used to avoid collapse, including stop gradients [5], momentum encoders [6], asymmetry, redundancy reduction [11], feature decorrelation [12], and distillation [7].

1.1 Properties of learned representations

The primary goal of most self-supervised models is to learn rich representations of their input data. Good representations can enable superior performance on a wide range of downstream tasks. For example, self-supervised representations outperform supervised methods on transfer learning for image classification, despite not being specifically targeted towards image classification [3, 7]. These methods are trained on a simple linear classifier on top of the learned representations.

An example of the performance differences between self-supervised and supervised representations is given in Caron *et al.* [7], where the self-attention maps inside both supervised and self-supervised vision transformers are analyzed to produce image segmentations. The self-supervised segmentation is noticeably better, with considerably less noisy extracted regions. Moreover, Caron *et al.* describe good results for image classification achieved with a nearest-neighbors classifier on the self-supervised representation, where outputs are sourced from other data points whose representations are close by [7]. This indicates that the geometry of the learned representation is semantically meaningful.

Despite their differences, self-supervised and supervised representations still share significant similarities. Bansal *et al.* [13] used model stitching to compare the representations learned by self-supervised and supervised models, and found that the learned representations

were similar enough that a composite model did not suffer a significant drop in accuracy.

1.2 Our contributions

Most studies of self-supervised models, to date, use natural data, such as human-generated text or images of objects [2, 4]. While such data is the most realistic, complex trends in this data can lead to obfuscation of the model mechanisms. We study self-supervised models in the context of a completely understood, well-controlled ideal dataset. This allows us to observe the behavior of self-supervised models without potential interference from unknown deficits in the dataset.

Our dataset is based off a rigid pendulum, one of the simplest physical systems. The pendulum's motion is well-understood and can be described exactly with an elliptic integral. Instead of using visual augmentations like cropping and color shift, our primary augmentation is moving the system backwards or forwards through time. This encourages the model to learn a time-invariant representation. Furthermore, given fixed mass and arm length, the pendulum has only one conserved quantity, energy. Therefore, the model should learn a one-dimensional representation which is well-correlated with the pendulum energy. Because we only have a one-dimensional dataset, we can objectively quantify how good our learned representations are.

Unlike Wetzel *et al.* [14], who analyzed physical systems using a numerical dataset, where position, velocity, momentum, etc. were directly fed into the model, we use a graphical dataset composed of generated pendulum images. With a graphical dataset, the problem becomes considerably harder and more realistic as velocity and position are not directly encoded in the input, and there are many other superfluous variables that may confuse the model.

For our encoder, we use ResNet-18, a convolutional network that has skip connections [15]. Our contrastive learning method is based off of SimCLR [4], a simple and intuitive way of contrastive learning that does not need anything except an encoder.

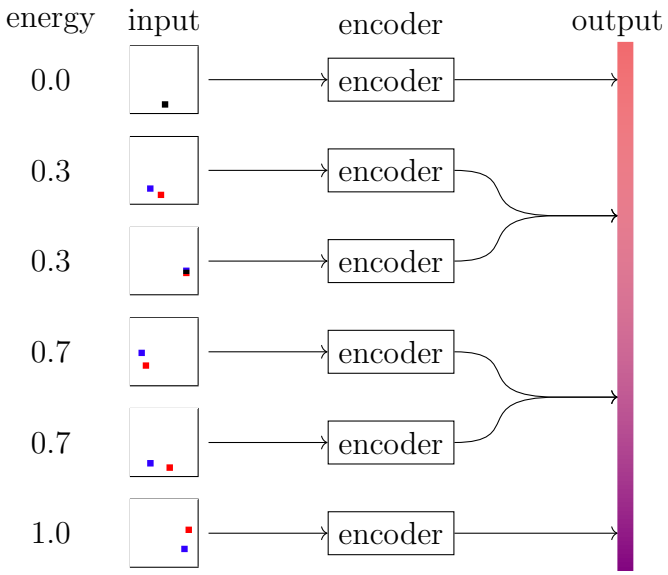

Figure 1: A schematic explaining how our self-supervised model works. The model is composed of an encoder, which converts images (left; see Figure 2 for how image generation works) to one-dimensional outputs (right). The encoder is trained to distinguish images of pendulums with different energies from one another, but also to recognize when two images come from the same pendulum (i.e. have the same energy encoded in them). Images from the same pendulum are encouraged to have the same output. Images from different pendulums are encouraged to have different outputs.

In section 3, we find that:

(i) Our ResNet-18 [15] encoder with custom setup inspired by SimCLR [4] determines the energy with high precision.

(ii) The encoder is noise-resistant.

(iii) The encoder has a limited capacity to interpolate and generalize beyond training data.

(iv) **In most of our experiments, the self-supervised model outperforms a supervised baseline.**

In section 4, we explore when self-supervised representations break down (Subsection 4.1) and some of the properties of self-supervised models (Subsection 4.2, Subsection 4.3).

These results show the power of contrastive learning, even compared to a supervised baseline that was given the ground truth. Furthermore, the self-supervised model is able to withstand some degree of distortion in the data (Subsection 3.2, Subsection 3.3). Our study suggests that, compared to self-supervised models, supervised models may have intrinsic limitations.

We acknowledge that the last result in particular is quite strange and unexpected, given that the supervised model should be—in theory—optimized to predict the energy. We were not able to find out why the supervised model underperformed the supervised model. It is probably not because of conventional overfitting or low resolution (see Appendix 1). We suspect it may be due to a non-optimal loss function, but the losses we used were rather simple and standard, so it is difficult to see why the loss may have issues. Furthermore, the effect is reproducible in a number of variations on our original dataset. We are rather sure that this phenomenon is genuine, but still do not really understand why it occurs. Finding out the reason for this difference will be a great starting point for future work.

2 Methods

2.1 Dataset

The dataset contains 20 pictures each of 5120 pendulums, and is entirely self-generated. The pendulum energy is uniformly distributed between 0 (no energy) and the level of energy needed to reach the top without going over, which we scale to 1. Note that in the limiting case of energy 1, the pendulum has infinite period and never actually reaches the top but approaches it with a continually decreasing velocity.

For each pendulum, 20 times are picked within a constant range of time (200 units). For the vast majority of observed pendulums, this range is large enough to contain several periods and thus it may be considered a good approximation of uniformly distributed time. For very high energy pendulums which have arbitrarily long periods, this may reduce the efficacy of the model. Images are generated using the time-energy pairs. Images are 32 by 32 pixels and heavily simplified.

Image generation is explained in Figure 2. Each image has a white background. For a time-energy pair (t, k), two bobs are drawn: a blue square bob of width 3 pixels for the pendulum with energy k at time t, and a red square bob of width 3 pixels for the same pendulum a small amount of time (0.5 units) later. Overlapping areas are drawn black. The distance between the two bobs approximates velocity, while their position encodes the position; these two variables (velocity and position) are sufficient to calculate the energy of the pendulum. Note that position and velocity, which are continuous, must be approximated in the image due to the discreteness constraints forced by the discrete pixels within the image.

This is what we call the *vanilla* setup, without any perturbations. We then introduce five harder variations on this dataset. For some graphical illustrations of these datasets as well as more examples from the dataset, see Appendix 2.

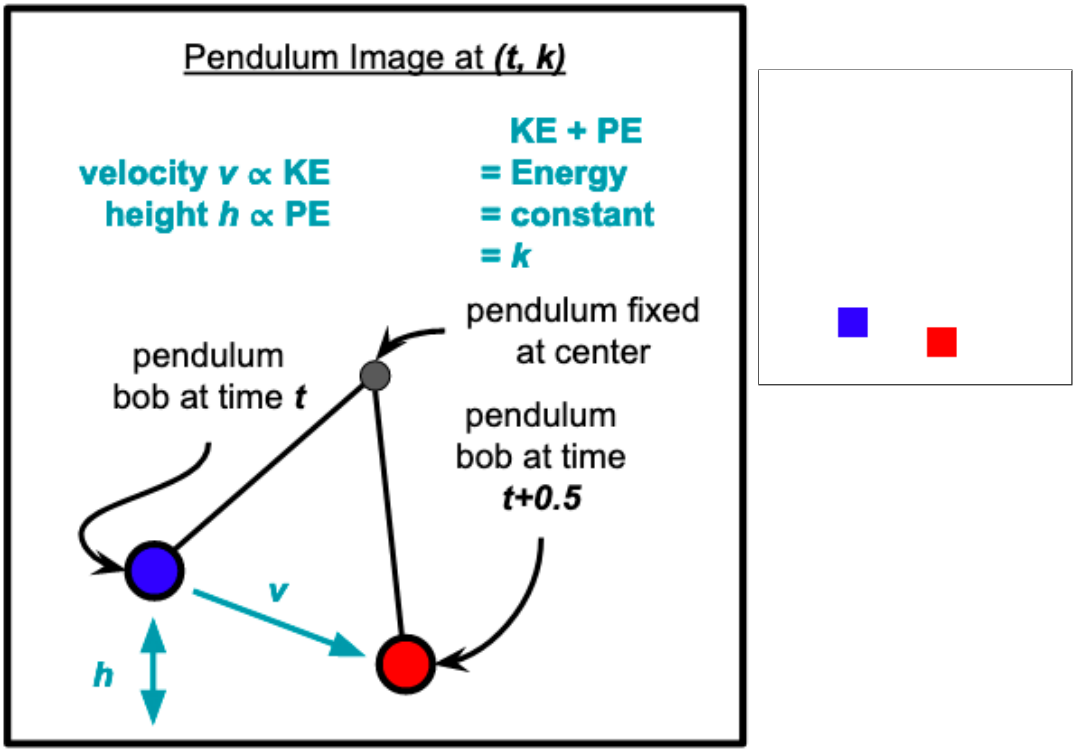

Figure 2: *(Left)* An overview of how the pendulum image generation works. The module is fed with a time t and an energy k. It then draws the position of the bob at time t in blue, and $t + 0.5$ in red. The distance between the two approximates velocity; the height approximates potential energy. Therefore, the total energy can roughly be derived from the image. *(Right)* A sample image from the dataset. Neither the rod not center is available, but the blue and red pendulum bobs (which belong to the same pendulum, 0.5 time units apart) are visible.

Large images. Images are 100 by 100 pixels instead of 32 by 32 pixels.

Numerical noise. In this dataset, each pendulum's position, as measured by its angle θ from the vertical, is perturbed by a value drawn from a zero-centered Gaussian with standard deviation ε_n, a parameter controlling the strength of the noise which we vary.

Graphical noise. In this dataset, three graphical perturbations are introduced. The severity of the perturbations is controlled by a parameter ε_g. The first is translational noise, whereby with probability ε_g the image is shifted one pixel left or right. The second is background noise, where each color channel of each pixel is perturbed by a value drawn from a zero-centered Gaussian with standard deviation $\varepsilon_g/4$. The third is tint noise, where the entire image is perturbed by a value drawn from a zero-centered Gaussian with standard deviation $\varepsilon_g/8$. None of these change the energy encoded in the image; they only change its appearance.

Energy gaps. Three uniformly-spaced *gaps* in the energy totaling to a length α_g are removed from the training set. Then, the missing energies are reintroduced during testing, forcing the model to interpolate on data it has not previously seen before.

Cropping. Given a parameter α_c, we generate an image of size $32/\alpha_c$, then reduce the height and width of the image by a factor of α_c, producing a 32×32 cropped image. We choose the cropped image to share the same bottom-right corner as the original. This produces a number of blank images in which the pendulum bob is not visible.

2.2 Encoder and loss function

We use a setup similar to SimCLR, with input batches of size 512, each containing a pair of images from the same pendulum. SimCLR is a contrastive learning approach where we have two identical branches both receiving input and numerous negative samples [4]. Unlike many other methods, the base encoder is sufficient without an additional downstream projector or predictor.

We use a version of the InfoNCE loss, with negative Euclidean distance in place of the typical cosine similarity [10]. Let $\text{dist}(\mathbf{x}, \mathbf{y})$ denote Euclidean distance, and let $b = 512$ denote the batch size. Let $f(x, y)$ be a **similarity** function. Given a temperature paramater τ, we define $f(x, y)$ to be precisely the following:

$$f(x, y) = e^{-\text{dist}(x,y)^2/\tau},$$

where $\text{dist}(x, y)$ is Euclidean distance.

Then, let $\{x_i\}_{i=1}^b = X$ be the outputs on one branch, and $\{y_i\}_{i=1}^b = Y$ the corresponding outputs from the other. If we let $\mathbb{E}$ denote expectation, then the InfoNCE loss is

$$\ell = -\mathbb{E}_X \left[\log \frac{f(x_i, y_i)}{\sum_{j \neq 1}^b f(x_i, y_j)} \right].$$

Mathematically, this is equivalent to a cross-entropy loss where the logits are negative Euclidean distance, and the target is 0 everywhere except where the indices match. This encourages x_i to be pushed near y_i, and for all $j \neq i$, away from x_j.

We note that instead of using cosine similarity as with all other version of this loss, we use negative Euclidean distance. Cosine similarity produces a *wrap-around* effect around $\theta = 2\pi = 0$ which is undesirable in our dataset because pendulums at the two energy extremes are very different. Therefore, the representations generated by the Euclidean distance are more faithful to the original dataset. This is not to say that Euclidean distance is better for all variables; for variables which have wrap-around (such as angles), cosine similarity is preferable. While it may be argued that this adjustment can only be made based on pre-existing knowledge of the conserved quantities, we note that our model learns well even with cosine distance (see Appendix 3) and that this change is mostly made to facilitate easier analysis of the representations.

The encoder itself is a ResNet-18 with the final output layer replaced with a small number of output neurons which determine the dimensionality of the representation [15]. Most of our tests were on one-dimensional representations, where the encoder had one final output neuron.

2.3 Testing and analysis

We train with 100 epochs, using stochastic gradient descent with momentum 0.9, warmup, and a cosine decay rate. We use gradient clipping, which is necessary to prevent exploding gradients. Each test is run 5 times and the reported result is the average, except for the vanilla 1D test which was run 25 times. Only results on testing sets are reported. Testing is conducted with a same size dataset (5120 pendulums with 20 images each).

The supervised model which we use as a baseline has identical ResNet-18 structure, the same number of epochs, the same number of data inputs, and with the same stochastic gradient descent optimizer. We use mean squared error loss against the true value of energy. It is trained under as similar conditions to the self-supervised model as possible. Exact training parameters for both models are in Appendix 4.

To quantify the strength of the representations learned, we use the Spearman correlation, which is the Pearson correlation except on the ranks of data. The Spearman correlation is well-suited to 1D nonlinear correlations like ours. We define two metrics using the Spearman correlation. The first is the *global Spearman correlation*, in which we use the Spearman correlation across all pendulums and all energy values. The second is the *local Spearman correlation*, in which we split the data into seven equal segments along the energy values and calculate the Spearman correlation within each segment. The global Spearman correlation quantifies how well the representation distinguishes low energy pendulums from high energy pendulums, while the local Spearman correlation focuses on whether the representation can distinguish pendulums with similar energy values from each other. In general, the local Spearman correlation is lower because the corresponding problem (distinguishing pendulums of a similar energy level from one another) is more difficult.

Image Size	32×32	100×100
Global Spearman	**.9938**	**.99958**
Supervised Baseline	.9937	.99888
Avg. Local Spearman	**.809**	**.9812**
Supervised Baseline	.804	.951

Figure 3: Main result. The number of digits specified in each result is the same as the number of digits needed to specify a 95% confidence interval. The self-supervised model is superior to the supervised baseline on both global Spearman correlation and average local Spearman correlation. The performance difference gets more pronounced when the image size increases; it may be better at interpolating within the larger possible input space (see Subsection 3.3 for interpolation results). This result suggests self-supervised learning may be intrinsically advantageous to supervised learning.

3 Results

3.1 Vanilla

In the 1D case, both the global (.9938) and local (.8093) Spearman correlations for the self-supervised model are higher than for the supervised model (.9937, .8044)—see Figure 3.

Furthermore, in higher dimensions, the representation is still primarily one-dimensional (see Figure 4). The other dimensions do encode meaningful, but not time-invariant, information such as position; for a more detailed analysis including accuracy, see Appendix 5. For the self-supervised model, the exact relationship between the energy and the representation is nonlinear; this is further analyzed in Subsection 4.2.

3.2 Noise

Numerical noise. Numerical noise gives the model truly misleading information; with high levels of numerical noise, the meaningful pattern associated with conservation of energy becomes obfuscated. For all levels of noise except the highest ones ($e_n \geq 0.2$), the self-supervised model outperforms the supervised model (Figure 5). Therefore, the self-supervised

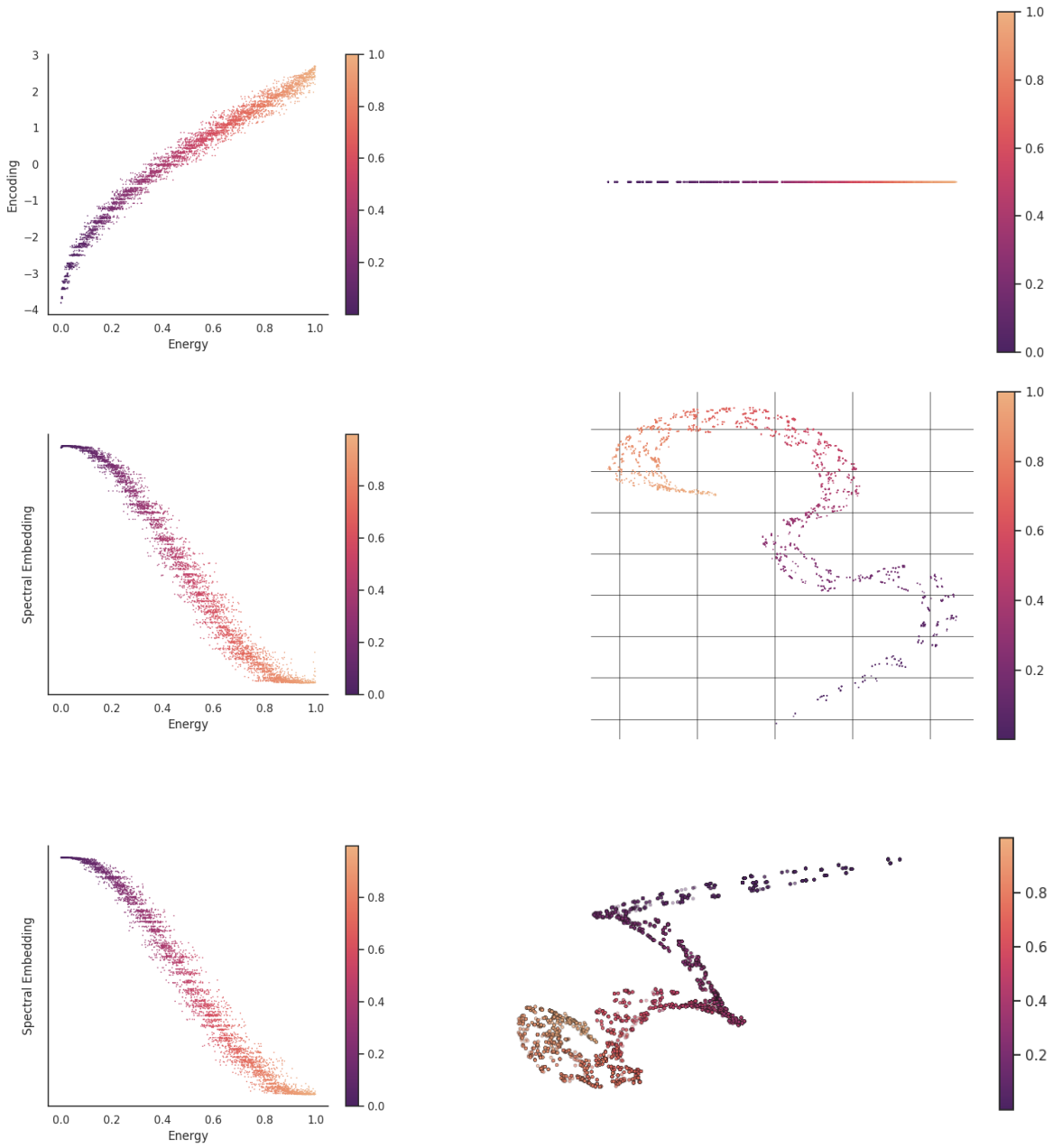

Figure 4: *(Left)* Correlation between energy and either the 1D representation itself, or the spectral embedding from the representation, whichever is applicable. Top is 1D; middle is 2D; bottom is 3D. Very strong correlation is notable in all three tests. *(Right)* Examples of learned 1D, 2D and 3D representations (top to bottom). Each axis corresponds to one of the outputs of the encoder. The representation stays mostly 1D even in higher dimensions, so the model has learned to model the true dimensionality of the dataset. We can also observe some curious qualities of the 2D and 3D representations, such as their tendency to twist about in space.

model is more robust than the supervised model against noise. For both models, the local Spearman correlation starts sharply decreasing before than the global Spearman correlation, indicating models unlearn local relationships before global relationships.

Also, as noise increases, the density of the representation increases. See Subsection 4.2.

Graphical noise. Graphical noise introduces superfluous noisy data but does not corrupt the position of the pendulum. The model is much more resistant to graphical noise. Again, the self-supervised model outperforms the supervised model (Figure 5). Here, the strengths of the self-supervised model become clear: it is much better at adapting to the random noise present in the image *as long as* the true information in the image is not distorted.

3.3 Energy gaps

In the gaps experiment, we introduced three gaps within the range of possible energies whose lengths summed up to a parameter α_g. In training, the model did not see any pendulums at all within those energy ranges. In testing, we reintroduced those energies and measured the model's ability to interpolate on completely new, previously unseen data. For $\alpha_g \leq 0.4$, the self-supervised model had superior performance, where as for $\alpha_g \geq 0.6$, the self-supervised model's global Spearman decreased considerably while the Spearman within interpolation/gap regions continued to be better than the supervised baseline (Figure 6).

The reason for this behavior is due to a phenomenon we call *fracturing*, when the model learns separate, discontinuous embeddings for different regions of the data and is unable to piece together a global structure. We discuss fracturing further in Subsection 4.1. If we remove the trials in which fracturing obviously occurred, the self-supervised model sees dramatic improvements.

We see that when the model does not fracture, it has especially good results on $\alpha_g \geq 0.6$ (Figure 7). Therefore, the behavior with large gaps can be characterized as either *fracturing*, where the representation becomes non-intepretable—but not meaningless—or *non-*

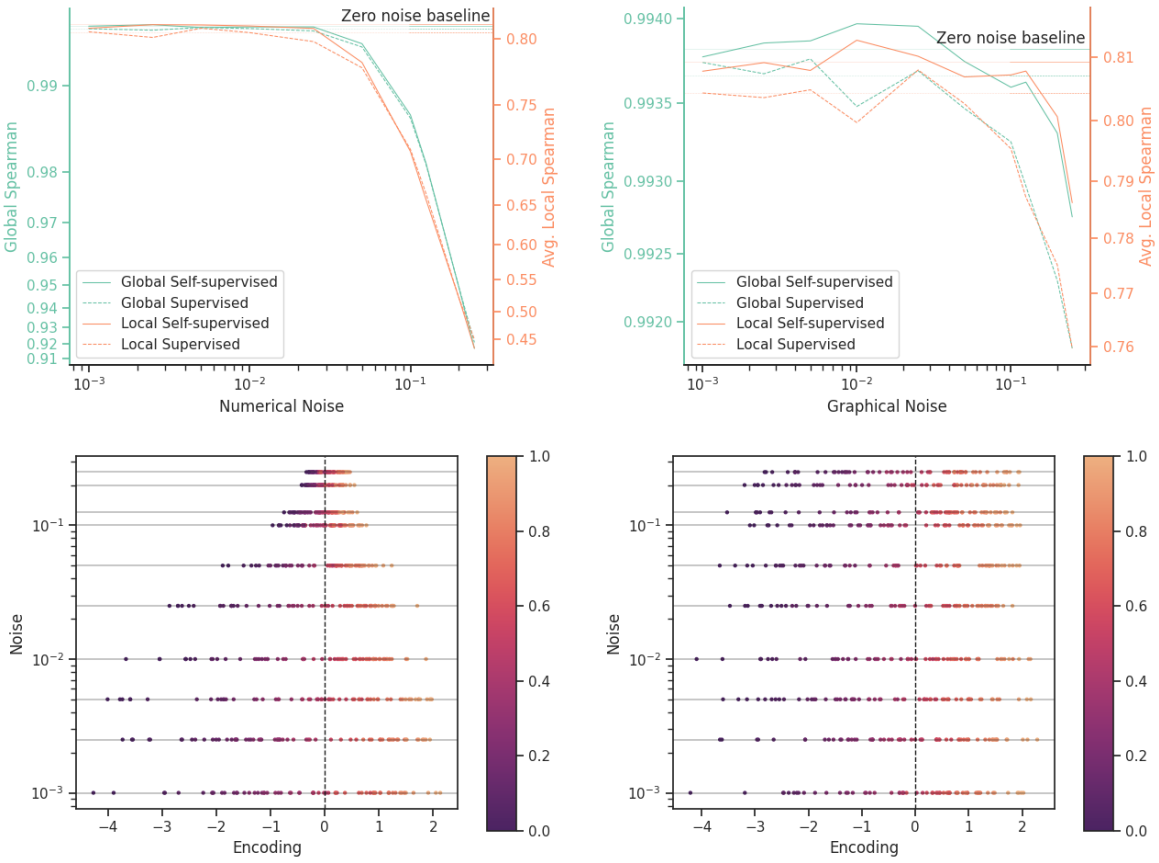

Figure 5: *(Top)* Global and local Spearman plotted against increasing numerical noise level. Dashed lines indicated supervised model; solid lines indicate self-supervised model; horizontal lines are zero noise baselines for comparison. Left–numerical noise; Right–graphical noise. Note that the scales for the two plots are different. *(Bottom)* 1D plots of the representations with increasing numerical noise level. Representations were reflected to align with each other and were translated to have zero median. Color corresponds to the energy of the associated pendulum. Left–numerical noise; Right–graphical noise.

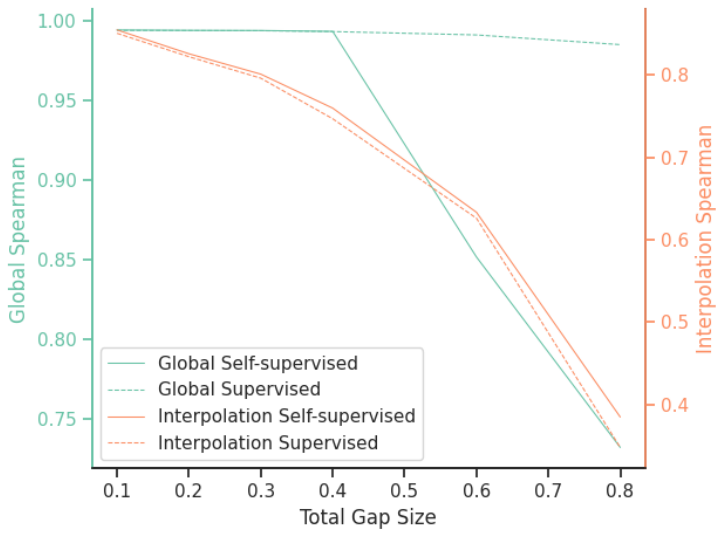

Figure 6: Global and interpolation Spearman correlations for the data with gaps. As the gaps increase in size, both correlation decrease as expected. Interpolation Spearman is calculated as the average Spearman correlation within each gap and measures how good the model is at interpolating. At $\alpha_g \geq 0.6$, we see an interesting phenomenon where the global Spearman decreases considerably, indicating that the global representation is no longer monotonic, but the interpolation Spearman is still higher than the supervised baseline.

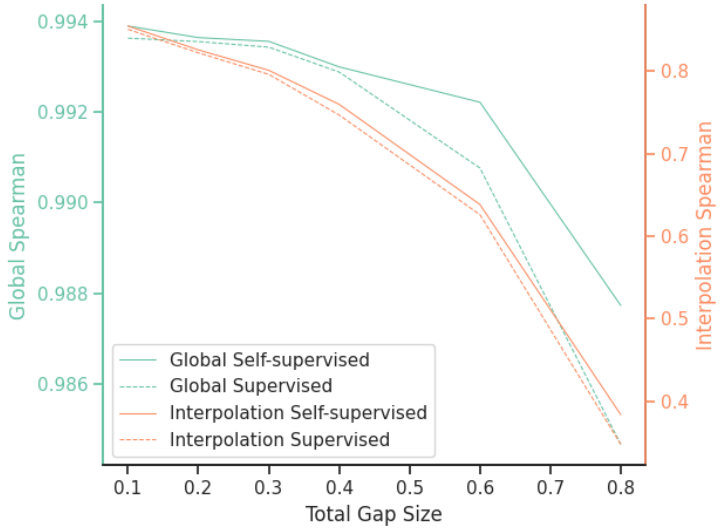

Figure 7: Global and interpolation Spearman correlations for the data with gaps, with some models that fractured removed. Now, we see that the global Spearman for high α_g is considerably higher than the supervised model. This suggests that the self-supervised model either fractured, causing its learned representation to become non-interpretable, or it did not fracture and learned a better representation than the supervised model.

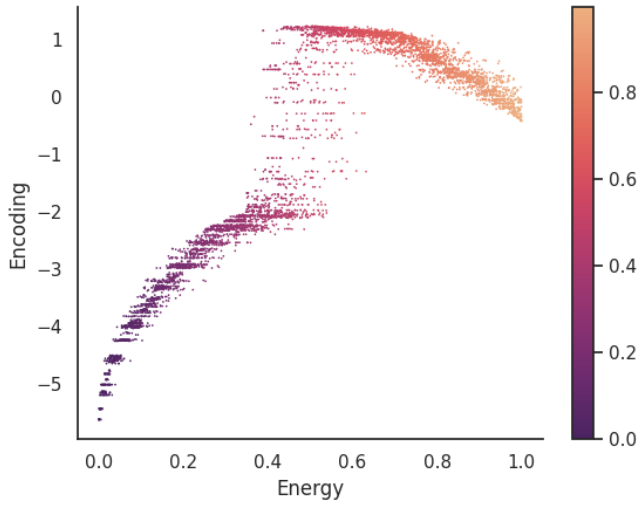

Figure 8: An example of fracturing. Note that two separate (low-energy and high-energy) embeddings are learned; there is a discontinuity between the two embeddings in the center. The model successfully interpolated each of the two embeddings until the discontinuity point.

fracturing, where the pre-existing performance difference becomes even more pronounced (see Subsection 4.1).

4 Discussion

In the previous section, we showed that self-supervised representations are of high quality (i.e. they map closely to the target, energy). Now we provide some more analysis of the actual representations learned as well as the model's training dynamics.

4.1 Fracturing

In Figure 8, we see an example of fracturing. In this sample, the model had a large gap of size 0.4 spanning energies from 0.3 to 0.7. The model learned two different embeddings: one low-energy embedding in which the encoding was positively correlated with the energy, and one high-energy embedding in which the encoding was negatively correlated with the

energy.

In the interpolated gap region, both embeddings continue—and are interpolated with some degree of success—until a sudden discontinuity in the encoding as the model switches from one embedding to the other. Because of this discontinuity and the resulting nonlinearity, the Spearman correlation for this example was quite low. However, our inspection of this particular case suggests that the model is actually interpolating quite well, given that there must be a discontinuity in the middle as the model switches from one embedding to the other.

Fracturing is more likely to happen with a larger total gap size or a smaller number of gaps, i.e. larger individual gap size. It is not always clear when fracturing occurred, but one sure indicator is when different visible (non-gap) regions from the training set are oppositely correlated with the energy. Note that it is possible for two different embeddings to be correlated with energy in the same direction, so the different-signs behavior is sufficient but not necessary for fracturing. According to this indicator, fracturing only occurred in 2 out of 5 trials, or 40% of the time. If each visible segment was assigned a positive or negative correlation at random, we expect this indicator to be positive seven out of eight times, or 87.5% of the time. Therefore, the model is still performing much better than chance, suggesting it has partial ability to recover the global structure, even when gaps cover 80% of the original input space.

We note that fracturing is not always bad behavior on the model's part. There are situations where the model should truly learn multiple discontinuous embeddings. A simple, although unrealistic, example is to take two very different datasets, such as ImageNet [16] and then our own pendulum dataset, and combine them into a composite dataset. ImageNet is composed of high-quality images of the natural world, and the views would be generated by standard computer vision augmentations such as cropping. This data would have a very different structure than the pendulum dataset, which is time-based and is visually extremely

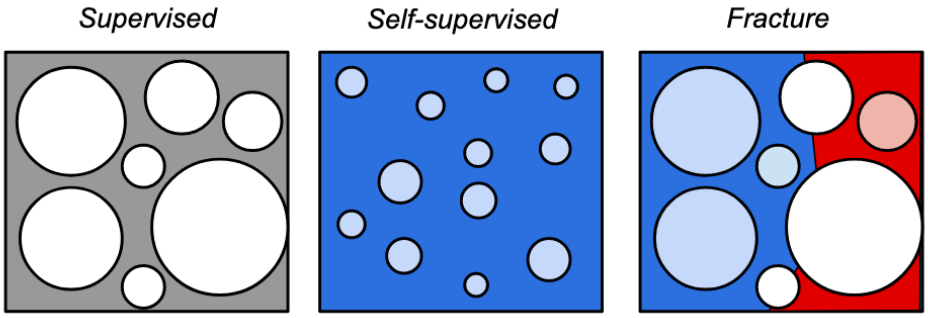

Figure 9: Certain regions of the input regions are missing from the training set (circles) which the model must learn to interpolate, akin to swiss cheese. *(Left)* A supervised model is able to interpolate even with large missing regions, although not that well. *(Center)* A self-supervised model interpolates very well when the missing regions are small. *(Right)* The self-supervised model fractures when the missing regions are too large, and learns two separate embeddings (blue and red). Within each embedding, it is still able to interpolate; however, interpolating between embeddings completely is impossible.

different. In this case, the model should learn a different embedding for ImageNet images than for the pendulum images. Forcing them onto one continuous embedding would likely give misleading results. Therefore, the question becomes one of balance, not of preventing fracturing entirely.

4.2 Relative density

We observe that the representation tends to get more dense wherever the Spearman correlation decreases. This is very obvious in the numerical noise test, where the range of the representations goes from roughly 7 for noiseless representations to less than 1 for very noisy representations. Moreover, we observed that the range of the representation is extremely consistent across trials and has no relation to the strength of gradient clipping. The representation range seems to be an intrinsic property of the dataset itself.

The range does change quite a bit, however, when the loss function temperature is changed. Because of this, we think this is a property that arises from the InfoNCE loss

we used. Therefore, we present a heuristic argument suggesting why density should increase in areas of low accuracy.

First, we will define density a little more rigorously. Let $F : [0, 1] \to \mathbb{R}$ be our encoder with a one-dimensional representation[1]. Over an interval, we can write

$$\begin{aligned}\text{density}([e_1, e_2]) &= \frac{|e_2 - e_1|}{\max_{e_1 \le e \le e_2} F(e) - \min_{e_1 \le e \le e_2} F(e)} \\ &\approx \frac{0.5 \cdot |e_2 - e_1|}{Q_3(\{F(e) : e_1 \le e \le e_2\}) - Q_1(\{F(e) : e_1 \le e \le e_2\})},\end{aligned}$$

where Q_1 and Q_3 denote the first and third quartiles, respectively. The numerator is simply to normalize the density; this density can be understood as inverse of the the range of F on the interval $[e_1, e_2]$. In practice, we preferred the second approximate formula as it is a bit more stable for outliers.

As its name suggests, the InfoNCE loss has information theoretic properties. In particular, when optimized, the similarity function $f(x, y)$ (see InfoNCE definition) should satisfy, for some constant c not depending on x, y,

$$-\text{dist}(x, y)^2/\tau = \log f(x, y) = c + \log \frac{p(x, y)}{p(x)p(y)} = c + \log \frac{p(x|y)}{p(x)},$$

where the expectation of the last quantity is called *mutual information* [10]. Mutual information is a measure of how much the occurrence of event x will shed light on the probability of event y happening, or vice versa; in other words, it is roughly a measure of how much information x and y share. It is 0 when x and y are completely independent. Given that $x \neq y$, in a very noisy setting, x and y have much more overlap and therefore share more information than in a zero noise setting. Therefore, when the noisiness increases, the mutual information increases, and thus $-\text{dist}(x, y)^2/\tau$ should increase, or x and y should become closer in the representation. This means the range of F on any given interval gets smaller,

[1]Since in general the encoder does not perfectly map all images of a pendulum with a given energy to the same point, no such F actually exists. However, F is a good construct and approximation for the behavior of the encoder which is mostly time-invariant. This statement is actually slightly incorrect since the encoder takes *images* as inputs; it's more correct to understand F as a combination of a stochastic image generation function I and the encoder, which is not stochastic. These two together map energies to points within the representation. We stated it this way for brevity.

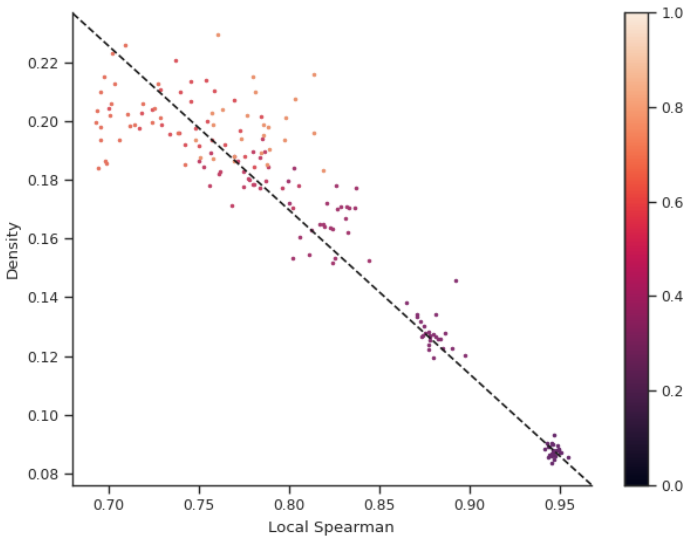

Figure 10: The local Spearman correlations (7 per trial) are plotted against the density on that seventh of the representation. Note the very strong inverse correlation between Spearman correlation and density here. Additionally, the points are colored by the energy in the seventh they came from. Higher energy pendulums have lower Spearman correlations and higher densities; this happens all within a single representation

and so the density becomes higher.

In fact, this pattern even appears *within individual representations*, where regions of lower accuracy tend to be denser. In our vanilla trials, we observed that the region corresponding to low-energy pendulums always had lower density than the region corresponding to high-energy pendulums (see Figure 10). At the same time, the Spearman correlations for high-energy pendulums was always significantly lower, which we believe may be due to the intrinsic local properties of the convolutional ResNets we used. Within a single representation, the most difficult area (high noise) corresponded to the densest area in that representation.

If this pattern holds true for higher-dimensional representations generated using InfoNCE or similar losses, there are some notable implications.

- First, from a practical point of view, we can understand where self-supervised models are struggling by identifying regions of high density and seeing which inputs they correspond to.

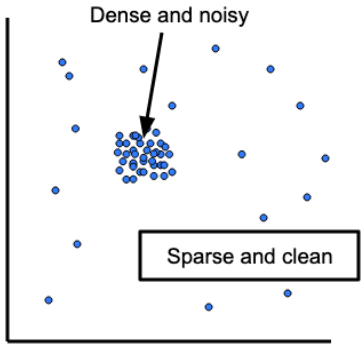

Figure 11: What the geometry of more complicated representations may look like: dominated by sparse regions of low noise, with dense regions of high noise interspersed throughout.

- Second, as long as density can be used as a reproducible metric of difficulty, measuring the density of different regions could be used to study the intrinsic difficulty of various related problems within a single self-supervised representation.

- Geometrically, the representation will be dominated by areas of high accuracy; this is more efficient spatially and information-wise than an even distribution.

- Humans do not pay much attention to things they do not understand, likely because it is more efficient time- and energy-wise. This behavior seems analogous to the density phenomenon here.

4.3 Cropping and ambiguous data

In the crop test, the model receives a large number of blank images. These could correspond to a large number of possible energies. Therefore, the model has to select a particular value to assign these blank images to.

Theoretically, if we had a simple mean squared error loss instead of the InfoNCE loss, the optimal blank image energy assignment should be the expectation of the energy across all blank images. It is unclear if this is still optimal in the InfoNCE case, but empirically

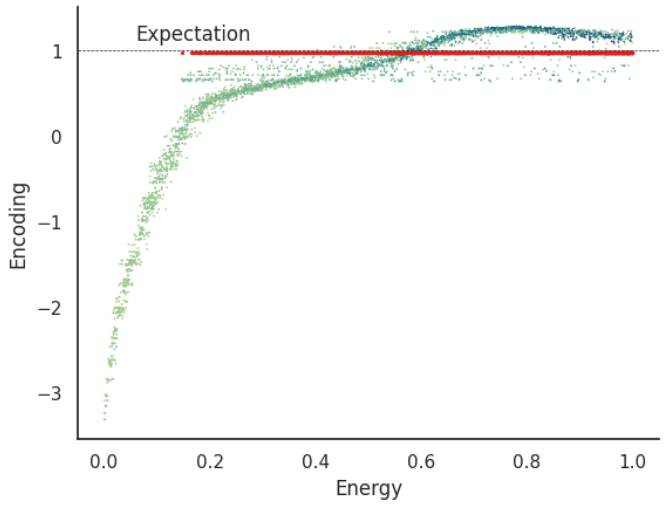

Figure 12: A plot of embedding against energy in a model which had a quarter (length-wise) of the inputs cropped. Blank images correspond to the red points along the horizontal line, since they must all have the same embedding. The expectation amongst blank images for the true encoding is plotted as a dashed horizontal line; note how close it is to the encoding of the blank images. This suggests that the model sends ambiguous data to the average. Additionally, we may note the range of energies with blank images (on the right) has a much denser encoding than the energies without blank images (on the left)—another example of the density effect (see Subsection 4.2).

the model does seem to assign the blank images to the expectation. This reveals another property of these self-supervised representations: *ambiguous data is sent to the average.*

We note that the presence of this ambiguous data reduces performance on the rest of the dataset only in the self-supervised model. Thus, while the supervised model is rather resistant against ambiguous data, the same is not true for the self-supervised model. We can also observe how the density also increases significantly, but only at energy levels where there are blank images. This provides us with another example of how more difficult parts of the representation tend to be denser.

5 Future Work

The biggest remaining question is why exactly is self-supervised learning better, given that supervised learning *should* be optimal? We tried to find out the reason, but were not able to (see Appendix 1). We also have studied only one-dimensional representations in depth. It remains to be seen whether our results will generalize to two or three dimensions, or to extremely high dimensional data such as natural images.

Moreover, given that we have identified both some of the strengths and pitfalls of contrastive learning (i.e. fracturing), how do we engineer models that maintain its strengths while overcoming its pitfalls? For example, what could prevent a model from fracturing? Using our insights to improve self-supervised models is a clear next step.

Finally, two related methods to the contrastive learning we used here are non-contrastive learning and semi-supervised learning [5, 17]. These are both newer techniques which seem to hold greater promise than the contrastive approach, so it is important to study the representations generated by these methods as well. Given that a lot of our results seem to be related precisely to the InfoNCE loss we used (although we do not think they are specific to InfoNCE), it may be that our results no longer hold true for non-contrastive or semi-supervised approaches.

6 Conclusion

In representation learning for computer vision, self-supervised models already outperform supervised models. One hypothesis to explain this phenomenon was that the labels used for supervised learning were simply inadequate to describe images. However, in our study, we found that even when the supervised model is given the ground truth, it still does not perform as well as the self-supervised model. This suggests that non-optimal labels may not be the main reason why self-supervised models outperform supervised ones.

When the task is made moderately more difficult, such as the presence of a moderate level of noise, or moderately sized gaps in the training dataset, the performance difference between the self-supervised model and its supervised counterpart are even more pronounced. We think this is due to a better representation learned by the self-supervised model. It also suggests that self-supervised models may ultimately be more powerful than supervised models in rather ideal settings like ours.

Furthermore, representations learned by self-supervised models have interesting properties. They are locally denser in regions of higher difficulty or noise. They also map ambiguous data to the expected input. These two results suggests that investigations of the geometry of learned representations on larger and more realistic datasets like ImageNet [16] may be fruitful.

Finally, we also see evidence of the limits of self-supervised learning. At high levels of noise or large amounts of ambiguous or missing data, the self-supervised model learns a considerably worse representation than its supervised counterpart. In practice, self-supervised models have been found to be data-hungry, performing better with vast datasets (for example, the 1-billion image Instagram dataset [3]). Our study suggests that without a lot of data to learn from, self-supervised representations break down or fracture while their supervised counterparts do not. This may be the reason why self-supervised methods seem to need significantly larger datasets than supervised methods.

7 Acknowledgments

Thank you to my mentors Rumen Dangovski and Peter Lu for this project, all of their time spent guiding me, and being so thoughtful and patient with all my questions. Thank you to my PI Marin Soljačić. Thank you to my tutor John Rickert for suggestions, and my first week TAs Lucy Cai and Ishan Khare for LaTeX help. Thank you to Sonia Smith, Reagan

Choi, and my last week TA Yunseo Choi for reading through this paper. Thank you to CEE, MIT, RSI and its sponsors for providing and organizing this research opportunity. Thank you to my sponsors, CACI International Inc., Arvind Parthasarathi, Jiajun Li, June Sabo, Katherine A. Paur, and Matt DeBergalis, for making this possible.

References

[1] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive, 2020.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning, 2019.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020.

[5] X. Chen and K. He. Exploring simple siamese representation learning, 2020.

[6] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[7] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers, 2021.

[8] A. Tamkin, M. Wu, and N. Goodman. Viewmaker networks: Learning views for unsupervised representation learning, 2021.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[10] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding, 2018.

[11] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021.

[12] T. Hua, W. Wang, Z. Xue, Y. Wang, S. Ren, and H. Zhao. On feature decorrelation in self-supervised learning, 2021.

[13] Y. Bansal, P. Nakkiran, and B. Barak. Revisiting model stitching to compare neural representations, 2021.

[14] S. J. Wetzel, R. G. Melko, J. Scott, M. Panju, and V. Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks. *Physical Review Research*, 2(3), Sept. 2020.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[17] M. Assran, M. Caron, I. Misra, P. Bojanowski, A. Joulin, N. Ballas, and M. Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples, 2021.

A Appendix

A.1 The reason for self-supervised vs. supervised differences

We did several tests to try to determine why the self-supervised model outperformed the supervised model. We ultimately did not find a complete explanation, but here is some discussion on some possibilities.

Overfitting. Supervised networks are known for a tendency to overfit. We tried to see if the difference was because of overfitting by testing out the Spearman correlation on the original training set for the encoder. When using training data to evaluate the supervised model, we got even worse results than on the testing set (.9936 vs .9937, .8030 vs. .8044), likely because the model seemed to be optimizing for loss, which at some point seems to force a decreased Spearman correlation. This suggests that some combination of a nonoptimal loss function plus overfitting may be at work.

Loss/metrics. We tried using ℓ_1 loss, which encourages the neural network to learn the mode, as opposed to mean-squared error loss, which encourages the network to learn the mean. This did not improve performance—it actually seemed to slightly decrease performance, but we did not run enough trials to see whether there was a clear difference once it was apparent that it was not better than the self-supervised model.

It is possible that with a loss function optimized to improve the Spearman correlation, we may see higher performance from the supervised network. Especially in light of the fact that the Spearman correlations are worse on the training set than the testing set, this seems like a good explanation. However, we saw in tests that using the random forest inverse regressor (see Appendix 5 for more details), that for large images in particular, the self-supervised model still outperformed the supervised model. Therefore, it seems likely to not be Spearman-correlation specific.

Dataset: low resolution and discretization. Another possibility is that this is a

property of the dataset we used. One concern in particular is the low image resolution we used; that because there were so few possible output images relative to data size, the supervised model was incentivized to overfit. However, given that in the large-images dataset the self-supervised model did *even better*, this is not the case.

It is possible that the discretization of the data may be an issue. Since the labels given to the supervised model are continuous, while the pendulum bobs have hard stops (i.e. are either blue or not blue, or red or not red, with no in-betweens), there may be a fundamental mismatch between labels and data. If this were truly the source of the difference, though, it seems like it would be at least partially remedied with larger resolutions, which it was not. We were not able to conduct any tests with truly continuous data though.

Labels. Since the main difference between the supervised and self-supervised model is the labels, perhaps the issue may be with the labels. Given that the labels are ground truth, however, it is hard to imagine what could be wrong with the labels (other than they are continuous and the data is discrete).

Tuned training parameters. We note that we did not really try to tune the training parameters on the supervised model to optimize performance. However, given some rather standard values, we did not really try to tune the training parameters on the self-supervised model either, other than introducing gradient clipping because of exploding gradients. It's possible, but unlikely, that we just had some rather ideal parameters for the self-supervised model by a lucky accident.

A.2 Image generation

See Figure 13 and Figure 14.

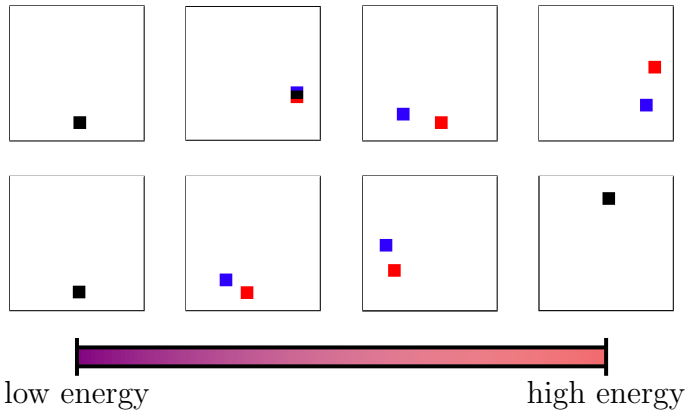

Figure 13: The pendulum graphical dataset (vanilla).

A.3 Cosine similarity

Figure 15 is a plot of a ℓ_2-normalized representation generated using cosine similarity. Note that it appears 2-dimensional but in effect is actually only 1-dimensional because of the initial normalization step in the loss. A very strong correlation between energy and position can be observed.

A.4 Exact training setup

See Figure 16.

A.5 2D and 3D analysis

As can be seen in Subsection 3.1, the dimensionality of the data stays constant even if the representation is allowed to be two or three dimensions.

In Figure 4, we plotted the spectral embedding of the representation against energy. While there is a clear correlation in the examples we plotted, in some cases, the spectral embedding does not really work on the learned representations. Other methods we tried, such as isomap or locally linear embedding, gave incoherent results.

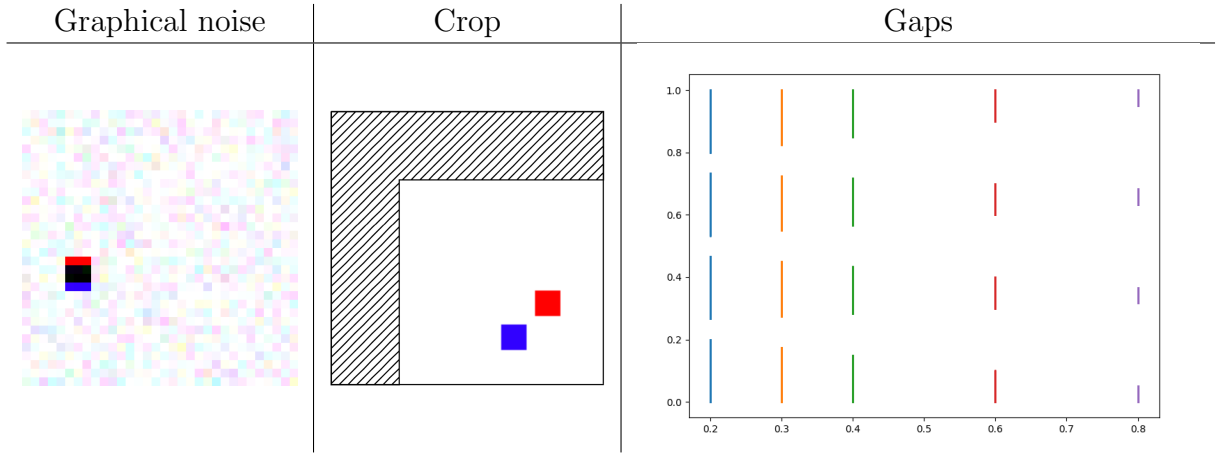

Figure 14: *Graphical noise.* Here is an example from a dataset with $\varepsilon_g = 0.25$. The pendulum itself is unchanged. *Cropping.* Here is a schematic of what cropping does to images. Note that it is possible for the bob to be outside of the image; the model then receives an ambiguous blank input image. *Gaps.* For total gap length, α_g, equal to $0.2, 0.3, 0.4, 0.6$, and 0.8 (x-axis), the five values we used in testing, we plot which energy levels allowed during testing (y-axis). Observe that the number of gaps are constant and that they are uniformly spaced, and as the total gap size increases the range of energies becomes dominated by pendulums that are *not* seen during testing.

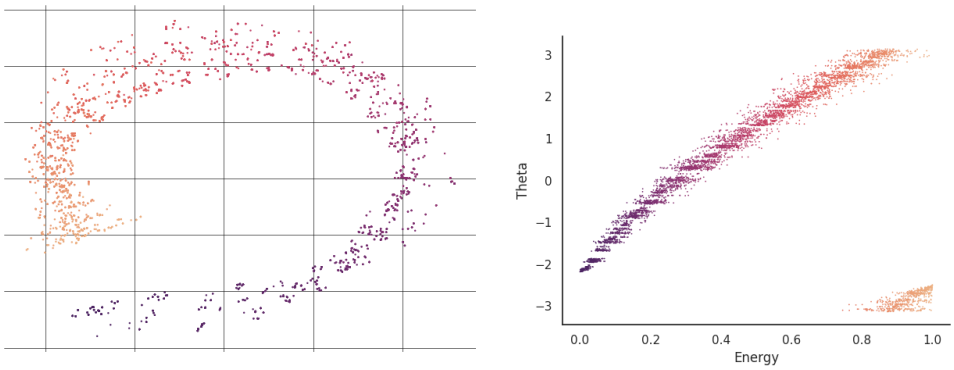

Figure 15: The model also learns well with cosine similarity (instead of negative Euclidean distance) as a measure of similarity between representations. *(Left)* The actual 2D representation. It is colored by energy values. A very strong correlation between position and energy is evident. Note the wrap-around effect, where pendulums of energy 0 and 1 are forced to be close together. *(Right)* A plot of θ (the angle a point forms against the positive x-axis) against the energy. Here, the correlation becomes even more obvious.

Variable	Value
Training dataset size	5120×20
Testing dataset size	5120×20
Image size	32×32
Time difference between images	0.5
Minimum energy	0
Maximum energy	reach the top; scaled to 1
Noise	0
Bob size	3×3
Epochs	100
Optimizer	SGD
Batch size	512
Warmup epochs	5
Learning rate	0.02
Final learning rate	0.0
Decay scheduler	Cosine
Weight decay	0.001
Distance metric	Euclidean
Temperature	0.1
Gradient clipping	3

Figure 16: Exact training setup used.

Dimensions	2	3
Self-supervised	.00322	.00323
Supervised Baseline	.00324	.00324

Figure 17: Root-mean-square error of a random forest inverse regressor, which predicts energies from the representation. The self-supervised and supervised models are closely matched; therefore, the self-supervised model does not have significantly degraded performance if the number of dimensions in the output representation is mismatched with the true dimensionality of the input dataset. We note that in testing these numbers seemed to be rather noisy so this data should be considered with a grain of salt.

Despite not being able to reduce the data to two dimensions, we can see whether the energy data is still well-preserved in the 2D and 3D representations by training an *inverse regression* that takes the representation and attempts to map them to energy levels. If the energy level was lost in the 2D or 3D representations, no regression should be able to capture the lost information. This also mimics, to some degree, adding a downstream task to the representation.

Our regressor of choice is a random forest regressor for simplicity. It is trained on one half of a 5120-pendulum dataset and predicts on the other half. We use root-mean-square error. Figure 17 displays the results (with 10 trials of the regressor against 5 trials for the encoder).

We also observe that the representations in 2D and 3D do have some additionally dimensionality to them, not being entirely one-dimensional. Surprisingly, the other dimensions seem to encode meaningful information. For example, in Figure 18, the second dimension of the 2D embedding seems to correspond to vertical position. It is not clear why this is the case, especially given that vertical position is certainly not invariant with time. More investigations into this phenomenon will be needed.

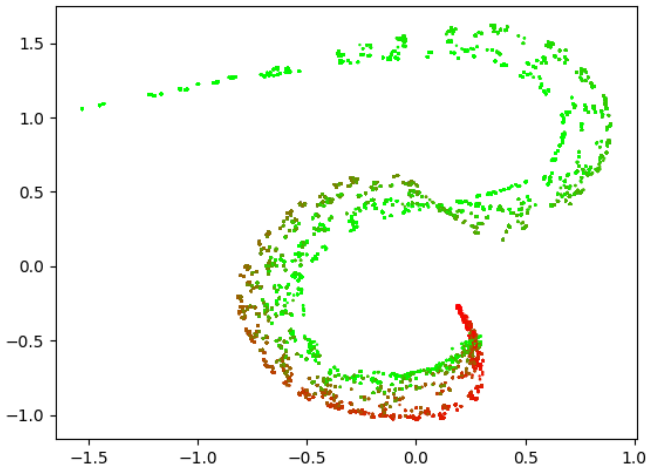

Figure 18: Coloring a 2D representation (red to green) by vertical position instead of energy. The first dimension, along the length of this "ribbon", is known to correlate with energy (not pictured). The second dimension appears to correlate with vertical position. This suggests the second and additional dimensions in higher dimensional representations is meaningful although not time-invariant.