# Unsupervised Representation Learning via Neural Activation Coding

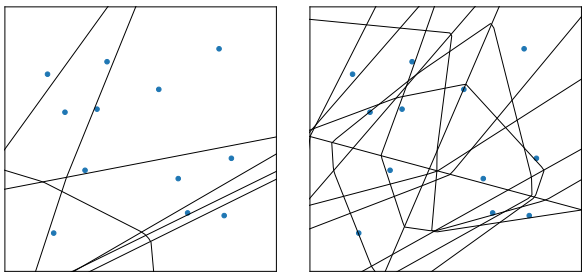
**Yookoon Park** [1] **Sangho Lee** [2] **Gunhee Kim** [2] **David M. Blei** [1]

## Abstract

We present *neural activation coding* (NAC) as a novel approach for learning deep representations from unlabeled data for downstream applications. We argue that the deep encoder should maximize its nonlinear expressivity on the data for downstream predictors to take full advantage of its representation power. To this end, NAC maximizes the mutual information between activation patterns of the encoder and the data over a noisy communication channel. We show that learning for a noise-robust activation code increases the number of distinct linear regions of ReLU encoders, hence the maximum nonlinear expressivity. More interestingly, NAC learns *both* continuous and discrete representations of data, which we respectively evaluate on two downstream tasks: (i) linear classification on CIFAR-10 and ImageNet-1K and (ii) nearest neighbor retrieval on CIFAR-10 and FLICKR-25K. Empirical results show that NAC attains better or comparable performance on both tasks over recent baselines including SimCLR and DistillHash. In addition, NAC pretraining provides significant benefits to the training of deep generative models. Our code is available at `https://github.com/yookoon/nac`.

## 1. Introduction

High dimensional data pose fundamental challenges for machine learning such as the *curse of dimensionality*, and thus often require tailored domain-specific architectures with a large amount of supervision (Krizhevsky et al., 2012; Vaswani et al., 2017). A good representation alleviates such challenges by providing a low-dimensional view of the data that captures high-level semantics and by rendering such information more easily accessible to downstream predictors (Bengio et al., 2013). Especially, unsupervised representation learning possesses a great potential since it provides a means of exploiting abundant unlabeled data for enhancing the performance on downstream applications (Devlin et al., 2018; Chen et al., 2020a), even with limited amounts of labels (Chen et al., 2020b).



(a) At initialization (b) After NAC training

*Figure 1.* Distinct linear regions of a simple ReLU network with 2 layers of width 4 on 2D toy data. The lines represent the activation boundaries that divide the input space into distinct linear regions. NAC maximizes the number of linear regions of the network on the data, hence the maximum nonlinear expressivity.

We focus on the problem of learning deep representations from unlabeled data for downstream predictors, a popular scenario in unsupervised representation learning literature (Wu et al., 2018; He et al., 2020; Chen et al., 2020a). In this setting, the deep encoder network is pretrained on an unlabeled dataset. The learned representation is then fed into subsequent predictors for downstream tasks such as classification. Most often, simple linear models are chosen as the predictors (Wu et al., 2018; He et al., 2020; Chen et al., 2020a) and the quality of the representation is evaluated by how well these models perform on the downstream applications. This evaluation protocol encodes the belief that a good representation should disentangle complex high-level semantics of the data and deliver them in a linearly accessible way. The key question here is: how can we learn the deep encoder to benefit the downstream predictors?

Self-supervised learning introduces *pretext* tasks with artificially generated pseudo-labels from unlabeled data (Doersch et al., 2015; Noroozi & Favaro, 2016; Gidaris et al., 2018) to train the encoder, expecting that the encoder would learn useful structures of the data to better solve such tasks. Recently,

[1]Computer Science Department, Columbia University, New York, USA [2]Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea. Correspondence to: David M. Blei <david.blei@columbia.edu>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

*contrastive learning* of representation (Oord et al., 2018; Wu et al., 2018; Chen et al., 2020a) based on the information maximization (InfoMax) principle has quickly gained popularity, leading significant improvements in learning unsupervised representations of natural images. Specifically, it formulates an instance-wise classification problem; as the encoder learns to identify whether a pair of inputs is from the same sample or not, the mutual information between the representation and the data is maximized.

In this work, we present a novel perspective for unsupervised representation learning: the encoder should attain maximum nonlinear expressivity on the data in order for downstream predictors to take full advantage of the encoder's nonlinear power. For a rectified activation (ReLU) network which is piece-wise linear, the nonlinear expressivity of the network is defined in terms of the number of distinct linear regions it defines on the input domain (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017), where each linear region is associated with an *activation pattern* of the encoder's hidden units. Based on this observation, *neural activation coding* (NAC) maximizes the mutual information between the activation code and the data over a noisy communication channel. We show that learning of a noise-robust activation code for communication increases the number of distinct linear regions of the network (Figure 1) and therefore maximizes its nonlinear expressivity. Moreover, NAC learns *both* continuous and discrete representations of data which we respectively evaluate on linear classification and nearest neighbor retrieval on natural image datasets. Finally, we show that NAC pretraining improves the training of deep generative models by enhancing the encoder expressivity.

Our main contributions are summarized as follows:

- We propose *neural activation coding* (NAC) as a novel approach for unsupervised representation learning. In contrast to contrastive learning approaches that are based on InfoMax principle, NAC maximizes the nonlinear expressivity of the encoder by formulating a communication problem over a noisy channel using the *activation code* of the encoder.
- NAC is able to learn *both* continuous and discrete representations of data, which we respectively evaluate on (i) linear classification on CIFAR-10 and ImageNet-1K and (ii) nearest neighbor search on CIFAR-10 and FLICKR-25K. We show NAC attains comparable or better performance to recent competitive methods, including SimCLR (Chen et al., 2020a) and DistillHash (Yang et al., 2019).
- By maximizing the nonlinear expressivity of encoder, we demonstrate that NAC pretraining significantly benefits the training of variational autoencoders.
- NAC does not require $\ell_2$-normalization for learning good representations, questioning the prevalent belief (Wu et al., 2018; Wang & Isola, 2020) that $\ell_2$-normalization plays a key role in unsupervised representation learning.

## 2. Related Works

**Nonlinear complexity of deep neural networks**. Pascanu et al. (2013); Montufar et al. (2014); Raghu et al. (2017); Serra et al. (2018); Arora et al. (2018); Hanin & Rolnick (2019) have studied nonlinear expressivity of deep neural networks (DNNs). In particular, a network with rectified activation ($\text{ReLU}(x) = \max(0, x)$) is a piece-wise linear function and divides the input space distinct locally linear regions. Accordingly, the nonlinear expressivity of the network is represented by the number of distinct linear regions it defines on the input domain. In contrast to the previous works that have either sought theoretical bounds on the the number of linear regions in DNNs or empirically analyzed the nonlinear expressiveness of DNNs when being trained on a supervised task, we propose a way to explicitly maximize the nonlinear expressivity of the encoder for representation learning.

**Self-supervised representation learning**. A prevalent approach for representation learning is to formulate *pretext* tasks with pseudo labels generated from the unlabeled data (Doersch et al., 2015; Noroozi & Favaro, 2016; Gidaris et al., 2018). Among others, contrastive learning (Wu et al., 2018; Oord et al., 2018; Tian et al., 2019; He et al., 2020; Chen et al., 2020a;c) methods have recently led the state-of-the-art advances in linear classification and transfer learning on natural image datasets. Specifically, the approach of Wu et al. (2018); He et al. (2020); Chen et al. (2020a) generates two different views of an example (e.g. random crops of an image); the views of the same example are treated as *positive samples* while the views generated from distinct examples are treated as *negative samples*. The encoder is trained to solve the instance discrimination problem of classifying the positive pair from the negatives. This maximizes the cosine similarity of positive pairs in the representation space, while pushing the representations of different examples away from each other.

Although contrastive representation learning has achieved significant advances on natural image data, it is not yet well understood exactly why it has been so successful in learning representations for downstream predictors. One explanation is the information maximization (InfoMax) principle (Oord et al., 2018; Bachman et al., 2019), which states that the representation should contain maximum information about the data. Notably, contrastive learning optimizes a lower-bound to the mutual information (MI) between the representation and the data (Poole et al., 2019). However, Tschannen et al.

(2020) argue the success of contrastive approach cannot be attributed to the InfoMax principle alone but strongly rely on the properties of MI estimators and architectural choices. In this work, we present a new approach for representation learning that maximizes the nonlinear expressivity of the encoder for downstream predictors.

**Unsupervised deep hashing**. Deep hashing aims to learn binary representations (i.e., *hash codes*) of data using deep neural networks (Salakhutdinov & Hinton, 2009; Krizhevsky & Hinton, 2011; Lin et al., 2016; Hu et al., 2017; Yang et al., 2018; 2019). The binary nature of the code admits efficient computation of nearest neighbor search algorithms for large-scale data, with minimal memory footprint. Lin et al. (2016) treat images and their rotated ones as similar pairs and learn rotation invariant hash mapping. Hu et al. (2017) is similar to NAC in that it maximizes the mutual information between the hash code and the data. However, they ignore higher-order interactions between the code bits in order to derive an approximation to the MI. On the other hand, NAC lower-bounds the MI using variational inference and subsampling, and further promotes noise-robustness of the code by introducing a noisy communication channel. More recently, Yang et al. (2018; 2019) exploit similarity of deep features to construct pseudo labels for hash code learning and achieve the state-of-the-art performance on natural image datasets.

## 3. Approach

### 3.1. Activation Code in ReLU Networks

Consider a deep neural network (DNN) with ReLU activation $\text{ReLU}(x) = \max(0, x)$:

$$\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \tag{1}$$

$$\mathbf{h}^{(l)} = \text{ReLU}(\mathbf{a}^{(l)}), \quad l = 1, 2, \ldots, L \tag{2}$$

where ReLU is applied element-wise and $L$ is the number of layers. We set $\mathbf{h}^{(0)} = \mathbf{x}$. The DNN is a piece-wise linear function that segments the input space into a set of distinct locally linear regions (Figure 1) (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017).

For layer $l$, we define the *activation code* as the binary string

$$\mathbf{c}^{(l)} = \text{sgn}(\mathbf{a}^{(l)}) \in \{-1, 1\}^D, \tag{3}$$

where $D$ is the number of hidden units in a layer. The activation code represents the *activation pattern* of the network that uniquely identifies a linear region. Hereafter, we focus on the last layer activation code $\mathbf{c}^{(L)}$ and will drop the superscript when there is no ambiguity.

The DNN maps each training data point $\mathbf{x}_i \in \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ to an activation codeword $\mathbf{c}_i$ (Equations (1) to (3)). The distance between two codewords $\mathbf{c}_i, \mathbf{c}_j \in \{-1, 1\}^D$ is measured using the *Hamming distance*:

$$d_H(\mathbf{c}_i, \mathbf{c}_j) = \frac{D - \mathbf{c}_i \cdot \mathbf{c}_j}{2}, \tag{4}$$

which counts the number of different bits between the two codewords. $\mathbf{c}_i \cdot \mathbf{c}_j$ denotes the dot product of the two codewords. The distance is the minimum number of distinct linear regions that one has to traverse along a path from $\mathbf{x}_i$ to $\mathbf{x}_j$. Therefore, the average distance or separability between the codewords serves as a measure of the effective number of linear regions on the data (Raghu et al., 2017).

While we limit our attention to ReLU in this work, similar analyses may apply to a broad class of other activation functions. For example, Leaky ReLU and MaxOut are already piece-wise linear functions. Exponential linear units (ELUs) and Gaussian error linear units (GELUs) can be seen as smooth approximations to ReLU. Sigmoid and hyperbolic tangent (tanh) activations also have distinct modes of operation; they behave like a linear function near the origin but gradually saturate to constant functions further away from the origin. Hence, they can be considered as smooth approximations to piece-wise linear functions too.

### 3.2. Neural Activation Coding

For a ReLU encoder, the distribution of that activation codewords $\{\mathbf{c}_1, ..., \mathbf{c}_N\}$ has significant implications for downstream applications as the nonlinear expressivity of the encoder is determined as the number of distinct activation codewords (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017), For example, if a set of data are mapped to the same codeword on the DNN, it means that they lie in the same linear region of the encoder. Therefore, downstream linear models won't be able to express any nonlinear relationships between these examples. This suggests that a good encoder network should attain high nonlinear expressivity by mapping the data to as many unique activation codewords as possible. This is the key motivation behind *Neural Activation Coding* (NAC) which proposes to maximize the nonlinear expressivity of the encoder for representation learning.

NAC maximizes the mutual information (MI) between the *activation code* and the data over a noisy communication channel $\mathbf{X} \rightarrow \mathbf{C} \rightarrow \widetilde{\mathbf{C}}$. Suppose the message $\mathbf{x}_i \in \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is selected uniformly at random from the dataset. The sender first encodes the message $\mathbf{x}_i$ into the activation codeword $\mathbf{c}_i$ (Equations (1) to (3)) and transmits it through the noisy channel. The receiver tries to reconstruct the message from the noisy code $\widetilde{\mathbf{c}}_i$. In order for the receiver to correctly decode the message with high probability, the codewords $\{\mathbf{c}_1, ..., \mathbf{c}_N\}$ should be easily separable; in other words maximally distant from each other (MacKay, 2003). The amount of information that the receiver gains from the

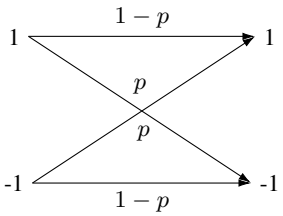



*Figure 2.* A symmetric noise channel used in NAC. Each bit of code is independently flipped with probability $p$.

communication is quantified as the mutual information between the noisy activation code and the data: $I(\mathbf{X}, \widetilde{\mathbf{C}})$. Thus maximizing $I(\mathbf{X}, \widetilde{\mathbf{C}})$ leads to noise-robust codewords that are maximally distant from each other, which translates to maximum nonlinear expressivity of the the encoder.

**Symmetric noise channel**. We consider a symmetric noise channel where the bits of $\mathbf{c}$ are randomly flipped with probability $p$ (Figure 2) to create a noisy code $\tilde{\mathbf{c}}$. The total number of flipped bits is given by the Hamming distance $d_H(\tilde{\mathbf{c}}, \mathbf{c}) = (D - \tilde{\mathbf{c}} \cdot \mathbf{c})/2$. The conditional probability of transmitted message $\tilde{\mathbf{c}}$ given a codeword $\mathbf{c}_i$ is

$$P(\tilde{\mathbf{c}}|\mathbf{c}_i) = p^{d_H(\tilde{\mathbf{c}},\mathbf{c}_i)}(1-p)^{D-d_H(\tilde{\mathbf{c}},\mathbf{c}_i)} \tag{5}$$

$$= p^{(D-\tilde{\mathbf{c}}\cdot\mathbf{c}_i)/2}(1-p)^{(D+\tilde{\mathbf{c}}\cdot\mathbf{c}_i)/2} \tag{6}$$

$$= \exp((\tilde{\mathbf{c}} \cdot \mathbf{c}_i)\frac{1}{2}\log\frac{1-p}{p} + \frac{D}{2}\log p(1-p)). \tag{7}$$

Accordingly, the marginal distribution of the message is

$$P_\theta(\tilde{\mathbf{c}}) = \sum_{j,\mathbf{c}} P_{\text{data}}(\mathbf{x}_j)P_\theta(\mathbf{c}|\mathbf{x}_j)P(\tilde{\mathbf{c}}|\mathbf{c}) \tag{8}$$

$$= \frac{1}{N}\sum_{j=1}^{N}\exp((\tilde{\mathbf{c}} \cdot \mathbf{c}_j)\frac{1}{2}\log\frac{1-p}{p} + \frac{D}{2}\log p(1-p)),$$

where $\theta$ denotes the parameters of the encoder and the data distribution is assumed to be uniform ($P_{\text{data}}(\mathbf{x}_j) = 1/N$) over the training examples. The activation code is a deterministic function of the input i.e., $P_\theta(\mathbf{c}_j|\mathbf{x}_j) = 1$.

The mutual information between the message and data is

$$I(\mathbf{X}, \widetilde{\mathbf{C}}) = \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}\left[\log\frac{P_\theta(\tilde{\mathbf{c}}|\mathbf{c}_i)}{P_\theta(\tilde{\mathbf{c}})}\right] \tag{9}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\tilde{\mathbf{c}}|\mathbf{c}_i}\left[\log\frac{\exp((\tilde{\mathbf{c}} \cdot \mathbf{c}_i)\frac{1}{2}\log\frac{1-p}{p})}{\frac{1}{N}\sum_j \exp((\tilde{\mathbf{c}} \cdot \mathbf{c}_j)\frac{1}{2}\log\frac{1-p}{p})}\right],$$

which follows from Equations (7) and (8). Note that $\log\frac{1-p}{p} > 0$ for $p < 0.5$ (i.e., the channel has non-zero capacity). From the denominator, we see maximizing $I(\mathbf{X}, \widetilde{\mathbf{C}})$ will minimize the similarity (i.e., maximize the Hamming distance) between the codewords and consequently maximize the number of distinct linear regions of the encoder.

### 3.3. Data Augmentation

Data augmentations play a significant role in self-supervised learning of natural image representations (Chen et al., 2020a; Hénaff et al., 2020; Wu et al., 2018). Common augmentation methods for images include horizontal flipping, random cropping, color jittering, etc. We incorporate data augmentations into NAC by modifying the communication channel as $\mathbf{X} \to \widetilde{\mathbf{X}} \to \mathbf{C} \to \widetilde{\mathbf{C}}$, where $\widetilde{\mathbf{X}}$ is the augmented version of $\mathbf{X}$ as a result of applying stochastic data augmentations. The mutual information is now

$$I(\mathbf{X}, \widetilde{\mathbf{C}}) = \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}\left[\log\frac{P_\theta(\tilde{\mathbf{c}}|\mathbf{x})}{P_\theta(\tilde{\mathbf{c}})}\right]. \tag{10}$$

Both the numerator and the denominator in Equation (10) is no longer tractable since it requires marginalization over $\tilde{\mathbf{x}}$. We therefore construct lower-bounds for each term using (i) variational inference and (ii) subsampling, respectively as described below.

**Variational inference**. We lower-bound the numerator using an amortized variational distribution $Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})$:

$$\mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}[\log Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})] \tag{11}$$

$$= \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}[\log P_\theta(\tilde{\mathbf{c}}|\mathbf{x})] - D_{KL}(P_\theta(\tilde{\mathbf{c}}|\mathbf{x})\|Q_\phi(\tilde{\mathbf{c}}|\mathbf{x}))]$$

$$\leq \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}[\log P_\theta(\tilde{\mathbf{c}}|\mathbf{x})], \tag{12}$$

where the expectation is taken over $P_\theta(\mathbf{x}, \tilde{\mathbf{c}})$, and the inequality stems from the non-negativity of KL-divergence. From the bound, see that maximizing Equation (11) in turn minimizes $D_{KL}(P_\theta(\tilde{\mathbf{c}}|\mathbf{x})\|Q_\phi(\tilde{\mathbf{c}}|\mathbf{x}))]$, driving the variational distribution closer to the true conditional. We adopt a mean-field approach by setting $Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})$ as a product of Bernoulli distributions (mind that here $\tilde{c}_d \in \{-1, 1\}$):

$$Q_\phi(\tilde{\mathbf{c}}|\mathbf{x}) = \prod_{d=1}^{D} q_d^{(1+\tilde{c}_d)/2}(1-q_d)^{(1-\tilde{c}_d)/2}, \tag{13}$$

where $D$ is the dimension of the code. We introduce an *inference network* $\phi$ to output the logit $r_d = \log\frac{q_d}{1-q_d}$ for each bit and then apply sigmoid function to obtain $q_d = \sigma(r_d)$. The expectation of Equation (11) is evaluated as

$$\mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}[\log Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})] \tag{14}$$

$$= \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}\left[\frac{1}{2}\sum_{d=1}^{D}\tilde{c}_d\log\frac{q_d}{1-q_d} + \log q_d(1-q_d)\right] \tag{15}$$

$$= \frac{1}{2}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}}}\left[\tilde{\mathbf{c}} \cdot \mathbf{r}_\phi(\mathbf{x}) + \mathbf{1} \cdot \log\sigma(\mathbf{r}_\phi(\mathbf{x}))(\mathbf{1} - \sigma(\mathbf{r}_\phi(\mathbf{x})))\right],$$

where $\mathbf{r}_\phi(\mathbf{x})$ is the $D$-dimensional logit vector, and $\mathbf{1}$ is the same-sized one vector. The sigmoid and the log functions are applied element-wise.

**Subsampling**. On the other hand, the denominator of Equation (10) can be lower-bounded using $2K$ subsamples $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2K}$ (Poole et al., 2019; Chen et al., 2020a). Specifically, we first sample $K$ examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K \sim P_{\text{data}}(\mathbf{x})$ and draw two augmented versions per each example $\tilde{\mathbf{x}}_{2k-1}, \tilde{\mathbf{x}}_{2k} \sim P_{\text{aug}}(\tilde{\mathbf{x}}|\mathbf{x}_k)$. Finally, the encoder network maps the augmented samples $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_{2K}$ to activation codewords $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2K}$. The bound is constructed as

$$\mathbb{E}_{\tilde{\mathbf{c}}}\left[\log \frac{1}{P_\theta(\tilde{\mathbf{c}})}\right] \tag{16}$$

$$\leq \mathbb{E}_{\tilde{\mathbf{c}},\mathbf{c}_1,\ldots\mathbf{c}_{2K}}\left[\log \frac{1}{\frac{1}{2K}\sum_{k=1}^{2K} P(\tilde{\mathbf{c}}|\mathbf{c}_k)}\right] \tag{17}$$

$$= \mathbb{E}_{\tilde{\mathbf{c}},\mathbf{c}_1,\ldots\mathbf{c}_{2K}}\left[\log \frac{1}{\frac{1}{2K}\sum_{k=1}^{2K} \exp((\tilde{\mathbf{c}}\cdot\mathbf{c}_k)\frac{1}{2}\log\frac{1-p}{p})}\right].$$

**NAC objective**. Combining the two bounds above (Equations (11) and (16)), we arrive at our objective

$$\mathcal{L}_{\text{NAC}} = \mathbb{E}_{\mathbf{x},\tilde{\mathbf{c}},\mathbf{c}_1,\ldots,\mathbf{c}_{2K}}\left[\log \frac{Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})}{\frac{1}{2K}\sum_{k=1}^{2K} P_\theta(\tilde{\mathbf{c}}|\mathbf{c}_k)}\right], \tag{18}$$

which is a lower-bound to the mutual information $I(\mathbf{X}, \widetilde{\mathbf{C}})$.

### 3.4. Optimization

The objective of Equation (18) involves discrete codewords and does not admit gradient-based optimization. To circumvent this issue, we adopt continuous relaxation (Cao et al., 2017) and replace the discrete code $\mathbf{c} = \text{sgn}(\mathbf{a}) \in \{-1, 1\}^D$ with a soft approximation $\mathbf{z} = \tanh(\mathbf{a}) \in [-1, 1]^D$ where $\mathbf{a}$ is the last preactivation of the encoder:

$$\widetilde{\mathcal{L}}_{\text{NAC}} \tag{19}$$

$$= \frac{1}{2}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}\cdot\mathbf{r}_\phi(\mathbf{x}) + \mathbf{1}\cdot\log\sigma(\mathbf{r}_\phi(\mathbf{x}))(\mathbf{1}-\sigma(\mathbf{r}_\phi(\mathbf{x})))\right]$$

$$- \mathbb{E}_{\tilde{\mathbf{z}},\mathbf{z}_1,\ldots,\mathbf{z}_{2K}}[\log\frac{1}{2K}\sum_{k=1}^{2K}\exp((\tilde{\mathbf{z}}\cdot\mathbf{z}_k)\frac{1}{2}\log\frac{1-p}{p})].$$

We apply stochastic gradient optimization by sampling a mini-batch of examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K \sim P_{\text{data}}(\mathbf{x})$ at each iteration. The gradients with respect to the parameters of the encoder $\theta$ and the inference network $\phi$ are computed using backpropagation on Equation (19).

### 3.5. Model Architecture

Figure 3 overviews the NAC architecture. The encoder takes the augmented data $\tilde{\mathbf{x}}$ as input and produces the representation $\mathbf{h}$. Following Chen et al. (2020a), Chen et al. (2020c), we attach a projection head at the end of the encoder. It is an MLP with one hidden layer that maps the encoder presentation $\mathbf{h}$ to the lower-dimensional feature $\mathbf{z}$. The activation code $\mathbf{c}$ is obtained by applying a sign function on $\mathbf{z}$. Similarly, the inference network shares the same encoder backbone and predicts the logit vector $\mathbf{r}$ from the encoder representation $\mathbf{h}$. The logit $\mathbf{r}$ defines the variational distribution $Q_\phi(\tilde{\mathbf{c}}|\mathbf{x})$ (Equation (13)). However, directly using the encoder representation $\mathbf{h}$ from the same path allows the inference network to easily cheat. Therefore, we build two independent pathways by sampling two augmented versions of data $\tilde{\mathbf{x}}_{2k-1}, \tilde{\mathbf{x}}_{2k} \sim P(\tilde{\mathbf{x}}|\mathbf{x}_k)$. The inference network takes the encoder representation of one pathway (e.g., $\mathbf{h}_{2k-1}$) and outputs the logits for the other pathway (e.g. $\mathbf{r}_{2k}$) and vice versa (Figure 3).

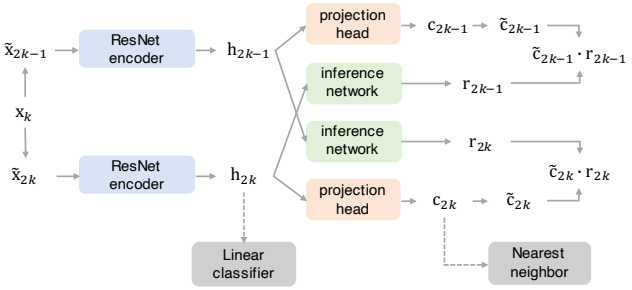



*Figure 3.* The NAC architecture. The inference network takes the encoder representation from one pathway and predict the logits for the other pathway. The negative samples are not depicted here. After training, the encoder representation and the activation code are respectively applied to linear classification and nearest neighbor search.

For ImageNet experiments, we incorporate the momentum queue (MQ) (He et al., 2020) in order to reduce the memory overhead. The momentum queue maintains $M$ momentum features $\mathbf{v}_1, ..., \mathbf{v}_M$ from previous iterations. In addition, we introduce a momentum model $\widehat{\mathbf{r}}_\phi(\mathbf{x})$ for the inference network as well. In our experiments, we find that the discrepancy between the norms of current model features and momentum features causes instability during the training of NAC-MQ models and add $\ell_2$-regularization term on the norm of the feature $\mathbf{z}$ in order to stabilize the training:

$$\widetilde{\mathcal{L}}_{\text{NAC-MQ}} \tag{20}$$

$$= \frac{1}{2}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{z}}}\left[\tilde{\mathbf{z}}\cdot\widehat{\mathbf{r}}_\phi(\mathbf{x}) + \mathbf{1}\cdot\log\sigma(\widehat{\mathbf{r}}_\phi(\mathbf{x}))(\mathbf{1}-\sigma(\widehat{\mathbf{r}}_\phi(\mathbf{x})))\right]$$

$$- \mathbb{E}_{\mathbf{z},\tilde{\mathbf{z}}}\left[\log\frac{1}{M}\sum_{m=1}^{M}\exp((\tilde{\mathbf{z}}\cdot\mathbf{v}_m)\frac{1}{2}\log\frac{1-p}{p}) + \lambda\|\mathbf{z}\|_2^2\right],$$

where $\lambda$ controls the strength of $\ell_2$ regularization. We use $\lambda = 0.1$ in our experiments.

### 3.6. Comparison to Contrastive Learning

We compare NAC to the contrastive learning objective of SimCLR (Chen et al. (2020a)) and highlight a few distinguishing traits. The SimCLR objective for example $\mathbf{x}_i$ is

$$\mathcal{L}^{(i)}_{\text{SimCLR}} = \log \frac{\exp((\mathbf{u}_i \cdot \mathbf{u}'_i)/\tau)}{\sum_{k=1}^{2K} \mathbb{I}_{[k\neq i]} \exp((\mathbf{u}_i \cdot \mathbf{u}_k)/\tau)}, \quad (21)$$

$$\text{where } \mathbf{u}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|} \text{ for } k = 1, \dots, 2K. \quad (22)$$

where $\mathbf{u}_i, \mathbf{u}'_i$ are the features of two augmented versions of the same image (i.e., a *positive pair*) and $\mathbf{u}_1, \dots, \mathbf{u}_{2K}$ are the *negative samples*. $\mathbb{I}$ is an indicator function. Notably, contrastive learning $\ell_2$-normalizes the features so that they lie on the unit hypersphere and use a temperature parameter $\tau$ to control the concentration of the distribution.

For comparison, we can rewrite the relaxed NAC objective in Equation (19) as

$$\widetilde{\mathcal{L}}^{(i)}_{\text{NAC}} = \log \frac{\exp((\tilde{\mathbf{z}}_i \cdot \mathbf{r}_\phi(\mathbf{x}_i))/2)}{\sum_{k=1}^{2K} \exp((\tilde{\mathbf{z}}_i \cdot \mathbf{z}_k)\frac{1}{2}\log\frac{1-p}{p})}. \quad (23)$$

We see that both methods take similar forms that incentivize minimizing the feature similarity between the negatives and maximizing the similarity for the positive pair. However, SimCLR uses the sample feature of the same example as the positive, while NAC lets the inference network to predict the distribution $P_\theta(\tilde{\mathbf{z}}|\mathbf{x})$ to guide the feature.

Moreover, NAC does not apply $\ell_2$-normalization to the features (Equation (22)) but still learns good representations of data, even though the role of $\ell_2$-normalization has been considered crucial in self-supervised representation learning (Wu et al., 2018; Chen et al., 2020a; Wang & Isola, 2020). We hypothesize that leaving out explicit normalization may be beneficial since it allows the model to represent uncertainty of its predictions in the norm of its features. To see this, we rewrite the NAC objective (Equation (23)) using the $\ell_2$-normalized features as

$$\widetilde{\mathcal{L}}^{(i)}_{\text{NAC}} = \log \frac{\exp((\tilde{\mathbf{u}}_i \cdot \mathbf{r}_\phi(\mathbf{x}_i))/2)^{\|\tilde{\mathbf{z}}_i\|}}{\sum_{k=1}^{2K} \exp((\tilde{\mathbf{u}}_i \cdot \mathbf{z}_k)\frac{1}{2}\log\frac{1-p}{p})^{\|\tilde{\mathbf{z}}_i\|}} \quad (24)$$

$$\text{where } \tilde{\mathbf{u}}_k = \frac{\tilde{\mathbf{z}}_k}{\|\tilde{\mathbf{z}}_k\|} \text{ for } k = 1, \dots, 2K. \quad (25)$$

Equation (24) can be interpreted as a softmax distribution where the norm of feature $\|\tilde{\mathbf{z}}_i\|$ dynamically controls the concentration of the distribution. When the encoder is confident, it outputs large $\|\tilde{\mathbf{z}}_i\|$ to make the distribution sharp. Otherwise, it assigns small $\|\tilde{\mathbf{z}}_i\|$ to smooth the distribution reflecting its uncertainty. This is not possible for SimCLR that enforces the features to be on the unit hypersphere; thus, its performance is sensitive to tuning of the temperature parameter $\tau$ (Wu et al., 2018; Chen et al., 2020a).

Finally, contrastive representation learning (Oord et al., 2018; Poole et al., 2019; Chen et al., 2020a) maximizes the MI between the representation and the data $I(\mathbf{X}, \mathbf{Z})$. In contrast, the goal of NAC is in maximizing the nonlinear expressivity of the encoder for downstream predictors. This is achieved by maximizing the MI between the activation code and the data $I(\mathbf{X}, \widetilde{\mathbf{C}})$ over a noisy communication channel. The noisy channel promotes noise-robust codewords which leads to high nonlinear encoder expressivity.

## 4. Experiments

We assess the quality of continuous and discrete representations learned by NAC on two downstream tasks respectively: (i) linear classification on CIFAR-10 and ImageNet-1K. (ii) nearest neighbor search using the activation hash code on CIFAR-10 and FLICKR-25. In addition, we explore whether deep generative models can benefit from enhanced encoder expressivity from NAC pretraining. We show that NAC attains better or comparable performance to recent methods including SimCLR (Chen et al., 2020a) and DistillHash (Yang et al., 2019) on the downstream tasks and provides significant improvement for the training of variational autoencoders (VAEs) (Kingma & Welling, 2014).

### 4.1. Experimental Details

Following previous works (Chen et al., 2020a; He et al., 2020; Chen et al., 2020c), we use ResNet architecture with ReLU activation as our encoders. The projection head is an MLP with one hidden layer with ReLU activation. The feature/code dimension is set to 128. The inference network has the identical structure to the projection head. For optimization, we use LARS optimizer (You et al., 2017) with linear warmup for the first 10 epochs followed by cosine learning rate decay. We apply the same set of data augmentations including horizontal flipping, random cropping and resizing, color distortions and Gaussian blur used in Chen et al. (2020a;c). We set weight decay to $10^{-6}$. For multi-GPU training, we adopt batch shuffling (He et al., 2020) to prevent the information leak in batch normalization layers.

**CIFAR-10**. We use a batch size of 1000 and train the encoder for 1000 epochs. Following Chen et al. (2020a), we exclude Gaussian blur in CIFAR-10 experiments. The learning rate is set to 3.0 with momentum 0.9. For the models with momentum queue, we set the size of the queue to 50000 and the moving average decay to 0.99.

**ImageNet**. We use a batch size of 512 and train the encoder for 200 epochs. The learning rate is set to 1.7 following the square root scaling rule ($0.075 \times \sqrt{\text{batch size}}$) (Chen et al., 2020a) with momentum 0.9. For the models with momentum queue, we set size of the queue to 65536 and the moving average decay to 0.999, following (He et al., 2020).

*Table 1.* Linear evaluation accuracy (top-1) on CIFAR-10 dataset using ResNet-50 encoders. Trained for 1000 epochs.

| Model | Accuracy (%) |
|---|---|
| *Contrastive Learning Methods:* | |
| InsDis (Wu et al., 2018) | 80.8 * |
| SimCLR (Chen et al., 2020a) | 92.8 † |
| MoCo-v2 (Chen et al., 2020c) | 91.6 † |
| NAC | **93.9** |
| NAC + Momentum Queue | 93.8 |

* Obtained using ResNet-18 encoder and kNN classifier
† Re-implemented for multi-GPU training.

*Table 2.* Linear evaluation accuracy (top-1) on ImageNet 1K dataset using ResNet-50 encoders. Trained for 200 epochs.

| Model | Accuracy (%) |
|---|---|
| *Contrastive Learning Methods:* | |
| InsDis (Wu et al., 2018) | 54.0 |
| CMC (Tian et al., 2019) | 60.0 |
| LocalAgg (Zhuang et al., 2019) | 60.2 |
| Moco (He et al., 2020) | 60.6 |
| SimCLR (Chen et al., 2020a) | 66.6 |
| Moco-v2 (Chen et al., 2020c) | **67.5** |
| NAC + Momentum Queue | 65.0 |

### 4.2. Linear Image Classification

For downstream classification, the projection head is detached from the encoder, and the encoder representation is fed into a linear classifier. The encoder network is kept fixed and only the linear model is learned using the supervision. We measure the top-1 classification accuracy of the classifiers on the test set. It is a popular evaluation procedure for assessing the quality of continuous deep representations (Hénaff et al., 2020; He et al., 2020; Wu et al., 2018).

The linear classifier is trained using Nesterov optimizer with momentum 0.9 for 100 epochs where the learning rate is searched among $\{0.01, 0.1, 1.0, 10.0\}$. We do not apply any regularization on the classifier. For CIFAR-10 experiments, we re-implement SimCLR (Chen et al., 2020a) and MoCo-v2 (Chen et al., 2020c) for multi-GPU training and report the corresponding results for fair comparison.

Table 1 summarizes the downstream linear classification results on CIFAR-10. The NAC outperforms the state-of-the-art baselines by over 1%p. We find that the MoCo-v2 attains slightly worse results than the SimCLR on CIFAR-10, while the NAC + Momentum Queue shows comparable performance to the vanilla NAC.

Table 2 shows the linear classification results on ImageNet. NAC falls slightly behind the state-of-the-art baselines in ImageNet 1K but there is room for additional improvements as we have not run extensive hyperparameter search due to computational constraints.

### 4.3. Nearest Neighbor Search Using Deep Hash Codes

The goal of deep hashing is to utilize deep neural networks to learn binary vector code $\mathbf{c}$ (i.e., hash codes) of high-dimensional data $\mathbf{x}$ for efficient nearest neighbor retrieval (Krizhevsky et al., 2012; Erin Liong et al., 2015; Do et al., 2016). Hamming distance is used for ranking and the binary nature of the code admits efficient nearest neighbor search for large-scale datasets with minimal memory footprint. The retrieved image is considered relevant if it belongs to the same class as the query image. The hash code table is populated using the training images and the test images are used as queries. The performance is measured using mean Average Precision (mAP) (Luo et al., 2020), which computes the average area under the precision-recall curve. For NAC, the activation code of the projection head is used as the hash code. We compare NAC against recent deep hashing methods (Lin et al., 2016; Yang et al., 2018; 2019) .

The hash code performance on CIFAR-10 is summarized in Table 3. For reference, we also evaluate the performance of contrastive methods of SimCLR (Chen et al., 2020c) and MoCo-v2 (Chen et al., 2020c) by discretizing the models' output using a sign function. Following Yang et al. (2019), we use a VGG16 encoder and train the models on 10% of the dataset for fair comparison. We train the encoder from scratch, while the deep hashing baselines finetune a pre-trained VGG16 encoder. NAC outperforms all the baselines by significant margins as it learns maximally *separable* codewords by promoting noise-robustness. Interestingly, the contrastive learning methods of SimCLR and Moco-v2 surpass the deep hashing baselines, even though they are not explicitly designed for learning hash codes. This may be attributed to the use of strong data augmentations that these models incorporate during training.

Table 4 shows the results on FLICKR-25K. Following Yang et al. (2019), we start from a VGG16 encoder pretrained on ImageNet-1K classification and finetune the model using NAC. Against the deep hashing baselines, NAC attains the highest mAP. However, as ImageNet pretraining already provides a strong baseline for FLICKR-25K, we find that the performance margins are not as significant as in the CIFAR-10 results.

### 4.4. Encoder Pretraining for Deep Generative Models

The deep generative models of variational autoencoders (VAEs) (Kingma & Welling, 2014) take the encoder-decoder architecture where the decoder defines the generative distri-

*Table 3.* Retrieval performance of unsupervised hash code (128 bit) on CIFAR-10. The baselines results for deep hashing methods are excerpted from (Yang et al., 2019). The models are trained on 10% of the data using VGG16 encoders.

| Model | mAP (%) |
|---|---|
| *Deep hashing methods* | |
| DeepBit (Lin et al., 2016) | 25.3 |
| SSDH (Yang et al., 2018) | 26.0 |
| DistillHash (Yang et al., 2019) | 29.0 |
| *Contrastive learning methods* | |
| MoCo-v2 (Chen et al., 2020c) | 32.3 |
| SimCLR (Chen et al., 2020a) | 34.2 |
| NAC | **40.5** |

*Table 4.* Retrieval performance of unsupervised hash code (128 bit) on FLICKR-25K. The baselines results for deep hashing methods are excerpted from (Yang et al., 2019). The models are trained on 5000 images using VGG16 encoders pretrained on ImageNet-1K.

| Model | mAP (%) |
|---|---|
| DeepBit (Lin et al., 2016) | 59.3 |
| SSDH (Yang et al., 2018) | 66.2 |
| DistillHash (Yang et al., 2019) | 70.0 |
| NAC | **70.8** |

*Table 5.* Comparison of VAE performance using random initialization and unsupervised pretraining on CIFAR-10. The loglikelihoods are estimated with importance sampling. Trained for 100 epochs.

| Encoder | Loglikelihood | KL divergence |
|---|---|---|
| Random init. | -3202 | 33.0 |
| SimCLR + finetune | -3174 | 38.9 |
| MoCo-v2 + finetune | -3103 | 32.2 |
| NAC + finetune | **-2865** | **71.8** |

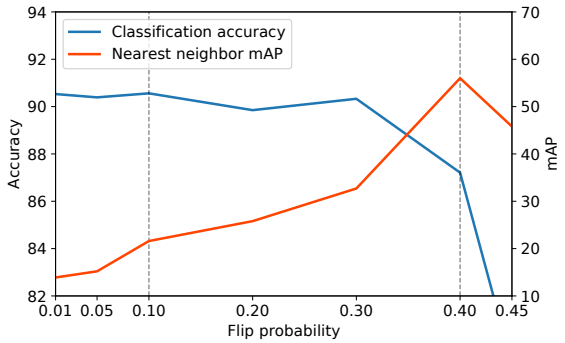



*Figure 4.* CIFAR-10 downstream performance using ResNet-18 encoders with different flip probabilities. The dashed lines denote the values of $p$ that attains the best accuracy on respective tasks.

bution given a latent variable, and the encoder predicts the posterior distribution of the latent variable. However, VAEs suffer from suboptimality of the encoder (Cremer et al., 2018; Marino et al., 2018; Kim et al., 2018), which leads to biased learning signals. This problem is exacerbated by several factors, namely: (i) the encoder is randomly initialized, leading to the *cold start* problem. (ii) the encoder is never trained to the optimality. This may be mitigated with additional optimization steps, but they come with significant computational overheads. (iii) To make the problem worse, the learning target for the encoder constantly changes as the decoder is jointly updated with the encoder.

Although even linear models can learn complex functions when combined with pretrained encoders, we hypothesize that NAC pretraining for the encoder can benefit VAEs by maximizing the encoder's nonlinear expressivity. Specifically, we pretrain the encoder using NAC and only randomly initialize the linear output layer to predict the posterior distribution of the latent variable. This allows us to apply higher learning rates on the linear output layer to speed up the training of the encoder and consequently improve the quality of amortized inference. We also finetune the encoder by backpropagating through the linear output layer.

Table 5 summarizes the results on CIFAR-10 using ResNet-18 architecture. We compare the NAC pretraining against random initialization and pretraining using self-supervised methods. Using the NAC pretraining achieves significantly higher loglikelihood and KL divergence compared to the baselines. This suggests that maximizing the encoder expressivity using NAC facilitates more active use of latent variable and significantly enhances the training of VAEs.

### 4.5. The Effect of Noisy Communication Channel

The noisy communication channel in NAC controls the overall difficulty of task with the bit flip probability $p$ of the channel. Figure 4 plots the effect of the flip probability on the downstream classification and nearest neighbor search on CIFAR-10 . Interestingly, while the smaller noise probability of $p = 0.1$ is favorable for linear classification, the nearest neighbor performance peaks at the higher noise level of $p = 0.4$, suggesting discrete hash code benefits more from improved noise-robustness. On the other hand, the continuous representation suffers as the task becomes too difficult, agreeing with the findings of Chen et al. (2020a). Note that in the limit $p \to 0.5$, the transmitted code becomes completely random, making the task impossible to solve.

## 5. Conclusion

We proposed Neural Activation Coding (NAC) for unsupervised representation learning. NAC maximizes the nonlinear expressivity of the encoder by formulating a communication problem over a noisy communication channel using the activation code of the encoder. Through empirical evaluations, we demonstrated that NAC can improve the performance of downstream applications as well as enhance the training of deep generative models. As future work, it is worthwhile to explore the use of NAC on other data domains such as natural language.

## Acknowledgements

This work is supported by ONR N00014-17-1-2131, ONR N00014-15-1-2209, NIH 1U01MH115727-01, NSF CCF-1740833, DARPA SD2 FA8750-18-C-0130, Amazon and Simons Foundation. Sangho Lee and Gunhee Kim are supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01772, Video Turing Test, No.2019-0-01082, SW StarLab).

We thank Christian A. Naesseth for helpful discussion.

## References

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *ICLR*, 2018.

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pp. 15535–15545, 2019.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *CoNLL*, 2016.

Cao, Z., Long, M., Wang, J., and Yu, P. S. Hashnet: Deep learning to hash by continuation. In *CVPR*, 2017.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *ICML*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Do, T.-T., Doan, A.-D., and Cheung, N.-M. Learning to hash with binary deep neural network. In *ECCV*, 2016.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

Erin Liong, V., Lu, J., Wang, G., Moulin, P., and Zhou, J. Deep hashing for compact binary codes learning. In *CVPR*, 2015.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Hanin, B. and Rolnick, D. Complexity of linear regions in deep networks. In *ICML*, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Hénaff, O. J., Srinivas, A., De Fauw, J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. Learning discrete representations via information maximizing self-augmented training. In *ICML*, 2017.

Kim, Y., Wiseman, S., Millter, A. C., Sontag, D., and Rush, A. M. Semi-amortized variational autoencoders. In *ICML*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR*, 2014.

Krizhevsky, A. and Hinton, G. E. Using very deep autoencoders for content-based image retrieval. In *ESANN*, volume 1, pp. 2. Citeseer, 2011.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Lin, K., Lu, J., Chen, C.-S., and Zhou, J. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, 2016.

Luo, X., Chen, C., Zhong, H., Zhang, H., Deng, M., Huang, J., and Hua, X. A survey on deep hashing methods. *arXiv preprint arXiv:2003.03369*, 2020.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Marino, J., Yisong, Y., and Mandt, S. Iterative amortized inference. In *ICML*, 2018.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *NeurIPS*, 2014.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Pascanu, R., Montufar, G., and Bengio, Y. On the number of response regions of deep feed forward networks with piece-wise linear activations. In *ICLR*, 2013.

Poole, B., Ozair, S., Oord, A. v. d., Alemi, A. A., and Tucker, G. On variational bounds of mutual information. In *ICML*, 2019.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. In *ICML*, 2017.

Salakhutdinov, R. and Hinton, G. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7): 969–978, 2009.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. Bounding and counting linear regions of deep neural networks. In *ICML*, 2018.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. On mutual information maximization for representation learning. In *ICLR*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

Yang, E., Deng, C., Liu, T., Liu, W., and Tao, D. Semantic structure-based unsupervised deep hashing. In *IJCAI*, 2018.

Yang, E., Liu, T., Deng, C., Liu, W., and Tao, D. Distillhash: Unsupervised deep hashing by distilling data pairs. In *CVPR*, 2019.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. In *CVPR*, 2019.