Transformation-Equivariant Representation Learning with Barber-Agakov and InfoNCE Mutual Information Estimation

Marshal Arijona Sinaga, T. Basarrudin and Adila Alfa Krisnadhi
Faculty of Computer Science, University of Indonesia, Depok, Indonesia

Keywords: Representation Learning, Transformation-Equivariant, Mutual Information Estimation, Barber-Agakov, InfoNCE.

Abstract: The success of deep learning on computer vision tasks is due to the convolution layer being equivariant to the translation. Several works attempt to extend the notion of equivariance into more general transformations. Autoencoding variational transformation (AVT) achieves state of art by approaching the problem from the information theory perspective. The model involves the computation of mutual information, which leads to a more general transformation-equivariant representation model. In this research, we investigate the alternatives of AVT called variational transformation-equivariant (VTE). We utilize the Barber-Agakov and information noise contrastive mutual information estimation to optimize VTE. Furthermore, we also propose a sequential mechanism that involves a self-supervised learning model called predictive-transformation to train our VTE. Results of experiments demonstrate that VTE outperforms AVT on image classification tasks.

1 INTRODUCTION

The success of convolutional neural networks (CNN) in computer vision tasks is due to the equivariant property (Hinton et al., 2011; Cohen and Welling, 2016). Specifically, the CNN extracts features/representations that are equivariant to the translation. In general, transformation-equivariant guarantees the obtained representation changes in the same way as we transform the image. Figure 1 shows the illustration of transformation-equivariant. This property enables CNN to extract a better representation structure from the given image. Some efforts have been made such that CNN can handle various types of transformations. However, current methods are restricted to discrete transformations. Such circumstance limits the capacity of CNN to capture visual structure under more complex transformation, including continuous and non-linear transformations.

An unsupervised approach solves the limitation of the transformations. State of the art utilizes an autoencoder called autoencoding transformation (AET) (Zhang et al., 2019). This autoencoder reconstructs the transformation **t** given the original image **x** and the transformed image **tx**. The transformation **t** is drawn from the affine and projective family of transformations. Another work extends the AET to an information theory perspective, resulting in autoencoding

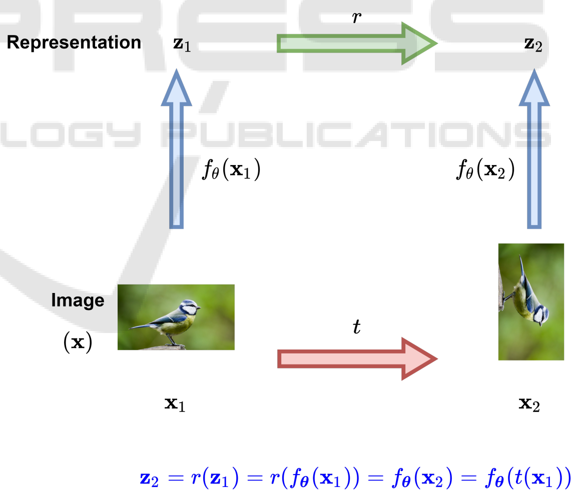


Figure 1: The illustration of transformation-equivariant. The representation $\mathbf{z}_2$ can be obtained in two ways. The first approach is to feed a transformed image $x_2 = t(\mathbf{x}_1)$ through a function f_θ, with t is the transformation in image space. Another approach is to transform representation $\mathbf{z}_1$ through function r.

variational transformation (AVT) (Qi et al., 2019).

AVT adopts the notion of steerability and extends it to an information theory perspective. The steerability guarantees that we can transform the representation $\hat{\mathbf{z}}$ the same way we transform the image $\mathbf{x}$ without requiring $\mathbf{x}$ (Cohen and Welling, 2017). From an information theory point of view, we can view the steerability property as the MI between the representation of the original image $\hat{\mathbf{z}}$, the representation of the transformed image $\mathbf{z}$, and the transformation $\mathbf{t}$ (Qi, 2019). The goal of AVT is to maximize MI $I(\mathbf{z};\hat{\mathbf{z}},\mathbf{t})$. However, computing the closed-form of MI is often intractable, especially for high-dimensional data. Instead, AVT estimates the MI by decomposing the MI into two terms and maximizing the lower bound of one of them.

$$I(\mathbf{z};\hat{\mathbf{z}},\mathbf{t}) = I(\hat{\mathbf{z}};\mathbf{z}) + I(\mathbf{z};\mathbf{t}|\hat{\mathbf{z}})$$

AVT maximizes $I(\mathbf{z};\mathbf{t}|\hat{\mathbf{z}})$ by using Barber-Agakov estimation (Agakov, 2004). Results show that AVT achieves promising results on image classification tasks. However, It remains unclear whether term $I(\hat{\mathbf{z}};\mathbf{z})$ gives the same performance as AVT.

This research investigates $I(\hat{\mathbf{z}};\mathbf{z})$ as an alternative objective to train an unsupervised transformation-equivariant representation. Later on, we call the alternative models as variational transformation-equivariant (VTE). Our finding shows that maximizing $I(\hat{\mathbf{z}};\mathbf{z})$ without prior information is not feasible. Instead, we train VTE into two stages. In the first stage, we build a self-supervised learning model that maximizes MI between transformation $\mathbf{t}$ and the representation of the transformed image $\mathbf{z}$. In the second stage, we build VTE model and incorporate the previous self-supervised learning model to maximize $I(\hat{\mathbf{z}};\mathbf{z})$. Moreover, we apply Barber-Agakov (Agakov, 2004) and InfoNCE (van den Oord et al., 2018) MI estimation to maximize the MI, resulting in three different models. Finally, we evaluate the proposed models on image classification tasks. We conduct the classification on two datasets: CIFAR-10 (Krizhevsky et al., 2009) and STL-10 (Coates et al., 2011) datasets. The main contributions of this paper are as follow:

- We design a mechanism to train VTE, an alternative version of AVT.
- We build a self-supervised model to help training the VTE.
- We utilize Barber-Agakov and InfoNCE MI estimation to train our VTE.
- We apply the proposed models as feature extractor on image classification tasks.

We make our code available on Github.[1]

The rest of this paper is organized as follows. In Section 2, we cover works related to self-supervised learning, transformation-equivariant representation, and MI estimation methods. Section 3 gives a detailed explanation of how to train VTE. In Section 4, we discuss the settings and results of experiments. Finally, Section 5 gives the conclusion of this research.

2 RELATED WORKS

2.1 Transformation-Equivariant Representation

The capsule net initiated the idea of general transformation-equivariant representation (Hinton et al., 2011; Wang and Liu, 2018). Capsule net takes groups of neurons, which are responsible to capture specific information of the image. Each capsule is designed to be equivariant to specific transformations. However, there was no rigorous algorithm to control and guarantee the equivariance property for each capsule.

Several works attempted to build a special convolution network that captures more types of transformation operations. Group equivariant convolution network (Cohen and Welling, 2016) introduces $p4$ and $p4m$ groups to handle the equivariance for rotation, translation, and reflection. This network produces a more complex visual structure which is beneficial for the classification layer. Steerable CNN (Cohen and Welling, 2017) utilizes a filter bank that is responsible to capture the equivariant property. Another work proposed dynamic routing for capsule net (Lenssen et al., 2018) with concepts of equivariant pose vectors and invariant agreements.

2.2 Self-supervised Learning

In general, self-supervised learning attempts to generate a surrogate label to enable supervised learning. The surrogate label is synthesized from the part of the data. (Noroozi and Favaro, 2016) divided the image into several patches and permute the order of the patches. Subsequently, they build a context-free network to predict the index of the permutation. (Doersch et al., 2015) also treat the images as grids/patches. The idea is to predict the relative position of a random grid given another grid that acts as the context. (Noroozi et al., 2017) attempt to count the number of features of the transformed images. In this research, they restrict the transformation operation into scaling and tiling. (Gidaris et al., 2018) apply discrete rotation on the image and ask the model to predict the type of rotation. (Dosovitskiy et al., 2016) apply several transformations to the image. Subsequently, they assign the same surrogate label to the original

[1]https://github.com/MarshalArijona/VTE

image and the transformed images. Finally, a classifier is asked to predict the surrogate class. Most self-supervised methods require some transformation operations to train the model. However, the transformations are restricted to the pseudo label on which the model is trying to solve. Furthermore, there is no explicit algorithm that guarantees the model preserves the equivariant property.

2.3 Mutual Information Estimation

Given two random variables $\mathbf{x}$ and $\mathbf{y}$, the mutual information I quantifies the amount of information (in nat or bit) obtained about $\mathbf{x}$ after observing y or vice versa. Mathematically, MI is defined by:

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &= \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right] \\ &= \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\frac{p(\mathbf{x}|\mathbf{y})}{p(\mathbf{x})}\right] = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}\right] \end{aligned} \tag{1}$$

Computing MI is often intractable. Specifically, the source of intractability is due to the unavailability to the conditional distribution $p(\mathbf{x}|\mathbf{y})$. Furthermore, it is often that we only have samples from the joint distribution. Therefore, sample-based methods are developed to estimate MI. In general, there are two approaches to estimating MI: taking the variational lower bound of MI and taking the variational upper bound of MI.

Barber-Agakov MI estimation (Agakov, 2004) approximates the $p(\mathbf{x}|\mathbf{y})$ with a variational distribution $q(\mathbf{x}|\mathbf{y})$ to obtain the lower bound of MI. This approach is relatively easy to compute but gives a high bias. Another approaches transformed $q(\mathbf{x}|\mathbf{y})$ into an unnormalized form by introducing a partition function $\mathbf{Z}$ (Nguyen et al., 2010; Donsker and Varadhan, 1975). However, those approaches require the samples from the marginal distribution, which we want to avoid. InfoNCE (van den Oord et al., 2018) obtains the lower bound by incorporating a contrastive loss approach. The advantage of this method is the low variance of the estimation. This method requires the positive pair and the negative pairs of samples. However, the method is heavily dependent on the number of samples.

We can adopt the Barber-Agakov method to obtain the upper bound of MI. (Alemi et al., 2017) approximate the marginal distribution $p(\mathbf{x})$ with a variational distribution $q(\mathbf{x})$. However, approximating the marginal distribution without prior information is challenging, especially on high dimensional data. The Leave-one-out (Poole et al., 2019) method attempts to approximate $p(\mathbf{x})$ by taking the sum of $p(\mathbf{x}^i|\mathbf{y}^j)$ over the samples $(\mathbf{x}^i,\mathbf{y}^j)$, except $\mathbf{y}^i$. $\mathbf{y}^i$ is the corresponding pair of $\mathbf{x}^i$. Another approach incorporates a contrastive method to derive the variational upper bound of MI (Cheng et al., 2020).

In this research, we perform MI maximization with the help of Barber-Agakov and InfoNCE estimation. Both methods only require samples from the joint distribution, which fit the problem we aim to solve.

3 VARIATIONAL TRANSFORMATION-EQUIVARIANT

3.1 The Generalization of Transformation-Equivariant Representation

Let $\hat{\mathbf{z}} \in \mathcal{Z}$ be the representation of image $\mathbf{x} \in \mathcal{X}$ and $t : \mathcal{X} \times \mathcal{T} \to \mathcal{X}$ be a transformation that involves image $\mathbf{x}$ and a transformation operation $\mathbf{t}$. We have $\mathbf{z} \in \mathcal{Z}$ be the representation of a transformed image $t(\mathbf{x},\mathbf{t}) = \mathbf{tx}$. We can view the transformation $\mathbf{t}$ as a matrix. Representations $\hat{\mathbf{z}}, \mathbf{z}$, and transformation $\mathbf{t}$ satisfy the transformation-equivariant property if there exists a function $r : \mathcal{Z} \times \mathcal{T} \to \mathcal{Z}$, such that

$$\mathbf{z} = r(\mathbf{z},\mathbf{t}) = \tau(\mathbf{t})(\hat{\mathbf{z}}) \tag{2}$$

where $\tau(\mathbf{t})$ denotes a function that enable $\mathbf{t}$ to be applied in $\mathcal{Z}$. Note that the representation $\mathbf{z}$ is completely determined by $\mathbf{t}$ and $\hat{\mathbf{z}}$ (no need access to $\mathbf{x}$). This notion is called steerability (Cohen and Welling, 2017; Qi, 2019), which enables computing $\mathbf{z}$ by applying an independent transformation $\tau(\mathbf{t})$ to $\hat{\mathbf{z}}$.

From the information theory perspective, we can model the notion of steerability as the MI between $\mathbf{z}$, and $(\hat{\mathbf{z}},\mathbf{t})$ (Qi, 2019). Here the MI is parameterized by θ. Therefore, the goal is to find θ that maximizes $I_\theta(\mathbf{z};\hat{\mathbf{z}},\mathbf{t})$.

$$\theta^* = \max_\theta I_\theta(\mathbf{z};\hat{\mathbf{z}},\mathbf{t}) \tag{3}$$

The form $I_\theta(\mathbf{z};\hat{\mathbf{z}},\mathbf{t})$ is not feasible to compute. To enable the training, we decompose $I_\theta(\mathbf{z};\hat{\mathbf{z}},\mathbf{t})$ into:

$$\begin{aligned} I_\theta(\mathbf{z};\hat{\mathbf{z}},\mathbf{t}) &= \mathop{\mathbb{E}}_{p_\theta(\mathbf{z},\hat{\mathbf{z}},\mathbf{t})}\left[\frac{p_\theta(\mathbf{z},\hat{\mathbf{z}},\mathbf{t})}{p_\theta(\mathbf{z})\,p_\theta(\hat{\mathbf{z}},\mathbf{t})}\right] \\ &= \mathop{\mathbb{E}}_{p_\theta(\mathbf{z},\hat{\mathbf{z}},\mathbf{t})}\left[\frac{p_\theta(\hat{\mathbf{z}})p_\theta(\mathbf{z}|\hat{\mathbf{z}})p_\theta(\mathbf{t}|\mathbf{z},\hat{\mathbf{z}})}{p_\theta(\mathbf{z})\,p_\theta(\mathbf{t}|\hat{\mathbf{z}})p_\theta(\hat{\mathbf{z}})}\right] \\ &= \mathop{\mathbb{E}}_{p_\theta(\mathbf{z},\hat{\mathbf{z}},\mathbf{t})}\left[\frac{p_\theta(\mathbf{z}|\hat{\mathbf{z}}}{p_\theta(\mathbf{z})}\right] + \mathop{\mathbb{E}}_{p_\theta(\mathbf{z},\hat{\mathbf{z}},\mathbf{t})}\left[\frac{p_\theta(\mathbf{t}|\mathbf{z},\hat{\mathbf{z}})}{p_\theta(\mathbf{t}|\hat{\mathbf{z}})}\right] \\ &= I_\theta(\mathbf{z};\hat{\mathbf{z}}) + I_\theta(\mathbf{z};\mathbf{t}|\hat{\mathbf{z}}) \end{aligned} \tag{4}$$

AVT aims to maximize $I_\theta(\mathbf{z};\mathbf{t}|\hat{\mathbf{z}})$ as the objective function. In this paper, we investigate the performance of an transformation-equivariant repersentation model by maximizing $I_\theta(\hat{\mathbf{z}};\mathbf{z})$. We name this model as variational transformation-equivariant (VTE). Both AVT and VTE encounter the intractable computation of MI. AVT needs to compute the intractable posterior $p_\theta(\mathbf{t}|\hat{\mathbf{z}},\mathbf{z})$, while VTE needs to compute intractable posterior $p_\theta(\mathbf{z}|\hat{\mathbf{z}})$. Therefore we need to estimate the MI. AVT utilizes Barber-Agakov estimation to maximize the MI. In this research, we maximize MI by using Barber-Agakov (Agakov, 2004) and InfoNCE MI estimation (van den Oord et al., 2018).

3.2 Transformation as Inductive Bias

Our preliminary experiments showed that maximizing MI $I_\theta(\hat{\mathbf{z}};\mathbf{z})$ without prior results in the model fails to learn. Specifically, the model will assign a trivial posterior probability given any pair of $(\hat{\mathbf{z}},\mathbf{z})$. Therefore, we need to explicitly involve the transformation $\mathbf{t}$ to train VTE.

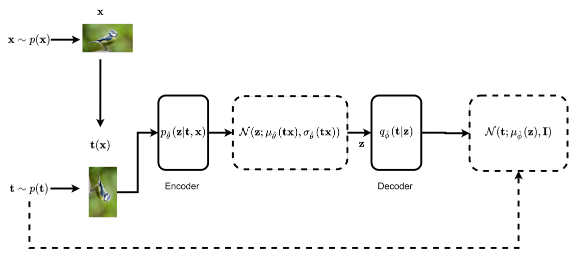
We propose sequential mechanism to train the VTE. The training comprises two phases of training. In the first phase, we model the distribution of $\mathbf{z}$. Recall that $\mathbf{z}$ is the representation of the transformed image. We involve the transformation $\mathbf{t}$ the train the model. Specifically, we build a self-supervised learning model that maximizes the MI $I_{\hat{\theta}}(\mathbf{z};\mathbf{t})$, parameterized by $\hat{\theta}$. We call this model as the predictive-transformation. Note that this objective function is similar to AVT ($I_\theta(\mathbf{z};\mathbf{t}|\hat{\mathbf{z}})$), except without $\hat{\mathbf{z}}$. Due to the absence of $\hat{\mathbf{z}}$, the obtained representation is not guaranteed to be equivariant anymore. We maximize the MI by using Barber-Agakov lower bound MI estimation. Barber-Agakov estimation introduces a variational distribution $q_{\check{\phi}}(\mathbf{t}|\mathbf{z})$, parameterized by $\check{\phi}$ to approximate $p_{\hat{\theta}}(\mathbf{t}|\mathbf{z})$.

$$\begin{aligned} I_{\hat{\theta}}(\mathbf{t};\mathbf{z}) &= H(\mathbf{t}) - H(\mathbf{t}|\mathbf{z}) \\ &= H(\mathbf{t}) + \mathbb{E}_{p_{\hat{\theta}}(\mathbf{t},\mathbf{z})} \log p_{\hat{\theta}}(\mathbf{t}|\mathbf{z}) \\ &= H(\mathbf{t}) + \mathbb{E}_{p_{\hat{\theta}}(\mathbf{t},\mathbf{z})} \log q_{\check{\phi}}(\mathbf{t}|\mathbf{z}) \\ &\quad + \mathbb{E}_{p(\mathbf{z})} \mathrm{KL}(p_{\hat{\theta}}(\mathbf{t}|\mathbf{z}) \,||\, q_{\check{\phi}}(\mathbf{t}|\mathbf{z})) \\ &\geq H(\mathbf{t}) + \mathbb{E}_{p_{\hat{\theta}}(\mathbf{t},\mathbf{z})} \log q_{\check{\phi}}(\mathbf{t}|\mathbf{z}) \end{aligned} \tag{5}$$

$H(.)$ and $\mathrm{KL}(.||.)$ denote the entropy and the Kullback-Leibler divergence between two distributions, respectively. Since $H(\mathbf{t})$ does not depend on $\hat{\theta}$ and $\check{\phi}$, we simply maximize

$$\max_{\hat{\theta},\check{\phi}} \mathbb{E}_{p_{\hat{\theta}(\mathbf{t},\mathbf{z})}} \log q_{\check{\phi}}(\mathbf{t}|\mathbf{z}) \tag{6}$$

We implement the predictive-transformation in the framework of autoencoder. Figure 2 shows the architecture of predictive-transformation.



$\mathbf{x}$
$\mathbf{x} \sim p(\mathbf{x})$
$\mathbf{t}(\mathbf{x})$
$\mathbf{t} \sim p(\mathbf{t})$
$p_{\hat{\theta}}(\mathbf{z}|\mathbf{t},\mathbf{x})$
Encoder
$\mathcal{N}(\mathbf{z};\mu_{\hat{\theta}}(\mathbf{tx}),\sigma_{\hat{\theta}}(\mathbf{tx}))$
$\mathbf{z}$
$q_{\check{\phi}}(\mathbf{t}|\mathbf{z})$
Decoder
$\mathcal{N}(\mathbf{t};\mu_{\check{\phi}}(\mathbf{z}),\mathbf{I})$

Figure 2: The architecture of predictive-transformation model. Transformed image is fed through the encoder $p_{\hat{\theta}}$. The output of encoder is the mean $\mu_{\hat{\theta}}$ and the standard-deviation $\sigma_{\hat{\theta}}$. The representation $\mathbf{z}$ is sampled and fed through the decoder $q_{\check{\phi}}$. The output of the decoder is the mean $\mu_{\check{\phi}}$, which corresponds to the transformation $\mathbf{t}$.

The encoder $E_{\hat{\theta}}$ represents $p_{\hat{\theta}}(\mathbf{z}|\mathbf{tx})$. We assume that $p_{\hat{\theta}}(\mathbf{z}|\mathbf{tx})$ is following a factored multivariate Gaussian distribution $\mathcal{N}(\mathbf{z};\mu_{\hat{\theta}},\sigma_{\hat{\theta}})$. Therefore, the output of the encoder is the mean $\mu_{\hat{\theta}}$ and the standard-deviation $\sigma_{\hat{\theta}}$. The decoder $D_{\check{\phi}}$ represents the variational distribution $q_{\check{\phi}}(\mathbf{t}|\mathbf{z})$. We also assume that $q_{\check{\phi}}(\mathbf{t}|\mathbf{z})$ is a factored multivariate Gaussian distribution, with a constraint that the standard-deviation sets to one: $\mathcal{N}(\mathbf{t};\mu_{\check{\phi}},\mathbf{I})$. $\mathbf{I}$ denotes the identity matrix. Thus, the output of the decoder is the mean $\mu_{\check{\phi}}$.

For the second phase of training, we capture the distribution of $\hat{\mathbf{z}}$ by training the VTE network. The goal is to maximize the MI $I_\theta(\hat{\mathbf{z}};\mathbf{z})$. We estimate $I_\theta(\hat{\mathbf{z}};\mathbf{z})$ by using Barber-Agakov and InfoNCE estimation. The optimization can be done through a gradient-based method such as stochastic gradient descent.

3.3 Barber-Agakov Lower Bound Estimation

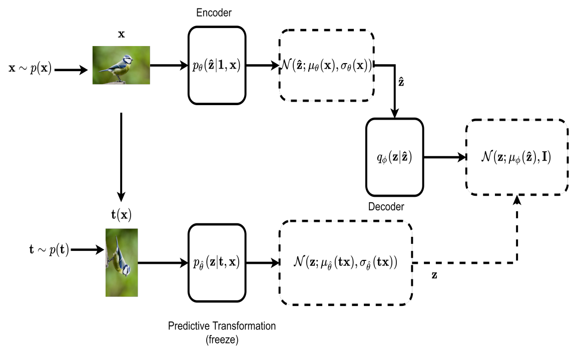
Following Equation 5, we derive the lower bound of $I_\theta(\hat{\mathbf{z}};\mathbf{z})$ by introducing a variational distribution $q_\phi(\mathbf{z}|\hat{\mathbf{z}})$, parameterized by ϕ to approximate $p_\theta(\mathbf{z}|\hat{\mathbf{z}})$. Furthermore, we incorporate the predictive-transformation from the first phase of training to obtain the representation $\mathbf{z}$. We call this model as VTEBArber-Agakov (VTEBA).

$$\begin{aligned} I_{\theta,\hat{\theta}}(\hat{\mathbf{z}};\mathbf{z}) &= H(\mathbf{z}) - H(\mathbf{z}|\hat{\mathbf{z}}) \\ &= H(\mathbf{z}) + \mathbb{E}_{p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log p_{\theta,\hat{\theta}}(\mathbf{z}|\hat{\mathbf{z}}) \\ &= H(\mathbf{z}) + \mathbb{E}_{p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log q_\phi(\mathbf{z}|\hat{\mathbf{z}}) \\ &\quad + \mathbb{E}_{p(\hat{\mathbf{z}})} \mathrm{KL}(p_{\theta,\hat{\theta}}(\mathbf{z}|\hat{\mathbf{z}}) \,||\, q_\phi(\mathbf{z}|\hat{\mathbf{z}})) \\ &\geq H(\mathbf{z}) + \mathbb{E}_{p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log q_\phi(\mathbf{z}|\hat{\mathbf{z}}) \end{aligned} \tag{7}$$

In this phase, we do not optimize $\hat{\theta}$ anymore. Therefore, the objective function of VTEBA is maximizing

$$\max_{\theta,\phi} \mathbb{E}_{p_{\hat{\theta},\theta(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})}} \log q_{\phi}(\mathbf{z}|\hat{\mathbf{z}}) \tag{8}$$

Just like predictive-transformation, we treat VTEBA as an autoencoder. Figure 3 shows the architecture of VTEBA.



Encoder
$\mathbf{x}$
$\mathbf{x} \sim p(\mathbf{x})$
$p_\theta(\hat{\mathbf{z}}|\mathbf{1},\mathbf{x})$
$\mathcal{N}(\hat{\mathbf{z}};\mu_\theta(\mathbf{x}),\sigma_\theta(\mathbf{x}))$
$\hat{\mathbf{z}}$
$q_\phi(\mathbf{z}|\hat{\mathbf{z}})$
$\mathcal{N}(\mathbf{z};\mu_\phi(\hat{\mathbf{z}}),\mathbf{I})$
Decoder
$\mathbf{t}(\mathbf{x})$
$\mathbf{t} \sim p(\mathbf{t})$
$p_{\hat{\theta}}(\mathbf{z}|\mathbf{t},\mathbf{x})$
$\mathcal{N}(\mathbf{z};\mu_{\hat{\theta}}(\mathbf{tx}),\sigma_{\hat{\theta}}(\mathbf{tx}))$
$\mathbf{z}$
Predictive Transformation
(freeze)

Figure 3: The architecture of VTEBA. The transformed image is fed through the predictive-transformation's encoder $p_{\hat{\theta}}$, while the original images is fed through the encoder p_{θ}. The output of the encoder is the mean μ_{θ} and the standard-deviation σ_{θ}. The representation $\hat{\mathbf{z}}$ is sampled and fed through the decoder q_{ϕ}. The output of the decoder is the mean μ_{ϕ}, which corresponds to the representation of transformed image $\mathbf{z}$, obtained from $p_{\hat{\theta}}$.

The encoder E_{θ} represents $p_{\theta}(\hat{\mathbf{z}}|\mathbf{z})$. We assume that $p_{\theta}(\hat{\mathbf{z}}|\mathbf{z})$ is following a factored multivariate Gaussian distribution $\mathcal{N}(\hat{\mathbf{z}};\mu_{\theta},\sigma_{\theta})$. Thus, the output of E_{θ} is the mean μ_{θ} and the standard-deviation σ_{θ}. The encoder $E_{\hat{\theta}}$ of the predictive-transformation is responsible to infer $\mathbf{z}$. Note that we freeze $\hat{\theta}$ while training VTE. We use the samples $\mathbf{z}$ to compute $q_{\phi}(\mathbf{z}|\hat{\mathbf{z}})$. The decoder D_{ϕ} represents $q_{\phi}(\mathbf{z}|\hat{\mathbf{z}})$ and takes $\hat{\mathbf{z}}$ as the input. We assume that $q_{\phi}(\mathbf{z}|\hat{\mathbf{z}})$ is following $\mathcal{N}(\mathbf{z};\mu_{\phi},\mathbf{I})$ to reduce the complexity of the model. Therefore, the output of D_{ϕ} is the mean μ_{ϕ}.

3.4 InfoNCE Lower Bound Estimation

The second estimation method is the InfoNCE. Note that we still utilize the predictive-transformation to infer $\mathbf{z}$. This method adopts the notion of contrastive learning to estimate MI. Recall that from Equation 1, we have the MI as the expectation of a density ratio between conditional distribution and the marginal distribution. Given $I_{\theta,\hat{\theta}}(\hat{\mathbf{z}};\mathbf{z})$, then:

$$I_{\theta,\hat{\theta}}(\hat{\mathbf{z}};\mathbf{z}) = \mathbb{E}_{p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log \frac{p_{\theta,\hat{\theta}}(\hat{\mathbf{z}}|\mathbf{z})}{p(\hat{\mathbf{z}})} \tag{9}$$

Furthermore, suppose that we have a batch of samples $\{(\hat{\mathbf{z}}^i,\mathbf{z}^i)\}_{i=1}^N$. For a particular pair $(\hat{\mathbf{z}}^i,\mathbf{z}^i)$, we can model the density ratio $\frac{p_{\theta,\hat{\theta}}(\hat{\mathbf{z}}=\hat{\mathbf{z}}^i|\mathbf{z}=\mathbf{z}^i)}{p(\hat{\mathbf{z}}=\hat{\mathbf{z}}^i)}$ as:

$$\exp(f(\hat{\mathbf{z}}^i,\mathbf{z}^i)) \propto \frac{p_{\theta,\hat{\theta}}(\hat{\mathbf{z}}=\hat{\mathbf{z}}^i|\mathbf{z}=\mathbf{z}^i)}{p(\hat{\mathbf{z}}=\hat{\mathbf{z}}^i)} \tag{10}$$

with f is an estimator function that takes $(\hat{\mathbf{z}}^i,\mathbf{z}^i)$ as the input. Note that this approximation form can be unnormalized, which means that the integral of the density ratio does not have to be 1. Therefore, for each pair of $(\hat{\mathbf{z}}^i,\mathbf{z}^i)$, we normalize it by a partition function that takes the sum of $(\hat{\mathbf{z}}^j,\mathbf{z}^i)$, $1 \le j \le N \land i \neq j$. We call $(\hat{\mathbf{z}}^i,\mathbf{z}^i)$ a positive pair, while $(\hat{\mathbf{z}}^j,\mathbf{z}^i)$ a negative pairs. Mathematically, we have

$$I_{\theta,\hat{\theta}}(\hat{\mathbf{z}},\mathbf{z}) \ge \mathop{\mathbb{E}}_{\prod_j p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log \frac{\exp(f(\hat{\mathbf{z}}^i,\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(f(\hat{\mathbf{z}}^j,\mathbf{z}^i))} \tag{11}$$

$$\ge \mathop{\mathbb{E}}_{\prod_j p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log \frac{\exp(g(\hat{\mathbf{z}}^i)^{\mathrm{T}}h(\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(g(\hat{\mathbf{z}}^j)^{\mathrm{T}}h(\mathbf{z}^i))} \tag{12}$$

It is known that neural network is an universal approximation for any function (Heaton, 2018). Therefore, we implement the estimator function f as neural network. We call this model as VTEInfoNCE concatenated version. Furthermore, we can decompose the function f into two different functions g and h, each takes $\hat{\mathbf{z}}$ and $\mathbf{z}$ as input separately. We combine the output by performing vector multiplication. We call this model as VTEInfoNCE separated version. Let $\hat{\phi}$ be the parameter of estimator function f. The objective function of VTEInfoNCE concatenated version is maximizing

$$\max_{\theta,\hat{\phi}} \mathop{\mathbb{E}}_{\prod_j p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log \frac{\exp(f(\hat{\mathbf{z}}^i,\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(f(\hat{\mathbf{z}}^j,\mathbf{z}^i))} \tag{13}$$

Subsequently, let $\tilde{\phi},\phi'$ be the parameter of the estimator function g and h, respectively. The VTEInfoNCE separated version aims to maximize

$$\max_{\theta,\tilde{\phi},\phi'} \mathop{\mathbb{E}}_{\prod_j p_{\theta,\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} \log \frac{\exp(g(\hat{\mathbf{z}}^i)^{\mathrm{T}}h(\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(g(\hat{\mathbf{z}}^j)^{\mathrm{T}}h(\mathbf{z}^i))} \tag{14}$$

Note that we do not optimize the parameter $\hat{\theta}$. In the next subsection, we provide the algorithm to train the predictive-transformation and the VTE models. Figure 4 shows the architecture of VTEInfoNCE separated.

3.5 Algorithm

3.5.1 Training Predictive-transformation

Suppose that we have a batch consists of N samples $\mathbf{X} = \{\mathbf{x}^i\}_{i=1}^N$. For each sample, we draw a

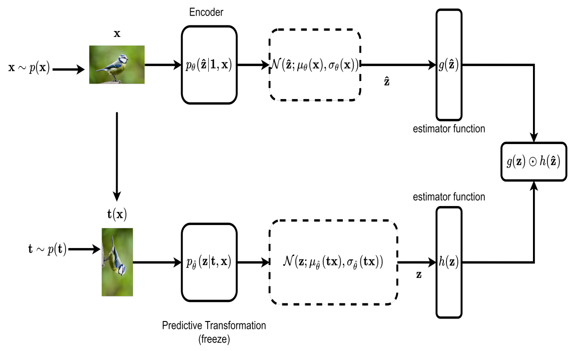

Encoder
$\mathbf{x}$
$\mathbf{x} \sim p(\mathbf{x})$
$p_\theta(\hat{\mathbf{z}}|\mathbf{1},\mathbf{x})$
$\mathcal{N}(\hat{\mathbf{z}};\mu_\theta(\mathbf{x}),\sigma_\theta(\mathbf{x}))$
$\hat{\mathbf{z}}$
$g(\hat{\mathbf{z}})$
estimator function
$g(\mathbf{z}) \odot h(\hat{\mathbf{z}})$
estimator function
$\mathbf{t}(\mathbf{x})$
$\mathbf{t} \sim p(\mathbf{t})$
$p_{\hat\theta}(\mathbf{z}|\mathbf{t},\mathbf{x})$
$\mathcal{N}(\mathbf{z};\mu_{\hat\theta}(\mathbf{tx}),\sigma_{\hat\theta}(\mathbf{tx}))$
$\mathbf{z}$
$h(\mathbf{z})$
Predictive Transformation
(freeze)

Figure 4: The architecture of VTEInfoNCE. The decoder is replaced with estimator functions g and h, parameterized by $\tilde{\phi}$ and ϕ', respectively.

transformation $\mathbf{t}^i$ from $p(\mathbf{t})$. We infer $p_{\hat\theta}(\mathbf{z}^i|\mathbf{t}^i\mathbf{x}^i)$ by feeding $\mathbf{t}^i\mathbf{x}^i$ through encoder $E_{\hat\theta}$. Note that the output of $E_{\hat\theta}$ is the parameters of a probability distribution. Therefore, we need to sample from the distribution to get $\mathbf{z}^i$ instance. However, performing ordinary sampling leads the model to fail to compute the gradient of the objective function w.r.t. the parameter $\hat\theta$ (encoder parameter). This condition is undesirable if we perform the optimization through gradient-based method. Therefore, we apply the reparameterization trick to solve the problem. This trick is renowned used by the variational autoencoder (Kingma and Welling, 2013). Given the mean $\mu_{\hat\theta}(\mathbf{t}^i\mathbf{x}^i)$ and standard-deviation $\sigma_{\hat\theta}(\mathbf{t}^i\mathbf{x}^i)$, we can write reparameterization trick as:

$$\mathbf{z}^i = \mu_{\hat\theta}(\mathbf{t}^i\mathbf{x}^i) + \sigma_{\hat\theta}(\mathbf{t}^i\mathbf{x}^i) \odot \varepsilon^i \tag{15}$$

with $\odot$ denotes the pointwise multiplication. Recall that μ and σ are obtained from the encoder $E_{\hat\theta}$. Furthermore, ε^i refers to a noise sampled from $\mathcal{N}(\varepsilon;\mathbf{0},\mathbf{I})$. Decoder $D_{\check\phi}$ takes $\mathbf{z}$ to output $q_{\check\phi}(\mathbf{t}|\mathbf{z})$. Since we only have the samples $\mathbf{z}$, we translate the expectation into an unbiased Monte Carlo estimation:

$$\min_{\hat\theta,\check\phi} \frac{1}{N}\sum_{i=1}^{N} -\log \mathcal{N}(\mathbf{t}^i;\mu_{\check\phi}(\mathbf{z}^i),\mathbf{I}) \tag{16}$$

By the property of the monotonic function (log), we minimize the negative of the objective function. Furthermore, the decoder $D_{\check\phi}$ only outputs $\mu_{\check\phi}$ (by the assumption of $q_{\check\phi}(\mathbf{t}|\mathbf{z})$).

3.5.2 Training Variational Transformation-Equivariant

We follow the same settings as the predictive-transformation. Given samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, we draw transformation $\mathbf{t}^i$ from $p(\mathbf{t})$ for each $\mathbf{x}_i$. Subsequently, we use encoder E_θ to infer $p_\theta(\hat{\mathbf{z}}^i|\mathbf{x}^i)$. and perform the reparameterization trick to generate $\hat{\mathbf{z}}$.

$$\hat{\mathbf{z}}^i = \mu_\theta(\mathbf{x}^i) + \sigma_\theta(\mathbf{x}^i) \odot \hat\varepsilon^i \tag{17}$$

The noise $\hat\varepsilon$ is drawn from a standard Gaussian distribution $\mathcal{N}(\hat\varepsilon;\mathbf{0},\mathbf{I})$.

We use the encoder $E_{\hat\theta}$ to infer $p_{\hat\theta}(\mathbf{z}^i|\mathbf{t}^i)$. Note that $E_{\hat\theta}$ is the encoder of predictive-transformation. We freeze the parameter $\hat\theta$ since we do not optimize $\hat\theta$. For the Barber-Agakov MI-estimation, we minimize

$$\min_{\theta,\phi} \frac{1}{N}\sum_{i=1}^{N} -\log \mathcal{N}(\mathbf{z}^i;\mu_\phi(\hat{\mathbf{z}}^i),\sigma(\hat{\mathbf{z}}^i)) \tag{18}$$

Here we translate the expectation into an unbiased Monte Carlo estimation method since we only have the samples of $\hat{\mathbf{z}}$ and $\mathbf{z}$. We take the benefit of the monotonic function's property by translating the objective function into a minimization problem. Decoder D_ϕ represents $q_\phi(\mathbf{z}|\hat{\mathbf{z}})$. By the assumption of $q_\phi(\mathbf{z}|\hat{\mathbf{z}})$, D_ϕ only outputs the mean μ_ϕ.

For the InfoNCE MI-estimation, we have two versions: the concated version and the separated version. The objective function of the concatenated version is as follows:

$$\min_{\theta,\hat\phi} \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp(f(\hat{\mathbf{z}}^i,\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(f(\hat{\mathbf{z}}^j,\mathbf{z}^i))} \tag{19}$$

$\hat\phi$ denotes the parameter of estimator function f. Subsequently, the objective function of the separated InfoNCE is as follows:

$$\min_{\theta,\tilde\phi,\phi'} \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp(g(\hat{\mathbf{z}}^i)^{\mathrm{T}}h(\mathbf{z}^i))}{\sum_{\hat{\mathbf{z}}^j}\exp(g(\hat{\mathbf{z}}^j)^{\mathrm{T}}h(\mathbf{z}^i))} \tag{20}$$

$\tilde\phi,\phi'$ denote the parameters of g and h, respectively. Both VTEBA and VTEInfoNCE can utilize a gradient-based method to optimize the objective function.

The difference between AVT and VTE lies in their output. The decoder of AVT estimates the probability distribution $p_\theta(\mathbf{t}|\hat{\mathbf{z}},\mathbf{z})$ through Barber-Agakov MI estimation. On the other hand, the decoder of VTEBA and the estimator function of VTEInfoNCE aim to estimate $p_\theta(\hat{\mathbf{z}}|\mathbf{z})$ through Barber-Agakov estimation and noise contrastive loss, respectively. As a result, AVT takes $\mathbf{z}$ and $\hat{\mathbf{z}}$ as the inputs, while VTEBA only takes one of either $\hat{\mathbf{z}}$ or $\mathbf{z}$. Furthermore, AVT requires one stage of training while VTE requires two stages of training. The latter is because VTE needs the predictive-transformation model as the inductive bias,

Table 1: The comparison of average error rate by different of each model with the various number of training data on CIFAR-10 image classification using MLP.

Model	50K	5K	0.5K
AVT	0.147 ± 0.0001	0.355 ± 0.0018	0.797 ± 0.0021
predictive-transformation	0.142 ± 0.0006	0.383 ± 0.0028	0.655 ± 0.0036
VTEBA	$\mathbf{0.140 \pm 0.0000}$	0.354 ± 0.0001	0.508 ± 0.0006
VTEInfoNCE (1)	0.148 ± 0.0005	$\mathbf{0.310 \pm 0.0015}$	0.550 ± 0.0015
VTEInfoNCE (2)	0.152 ± 0.0000	0.362 ± 0.0016	$\mathbf{0.476 \pm 0.0006}$

which requires separate training. From the optimization side, VTE and AVT depend on the gradient-based method and reparameterization trick to optimize their parameters.

4 EXPERIMENTS

For the experiments, we train predictive-transformation, VTEBA, VTEInfoNCE separated version, and VTEInfoNCE concatenated version. We also reproduce AVT to give a fair comparison. We then evaluate each of model on image classification tasks. We utilize multi-layer perceptron (MLP), K-nearest neighbor (K-NN), and multinomial logistic regression as the classifiers. In this experiment, we use CIFAR-10 (Krizhevsky et al., 2009) and STL-10 (Coates et al., 2011) datasets.

4.1 CIFAR-10 Experiment

Architecture. We follow the original architecture of AVT for each model. Specifically, we use Network-In-Network architecture (Lin et al., 2013) for the convolution blocks (encoder). We represent the distribution of $\mathbf{z}$ and $\hat{\mathbf{z}}$ as a MLP of size 1024. The first 512 neurons represent the mean and the rest represent the log variance. The idea of replacing the standard-deviation with the log variance is to preserve numerical stability. We can derive the standard-deviation by performing an exponentiation trick. We implement the Decoder $D_{\check{\phi}}, D_{\phi}$, and the estimator function f, g and h as MLP with three layers.

Implementation Details. All models are optimized by adaptive moment (Adam) (Kingma and Ba, 2015) with a learning rate of $1e-4$. We train the models for 200 epochs. For each iteration, we chunk the data into several mini-batches, each with size 256. We utilize 1 GPU Tesla V-100 as the source of computation. Furthermore, our models follow the same settings as AVT (Qi et al., 2019) for the type of transformation. For each image, we apply the projective transformation consists of random translation along horizontal line and vertical line by $[-0.125, 0.125]$ of the width and the height of the image, random scaling with the ratio $[0.8, 1.2]$, and the random rotation with an angle from $\{0°, 90°, 180°, 270°\}$.

Evaluation. We perform image classifications to evaluate the models. First, we feed the feature extracted by the encoder of each model through an MLP-based classifier. The MLP consists of three fully connected layers. The first two layers share the exact size of 2048 neurons, while the last layer has a size of 10. We train the classifier on the various number of training data. Specifically, we train the MLP on 50000, 5000, and 500 training data, respectively. We then test the MLP on 10000 images. Since the encoder is a probabilistic model, we perform the classification 5 times for each image and take the average of the error rate. This approach is a bit different with AVT since they only compute the error once.

Table 1 shows the classification results using MLP on CIFAR-10 dataset. VTEInfoNCE (1) and VTEInfoNCE (2) refer to VTEInfoNCE separated version and VTEInfoNCE concatenated version, respectively. The results show that VTEBA outperforms the other models on 50000 data with a 0.14 average error rate while VTEInfoNCE (2) gives the worst result. On 5000 data, VTEInfoNCE (1) outperforms the others with a 0.31 ± 0.0015 average error rate, while VTEInfoNCE (2) gives the worst result with a 0.355 ± 0.00018 average error rate. Finally, VTEInfoNCE (2) yields the best result with a 0.476 ± 0.0006 average error rate on 500 data, while AVT yields the worst result with a 0.797 ± 0.0021 average error rate. In this experiment, the best model is different for each number of data involved during the training. In general, the proposed models give more satisfying results compared to the baseline model.

Subsequently, we perform image classification task by using K-NN. In this experiment, we choose $K = 5$. All neighbors have an equal impact on the classification result. Table 2 shows the classification results using K-NN on CIFAR-10 dataset. From the table, VTEBA consistently outperforms other models for every number of dataset. Furthermore, AVT

Table 2: The comparison of average error rate by different of each model with the various number of training data on CIFAR-10 image classification using K-NN.

Model	50K	5K	0.5K
AVT	0.693 ± 0.050	0.743 ± 0.006	0.803 ± 0.009
predictive-transformation	0.527 ± 0.006	0.596 ± 0.002	0.69 ± 0.011
VTEBA	**0.396 ± 0.003**	**0.488 ± 0.003**	**0.578 ± 0.008**
VTEInfoNCE (1)	0.429 ± 0.030	0.508 ± 0.004	0.583 ± 0.004
VTEInfoNCE (2)	0.448 ± 0.003	0.519 ± 0.001	0.596 ± 0.006

Table 3: The comparison of average error rate by different of each model with the various number of training data on CIFAR-10 image classification using multinomial logistic regression.

Model	50K	5K	0.5K
AVT	0.622 ± 0.0023	0.759 ± 0.0023	0.840 ± 0.0029
predictive-transformation	0.539 ± 0.0025	0.698 ± 0.0018	0.793 ± 0.0046
VTEBA	**0.391 ± 0.0005**	**0.457 ± 0.0003**	**0.603 ± 0.0003**
VTEInfoNCE (1)	0.439 ± 0.0010	0.550 ± 0.0014	0.681 ± 0.0012
VTEInfoNCE (2)	0.423 ± 0.0015	0.523 ± 0.0011	0.656 ± 0.0028

Table 4: Average error rate on STL-10 dataset using different classifiers.

Model	PLB	K-NN	Logistic regression
AVT	0.522 ± 0.0000	0.707 ± 0.004	0.632 ± 0.0000
predictive-transformation	0.492 ± 0.0009	0.711 ± 0.002	0.544 ± 0.0012
VTEBA	0.460 ± 0.0007	0.607 ± 0.004	0.54 ± 0.0002
VTEInfoNCE (1)	0.365 ± 0.0000	0.475 ± 0.005	0.477 ± 0.0000
VTEInfoNCE (2)	**0.363 ± 0.0005**	**0.473 ± 0.003**	**0.462 ± 0.0003**

yields the highest average error rate for every number of dataset.

Finally, we evaluate the representation models using multinomial logistic regression. The classifier optimize the cross-entropy loss for 100 iterations using stochastic average gradient descent (SAG) (Schmidt et al., 2017). Furthermore, we involve l^2-norm to regularize the weights of classifier. Table 3 shows the classification results using multinomial logistic regression on CIFAR-10 dataset. We have VTEBA consistently outperforms other models for every number of dataset. Moreover, AVT also becomes the model with the highest average error rate for every number of dataset.

We argue that two main factors cause the inconsistency results in Table 5. The first factor is the tendency of MLP to suffer from over-fitting, especially if there is only a small amount of data. The second factor is related to the characteristic of the representation generated by VTEBA and VTEInfoNCE. We argue that the contrastive loss leads VTEInfoNCE to generate representations that lie sparsely one each other. On the other hand, the objective function of VTEBA does not consider the relation of the representation with its negative samples. Thus, the generated representations are naturally more dense one each other. VTEInfoNCE has an advantage on the image classification task, which involves a small amount of data since the sparsity might reduce the overfitting to some degree. However, this model performs poorly if we have a large dataset since the MLP has to find a more complex hypothesis (large and sparse). VTEBA performs worse than VTEInfoNCE on a small dataset since the MLP can fit the data too well. On the contrary, VTEBA can give a better result on a large dataset since the representations are more concentrated in some regions of the representation space.

4.2 STL-10 Experiment

Architecture. For STL-10 experiment, we adopt the Alexnet architecture (Krizhevsky et al., 2017) for the convolution blocks (encoder). Each block comprises a convolution and ReLU layer, followed by a pooling layer. For the mean and the standard-deviation, we adopt the same settings as the previous experiment.

Implementation Details. In this experiment, we train the proposed models and baseline model on 100000 unlabeled images, each with size 96×96. We first resize each image to 32×32. Subsequently, we apply the same transformation as the previous experiment. All models are trained for 200 epochs with a 512 batch size. Same as previous research, the models use 1 GPU Tesla V-100. We use the adaptive moment to optimize the model with a learning rate 1e-4.

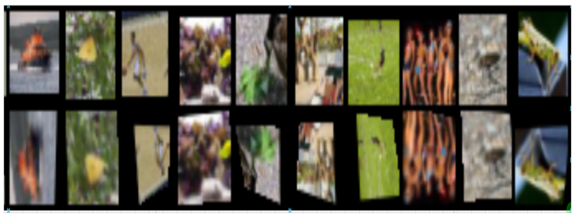


Figure 5: The first row shows sample images of the STL-10 dataset. The images in the second row are obtained by applying projective transformation on images in the first row.

Evaluation. We use the encoder of each model to extract the feature for the image classification task. We train an MLP, K-NN, and multinomial logistic regression on 5000 labeled images. All classifiers follow the same settings as the previous experiment. Finally, we ask each classifier to predict the class of 8000 unseen images. We also compute the average error rate instead of just the error rate. Table 4 shows the image classification results on the STL-10 dataset. The results show that VTEInfoNCE(2) yields the lowest average error-rate for each classifier. Furthermore, AVT achieve the highest average error-rate for each classifier. The results are quite surprising since we expect that AVT outperforms the predictive-transformation. Recall that the goal of AVT is to maximize $\mathbb{E}_{p_{\hat{\theta}}(\mathbf{t},\hat{\mathbf{z}},\mathbf{z})} q_\phi(\mathbf{t}|\hat{\mathbf{z}},\mathbf{z})$. We argue that combining $\mathbf{z}$ and $\hat{\mathbf{z}}$ directly to reconstruct $\mathbf{t}$ restricts the expressive power of $\mathbf{z}$ and $\hat{\mathbf{z}}$ mutually. Thus, it reduces the generalization of the model to extract the representation. In contrast, VTE models $\mathbf{z}$ and $\hat{\mathbf{z}}$ independently, allowing the representations to fully exploit the structure of data $\mathbf{tx}$ and $\mathbf{x}$, without losing the equivariant property.

5 CONCLUSIONS

In this research, we investigate the alternatives of autoencoding variational transformation (AVT). We call the models variational transformation-equivariant (VTE). We find that training VTE directly fails in the model to learn the representation of the data. The reason is due to the absence of the prior/inductive bias that gives the context of training. Instead, we propose training the model into two phases. In the first phase, we build a probabilistic self-supervised learning model to learn the representation of the transformed image. Theoretically, this model maximizes the mutual information (MI) between the transformation and the representation of the transformed image. We call the model predictive-transformation. In the second phase of training, we build a representation model that learns the representation of the original image. In theory, we maximize MI between the representation of the original image and the representation of the transformed image. We leverage the previous model to obtain the representation of the transformed image. However, computing MI directly is intractable. Therefore, we utilize Barber-Agakov and InfoNCE MI estimation method to maximize MI. Barber-Agakov estimation approximates the true posterior distribution with a variational distribution that is easy to compute. We call the model VTEBArber-Agakov. InfoNCE estimation method uses the deep learning network as an estimator function of the density ratio. In this research, we propose two versions of VTE with InfoNCE estimation. We call them VTEInfoNCE concatenated version and VTEInfoNCE separated version. Furthermore, we evaluate the proposed models and baseline on image classification tasks. Results on CIFAR-10 and STL-10 datasets show that our proposed models outperform the baseline.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the Tokopedia-UI AI Center of Excellence, Faculty of Computer Science, University of Indonesia, for allowing us to use its NVIDIA DGX-1 for running our experiments.

REFERENCES

Agakov, D. B. F. (2004). The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320):201.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2017). Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Cheng, P., Hao, W., Dai, S., Liu, J., Gan, Z., and Carin, L. (2020). CLUB: A contrastive log-ratio upper bound of mutual information. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1779–1788. PMLR.

Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In Gordon, G. J., Dunson, D. B., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 215–223. JMLR.org.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2990–2999. JMLR.org.

Cohen, T. S. and Welling, M. (2017). Steerable cnns. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1422–1430. IEEE Computer Society.

Donsker, M. D. and Varadhan, S. S. (1975). Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47.

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M. A., and Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(9):1734–1747.

Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning - the MIT press, 2016, 800 pp, ISBN: 0262035618. *Genet. Program. Evolvable Mach.*, 19(1-2):305–307.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In Honkela, T., Duch, W., Girolami, M. A., and Kaski, S., editors, *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, volume 6791 of *Lecture Notes in Computer Science*, pages 44–51. Springer.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Lenssen, J. E., Fey, M., and Libuschewski, P. (2018). Group equivariant capsule networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8858–8867.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Trans. Inf. Theory*, 56(11):5847–5861.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, volume 9910 of *Lecture Notes in Computer Science*, pages 69–84. Springer.

Noroozi, M., Pirsiavash, H., and Favaro, P. (2017). Representation learning by learning to count. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5899–5907. IEEE Computer Society.

Poole, B., Ozair, S., van den Oord, A., Alemi, A., and Tucker, G. (2019). On variational bounds of mutual information. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR.

Qi, G. (2019). Learning generalized transformation equivariant representations via autoencoding transformations. *CoRR*, abs/1906.08628.

Qi, G., Zhang, L., Chen, C. W., and Tian, Q. (2019). AVT: unsupervised learning of transformation equivariant representations by autoencoding variational transformations. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8129–8138. IEEE.

Schmidt, M., Roux, N. L., and Bach, F. R. (2017). Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1-2):83–112.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Wang, D. and Liu, Q. (2018). An optimization view on dynamic routing between capsules. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net.

Zhang, L., Qi, G., Wang, L., and Luo, J. (2019). AET vs. AED: unsupervised representation learning by autoencoding transformations rather than data. In *IEEE*

Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. Computer Vision Foundation / IEEE.

APPENDIX

5.1 Models Architecture on CIFAR-10 Dataset

Table 5 shows the architecture being used to build predictive-transformation, AVT, VTEBA, VTEInfoNCE separated, and VTEInfoNCE concatenated on CIFAR-10 dataset. The architecture of the encoder follows the implementation of Network-In-Network.

Table 5: Architecture being used on CIFAR-10 experiment.

Encoder	**Decoder, f, g, dan h**

`Block(3, 192, 5)`
`Block(192, 160, 1)`
`Block(160, 96, 1)`
`Max-Pool(3, 2, 1)`
	`Linear(512, 2048)`
	`ReLU`
	`Linear(2048, 512)`

`Block(96, 192, 5)`
`Block(192, 192)`
`Block(192, 8)`
`Avg-Pool(3, 2, 1)`	

$\mu_\phi \rightarrow$ `Linear(512, 512)` $\log \sigma_\phi \rightarrow$ `Linear(512, 512)`	

`Block(in, out, kernel)` is a module consists of `Conv2D(in, out, kernel, stride=1, padding=(kernel - 1) // 2)` $\rightarrow$ `Batch Norm 2D` $\rightarrow$ `ReLU`. Parameter `in` denotes the size of input channel, `out` denotes the size of output channel, and `kernel` denotes the size of kernel which owns the same height and width.

5.2 Models Architecture on STL-10 Dataset

Tabel 6 shows architecture being used to build *predictive-transformation*, AVT, VTEBA, and VTEInfoNCE on STL-10 *dataset*. In this experiment, the encoder adopts architecture of Alexnet. `Alex Block(in, out, kernel, stride, padding)` is a module consists of `Conv2D(in, out, kernel, stride, padding)` $\rightarrow$ `ReLU`.

Table 6: Architecture being used on STL-10 experiment.

Encoder	**Decoder, f, g, dan h**

`Alex-Block(3, 64, 11, 1, 2)`
`Max-Pool(3, 2, 0)`
	`Linear(512, 2048)`
	`BatchNorm`
	`ReLU`

`Alex-Block(64, 192, 5, 1, 2)`
`Max-Pool(3, 2, 0)`
	`Linear(2048, 1024)`
	`BatchNorm`
	`ReLU`

`Alex-Block(192, 384, 3, 1, 1)`	`Linear(1024, 512)`
`Alex-Block(96, 192, 5)`	

`Alex-Block(192, 192)`
`Max-Pool(3, 2, 0)`	

$\mu_\phi \rightarrow$ `Linear(1024, 512)` $\log \sigma_\phi \rightarrow$ `Linear(1024, 512)`	