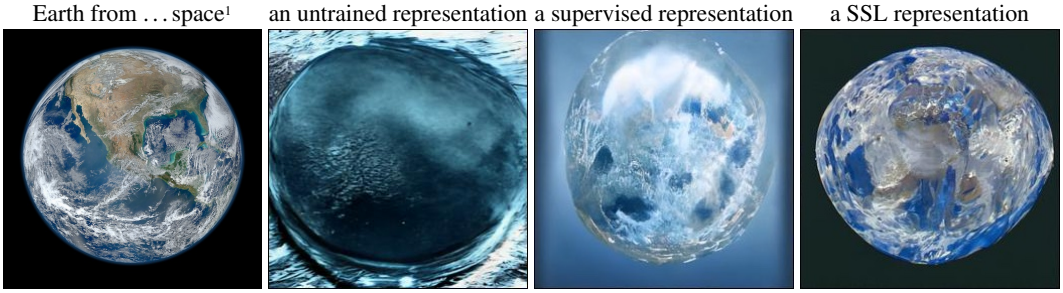
HIGH FIDELITY VISUALIZATION OF WHAT YOUR SELF-SUPERVISED REPRESENTATION KNOWS ABOUT

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering what is learned by neural networks remains a challenge. In self-supervised learning, classification is the most common task used to evaluate how good a representation is. However, relying only on such downstream task can limit our understanding of how much information is contained in the representation of a given input. In this work, we study how to visualize representations learned with self-supervised models. We investigate a simple gradient descend based method to match a target representation and show the limitations of such techniques. We overcome these limitations by developing a representation-conditioned diffusion model (RCDM) that is able to generate high-quality inputs that share commonalities with a given representation. We further demonstrate how our model's generation quality is on par with state-of-the-art generative models and how the representation conditioning brings new avenues to analyze and improve self-supervised models.

Earth from . . . space[1] | an untrained representation | a supervised representation | a SSL representation



1 INTRODUCTION AND MOTIVATION

Approaches aimed at learning useful representations, from unlabeled data, have a long tradition in machine learning. These include probabilistic latent variable models and variants of auto-encoders (Ackley et al., 1985; Hinton et al., 2006; Salakhutdinov et al., 2007; Vincent et al., 2008; Kingma & Welling, 2014; Rezende et al., 2014), that are traditionally put under the broad umbrella term of *unsupervised learning* (Bengio et al., 2013). More recent approaches, under the term of *self-supervised learning* (SSL) have used various kinds of "pretext-tasks" to guide the learning of a useful representations. Filling-in-the-blanks tasks, proposed earlier in (Vincent et al., 2008; 2010), later proved remarkably successful in learning potent representations for natural language processing (Vaswani et al., 2017; Devlin et al., 2019). Pretext tasks for the image domain include solving Jigsaw-puzzles (Noroozi & Favaro, 2016), predicting rotations or affine transformations (Gidaris et al., 2018; Zhang et al., 2019) or disriminating instances (Wu et al., 2018; van den Oord et al., 2018).

The latest, most successful, modern family of SSL approaches for images (Misra & Maaten, 2020; Chen et al., 2020; Chen & He, 2020; He et al., 2020; Grill et al., 2020; Caron et al., 2020; 2021; Zbontar et al., 2021; Bardes et al., 2021), have two noteworthy characteristics that markedly distinguish them from traditional unsupervised-learning models such as autoencoder variants or GANs

[1]We use representations of the real picture of Earth on the left (source: NASA) as conditioning for RCDM. We show samples (resolution 256×256) in cases where the representations (2048-dimensions) were obtained respectively with a random initialized ResNet50, a supervised-trained one, and a SSL-trained one. More samples in Figure 23.

(Goodfellow et al., 2014): a) their training criteria are not based on any input-space reconstruction or generation, but instead depend only on the obtained distribution in the representation or embedding space b) they encourage invariance to explicitly provided input transformations a.k.a. data-augmentations, thus injecting important additional domain knowledge.

Despite their remarkable success in learning representations that perform well on downstream classification tasks, rivaling with supervised-trained models (Chen et al., 2020), much remains to be understood about SSL algorithms and the representations they learn. How do the particularities of different algorithms affect the representation learned and its usefulness? What information does the learned representation contain? Empirical analyses have so far attempted to analyse SSL algorithms almost exclusively through the limited lens of the numerical performance they achieve on downstream tasks such as classification. Contrary to their older unsupervised learning cousins, due to characteristic a) highlighted above, modern SSL methods do not provide any direct way of mapping back the representation in image space, to allow *visualizing* it. This is the motivation and goal of our work: enable the visualization of representations learned by SSL methods, as a tool to improve our understanding.

More precisely, we suppose that we are given a mapping function f – a (part of) a SSL or otherwise trained neural network – that takes an input image $\boldsymbol{x} \in \mathbb{X}$ and maps it to a representation $\boldsymbol{h} \in \mathbb{H}$ as in $\boldsymbol{h} = f(\boldsymbol{x})$. The input space $\mathbb{X}$ will typically be RGB pixel space represented as $\mathbb{X} = [-1, 1]^D$, and the representation space $\mathbb{H}$ will be the output space of a deeper layer. Without loss of generality, we will always denote the representation space's dimension by K as in $\mathbb{H} = \mathbb{R}^K$. Now we want, when given a specific representation $\boldsymbol{h} \in \mathbb{H}$, to visualize what inputs $\boldsymbol{x}$ yield this representation. As f is typically *not* bijective, e.g. if it computes a higher level representation of reduced dimension, there may be many inputs that yield that same representation. We will examine two approaches for visualizing these: the first approach (Section 2) aims at finding inputs $\boldsymbol{x} \in \mathbb{X}$ that merely map very closely to a given target $\boldsymbol{h}$. But we will see that most of these will not resemble natural images. The second approach (Section 3), aims at finding inputs that not only map to the target $\boldsymbol{h}$ but are also visually recognizable images. For this we will build a conditional generative model that (implicitly) models $p(\boldsymbol{x}|\boldsymbol{h})$ and allows to sample diverse $\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{h})$. For reasons that we will explain later, we opted for a conditional diffusion model, inspired by Dhariwal & Nichol (2021), for our conditional generative model.

This paper's main contributions are:

- To study the interplay between the set of inputs that map to a given network embedding, the perceptual quality of those matched inputs and the model Jacobian matrices.
- To devise a conditional diffusion model architecture suitable for conditioning on large vector representations s.a. SSL representations.
- To showcase its usefulness for qualitatively analyzing SSL representations and embeddings, by shedding light on what information about the input image is or isn't retained in them.

As will become clear, the benefits of a conditional model are two-fold. First, by repeatedly sampling from the same conditioning, one can identify the factors that are identical through the samples: this identifies what is encoded in the representation. Second, the non-constant factors of variations represent what is not encoded in the representation.

2 Matching a representation by following input gradients

We want to visualize what kinds of input images would be mapped to the same representation $\boldsymbol{h}$ as that of an input image $\boldsymbol{x}$ by a given network function f. This section proposes to sample inputs that fall into that equivalence class by solving an explicit optimization problem, while the next section leverages a conditional diffusion model.

2.1 Peeking into the representation-matching input set

We call the set of inputs that a trained network function f maps to a given representation $\boldsymbol{h}$ the *representation-matching input set*, defined formally as

$$\mathcal{S}(\boldsymbol{h}) \triangleq \{\boldsymbol{x}' \in \mathbb{X} : d(\boldsymbol{h}, f(\boldsymbol{x}')) = 0\}, \tag{1}$$

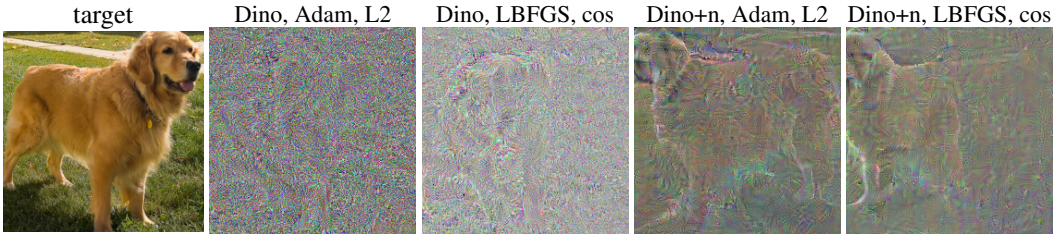


Figure 1: Gradient based samples from $\mathcal{S}(\boldsymbol{h})$. Leftmost image $\boldsymbol{x}$ is used to obtain target SSL representation $\boldsymbol{h}$ (2048 dimensions) with either a standard DINO-ResNet50 (Dino) or one trained with additive noise as extra augmentation (Dino+n). A random initialized input is moved so that its representation will match $\boldsymbol{h}$, by minimizing either L2 or cosine distance using Adam or L-BFGS respectively (indicated in column headers). We display the samples $\boldsymbol{x}^{(T)}$ obtained after $T = 10,000$ iterations obtained from the respective optimizers and distances, in all cases starting from a random Gaussian image as $\boldsymbol{x}^{(0)}$. These $\boldsymbol{x}^{(T)}$ have the same SSL representation as $\boldsymbol{x}$ or very close (relative distance $0.4\%, 0.1\%, 3.6\%, 3.3\%$ respectively, see details in appendix Tab. 1,2,3), but do not resemble natural images. Samples obtained from Dino+n look slightly more natural: we distinguish faint edges similarly shaped to the original. Similar experiments but applied on the lower-dimensional projection head embeding are reported in Figure 8 in appendix.

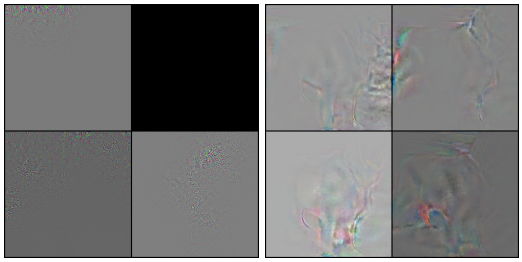


Figure 2: Depiction of four rows of the Jacobian matrix $\boldsymbol{J}_f(\boldsymbol{x})$ for the input $\boldsymbol{x}$ given in Figure 1, with f being a Resnet50 trained with standard DINO (**left**) and additive noise (DINO+n) (**right**). The Jacobian matrix in the noisy case shows more structures looking somewhat more natural-image like. Recalling (2), this observation justifies the more natural images observed in Figure 1. Additional rows provided in the appendix in Figure 9)

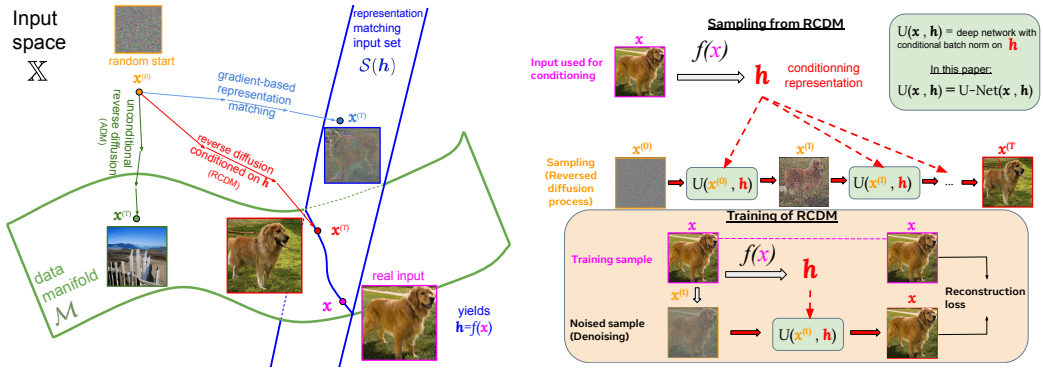
where d could be any desired distance[2]. We would like to see what kinds of "images" $\mathcal{S}(\boldsymbol{h})$ contains (besides the $\boldsymbol{x}$ that we may have used to obtain representation $\boldsymbol{h}$ to begin with). We tackle this problem through a simple gradient-based optimization. We start from a random input $\boldsymbol{x}^{(0)} \in \mathbb{X}$, sampled from a basic distribution (s.a. uniform). We then performing T gradient steps in input-space towards minimizing objective $d(f(\boldsymbol{x}), \boldsymbol{h})$, i.e. to "match the representation", yielding final sample $\boldsymbol{x}^{(T)}$. Note that the minimizer is usually not unique, so that we can obtain quite different $\boldsymbol{x}^{(T)}$ depending on the random $\boldsymbol{x}^{(0)}$ we started from.

2.2 Do samples from $\mathcal{S}(f(x))$ look like x?

We performed the above-described procedure to sample examples from $\mathcal{S}(f(\boldsymbol{x}))$, using for f the same ResNet50 backbone (He et al., 2016) trained with the DINO SSL criterion (Caron et al., 2021) on ImageNet (Russakovsky et al., 2015). We took $\boldsymbol{x}$ from the validation set of ImageNet. In addition to standard DINO training, we also trained a second SSL network (termed DINO+n) that uses independent additive noise as extra augmentation[3] which led to 71% top-1 Imagenet accuracy. Examples of obtained images are provided in Figure 1. Appendix A has more details and more results for these experiments. We see that even though gradient-based input optimization manages to produce samples that match a target embedding representation, this technique fails to produce realistic images. *Mapping to the same SSL representation as a natural image is not sufficient for being a similar realistic-looking image.*

[2]In practice, we may be content with finding elements of a relaxed representation-constrained set $\mathcal{S}_\epsilon(\boldsymbol{h}) \triangleq \{\boldsymbol{x}' \in \mathbb{X} : d(\boldsymbol{h}, f(\boldsymbol{x}')) \leq \epsilon\}$ allowing for a small tolerance ϵ

[3]The motivation was to learn a smoother map f for gradient-based representation matching to be easier. Although the optimization proved no more difficult in the non-noised case, DINO+n yields *qualitatively* markedly different samples, where one can more easily distinguish natural-image like edges. The reason is unclear; one hypothesis is that to reliably discriminate noised instances the representation must focus more on edges.



(a) (b)

Figure 3: (a) Illustration of considered image generation methods. A real input $\boldsymbol{x}$ yields representation $\boldsymbol{h}$. All methods start from a random noise image $\boldsymbol{x}^{(0)}$. Gradient-based representation matching (light blue arrows) will move it towards $\mathcal{S}(\boldsymbol{h})$ i.e. until its representation matches $\boldsymbol{h}$, but won't land on the data-manifold $\mathcal{M}$. Unconditional reverse diffusion (ADM model, green arrows) will move it towards the data manifold. Our representation-conditioned diffusion model (RCDM, red arrows) will move it towards $\mathcal{M} \cap \mathcal{S}(\boldsymbol{h})$, yielding a different natural-looking image with the same given representation. (b) Representation-Conditionned Diffusion Model (RCDM). From a diffusion process that progressively corrupts an image, the model learns the reverse process by predicting the noise that it should remove at each step. We also add as conditioning a vector h, which is the representation given by a SSL or supervised model for a given image c. Thus, the network is trained explicitly to denoise towards a specific example given the corresponding conditioning. The diffusion model used is the same as the one presented by Dhariwal & Nichol (2021) with the exception of the conditioning on the representations.

The gradient directions are not enough. The updates producing the sequence $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)}$ all follow a trajectory that only involves the Jacobian matrix of deep network f at each step. This is due to the chain rule of calculus and reads as

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \underbrace{\boldsymbol{J}_f(\boldsymbol{x}^{(t)})^T \boldsymbol{u}(\boldsymbol{x}^{(t)})}_{\text{linear combination of Jacobian matrix rows}} , \tag{2}$$

where $\boldsymbol{u}(\boldsymbol{x}^{(t)})$, the linear combination coefficients, is given by $\nabla_{d(f(\boldsymbol{x}),.)}(f(\boldsymbol{x}^{(t)}))$. As a result, it is clear that $\boldsymbol{x}^{(t+1)}$ is constrained to be within the affine space spanned by $\boldsymbol{J}_f(\boldsymbol{x}^{(t)})^T \boldsymbol{u}(\boldsymbol{x}^{(t)})$ and shifted by $\boldsymbol{x}^{(t)}$. Given that f is a mapping to $\mathbb{R}^S$, the dimension of that affine space is at most S.

We see that the representation and mapping function f obtained through SSL training are by themselves not sufficient to recover corresponding natural-image-like inputs.

3 Conditioning a diffusion model on representation h

In the previous sections we saw that gradient based model inversion was possible but lacked quality in the obtained samples. In this section, we instead propose to build a novel conditional diffusion process whose goal is to directly generate realistic images that match a given target representation.

Given a representation $\boldsymbol{h}$ the failure of the above method suggests we need a way to further constrain the type of samples we generate, beyond the mere constraint of belonging to $\mathcal{S}(\boldsymbol{h})$. More precisely, we want to be able to sample, among $\mathcal{S}(\boldsymbol{h})$, inputs that are more like the training data (here natural images), i.e. that are likely under the same distribution. That is we would like not merely to find $\boldsymbol{x}' \in \mathcal{S}(\boldsymbol{h})$ but rather to sample $\boldsymbol{x}' \sim p(\boldsymbol{x}|\boldsymbol{h})$. Informally we might picture the *set* of likely natural images (points whose density $p(\boldsymbol{x})$ is above some threshold) as a subset of $\mathbb{X}$ that we will loosely refer to as the "data manifold" $\mathcal{M}$. Where our first approach attempted to sample points more or less uniformly within $\mathcal{S}(\boldsymbol{h})$, modeling and sampling form $p(\boldsymbol{x}|\boldsymbol{h})$ will more likely produce points from $\mathcal{M} \cap \mathcal{S}(\boldsymbol{h})$. We propose to train a *conditional diffusion model* to implicitly model $p(\boldsymbol{x}|\boldsymbol{h})$ and allow sampling from it.

While we could have considered other conditional generative approaches (some of the alternatives will be discussed in section 4), the choice of the reverse diffusion approach (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song & Ermon, 2020b;a; Song et al., 2020; Nichol & Dhariwal, 2021) is not entirely arbitrary. It is motivated by the remarkable quality of image generation they recently proved capable of (Dhariwal & Nichol, 2021), as well as the closeness of their Langevin-MCMC-like generation process to the input-gradient-directed optimization we used to obtain samples in section 2. Indeed sampling from an reverse diffusion model similarly starts from a random noise image, and takes multiple steps in input space that can be thought of as (noisy) gradient steps on an implicit energy function[4]. Informally, one can think of these steps as progressively moving this initial random point closer to the "data manifold" $\mathcal{M}$. A reverse diffusion *conditioned* on $\boldsymbol{h}$ will move it towards $\mathcal{M} \cap \mathcal{S}(\boldsymbol{h})$. The three sampling approaches are depicted and contrasted in Figure 3a.

We base our work on the Ablated Diffusion Model (ADM) developed by Dhariwal & Nichol (2021) which uses a UNet architecture (Ronneberger et al., 2015) to learn the reverse diffusion process.Our conditional variant – called *Representation-Conditionned Diffusion Model* (RCDM) – is illustrated in Fig. 3b. To suitably condition on representation $\boldsymbol{h} = f(\boldsymbol{x})$, we replaced the Group Normalization layers of ADM by conditional batch normalization layers (Dumoulin et al., 2017) that take $\boldsymbol{h}$ as conditioning[5]. More precisely we apply a fully connected layer to $\boldsymbol{h}$ that reduces dimension to a vector of size 512. This vector is then given as input to multiple conditional batch normalization layers that are placed in each residual block of the diffusion model.

In contrast with Dhariwal & Nichol (2021) we don't use the input gradient of a classifier to bias the reversed diffusion process towards more probable images, nor do we use any label information for training our model – recall that our goal is building a visualization tool for SSL models that train on unlabeled data. Our batch normalization based conditioning is also different from the approach that was used by Dhariwal & Nichol (2021) when conditioning their super-resolution model on a low-resolution image. Their technique of upscaling and appending the conditioning image as extra channels to the input would not work for our application. Our representation $\boldsymbol{h}$ typically has 2048 "channels" with no spatial extent: upscaling it to the size of the input image would blow up memory constraints.

4 Related work

4.1 Gradient-based visualization techniques for Neural Networks

Many works (Erhan et al., 2009; Zeiler & Fergus, 2014; Simonyan et al., 2013; Selvaraju et al., 2016; Smilkov et al., 2017) used gradient based techniques to visualize what is learned by neural networks. The earliest of these (Erhan et al., 2009; Zeiler & Fergus, 2014) proposed the technique of activity-maximization to visualize what a specific neuron is sensitive to. The technique is similar to our first approach in 2, as it consists in optimizing a (randomly initialized) input $\boldsymbol{x}^{(0)}$ via consecutive input gradient steps, producing the inputs $\boldsymbol{x}^{(t)}, t = 1, \ldots, T$ such that $\boldsymbol{x}^{(T)}$ maximizes the activity of a specific unit in a DN. This led to successful interpretability of DNs internal features, especially when applied on a unit belonging to the first few layers of a DN (Cadena et al., 2018).

More recently, Caron et al. (2021) used the attention mask of transformers to perform unsupervised object segmentation. By contrast, our methods is not model dependant, we can plug any type of representation as conditioning for the diffusion model. Another possibility, explored in Zhao et al. (2020); Appalaraju et al. (2020); Ericsson et al. (2021), is to learn to invert the DN features through a Deep Image Prior (DIP) g_θ as in $\min_\theta d(g_\theta(f(\boldsymbol{x})), \boldsymbol{x})$. This however requires to train the DIP network for each feature-image pairs and only quantify how much information about $\boldsymbol{x}$ is retained in $f(\boldsymbol{x})$ while we are interested in finding all the $\boldsymbol{x}$s that are seen to have the same information content.

4.2 Conditional generative models

Several families of techniques have been developed as generative models, that can be trained on unlabeled data and then employed to generate images. These include auto-regressive models (Van Den Oord et al., 2016), variational auto-encoders (Kingma & Welling, 2014; Rezende et al., 2014),

[4]Diffusion models do not usually explicitly express an energy function, and cannot guarantee that their steps correspond to the derivative of an energy, but this remains a potent analogy.

[5]A similar technique was used by Casanova et al. (2021) for IC-GAN, discussed in the next section.

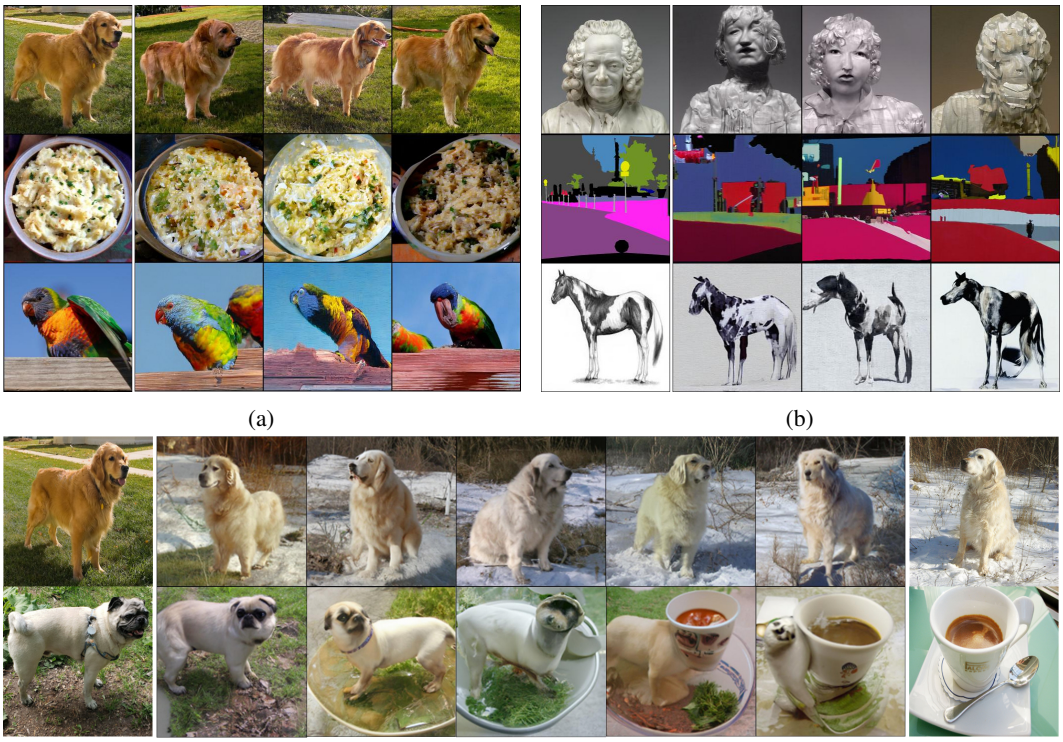
GANs (Goodfellow et al., 2014), autoregressive flow models (Kingma et al., 2016), and diffusion models (Sohl-Dickstein et al., 2015). Conditional versions are typically developed shortly after their unconditional versions (Mirza & Osindero, 2014; van den Oord et al., 2016). In principle one could envision training a conditional model with any of these techniques, to condition on an SSL or other representation for visualization purpose, as we are doing in this paper with a diffusion model. One fundamental challenge when conditioning on a rich representation such as the one produced by a SSL model, is that for a given conditioning $\boldsymbol{h}$ we will usually have available only a *single* corresponding input instance $\boldsymbol{x}$, precious few to learn a distribution. This can lead model training astray. By contrast a particularly successful model such as the conditional version of BigGAN (Brock et al.) conditions on a categorical variable, the class label, that for each value of the conditioning has a large number of associated $\boldsymbol{x}$ data.

One closely related work to ours is the recent work on Instance-Conditioned GANs (IC-GAN) of Casanova et al. (2021). Similar to us it also uses SSL or supervised representations as conditioning when training a conditional generative model, here a GAN (Goodfellow et al., 2014), specifically a variant of BigGAN (Brock et al.) or StyleGAN2 (Karras et al., 2020). However, their model is trained such that, from a specific representation $\boldsymbol{h}$, it learns to generate not only images that should map to this representation, but a much broader neighborhood of the training data. Specifically up to 50 training points that are nearest neighbors in representation space to $\boldsymbol{h}$. It is unclear whether such a GAN architecture could be trained successfully without resorting to a nearest neighbor set. IC-GAN is to be understood as a conditional generative model of an image's broad *neighborhood*, and the primary focus of this work was on developing a superior quality controllable generative model. By contrast we want to sample images that map as closely as possible to the original image in the representation space, as our focus is to build a tool to analyse SSL representations, to enable visualising what images correspond *precisely* to a representation.

As previously stated, our choice of a diffusion-based model rather than a GAN was motivated by this, by the simple stable training of such models, by the high quality of generated images demonstrated with the model we build on Dhariwal & Nichol (2021) that rivals that of GAN, and by the similarity of the input-space gradient-based sampling procedure with the simple approach we explored in section 2. While conditional versions of their diffusion model were already developed in Dhariwal & Nichol (2021), these were unsuitable for conditioning on high dimensional distributed representation s.a. those obtained with SSL models, as we discussed in details in section 3. This prompted us to develop the architecture variant of this paper. Despite their qualities, diffusion models also have drawbacks, in particular they are resource-hungry and slow for generation. It is thus very likely that alternative approaches for representation-conditioned generative models will be developed and employed for analysis and visualisation purposes in the future.

5 EXPERIMENTS USING RCDM TO MAP BACK REPRESENTATIONS TO IMAGES

Our first experiments aim at evaluating the abilities of our model to generate realistic-looking images whose representations are close to the conditioning. To do so, we trained our Representation-Conditionned Diffusion Model (RCDM), conditioning on the 2048 dimensional representation given by a Resnet50 (He et al., 2016) trained with Dino (Caron et al., 2021) on ImageNet (Russakovsky et al., 2015). Then we compute the representations of a set of images from ImageNet validation data to condition the sampling from the trained RCDM. Figure 4a shows it is able to sample images that are very close visually from the one that is used to get the conditioning. We also evaluated the generation abilities of our model on out of distribution data. We took images from Wikimedia Commons, PACS (Li et al., 2017) and Cityscape (Cordts et al., 2016). Figure 4b shows that our model is able to sample new views of these images while staying close to the original one even in the out of distribution scenario. This implies that there is much information kept inside the SSL representation so that the conditional generative model is able to reconstruct many characteristics of the original image. We also perform interpolations between two SSL representations in Figure 13b. This shows that our model is able to produce interpretable images even for SSL representations that correspond to an unlikely mix of factors. Both the interpolation and OOD generation clearly show that the RCDM model is not merely outputting training set images that it could have memorized. This is also confirmed by Figure 14 in the appendix that shows nearest neighbors of generated points.



(a) (b) (c)

Figure 4: a) In distribution conditional image generation. We take images from the validation set of ImageNet (first column) and we compute the corresponding representation given by a trained SSL model (Dino). We used these representations as conditioning for the diffusion model and show the resulting samples in the subsequent columns. We observe that our conditional diffusion model produces samples that are very close to the original image. b) Out of distribution (OOD) conditional image generation. How well does RCDM generalize when conditioned on representations given by images from a different distribution? First column sources: WikiMedia Commons, Cityscape (Cordts et al., 2016), PACS (Li et al., 2017). Samples generated by the diffusion models are in the following columns. We observe that even with OOD conditionnings, the images produced by our model are still very close to the original image. c) Interpolation between two set of images from ImageNet validation data. We apply a linear interpolation between the SSL representation of the images on the first column and the representation of the images on the last column. We use the interpolated vector as conditioning for our model that produce the samples that are showed in column 2 to column 7.

The conditional diffusion model might also serve as a building block to hierarchically build an unconditional generative model. Any technique suitable for modeling and sampling the distribution of (lower dimensional) representations could be used. As this is not our primary goal in the present study, we experimented only with simple kernel density estimation (see appendix for details). This allow us to quantify the quality of our generative process in an unconditional manner to fairly compare against state-of-the-art generative models such as ADM. We provide some generative model metrics in Table 5a along some samples in Figure 5b to show that with this basic approach our methods is still competitive with the current literature.

5.1 What are SSL representations really invariant to?

To better understand what information is stored in SSL representations, we trained several RCDM models using the representations given by SimCLR (Chen et al., 2020), Dino (Caron et al., 2021), Barlow Twins (Zbontar et al., 2021) and VicReg (Bardes et al., 2021). We then compare how the generated images differ depending on the kind of representation they were trained on. In Figure 6, using the same conditioning image, we sample several images with RCDMs trained on different kinds of representation. Using the projector head representation leads to a much larger variance in

(a) We report results for ImageNet to show that our approach is reliable for generating images which look realistic. Since the focus of our work is not generative modelling but to showcase and encourage the use of such model for representation analysis, we only show results for one conditional generative models. For each method, we computed FID and IS with the same evaluation setup in Pytorch.

Method	**Res.**	**↓FID**	**↑IS**
ADM (Dhariwal & Nichol, 2021)	256	26.8	34.5 ± 1.8
IC-GAN (Casanova et al., 2021))	256	20.8	51.3 ± 2.2
IC-GAN (Casanova et al., 2021) (KDE*)	256	21.6	38.6 ± 1.1
RCDM (ours)	256	19.0	51.9 ± 2.6

(b)

Figure 5: a) Table of results on ImageNet. We compute the FID (Heusel et al., 2017) and IS (Salimans et al., 2016) on 10 000 samples generated by each models with 10 000 images from the validation set of ImageNet as reference. KDE* means that we used our unconditional representation sampling scheme based on KDE (Kernel Density Estimation) for conditioning IC-GAN instead of the method based on K-means introduces by Casanova et al. (2021). b) Unconditional samples with KDE on representations (Dino) from a RCDM trained on 256x256 ImageNet images. For additional images, see Figure 21.

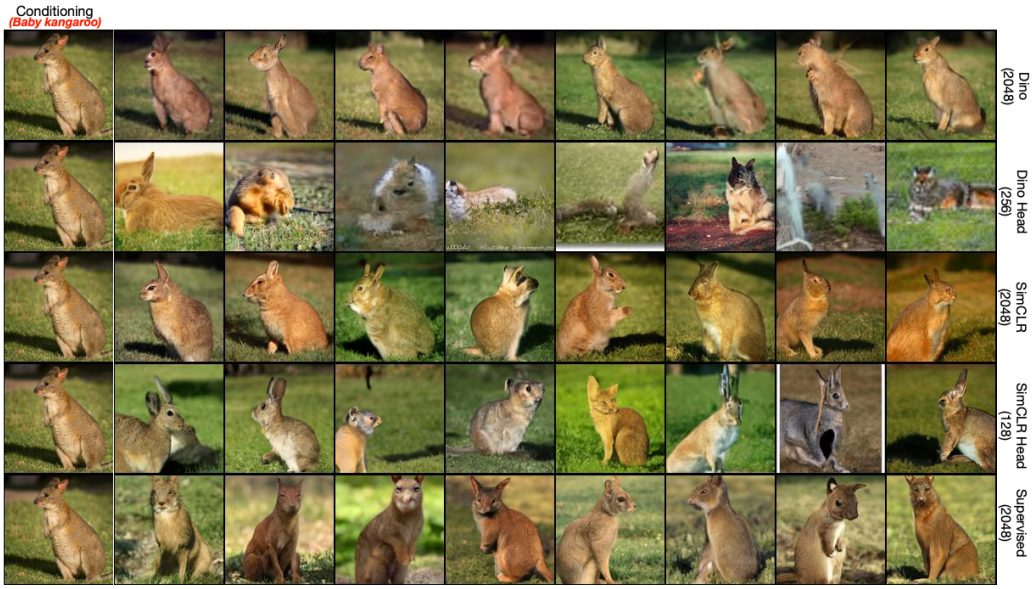


Figure 6: On the first and second row, RCDM samples conditioned on the usual backbone representation (size 2048) and projector representation (size 256) with Dino. Same on the third and forth row with SimCLR representations (2048 and 128). For comparison, we also added samples from RCDM trained on representation given by a supervised trained model. We clearly observe that the projector only keeps global information and not its context, as opposed to the backbone. This indicates that invariances in SSL models are mostly achieved in the projector representation, not the backbone. Additional comparisons provided in Figure 18,19,21.

the generated samples whereas using the traditional backbone (2048 dimesions) representation leads to samples that appear much closer to the original image, The supervised-trained 2048 dimensional representation constrains the appearance much less (larger variety). In Figure 7, we apply specific transformations (augmentations) to a test image and we check whether the samples generated by the diffusion model change accordingly. We also compare with the behavior of a supervised model. We note that despite their invariant training criteria, the 2048 dimensional SSL representations do

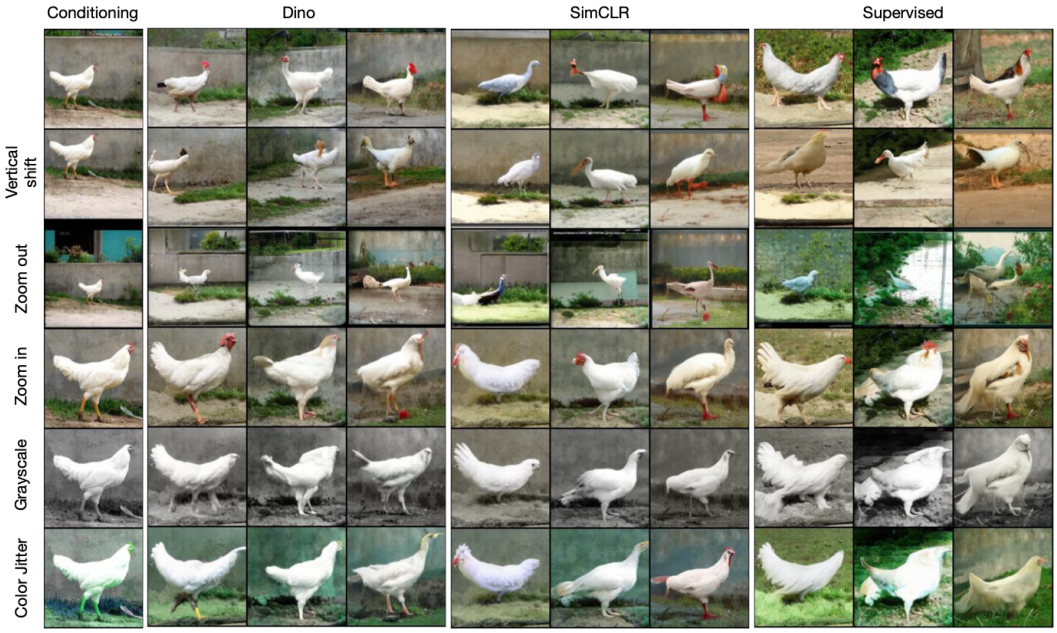


Figure 7: We use our conditional generative model to get insight on the invariance (or covariance) of representations with respect to several data augmentations. On an original image (top left) we apply specific transformations (visible in the first column). For each transformed image, we compute the SSL representation with Dino, SimCLR and a supervised network and condition the corresponding RCDM with that representation to sample 3 images. We see that despite their invariant training criteria, the 2048 dimensional SSL representations appear to retain information on object scale, grayscale vs color, and color palette of the background, much like the supervised representation. They do appear insensitive to vertical shifts. We also see that supervised representation constrain the appearance much less. Refer to Figure 20 in Appendix for a comparison with using the lower dimensional projector head embedding as the representation.

retain information on object scale, grayscale status, and color palette of the background, much like the supervised representation. They do appear invariant to vertical shifts. In the Appendix, Figure 20 applies the same transformations, but additionally compares using the 2048 representation with using the lower dimensional projector head embedding as the representation. There, we observe that the projector representation seems to encode object scale, but contrary to the 2048 representation, it appears to have gotten rid of grayscale-status and background color information.

6 Conclusion

We studied the interplay between the set of inputs that map to a given network representation and the perceptual quality of these matched inputs. We showed that gradient descent based optimization can find samples that are very closes to a given representation, but these won't be realistic-looking images. To remediate this issue, we introduce a conditional diffusion-based model (RCDM) that is suitable for conditioning on large vector representations. This model is able to generate samples that are close to a given representation in the representation space while producing realistic images. By training many RCDM models on different types of representations, we were able to qualitatively analyze SSL representations and their differences. A key benefit of conditional generation is that by looking at the commonalities between samples coming from the same conditioning, one can identify the information encoded in the representation; by looking at the differences between samples one can identify the information that is not present in the representation. Our main finding is that current SSL representations retain information about the background and the scale of a given image. We hope that the tools we introduced in this paper will allow researchers to better design and understand self-supervised models.

7 REPRODUCIBILITY STATEMENT

The data and images in this paper were only used for the sole purpose of exchanging reproducible research results with the academic community.

Our results should be easily reproducible as:

- RCDM, is based on the same code as Dhariwal & Nichol (2021) (https://github.com/openai/guided-diffusion) and uses the same hyper-parameters (See Appendix I of Dhariwal & Nichol (2021) for details about the hyper-parameters).
- To obtain our conditional RCDM, one just needs to replace the GroupNormalization layers in that architecture by a conditional batch normalization layer of Brock et al. (using the code from https://github.com/ajbrock/BigGAN-PyTorch).
- The self-supervised pretrained models we used to extract the conditioning representations were obtained from the model-zoo of VISSL (Goyal et al., 2021) (code from https://github.com/facebookresearch/vissl).
- The unconditonal sampling process is straightforward, as explained in Appendix C.
- We are working on cleaning and preparing to release any remaining code glue to easily reproduce the results in this paper.

REFERENCES

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

Srikar Appalaraju, Yi Zhu, Yusheng Xie, and István Fehérvári. Towards good practices in self-supervised representation learning. *arXiv preprint arXiv:2012.00868*, 2020.

Randall Balestriero and Richard Baraniuk. A spline theory of deep learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 374–383. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/balestriero18b.html.

Randall Balestriero, Romain Cosentino, Behnaam Aazhang, and Richard Baraniuk. The geometry of deep networks: Power diagram subdivision. *Advances in Neural Information Processing Systems*, 32:15832–15841, 2019.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. URL http://arxiv.org/abs/1206.5538. cite arxiv:1206.5538.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*. URL https://openreview.net/forum?id=B1xsqj09Fm.

Santiago A Cadena, Marissa A Weis, Leon A Gatys, Matthias Bethge, and Alexander S Ecker. Diverse feature visualizations reveal invariances in early layers of deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 217–232, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

Arantxa Casanova, Marlène Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero-Soriano. Instance-conditioned gan. 2021.

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

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2020.

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

Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear approximation and (deep) relu networks. *Constructive Approximation*, pp. 1–46, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. *arXiv preprint*, (arXiv:2105.05233), 2021.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv:1610.07629*, 2017.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

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

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

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014.

Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. https://github.com/facebookresearch/vissl, 2021.

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. In *NeurIPS*, 2020.

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

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

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv:2006.11239*, 2020.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *CVPR*, 2017.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*, (arXiv:1411.1784), 2014.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *arXiv preprint arXiv:1402.1869*, 2014.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv:2102.09672*, 2021.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML 2014)*, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pp. 234–241, 10 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 791–798, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273596. URL https://doi.org/10.1145/1273496.1273596.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv:1503.03585*, 2015.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv:2006.09011*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv:arXiv:1907.05600*, 2020b.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv:2011.13456*, 2020.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/b1301141feffabac455e1f90a7de2054-Paper.pdf.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1747–1756. JMLR.org, 2016.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010. URL http://jmlr.org/papers/v11/vincent10a.html.

Zhirong Wu, Yuanjun Xiong, Stella Yu, , and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. *arXiv preprint*, (arXiv:1901.04596), 2019.

Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*, 2020.

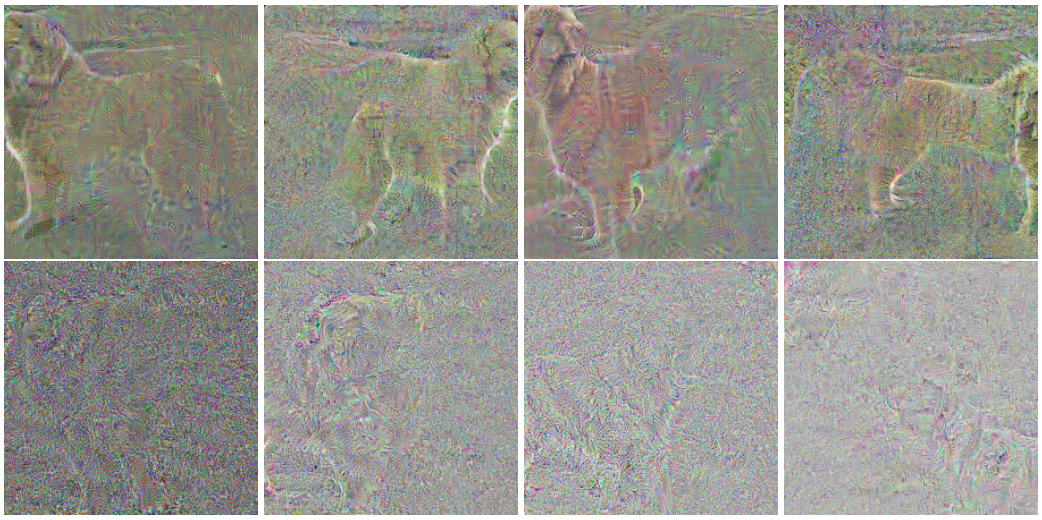


Figure 8: Reprise of Fig. 2 but now when considering the mapping to be the resnet50 backbone and the projection head of DINO. Top row is when using DINO+noise and the bottom row is when using standard DINO. As was the case when using the resnet50 backbone only, we do not obtain realistic inputs from $\mathcal{S}(\boldsymbol{x})$ when following gradient directions.

distance	ℓ_2	ℓ_1	cosine	relative ℓ_2 (%)	relative ℓ_1 (%)	relative cosine (%)
Adam plateau	0.8	30.0	0.0	0.1	5.5	0.1
Adam cosine	0.4	11.0	0.0	0.1	2.1	0.1
GD plateau	2.8	48.0	0.1	0.4	8.8	5.7
GD cosine	2.2	17.0	0.1	0.3	3.1	7.8
L-BFGS plateau	0.1	23.0	0.0	0.0	4.3	0.0
L-BFGS cosine	0.1	26.0	0.0	0.0	4.9	0.0

Table 1: We depict here the final value of the input optimization step ($\boldsymbol{x}^{(T)}$). We experiment with different distances (each column) and we provide the actual value of the distance along with a relative distance which is obtained by $100 - 100 \times |d(f(\boldsymbol{x}), f(\boldsymbol{x}^{(0)})) - d(f(\boldsymbol{x}), f(\boldsymbol{x}^{(T)}))| / d(f(\boldsymbol{x}), f(\boldsymbol{x}^{(0)}))$. That is, the relative distance gives a proportion of how close to the target is the obtained representation as a ratio with respect to the distance using the initial (random) image, value between 0 and 100. In this table, we are looking at the DINO model. We provide the noise models in the below tables.

distance	ℓ_2	ℓ_1	cosine	relative ℓ_2 (%)	relative ℓ_1 (%)	relative cosine (%)
Adam plateau	3.1	54.0	0.0	0.9	13.5	0.9
Adam cosine	3.6	41.0	0.0	1.0	10.2	0.8
GD plateau	12.2	107.0	1.3	3.5	26.6	91.8
GD cosine	14.0	90.0	1.3	4.1	22.5	95.1
L-BFGS plateau	2.2	50.3	0.0	0.6	12.4	0.7
L-BFGS cosine	3.3	560.0	0.0	0.9	61.6	0.5

Table 2: Reprise of Tab. 1 but with DINO noise

A More on $\mathcal{S}(f(x))$ sampling

In this section, we propose in Fig. 8 additional gradient based matching that employ the projector head of DINO. We also provide in Tab. 1,2, 3 the distances that those gradient based matched input can read in term of representation from a target one. In the next section we also provide additional theoretical arguments supporting the challenge of following gradient directions to obtain realistic samples from $\mathcal{S}$.

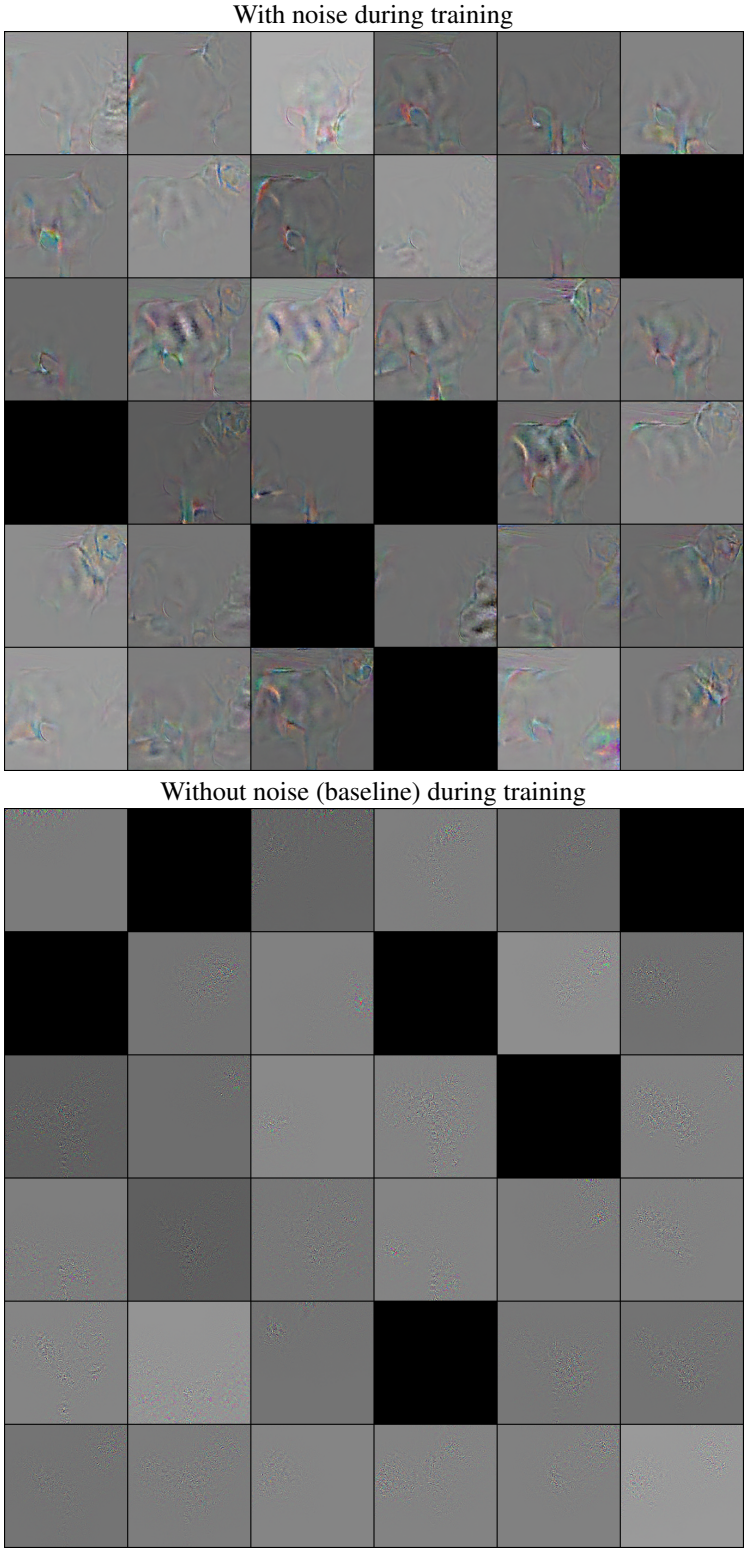


Figure 9: Depiction of 36 rows of the Jacobian matrix of a trained DINO model that either employed Gaussian noise on the images during training (top) or did not (bottom). Clearly the use of noise during training produce Jacobian matrices with more natural patterns.

distance	ℓ_2	ℓ_1	cosine	relative ℓ_2 (%)	relative ℓ_1 (%)	relative cosine (%)
Adam plateau	1.8	52.4	0.0	0.3	15.6	0.3
Adam cosine	2.8	39.3	0.0	0.5	11.7	0.5
GD plateau	27.2	131.0	0.7	4.5	39.3	85.4
GD cosine	50.1	170.0	0.7	8.4	50.9	89.9
L-BFGS plateau	3.6	53.7	0.0	0.6	16.0	0.5
L-BFGS cosine	3.4	52.3	0.0	0.6	15.6	0.5

Table 3: Reprise of Tab. 1 but with DINO noise ++

A.1 Deep Networks and CPAs

In this section we propose to further characterize what $\mathcal{S}(\boldsymbol{h})$ looks like by using a specific form for the DN input output mapping.

Without loss of generality we consider a mapping f that is continuous piecewise affine (CPA), as is the case for most DNs(Balestriero & Baraniuk, 2018). The DN input-output is then given by

$$f(\boldsymbol{x}) = \sum_{\omega \in \Omega} \left(\boldsymbol{A}_\omega \boldsymbol{x} + \boldsymbol{b}_\omega\right) 1_{\{\boldsymbol{x} \in \omega\}}, \tag{3}$$

with Ω a partition of the DN input space. In the case of f being smooth, a simple approximation argument will allow to fall back to the above setting (Daubechies et al., 2021). Using this formulation, we can now characterize more precisely the form of $\mathcal{S}(f(\boldsymbol{x}))$ as follows

$$\mathcal{S}(f(\boldsymbol{x})) = \cup_{\boldsymbol{x}' \in \mathcal{X}(\boldsymbol{x})} \Big(\omega(\boldsymbol{x}') \cap \underbrace{\left\{ \boldsymbol{x}' + \boldsymbol{u}, \boldsymbol{u} \in \ker\left(\boldsymbol{A}_{\omega(\boldsymbol{x}')}\right) \right\}}_{\text{linear subspace } \ker(\boldsymbol{A}_{\omega(\boldsymbol{x}')}) \text{ shifted by } \boldsymbol{x}'} \Big), \tag{4}$$

where $\omega(\boldsymbol{x}')$ is the region from Ω in which $\boldsymbol{x}'$ lives in, and $\mathcal{X}(\boldsymbol{x})$ is a finite set of inputs that depend on $\boldsymbol{x}$ such that each point lives in a separate region from the others as in $\forall \boldsymbol{u}, \boldsymbol{v} \in \mathcal{S}(f(\boldsymbol{x}))^2, \omega(\boldsymbol{u}) = \omega(\boldsymbol{v}) \iff \boldsymbol{u} = \boldsymbol{v}$.

The optimization problem needs to be constrained. The first challenge of our method comes from the fact that the set $\mathcal{S}(f(\boldsymbol{x}))$ consists of a union of affine subspace that are highly localized in the input space (recall (4)). In fact, the regions $\omega \in \Omega$ are often extremely localized in the input space, especially as the architecture involves many layers (Montúfar et al., 2014; Balestriero et al., 2019). In addition to that optimization difficulty, we have that the equivalence class does not constrain the inputs to lie within the data manifold. In fact, each affine subspace that form $\mathcal{S}(f(\boldsymbol{x}))$ is very high dimension as we emphasize below.

Proposition 1. *Given a model $f : \mathbb{X} \mapsto \mathbb{H}$, the set $\mathcal{S}(f(\boldsymbol{x}))$ is a union of $\mathrm{Card}(\mathcal{X}(\boldsymbol{x}))$ affine subspaces, each with dimension at least $D - K$.*

In other words, regardless of the chosen distance d, as soon as the dimensions of the affine subspaces forming $\mathcal{S}(f(\boldsymbol{x}))$ are greater than the dimension of the data manifold $\mathcal{M}$ (a sufficient condition being $D - S > \dim(\mathcal{M})$) we obtain that $\mathcal{S}(f(\boldsymbol{x}))$ contains samples that do not belong to $\mathcal{M}$. That is, performing gradient descent from randomly initialized samples $\boldsymbol{x}^{(0)}$ will almost surely produce samples $\boldsymbol{x}^{(T)} \notin \mathcal{M}$. This is particularly true for a case such as Imagenet in which $D = 150528$ and $S = 2048$.

B Conditional and super-resolution sampling with RCDM

As presented in the main text, RCDM generates samples that preserved well the semantics of the images used for the conditioning. We show in Figure 21 additional samples of RCDM when conditioning on the SSL representation of ImageNet validation set images (which were never used for training). We observe that the information hidden in the SSL representation is so rich that RCDM is almost able to reconstruct entirely the image used for conditioning. To further evaluate the abilities of this model, we present in Figure 11 a similar experiment except that we use out of distribution images as conditioning. We used cell images from microscope and a photo of a status (Both from Wikimedia Commons), sketch and cartoons from PACS (Li et al., 2017), image segmentation from Cityscape

(Cordts et al., 2016) and an image of the Earth by NASA. Even in the OOD scenario, RCDM is able to generate images that are very close to the one used as conditioning because of the richness of ssl representations.

We also use the super-resolution model presented by Dhariwal & Nichol (2021) to generate images of higher resolutions. In Figure 12, we use the small images on the top of the bigger images as conditioning for a RCDM trained on images of size 128x128. Then, we feed the 128x128 samples into the super-resolution model of Dhariwal & Nichol (2021) to get images of size 512x512. Since the model of Dhariwal & Nichol (2021) is conditional and need labels, we used a random label when upsampling from RCDM. Despite using the "wrong" label, the high resolution samples are still very close to the conditioning. This show that RCDM can be used jointly with a super-resolution model to sample high fidelity images in the close neighborhood of the conditioning.

To verify how well our model can produces realistic samples from different combinations of representations, we take two images from which we compute their representations and perform a linear interpolation between those. This give us new vectors of representation that can be used as conditioning for RCDM. We can see on Figure 13 that RCDM is able to generate samples that contains the semantic characteristics of both images.

Finally, in Figure 14, we search the nearest neighbors of a series of samples in the ImageNet training set. As demonstrated by Figure 14, RCDM samples images that are new and far enough from images belonging to the training set of ImageNet.

C A HIERARCHICAL DIFFUSION MODEL FOR UNCONDITIONAL GENERATION

We provided a novel and conditional generative model based on a given latent representation e.g. from a DN embedding, and a diffusion model. This provided tremendous insights into interpreting what is encoded in those representations. We can go one step further and augment this conditional model with an unconditional one that can generate those representations. This will provide us with the ability to generate new samples without the need to condition on a given input. As a by-product, it will allow us to quantify the quality of our generative process in an unconditional manner to fairly compare against state-of-the-art generative models.

We shall recall that our goal is to employ the conditional generative model to provide understanding into learned (SSL) representations. The unconditional model is only developed to compare our generative model and ensure that its quality is reliable for any further down analysis. As such, we propose to learn the representation distribution in a very simple manner via the usual Kernel Density Estimation (KDE). That is, the distribution is modeled as

$$p(\boldsymbol{h}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\boldsymbol{h}; f(\boldsymbol{x}_n); I\sigma)$$

with σ set to 0.01. By using the above distribution, we are able to sample representations $\boldsymbol{h}$ to then sample images $\boldsymbol{x}$ conditionally to that $\boldsymbol{h}$ using our diffusion model. We provide some samples in Figure 21 to show that even with our very simple conditioning, our methods is still able to generate realistic images.

D ON THE CLOSENESS OF THE SAMPLES IN THE REPRESENTATION SPACE

Even if we show that RCDM is able to generate images that seems visually close to the image used for the conditioning, it's still unclear how close those images are in the representation space. We can compute euclidean distances but to know how close the generated samples are to the conditioning, we need to have references that can be used to compare this distance with. As references, we compute the euclidean distance between a conditioning image and random images in the validation set of ImageNet, random images belonging to the same class as the conditioning, the closest images in the training set, the conditioning image on which we applied single data augmentations and the conditoning image on which we applied the data augmentation performed by Swav and Dino (Caron et al., 2020; 2021). The results can be seen in Figure 16 for a RCDM trained with Dino representations and in Figure 17 for a RCDM trained with SimCLR representations. On both Figure, we observe that

Figure 10: Generated samples from RCDM on 256x256 images trained with representations produced by Dino. We put on the first column the images that are used to compute the representation conditioning. On the following column, we can see the samples generated by RCDM. It is worth to denote our generated samples are qualitatively close to the original image.

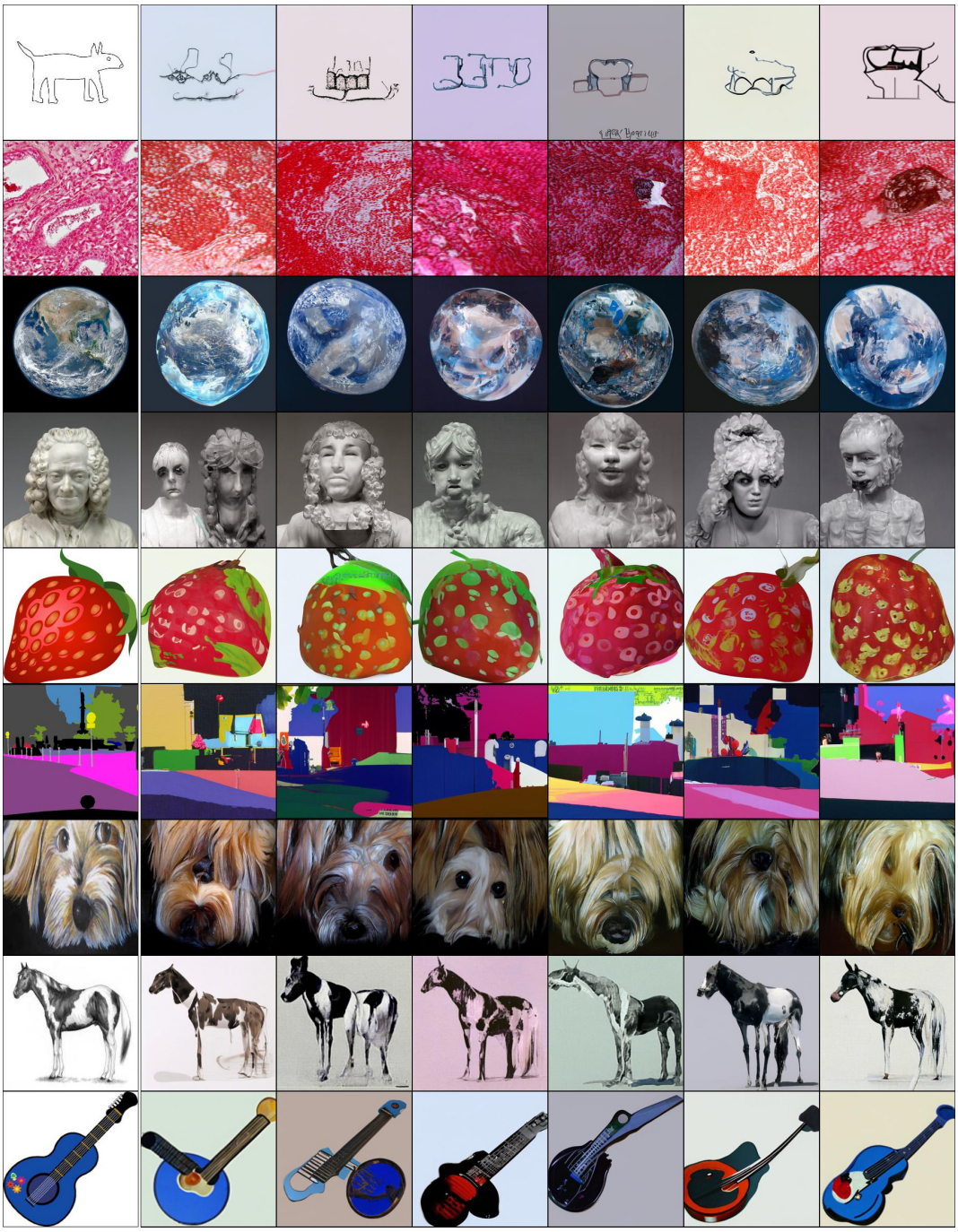


Figure 11: Generated samples from RCDM model on 256x256 images trained with representations produced by Dino on Out of Distribution data. We put on the first column the images that are used to compute the representation. On the following column, we can see the samples generated by RCDM. It is worth to denote our generated sample are close to the original image. The images used for the conditioning are from Wikimedia Commons, Cityscapes (Cordts et al., 2016), PACS (Li et al., 2017) and the image of earth from NASA.

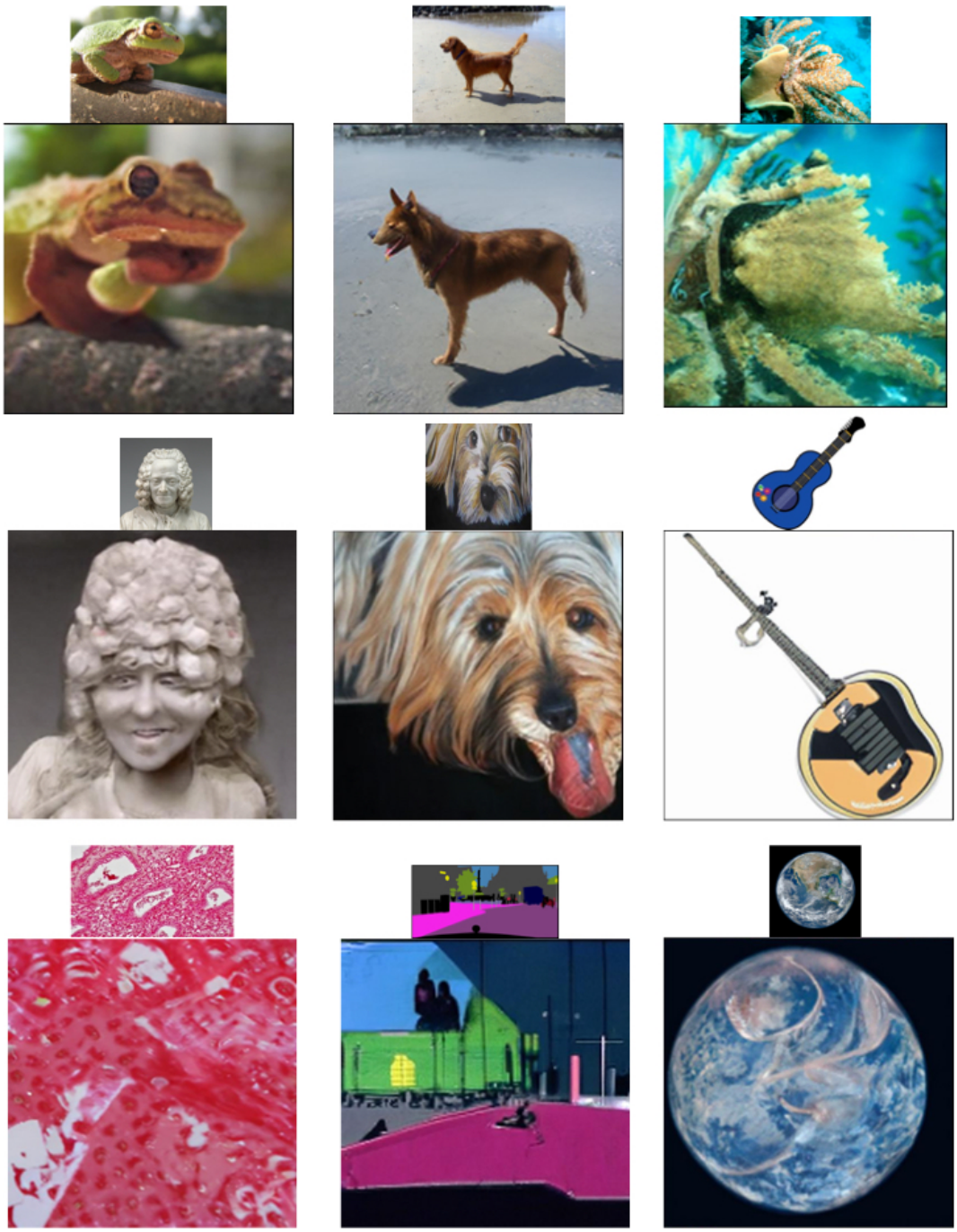
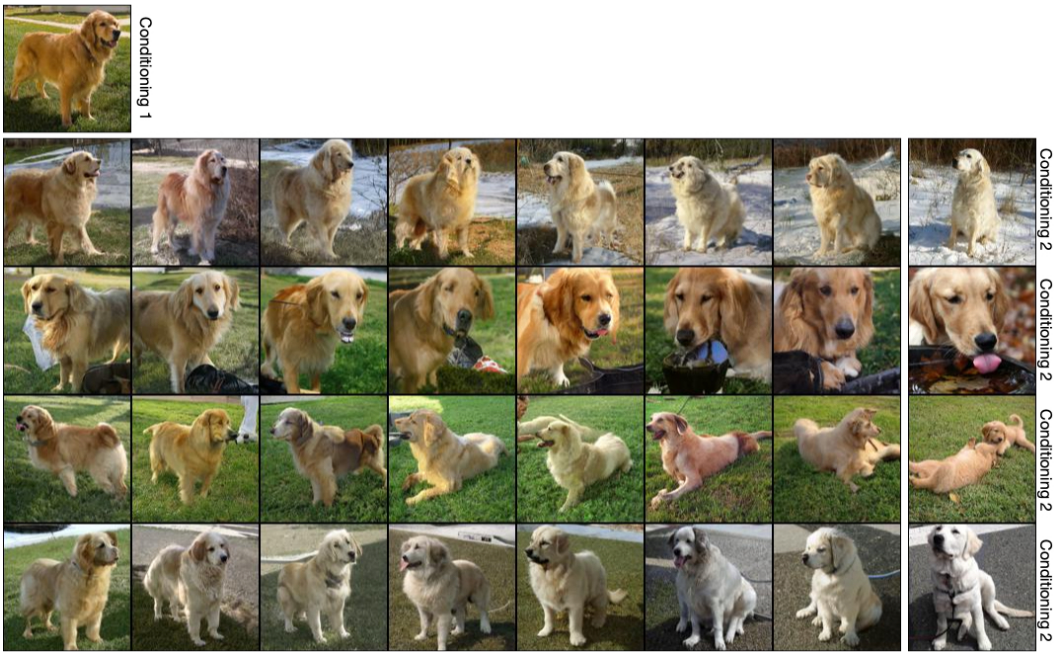
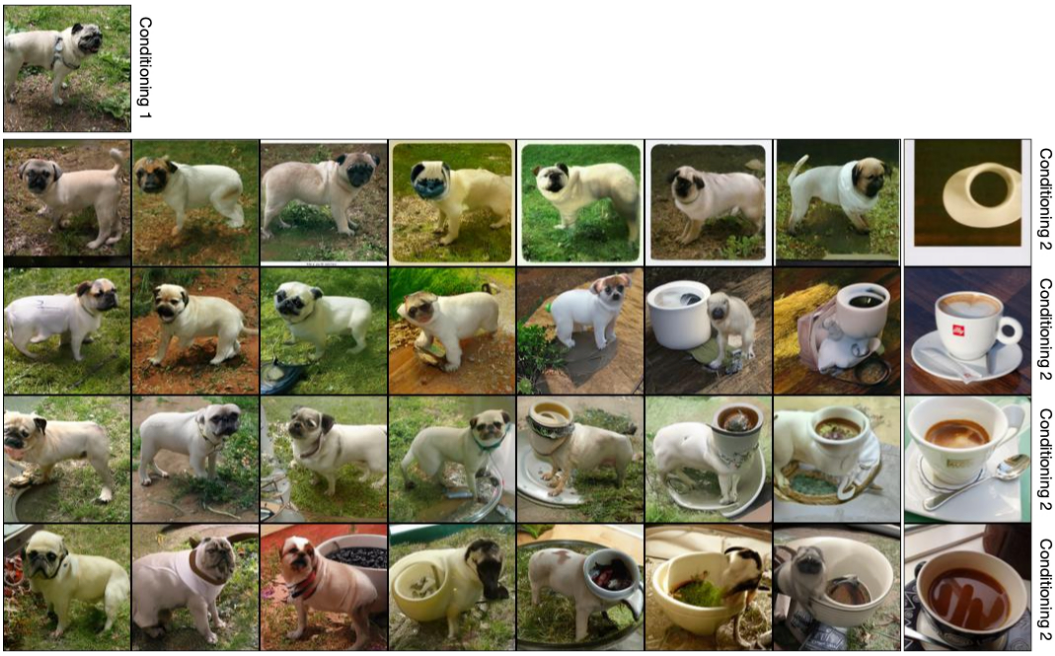


Figure 12: High resolution samples from our conditional diffusion generative model using the super resolution model of Dhariwal & Nichol (2021). We use the small images on the top of each bigger image as conditioning for a diffusion model trained with Dino representation on 128x128 images. Then, we feed the samples generated to the super resolution model of Dhariwal & Nichol (2021) which produces images of size 512x512. Since the super resolution model is conditional, we sample a random label. We note that the high resolution samples are still very close to the conditioning.



(a) Linear interpolation between the image of the golden retriever in conditioning 1 with various other images belonging to the same class as conditioning 2.



(b) Linear interpolation between the image of the pug in conditioning 1 with various other images belonging to the espresso class as conditioning 2.

Figure 13: Each vectors that result from the linear interpolation is feed to a RCDM trained with Barlow Twins representation.

the generated images with RCDM are closer to the conditioning than the closest neighbors in the entire training set of ImageNet.

E ANALYSIS OF REPRESENTATIONS LEARNED WITH SELF-SUPERVISED MODEL

Having generated samples that are close in the representation space to a conditioning image can give us an insight on what's hidden in the representations learned with self-supervised models. As demonstrated in the previous section, the samples that are generated with RCDM are really close visually to the image used as conditioning. This give an important proof of how much is kept inside a SSL representation. However, it's also important to consider how much this amount of "hidden" information varied depending on the SSL representation that is used. Therefore, we train several RCDM on SSL representations given by VicReg (Bardes et al., 2021), Dino (Caron et al., 2021), Barlow Twins (Zbontar et al., 2021) and SimCLR (Chen et al., 2020). In many applications that used self-supervised models, the representation that is used is the one that corresponds to the backbone of the ResNet50. Usually, the representation given by the projector of the SSL-model (on which the SSL criterion is applied) is discarded because the results on many downstream tasks like classification is not as good as the backbone. However, since our work is to visualize and better understand the differences between SSL representations, we also trained RCDM on the representation given by the projector of Dino, Barlow Twins and SimCLR. In Figure 18 and 19 we condition all the RCDM with the image labelled as conditioning and sample 9 images for each model. We observe that Dino representation does not allow much variance meaning that even information about the pose of the animal is kept inside the representation. In contrast, the SimCLR representation seems to be more invariant to the pose of the kangaroo, maybe too much because on many images the kangaroo become a rabbit. VicReg seems to be more robust in the sense that the animal doesn't cross the class boundary despite changes in the background. If we look at the projector of the SSL models, the generated samples have a higher variance except for Barlow Twins. This can be explained by the fact that the dimension of the representation given by the projector has a size of 8192 which is much bigger than the one used by other methods.

To further compare and analyse the different SSL models, we visualize how much SSL representations can be invariant with respect to a transformation that is applied on the conditioning image. In Figure 20, we apply several Data Augmentation: Vertical shift, Zoom out, Zoom In, Grayscale and a Collor Jitter on a given conditioning image. Then we compute the SSL representations of the transformed image with different SSL models and use our corresponding RCDM to see how much the samples have changed with respect to the samples generated on the vanilla conditioning image. We observe that the representation (the 2048 backbone one) of all SSL methods are not invariant to scale and change of colors. Whereas the representation of the projector doesn't seem to take into account any small transformation in the original conditioning outside the scale for Dino. For SimCLR, there is still some information about the background that is kept in the representation however the samples are not as close visually with respect to the 2048 representation. Barlow Twins is interesting because there isn't much differences between the backbone representation (2048) one and the representation of the projector (Size 8192). With the exception that this last representation seems to be more invariant to color shift than the backbone one.

E.1 MANIPULATION OF SSL REPRESENTATIONS

It is also possible to manipulate SSL representations to generate new images. We try to apply addition and subtractions over SSL representations (similarly to what has been done in NLP). From two different images, we compute the difference between the two corresponding representations and add the difference vector to a third image. Figure 22 shows that it is possible to apply such transformations meaningfully in the SSL space.

(a) Closest real images (ImageNet training set) from images sampled with RCDM trained on Dino (backbone) representation (2048).

(b) Closest real images (ImageNet training set) from images sampled with RCDM trained on Dino projector representation (256).

Figure 14: We find the nearest neighbors in the representation space of samples generated by RCDM. The images in the red squared are the ones used for conditioning.

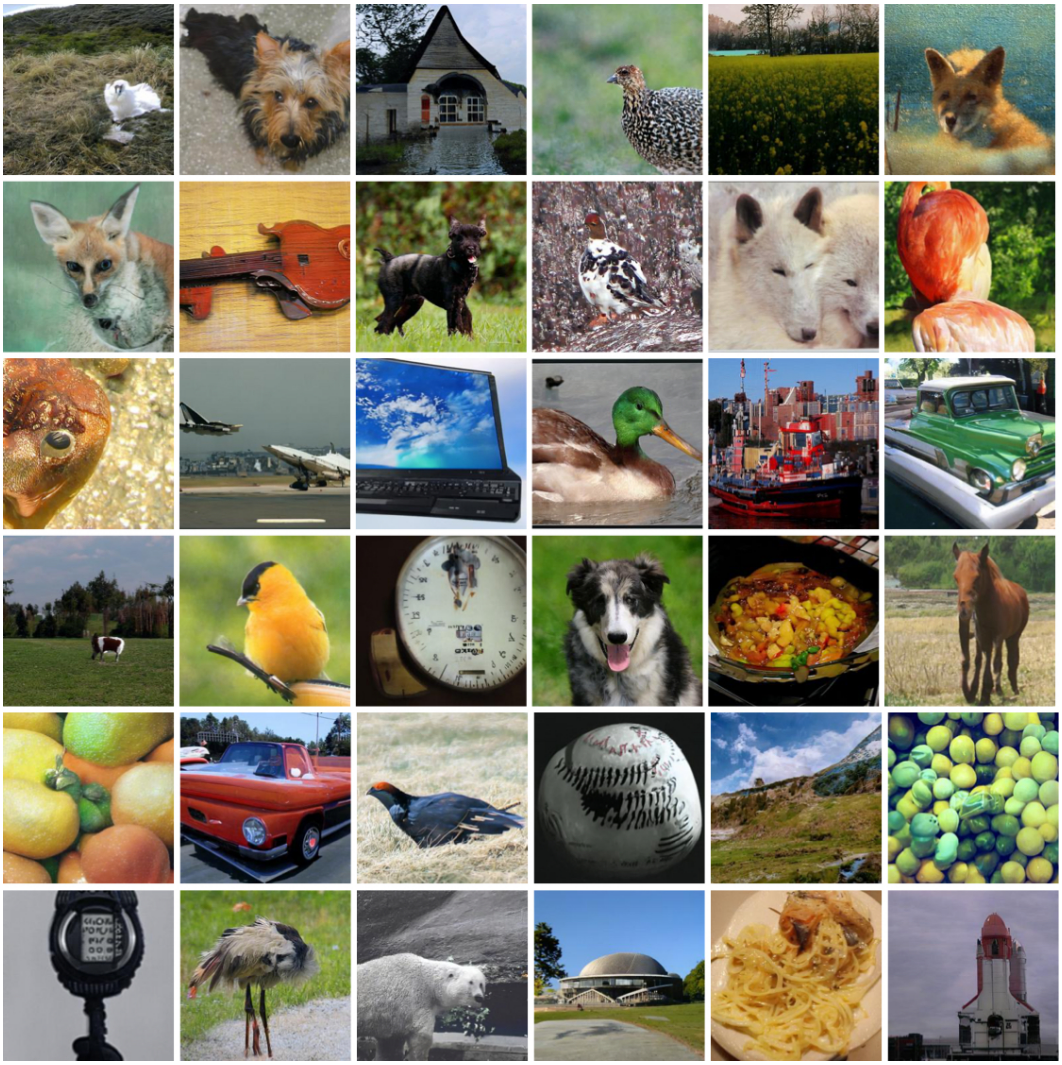


Figure 15: Unconditional generation following the protocol of section C. Our simple generative model of representations consists in applying a small Gaussian noise over representation computed from random training images of ImageNet. We use these noisy vector as conditioning for our 256x256 RCDM trained with Dino representations. We note that the generated images looks realistic despite some generative artefact like a two-headed dog and an elephant-horse.

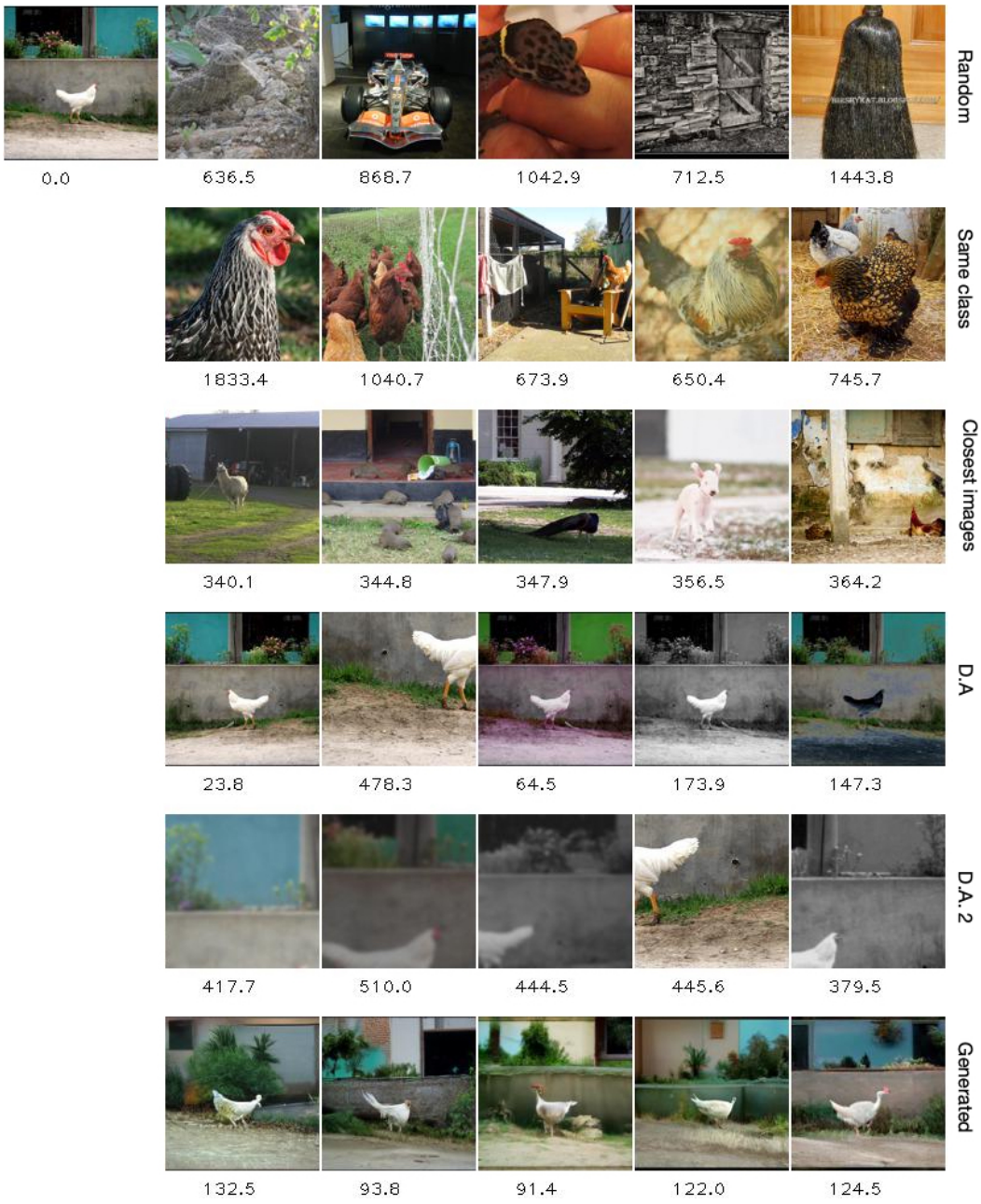


Figure 16: We show the squared euclidean distance between the conditioning image on the left most column on first row and different images to get an insight about how close the samples generated by the diffusion model stay close to the representation used as conditioning. The distances with the conditioning is printed below each images. On the first row, we show random images from the ImageNet validation data. On the second row, we take random validation examples belonging to the same class as the conditioning. On the third row, we find the closest training neighbors of the conditioning in the representation space. On forth row, D.A. means Data Augmentation which consist in horizontal flip, CenterCrop, ColorJitter, GrayScale and solarization. On fifth row (D.A. 2), we use the random data Augmentation used in the paper of (Caron et al., 2020; 2021). On the last row, we show the generated samples from our conditional diffusion model that use **Dino representation**. The samples produces by our model are much closer to the conditioning than other images.

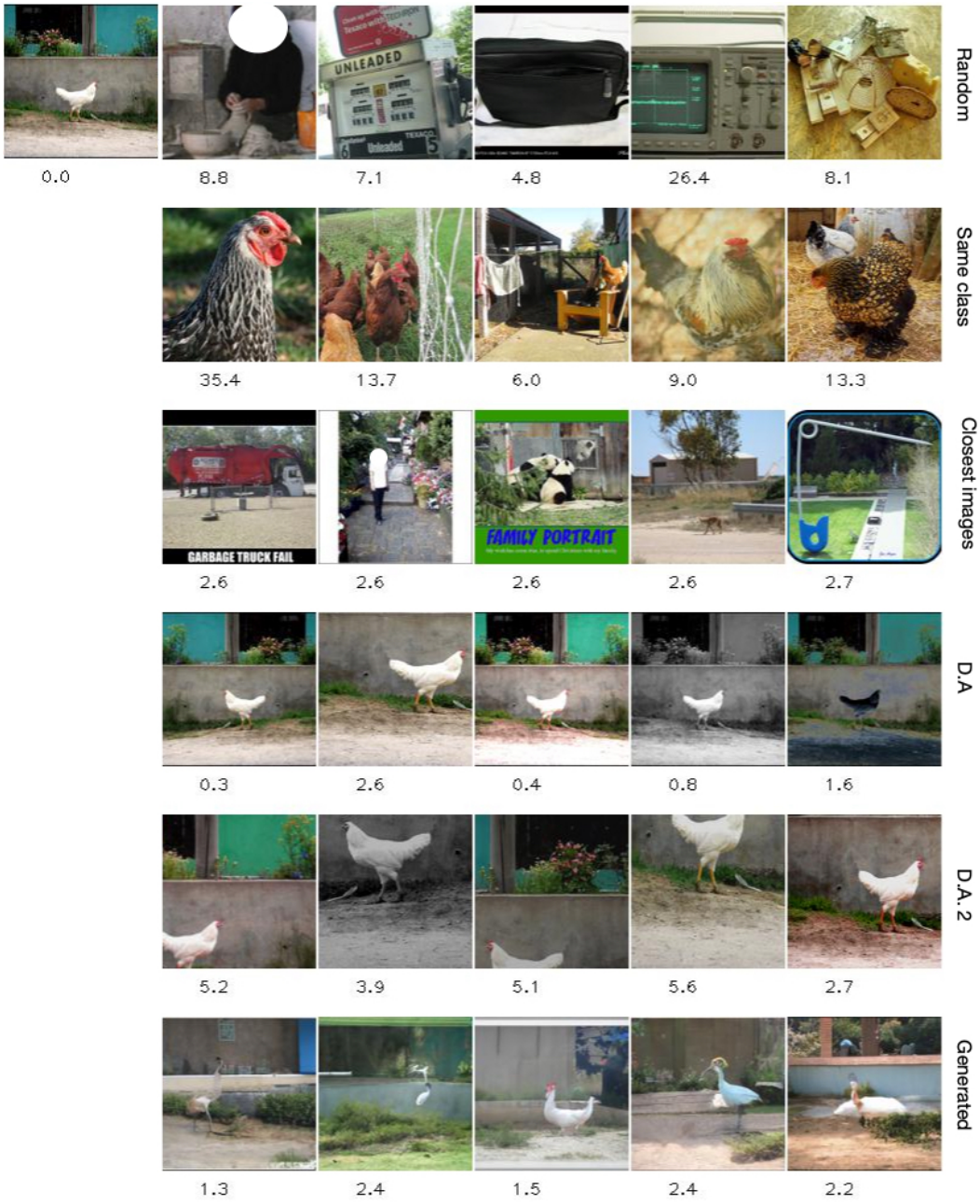


Figure 17: We show the squared euclidean distance between the conditioning image on the left most column on first row and different images to get an insight about how close the samples generated by the diffusion model stay close to the representation used as conditioning. The distances with the conditioning is printed below each images. On the first row, we show random images from the ImageNet validation data. On the second row, we take random validation example belonging to the same class as the conditioning. On third row, we find the closest training neigbords of the conditioning in the representation space. On forth row, D.A. means Data Augmentation which consist in horizontal flip, CenterCrop, ColorJitter, GrayScale and solarization. On fifth row (D.A. 2), we use the random data Augmentation used in the paper of (Caron et al., 2020; 2021). On the last row, we show the generated samples from our conditional diffusion model that use **SimCLR representation**. The samples produces by our model are much closer to the conditioning than other images.

Figure 18: Generated samples from RCDM trained with representation from various self-supervised models. The image used for conditioning is a baby kangaroo on the top row, left-most column. Then, we generate 9 samples for each model with different random seed. We observe that the representation given by dino isn't very invariant while the one given by SimCLR or VicReg show much better invariance. We also show the samples of RCDM trained on the representation given by the projector (The embedding on which is usually applied the SSL criterion). There is a much higher variability in the generated samples. Maybe too much to be used for a classification task since we can observe class crossing.

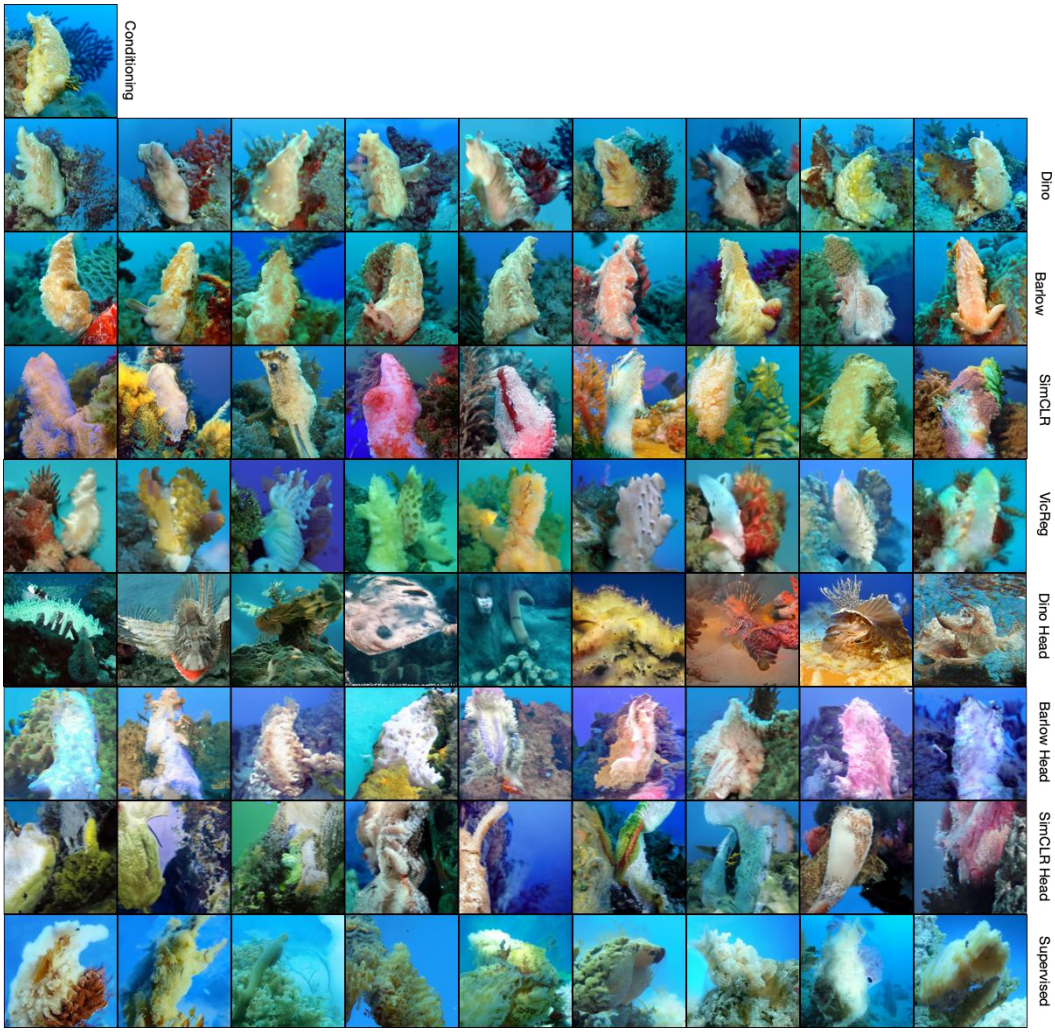


Figure 19: Generated samples from RCDM trained with representation from various self-supervised models. We generate 9 samples for each model with different random seeds. We observe that the representation given by dino isn't very invariant while the one given by SimCLR or VicReg show much better invariance. We also show the samples of RCDM trained on the representation given by the projector (The embedding on which is usually applied the SSL criterion). There is a much higher variability in the generated samples. Maybe too much to be used for a classification task since we can observe class crossing.

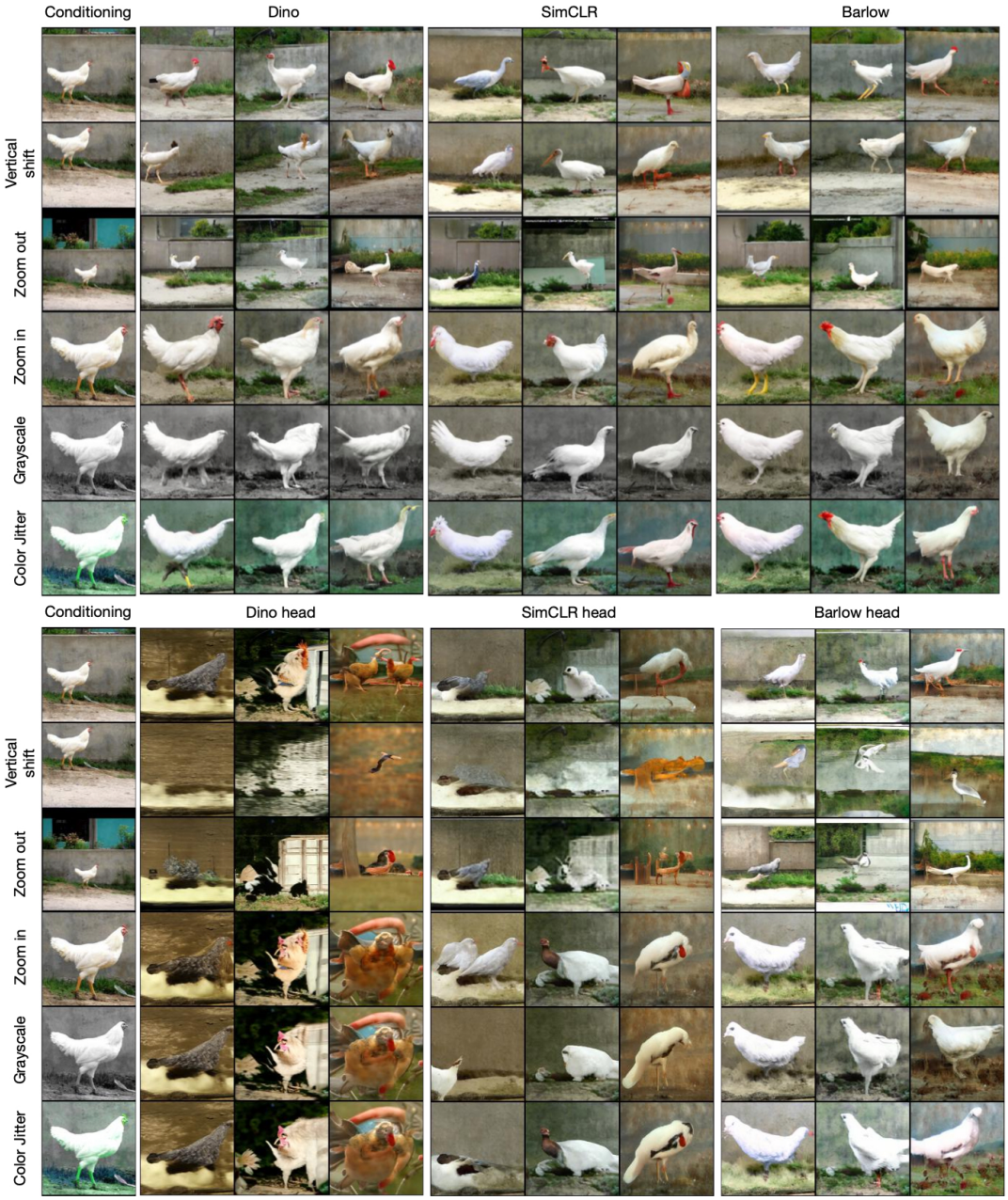


Figure 20: We compare how much the samples generated by RCDM change depending on different transformations of a given image and the model and layer used to produces the representation. Top half uses 2048 representation. Bottom half uses the lower dimensional projector head embedding. We observe that using the projector head representation leads to a much larger variance in the generated samples whereas using the traditional backbone (2048) representation leads to samples that are very close to the original image. We also observe that the projector representation seems to encode object scale, but contrary to the 2048 representation, it seems to have gotten rid of grayscale-status and background color information.

Figure 21: Generated samples from RCDM using the mean representation for a specific class (golden retriever) in ImageNet for various SSL models.

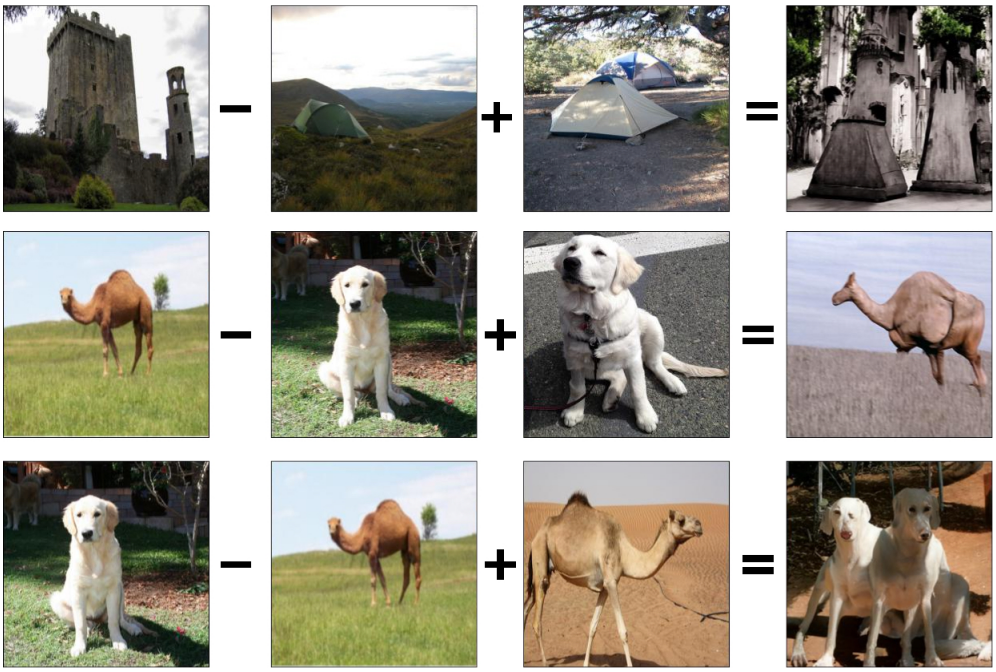
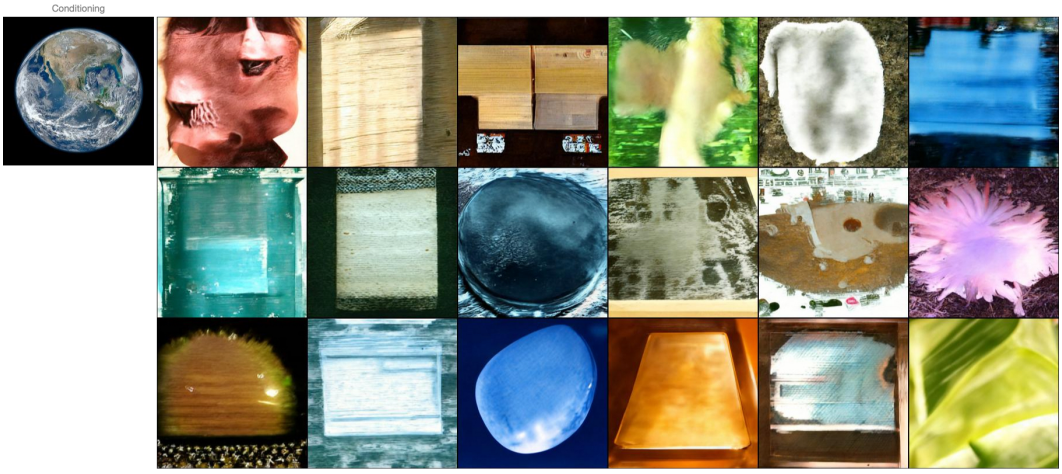
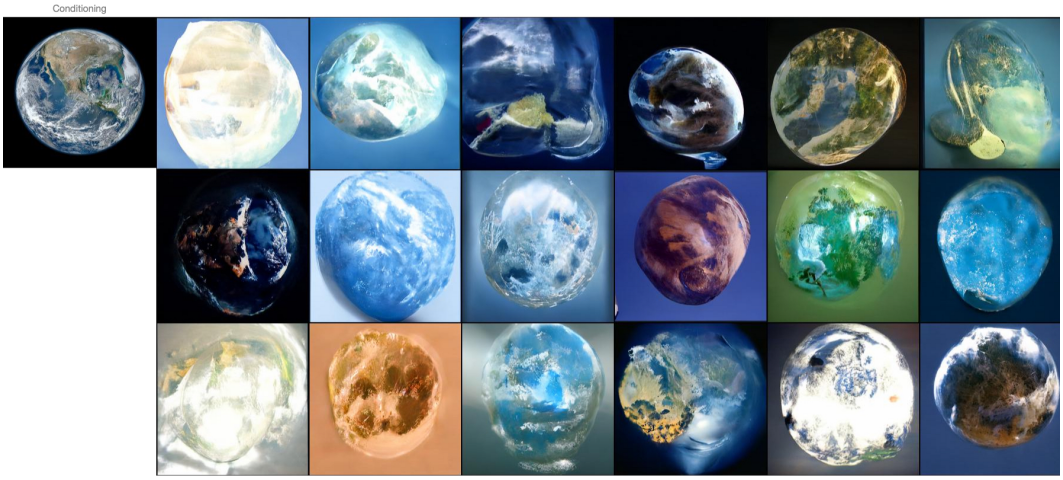


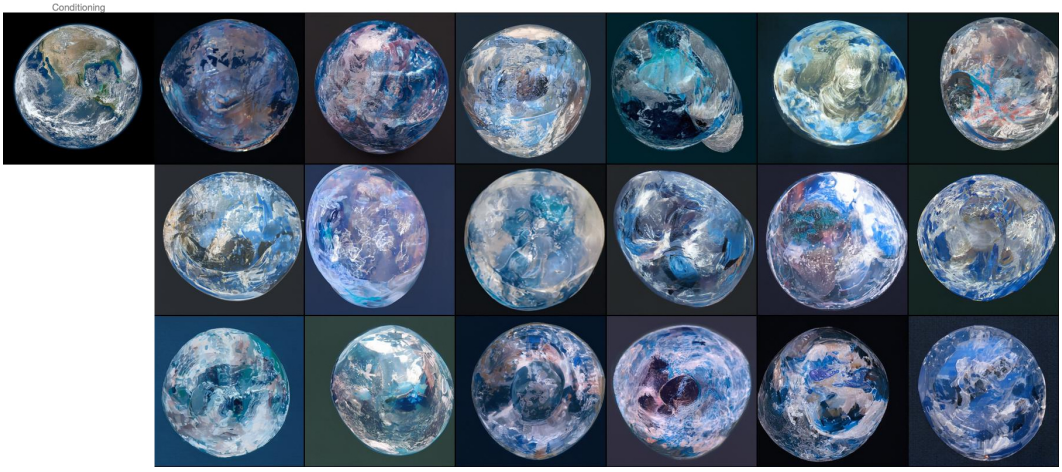
Figure 22: Algebraic manipulation of representations from real images (left-hand side of =) allows RCDM to generate new images with novel combination of factors. Here we use this technique with ImageNet images, to attempt background substitutions.



(a) Earth from an untrained representation (Random initialized Resnet 50).



(b) Earth from a supervised representation (Pretrained resnet50 on ImageNet)



(c) Earth from a SSL representation (Dino Resnet50 backbone).

Figure 23: Different samples of RCDM conditioned on a satellite image of the earth (source: NASA). We show the samples we obtained in a) when using a random initialized network to get representations, b) when using a pretrained resnet50, c) when using a self supervised model (Dino).