ECAI 2024
U. Endriss et al. (Eds.)
© 2024 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA240689

A Parallel Gumbel-Softmax VAE Framework with Performance-Based Tuning

Fangshi Zhou[a], Tianming Zhao[a], Luan Viet Nguyen[a] and Zhongmei Yao[a,*]

[a]Department of Computer Science, University of Dayton, Ohio, United States
ORCID (Fangshi Zhou): https://orcid.org/0000-0002-0021-4661, ORCID (Tianming Zhao): https://orcid.org/0000-0002-1177-6897, ORCID (Luan Viet Nguyen): https://orcid.org/0000-0001-5516-2443, ORCID (Zhongmei Yao): https://orcid.org/0000-0002-8055-7110

Abstract. Traditional training algorithms for Gumbel Softmax Variational Autoencoders (GS-VAEs) typically rely on an annealing scheme that gradually reduces the Softmax temperature τ according to a given function. This approach can lead to suboptimal results. To improve the performance, we propose a parallel framework for GS-VAEs, which embraces dual latent layers and multiple sub-models with diverse temperature strategies. Instead of relying on a fixed function for adjusting τ, our training algorithm uses loss difference as performance feedback to dynamically update each sub-model's temperature τ, which is inspired by the need to balance exploration and exploitation in learning. By combining diversity in temperature strategies with the performance-based tuning method, our design helps prevent sub-models from becoming trapped in local optima and finds the GS-VAE model that best fits the given dataset. In experiments using four classic image datasets, our model significantly surpasses a standard GS-VAE that employs a temperature annealing scheme across multiple tasks, including data reconstruction, generalization capabilities, anomaly detection, and adversarial robustness. Our implementation is publicly available at https://github.com/wxzg7045/Gumbel-Softmax-VAE-2024/tree/main.

1 Introduction

Variational Autoencoders (VAEs) [25] have recently achieved significant advancements in deep learning. They learn to transform input data via an encoder into its *latent space*, sample latent variables from this space, and then generate new, similar data via a decoder. This training is compelling as it does not require any label from input data, which belongs to *self-supervised* learning. VAEs have been successfully used in many applications [17, 20], including dimensionality reduction, data generation for virtual reality and artistic creation, image denoising, and anomaly detection in cyber-security.

In a *standard* VAE, the latent space is *continuous* (e.g., following a Gaussian distribution). However, for discrete data such as words in a sentence or pixels in an image, a *discrete* latent space is more appropriate, often described by a categorical distribution. The Gumbel-Softmax (GS) distribution [21] has been introduced to handle discrete latent variables, which is a *differentiable* approximation of the categorical distribution. Gumbel Softmax VAEs (GS-VAEs) use the GS distribution for sampling latent variables, allowing gradients to flow through the sampling process, which is crucial for training models using backpropagation.

Existing theoretical results [30] show that as the Softmax temperature τ of the GS distribution approaches zero, the GS distribution converges to the true categorical distribution that is learned by the VAE. In alignment with these, prior work [21, 18] suggests a temperature annealing scheme for training the model, which initiates τ at a relatively high value and gradually decreases it to a small but non-zero value. While various strategies have been employed to adjust τ in existing work [8, 37] and our experiments, we find that GS-VAE training is extremely sensitive to the temperature strategy and also highly dependent on datasets. To tackle the difficulty in training and improve model performance, we propose a new framework for GS-VAEs. Our key contributions and findings are summarized next.

First, without any prior knowledge about the data, we need a model that can quickly adapt to any given dataset and various tasks. To take this challenge, we introduce a multi-model exploration mechanism that improves adaptability and optimization. Our design consists of multiple GS-VAE instances in parallel. Individual sub-models employ a range of diverse temperature strategies and accommodate model adjustments based on the best sub-model evaluated per training epoch. At the end of training, the best sub-model is saved and then used for testing. We find that even with 5 or 8 sub-models for training, our design significantly improves model performance across different datasets.

Second, instead of relying on a fixed function for updating τ, our training algorithm uses loss difference as performance feedback to dynamically update each sub-model's temperature. This is inspired by the need to balance exploration and exploitation in learning. When a sub-model's performance is close to the best sub-model's performance, it is beneficial to strengthen its exploitation ability. Conversely, the sub-model may take larger temperature change, leading to more exploration to find better solutions. Our algorithm also utilizes a patience mechanism to ensure that a sub-model makes a big change only when its patience counter exceeds given threshold, in order to avoid hasty temperature adjustments due to short-term fluctuations.

Third, experiments show that our model surpasses a standard GS-VAE that uses a temperature annealing scheme in various important applications across four well-known image datasets. When evaluating generalization capabilities, we find that our model can reconstruct data of unfamiliar categories that never appear in the training

[*] Corresponding Author Emails: {zhouf4, zyao01}@udayton.edu

dataset whereas a standard GS-VAE fails this task (Figure 7-8). In anomaly detection, our model can identify altered (or anomaly) data with *more than* 80% *accuracy*, whereas a standard GS-VAE achieves *accuracy below* 40% under the same circumstances (Figure 9). When input data is under white-box adversarial attack, even advanced pre-trained deep learning models deteriorate significantly (e.g., VGG16 has *accuracy below* 20% for $\epsilon \geq 0.3$), whereas

with *our model for repairing/sanitizing data*, a basic pre-trained CNN can achieve *accuracy more than* 87% (Table 2-3).

Finally, while theoretical results show that as temperature $\tau \to 0$, the GS distribution converges to the categorical distribution, the optimal temperature τ discovered by our model is much bigger than zero for all datasets studied (Figure 5), which is in alignment with suggestions in [16]. This important finding demonstrates that instead of monotonically decreasing τ over iterations, *GS-VAE training should strike a balance between discretization and continuity of latent variables*, as continuous samples let gradients pass through the sampling process, which is crucial for the model to capture core hidden features of input data and hence generate output with smaller error.

We next discuss relation work in Section 2 and the basis of GS-VAEs in Section 3. We present our model and training algorithms in Section 4 and experiments in Section 5. Section 6 evaluates model robustness when data is compromised by various attacks and Section 7 concludes our work.

2 Related work

The Gumbel-Softmax distribution [21, 30] sheds light on learning categorical datasets and has led to significant improvements in VAEs. Recent research can be divided into two main categories: one that improved the performance tied to optimizing the Softmax temperature and the other that applied VAEs to various tasks and protection against adversarial attacks.

Early works in [3, 8, 11, 18, 37, 38, 42] have investigated either a constant Softmax temperature or a dynamic, model-specific tuning process. These studies led to a deeper understanding of how the Softmax temperature affects model training and performance. Similarly, [16] introduced the Opti-Softmax method, which aims to find an optimal temperature that minimizes information loss during model training. This method represents a significant advancement in understanding the trade-offs in temperature settings, suggesting that temperatures that are neither too high nor too low are beneficial for model accuracy and generalization.

A considerable amount of research has been devoted to refining the latent space architecture of VAEs. Recent studies in [1, 9, 12, 36, 43] have focused on improving the performance of VAEs in image reconstruction tasks. These studies have shown success in reconstructing clean data, but they also demonstrated a decline in performance when the datasets include data from previously unseen categories. Additionally, [19, 28] explored the use of VAEs to reconstruct images subjected to adversarial attacks. However, the application of VAEs against more challenging attacks, such as the Fast Gradient Sign Method (FGSM) [6, 24, 31, 40], remains unexplored.

Current academic literature lacks dedicated research that addresses temperature adjustment in GS-VAEs. Existing research often struggles to accurately capture data features, particularly in complex scenarios such as reconstructing novel, unseen data. Moreover, adversarial attacks, including the FGSM and Projected Gradient Descent (PGD) attacks, pose significant challenges to many existing models. Traditional methods, such as adversarial training [24], require significant time and computational resources and may overspecialize a model against certain known attacks. Another aspect involves detecting minuscule patch attacks where tiny, often imperceptible patches are added to image datasets. These slight modifications, which can deceive models more easily, are less explored than larger adversarial patch attacks in current literature.

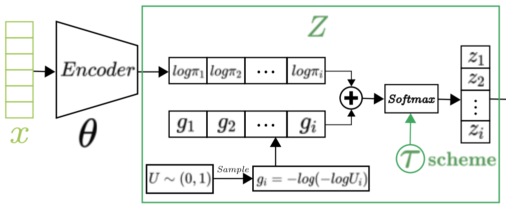

Figure 1. The structure of a standard Gumbel-Softmax VAE.

Our work aims to overcome the challenges and gaps discussed above.

3 Background: GS-VAEs

Given a dataset, let x be one of data points in the collection. As in Figure 1, a VAE consists of an encoder with parameters θ and a decoder with parameters ϕ, where encoder takes input x and transform it into a data point z in latent space in a different dimension. The *decoder* reconstructs x' from z, where x' is the output in the same dimension as x and more importantly, has the same distribution as x. We refer readers to [21, 25, 30] for derivations but focus on the *loss function* and the *sampling process* for creating latent data z in a GS-VAE.

The objective of training a VAE is to maximize the log-likelihood of data x, which can be reduced to *minimizing* the *negative* of ELBO (Evidence Lower Bound) [25]:

$$-ELBO := RL + D_{KL}, \text{ where} \tag{1}$$

$$RL := E_{z\sim q_\theta(z|x)}[-\log p(x|z)], \tag{2}$$

$$D_{KL} := D_{KL}[q_\theta(z|x)||p(z)], \tag{3}$$

RL (Reconstruction Loss) is expected negative log-likelihood of reconstructed data and D_{KL} is the Kullback-Leibler (KL) divergence measuring the difference between posterior distribution $q_\theta(z|x)$ learned by encoder and prior distribution $p(z)$ of latent space [26].

For a d-dimensional Bernoulli data point $x = [x_1, \ldots, x_d]$ where $d > 1$ and x_i is either 0 or 1 (e.g., black or white pixel in an image), RL (2) can be transformed into:

$$RL = -\sum_{j=1}^{d}[x_j \log x'_j + (1 - x_j)\log(1 - x'_j)], \tag{4}$$

which is a special case of the general cross-entropy loss, where x_j is the *true value* (0 or 1) and x'_j is the predicted *probability* of the class label being 1 from decoder.

The calculation of D_{KL} in (3) depends on the choice of latent space. For *discrete* data (e.g., pixels in image datasets), *categorical* latent space is more appropriate than Gaussian latent space (for continuous data). As shown in Figure 1, the parameters of the distribution of a categorical variable can be directly learned by encoder; i.e., $\log \pi = [\log \pi_1, \ldots, \log \pi_k]$ is output from encoder, where

$$\pi_j := P(C = j), \text{ for } j = 1, \ldots, k, \tag{5}$$

is the probability of categorical variable C taking class j among k distinct classes and $\sum_{j=1}^{k} \pi_j = 1$.

For the decoder to produce x' that is similar to x, it needs to *take a random sample* following (5) as its input. To this end, the

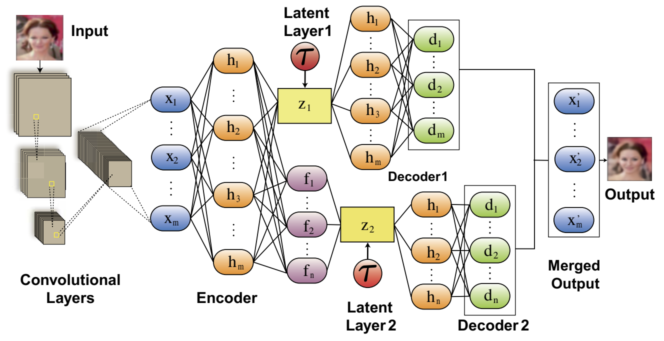

Figure 2. The structure of a GS-VAE incorporating dual latent layers z_1 and z_2 and dual decoders.

Gumbel-max trick [21] generates k Gumbel (0,1) random variables, $G_1, \ldots, G_k$, and then combines π_j to create a k-dimensional one-hot vector (i.e., a vector of length k with exactly one element set to 1 and all others set to 0):

$$Z := \text{one-hot}\left[\underset{j\in\{1,\ldots,k\}}{\arg\max}\,(\log \pi_j + G_j)\right], \qquad (6)$$

where the position of the 1 indicates the class that the data belongs to. Because the arg max in (6) is not differentiable, the Softmax function with temperature τ serves as a *differentiable* approximation for (6). The input to decoder thus becomes $Z := [Z_1, \ldots, Z_k]$, where each $Z_j \in [0, 1]$ given by:

$$Z_j := \frac{\exp((\log \pi_j + G_j)/\tau)}{\sum_{m=1}^{k} \exp((\log \pi_m + G_m)/\tau)}, \; j = 1, \ldots, k, \qquad (7)$$

and temperature parameter $\tau > 0$. This allows the gradients to be computed and back-propagated through the neural network. The Z_j follows the Gumbel-Softmax distribution [21].

We follow (7) to generate samples as inputs to the decoder (Figure 1). Temperature τ controls differentiability of GS samples. At high temperatures $\tau > 1$ the samples generated are more continuous, whereas at low temperatures they become more discrete. As $\tau \to 0$, the distribution of (7) converges to the categorical distribution [30]:

$$P\left(\lim_{\tau\to 0} Z_j = 1\right) = \pi_j, \quad P\left(\lim_{\tau\to 0} Z_j = 0\right) = 1 - \pi_j, \qquad (8)$$

for $j = 1, 2, \ldots, k$. Prior work [21, 10] suggests that one can start training from a higher temperature to benefit from the smoother gradient and slowly lower τ to make the distribution more discrete, thus more closely approximating the true categorical latent space.

Finally, assuming that $p(z)$ is a uniform distribution across k categories (a common practice), the KL-divergence (3) yields:

$$D_{KL}[q_\theta(z|x)||p(z)] = \sum_{j=1}^{k} \pi_j \log\left(\frac{\pi_j}{1/k}\right), \qquad (9)$$

where π_j as in (5) is learned by encoder.

To summarize, the loss that will be minimized by a GS-VAE is:

$$Loss := -EBLO = RL + D_{KL}. \qquad (10)$$

with RL in (4) and D_{KL} in (9), where RL ensures that the model learns to generate outputs x' that are close to the original inputs x, preserving the essential characteristics of the data, and D_{KL} ensures that the latent space distribution learned by the model approaches its true value but does not deviate too much from the prior uniform distribution $p(z)$.

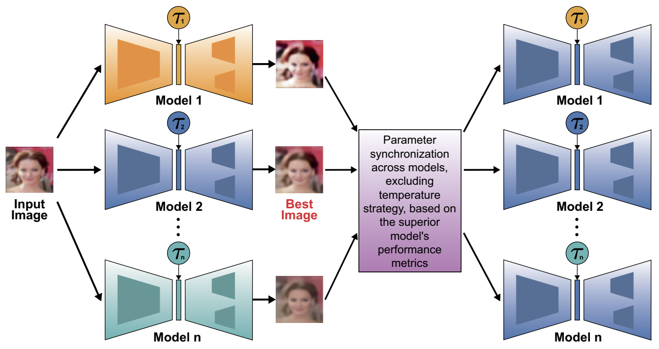
Input Image
Model 1
Model 2
Model n
Best Image
Parameter synchronization across models, excluding temperature strategy, based on the superior model's performance metrics
Model 1
Model 2
Model n

Figure 3. The parallel structure with performance-based model parameter adjustment per training epoch.

4 Parallel Model and Training Algorithms

Our model consists of n sub-models, $\{M_1, M_2, \ldots, M_n\}$, where n is a small number (e.g., $n = 5$ or 8) and each sub-model M_i is a GS-VAE with dual latent layers and dual decoders. We start from introducing M_i.

4.1 *Sub-model with Dual Latent Layers*

We add *dual* latent layers with their own decoders upon a standard GS-VAE as in Figure 2. Similar architectures of multi-level latent variable models have been applied in recent text generation applications [41]. Different from these work, our design aims to better extract hidden features of original data.

In sub-model M_i, input data is encoded through a series of convolutional layers into high-dimensional feature representations, here indicated by the blue x layers post-convolution. These represent the feature maps that capture essential data characteristics for further processing. The extracted feature maps are then passed into two separate fully connected layers, generating two sets of latent representations corresponding to two distinct latent spaces, denoted as z_1 and z_2 in Figure 2, where the first layer z_1 captures higher-level features and the second layer z_1 focuses on finer details. Dual layers allow the model to disentangle different factors of variation more effectively, leveraging the layered information for more precise output.

Subsequently, z_1 and z_2 are fed into their decoders, respectively. Each decoder is a mirrored encoder, and reconstructs outputs that match the dimension of input data. The notations f, h, and d respectively represent the fully connected, hidden, and decoded layers, respectively. The hidden layers h capture intricate data patterns, while the decoded layers d are used to reconstruct data from the latent space representations. Through these two channels, decoders learn to *reconstruct data from different perspectives of latent spaces*. Ultimately, the outputs from both decoders are averaged to generate the final reconstruction x'.

4.2 *Parallel Model with Adaptive Temperature Tuning*

Our parallel model $\{M_1, M_2, \ldots, M_n\}$ is illustrated in Figure 3, where sub-models M_i have their own strategies for adjusting the Softmax temperature τ. At the end of each training epoch, the best sub-model is discovered based on measurement of the performance and shares its model parameters with the other $n - 1$ models, getting ready for the next training epoch. The overall algorithm is shown in Algorithm 1, which we elaborate on in the following.

Algorithm 1 Overall Training Algorithm

```
Require: Number of Models $n$, Training Data
 1: Initialize $n$ GS-VAE models $\{M_1, M_2, \ldots, M_n\}$ and independent optimizers
 2: Assign temperature strategies to $\{M_1, M_2, \ldots, M_n\}$ according to Algorithm 2
 3: for each training epoch do
 4:     Train $\{M_1, M_2, \ldots, M_n\}$ in parallel on given data
 5:     Evaluate the performance of each model
 6:     for $i = 1$ to $n$ do
 7:         Compute loss $L_i$
 8:     end for
 9:     Find the best model $M_{\text{best}}$ with the minimum loss $L_{\text{best}}$
10:     for $i = 1$ to $n$ do
11:         if $M_i$ is not $M_{\text{best}}$ then
12:             Adjust temperature $\tau$ for $M_i$ by Algorithm 3
13:         end if
14:     end for
15:     Assign the encoder and decoders of $M_{best}$ to all other models,
        but keep the temperature strategy the same for individual $M_i$'s
16: end for
17: Save $M_{best}$ with its temperature strategy for various tasks
```

- *Initialization Phase*: First, we create multiple instances of the GS-VAE model with dual latent layers, each equipped with an independent optimizer (e.g., the Adam optimizer). Initial temperature strategies are assigned to models according to Algorithm 2, including five different schemes, i.e., fixed value, linear increase, linear decrease, exponential increase, and lastly exponential decrease. For $n > 5$, the repeated strategies start from linear increase. This ensures greater *diversity* in temperature strategy design and expands the scope for exploring parameters, so that our model can quickly adapt to underlying datasets.
- *Training and Performance Evaluation*: During this phase, we train all sub-models in parallel. The training process focuses not only on the loss of each model but also evaluates performance on specific tasks by calculating metrics such as log-likelihood, reconstruction error, or classification accuracy. These evaluation results provide a crucial base for subsequent model adjustments.
- *Dynamic Adjustment and Parameter Sharing*: At the end of each training epoch, we compare sub-models' performance and select the model with *the minimum loss* given by (10). The parameters of this model are then shared with other models to synchronize the optimal network learning state. Meanwhile, we keep the temperature strategy the same for each sub-model, but dynamically adjust its temperature value according to Algorithm 3.
- *Iterative Optimization*: In subsequent training epochs, this process continues, including parallel training, performance evaluation, selection of the best model, and dynamic temperature adjustment. This iterative process continually optimizes the model performance and progressively refines and adjusts the temperatures based on the real-time performance.
- *Final Model Selection and Application*: Finally, at the end of the training, we select the overall best model and its corresponding temperature strategy, applying it to different tasks such as image reconstruction, anomaly detection, and so on.

In Algorithm 3, we choose the best sub-model M_{best} based on the minimum loss among n sub-models. We then use loss difference between individual M_i and M_{best} as performance feedback, $\Delta\tau$, to adjust temperature τ_i (see lines $6-7$). Patience used in this algorithm keeps track of the number of training epochs that the training process continues *without* improvement. To achieve this, we adopt a patience counter that is increased when the current best model *does not* have a smaller loss than the loss from the previous best model, or reset otherwise. Line 12 ensures that temperature τ_i for each sub-model is kept within a defined range $[\tau_{min}, \tau_{max}]$. Utilizing the patience mechanism that aims to prevent hasty parameter changes due to short-term fluctuations, we consider two scenarios:

Algorithm 2 Assign Temperature Strategies to Sub-models

```
Require: Number of Models $n$, Initial Temperature $\tau_{\text{initial}}$, Maximum Temperature $\tau_{\text{max}}$,
         Minimum Temperature $\tau_{\text{min}}$, Rate of Increase $l_{\text{inc}}$, $r_{\text{inc}}$, Rate of Decrease $l_{\text{dec}}$, $r_{\text{dec}}$
 1: Initialize each $\tau_i$ using $\tau_{\text{initial}}$ at epoch $t = 0$
 2: for $i = 1$ to $n$ do
 3:     if $i == 1$ then
 4:         Fixed: $\tau_i^{t+1} = \tau_i^t$
 5:     else if $i \mod 4 == 2$ then
 6:         Linear increase: $\tau_i^{t+1} = \tau_i^t + (i-1) \cdot l_{\text{inc}}$
 7:     else if $i \mod 4 == 3$ then
 8:         Linear decrease: $\tau_i^{t+1} = \tau_i^t - (i-1) \cdot l_{\text{dec}}$
 9:     else if $i \mod 4 == 0$ then
10:         Exponential increase: $\tau_i^{t+1} = \tau_i^t \cdot ((i-1) \cdot r_{\text{inc}})$
11:     else
12:         Exponential decrease: $\tau_i^{t+1} = \tau_i^t / ((i-1) \cdot r_{\text{dec}})$
13:     end if
14:     $\tau_i = \max(\tau_{\text{min}}, \min(\tau_i, \tau_{\text{max}}))$
15: end for
```

Algorithm 3 Temperature Adjustment Per Training Epoch

```
Require: Loss $L_i$ of Model $M_i$ for $i = 1, \ldots, n$, Loss $L_{\text{best}}$ of Model $M_{\text{best}}$, Learning Rate $lr$,
         Maximum Temperature $\tau_{\text{max}}$, Minimum Temperature $\tau_{\text{min}}$, Patience Threshold $p_{\text{thresh}}$
 1: At epoch $t = 0$, initialize patience counter $p_{\text{counter}_i}$ for each $M_i$
 2: for $i = 1$ to $n$ do
 3:     if $L_i > L_{\text{best}}$ then
 4:         Increment patience counter: $p_{\text{counter}_i}^{t+1} = p_{\text{counter}_i}^t + 1$
 5:         if $p_{\text{counter}_i}^{t+1} \geq p_{\text{thresh}}$ then
 6:             Adjust temperature: $\Delta\tau_i = lr \cdot (L_i - L_{\text{best}})$
 7:             Update $\tau_i$ for $M_i$: $\tau_i^{t+1} = \tau_i^t + \Delta\tau_i$
 8:         end if
 9:     else
10:         Reset patience counter: $p_{\text{counter}_i}^{t+1} = 0$
11:     end if
12:     Ensure within bounds: $\tau_i^{t+1} = \max(\tau_{\text{min}}, \min(\tau_i^{t+1}, \tau_{\text{max}}))$
13: end for
```

- If sub-model M_i's patience counter is within the patience threshold, set feedback $\Delta\tau = 0$ and its temperature is updated according to its own strategy as in Algorithm 2.
- Otherwise, feedback $\Delta\tau$ is given by loss difference between M_i's own loss and the best model's loss, scaled by the learning rate lr. The $\Delta\tau$ is then incorporated as in line 7 to update the temperature of this sub-model besides following its own strategy in Algorithm 2. This leads to a more significant temperature change, if the model lags far behind the best model. Note that model M_i makes this big change only when its patience counter exceeds the patience threshold.

Algorithm 3 is inspired by the need for balancing exploration and exploitation in machine learning [35]. When sub-model M_i's per-

Table 1. MSEs on data reconstruction under different VAEs. Values in bold are the best results and Standard GS-VAE is the baseline.

VAEs	MNIST	Fashion-MNIST	CIFAR-10	CelebA-HD
VAE	0.0081	0.0192	0.0763	0.0478
β-TCVAE [9]	0.0021	0.0081	0.0412	0.0412
Soft-Intro-VAE [12]	0.0194	0.0257	0.0211	0.0247
Standard GS-VAE	0.0154	0.0354	0.0591	0.0474
Our model	**0.0002**	**0.0036**	**0.0012**	**0.0032**

formance is very close to the best sub-model's performance (i.e., $(L_i - L_{best})$ is small), it is beneficial to make a small temperature change and hence strengthen the model's exploitation ability. Conversely, the model may take significant temperature changes, leading to more exploration to find better solutions. In addition, adjusting the temperature parameter dynamically also helps M_i escape from local optima that may be caused by a fixed strategy given in Algorithm 2.

5 Experiments

5.1 *Experimental Environment and Datasets*

Experiments are conducted in the Google Colab environment in Python 3 using Nvidia K80/T4 GPUs, with 16 GB memory and a memory clock speed of 0.82GHz/1.59GHz.

Four classic image datasets are utilized in experiments. The MNIST (Modified National Institute of Standards and Technology) dataset [7] is a collection of hand-written digit images, including digits from 0 to 9. Fashion-MNIST [45] represents images of fashion items. It comprises 10 distinct categories, such as t-shirts, trousers, shoes, and more. CIFAR-10 [27] consists of 32×32 pixel color images, distributed across ten classes, including airplanes, cars, dogs, cats, and more. The CelebA-HD dataset [29] contains 1024×1024 pixel images of celebrities' faces, with multiple images for each celebrity.

5.2 *Model Architectures and Reconstruction Evaluation*

We implement our parallel model following Algorithms 1-3. Each sub-model M_i consists of an encoder, dual latent layers, and dual decoders. As an example, for CIFAR-10 dataset, the encoder comprises four convolutional layers with 32, 64, 128, and 256 neurons, respectively. Similar architectures are adopted in [39]. These convolutional layers transform an input image onto two latent variable layers, where the first latent layer (z_1 in Figure 2) has 256 neurons, dealing with lower-level features, and the second latent layer (z_2 in Figure 2) has 128 neurons for capturing higher-level features. These two latent layers are then connected with their decoders that are mirrored encoders.

We use MSE (Mean Squared Error) as evaluation metrics for the data reconstruction task [17]. MSE quantifies the pixel-level differences between reconstructed and original images. Smaller MSE means higher quality reconstructed data. As in Table 1, our model (with $n = 8$) produces the smallest MSE among closely related work, and greatly outperforms the baseline, i.e., a standard GS-VAE using a temperature annealing scheme, across these four datasets.

5.3 *Impact of the Number of Models*

Figure 4 shows the MSE (blue lines) and training times (red lines) as the number of models n changes. When $n = 1$, our model reduces to a single GS-VAE and the corresponding MSE is the worst. Once n increases, our model improves reconstruction with smaller MSE. Notably, complex datasets like CIFAR-10 and CelebA-HD benefit more from our parallel structure, as their MSE decrease more significantly when n rises from 4 to 8, demonstrating that our model can capture intricate latent space for complex images.

However, there's a trade-off between MSE and model training time. As in Figure 4, the time (in red lines) required for training the model may increase linearly in n. The model improvement peaks at a moderate number of models, beyond which the additional computational cost may outweigh the marginal gains in reconstruction accuracy. The best choice seems to be around $n = 8$, where the balance between accuracy and training time is most favorable.

For the rest of the results, we use our model with $n = 8$.

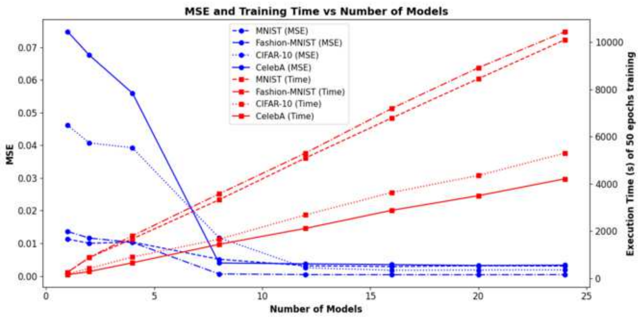

Figure 4. The impact of the number of models n on MSE values (blue lines) and training times (red lines) for four datasets.

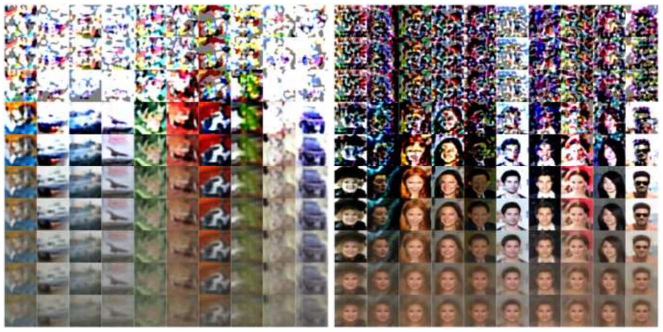

Figure 5. Impact of temperature τ on data generation. For CIFAR-10 (left), we compare results in a range of temperatures [0.5, 0.9, 10, 30, 60, 98.901, 120, 160, 200, 300], where 98.901 is the optimal temperature discovered by our model leading to quality reconstruction with the smallest MSE. For CelebA-HD (right), temperatures [0.5, 0.9, 0.95, 5, 10, 30, 51.35, 90, 200, 300] are tested, where 51.35 is the optimal temperature.

5.4 *Impact of Temperature*

Recall that the Softmax temperature τ plays a crucial role in training GS-VAEs, in that high τ increases continuity (benefiting gradient-based training), whereas low τ leads to more discrete latent variables (better approximating categorical latent space). However, discretization of latent variables deteriorates the decoder's ability to reconstruct data from the latent space. As displayed in Figure 5 with temperature τ configured at different values, there is a large variation among the images generated by our trained model. Neither very low nor very high temperatures lead to quality images. Figure 5 shows that the optimal temperature discovered by our model produces the best images with the smallest MSE. This demonstrates that striking

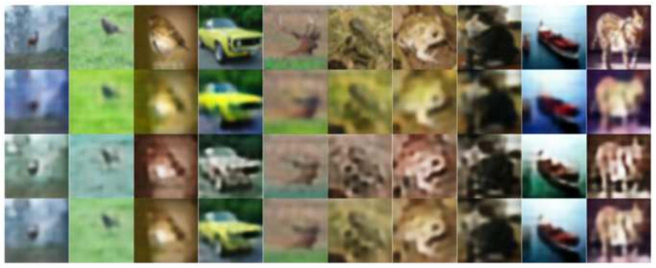
Figure 6. Impact of dual latent layers. The first row shows original CIFAR-10 images, the second displays reconstructions from the first latent space (z_1 in Figure 2), the third from the second latent space (z_2 in Figure 2), and the last merges outputs from both latent layers.

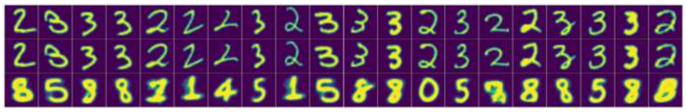
Figure 7. Data reconstruction of unfamiliar categories, where the model is trained *excluding* digits 2s and 3s from the MNIST training dataset. The top row shows original digits 2s and 3s from the MNIST *testing* dataset, the second showcases images generated by our model, and the third shows results using a standard GS-VAE.

a balance between discretization and continuity of latent variables is critical in training GS-VAEs, allowing our model to better capture the core features of input data and generate outputs with smaller MSE.

5.5 *Impact of Dual Latent Layers*

Figure 6 showcases the distinct features extracted by dual latent spaces z_1 and z_2 of our model on CIFAR-10. The last row shows images highlighting the model's ability to integrate distinct features into a cohesive whole. Figure 6 demonstrates that the details captured by each latent layer are indeed different. This layered approach to reconstruction allows our model to capture a richer representation of the data by utilizing the unique perspectives from dual latent layers.

5.6 *Data Reconstruction of Unfamiliar Categories*

This experiment evaluates the generalization capability of VAEs. We train our model upon the MNIST dataset and CIFAR-10, deliberately excluding *digits* 2 *and* 3 from the MNIST training set, and removing images $Automobiles$ from CIFAR-10 training set. After training, we aim to find whether the model is able to reconstruct these unfamiliar images as they do not appear during training.

Figure 7 shows that our model can successfully reconstruct data of unlearned categories, whereas a standard GS-VAE generate very blurry digits and some wrong ones. As *all digit images share standard basic strokes, such as curves and straight lines*, our model has learned these shared stroke features and is able to utilize them to generate new digits that are not seen during training. This remarkable learning capability is also observed in Figure 8 for *Automobile* images in CIFAR-10, where a standard GS-VAE fails the task.

6 Model Robustness

This section studies model robustness when data faces various attacks. In experiments, we first train our model with clean data and then utilize our trained model to process data tampered by attacks.

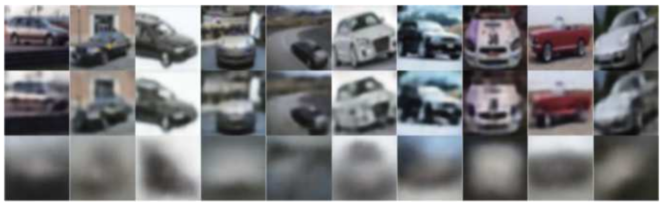
Figure 8. Reconstruction of unfamiliar data, where the model is trained *excluding* class $Automobile$ from the CIFAR-10 training dataset. The top row shows original images of $Automobile$ from the CIFAR-10 *testing* dataset, the second showcases images generated by our model, and the third shows results using a standard GS-VAE.

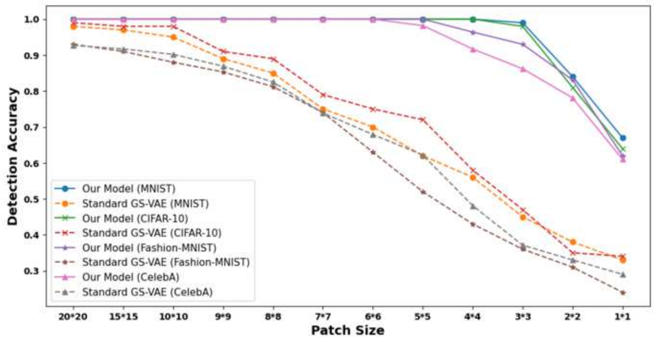

Figure 9. Performance comparison between our model and a standard GS-VAE in patch attack detection.

6.1 *Anomaly Data*

We investigate shaped adversarial patches [32], which are noise added onto images in specific shapes and positions. We leverage reconstruction errors, i.e., MSE, to detect anomaly data, because when anomaly data is fed into our trained model, it typically exhibits higher MSE. By appropriately selecting a threshold on reconstruction errors, our trained model is able to detect the presence of shaped patches in input data.

Detection results were visualized in Figure 9, where solid lines are from our model whereas dotted lines from a standard GS-VAE. Our model consistently outperforms a standard GS-VAE in detecting patch attacks across these four datasets, regardless of patch size, from large to minuscule.

6.2 *FGSM Attack*

In FGSM (Fast Gradient Sign Method) [14], the gradient of the loss with respect to the original image is used to create an adversarial image x^* in order to fool a classifier, where:

$$x^* = x + \epsilon \cdot \operatorname{sign}\left(\nabla_x J(\theta, x, y)\right), \tag{11}$$

x is the original input data with its label y, ϵ is defined as a multiplier used to control the magnitude of perturbations, θ represents the model's parameters, and J is the loss function of the model. It is well-known that convolutional neural networks (CNNs) are highly susceptible to FGSM attack.

Figure 10 demonstrates the robustness of our trained model, in that our model is still capable to generate clear handwritten digits given $\epsilon = 0.5$, whereas reconstructed images from a standard GS-VAE are increasingly blurry and unrecognizable.

Next, we feed reconstructed data generated by our model to a *basic pre-trained CNN* (consisting of 2 convolutional layers and 3 fully connected layers) to classify these reconstructed images.

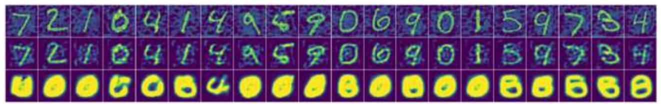

Figure 10. Reconstruction under FGSM attack on MNIST with $\epsilon = 0.5$. The top row is adversarial images created according to (11), the second is reconstructed images generated by our model, and the third is from a standard GS-VAE.

Table 2. Classification accuracy (in percent) under FGSM attack on three datasets, where CNN is a basic pre-trained CNN. The values in bold indicate the best results. Percent symbols are omitted due to limited space.

MNIST						
Model / ϵ	0.5	0.3	0.09	0.05	0.01	0.0
Our model	**87.28**	91.41	**96.54**	**98.17**	99.02	99.19
CNN + GS-VAE	11.40	25.39	61.32	84.66	94.26	95.82
AT [24]	55.52	**97.17**	-	98.91	99.16	-
Marina et al. [22]	11.31	17.51	78.67	91.13	**99.23**	**99.54**
Mirman et al. [15]	-	82.00	96.00	-	-	-
Ghosh et al. [13]	-	87.00	-	92.16	-	-
DCNN [40]	-	91.11	-	98.00	92.65	-
HDC [6]	-	-	-	-	63.00	92.00
Fashion-MNIST						
Model / ϵ	0.5	0.3	0.10	0.05	0.01	0.0
Our model	**88.44**	**88.97**	**89.59**	**90.12**	**90.53**	92.07
CNN + GS-VAE	7.51	8.30	60.75	72.02	80.21	89.54
Marina et al. [22]	-	-	-	-	54.00	88.00
VGG16 [4]	-	9.00	8.00	8.00	14.00	90.00
AlexNet [2]	-	0.92	11.28	36.33	-	87.63
ResNet-18 [2]	-	2.44	0.93	4.30	-	**93.52**
Resnet50 [4]	7.16	-	6.63	-	-	89.89
CBAM [4]	7.22	-	6.85	-	-	92.73
CIFAR-10						
Model / ϵ	0.5	0.3	0.10	0.05	0.01	0.0
Our model	**28.12**	**29.69**	**60.94**	**62.50**	**70.31**	79.84
CNN + GS-VAE	18.26	20.55	36.29	44.85	53.97	76.31
CNN	2.19	5.77	6.92	25.08	55.29	86.51
Madry et al. [31]	-	-	-	45.9	-	**87.30**
NT [44]	-	-	-	4.68	-	78.74
FGSM-AT [44]	-	-	-	40.51	-	77.10
AlexNet [2]	-	0.00	0.00	0.00	-	63.50
ResNet-18 [2]	-	3.15	2.43	1.94	-	83.43

As in Table 2, the row marked with bold "Our model" displays the best results obtained by using the basic pre-trained CNN with reconstructed images from our model, the next row "CNN + GS-VAE" is that with a standard GS-VAE, and the remaining rows are results from pre-trained advanced classifiers without using any VAE for reconstructed data. This shows that with our model for repairing data, even a basic CNN can be accurate and resilient to FGSM attack.

6.3 *PGD Attack*

PGD (Projected Gradient Descent) attack [31] is another white-box attack, where adversarial data is generated by:

$$x^{(0)} = x + \text{random noise within } \epsilon\text{-ball},$$
$$x^{(t+1)} = \Pi_{x+S}\left(x^{(t)} + \alpha \cdot \text{sign}\left(\nabla_x J(\theta, x^{(t)}, y)\right)\right), \qquad (12)$$

$x^{(0)}$ is the initial perturbed image, and $x^{(t+1)}$ is the adversarial example at iteration $t+1$, the operation Π_{x+S} projects the perturbed image back onto the ϵ-ball centered around the original image x, α is the step size for each iteration, and ϵ is the maximum allowed perturbation. Notably, PGD attack is more dangerous than FGSM, because it utilizes a gradient-descent approach and repeatedly searches for the optimal adversarial examples over iterations, generating more challenging adversarial data.

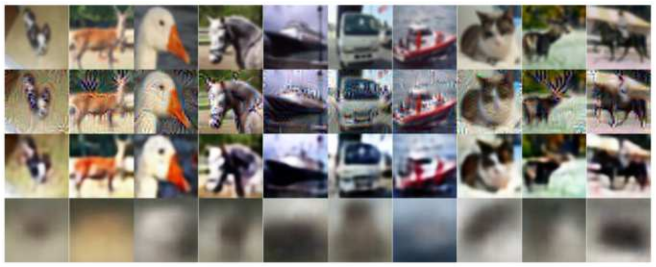

Figure 11. Reconstruction under FGSM attack on CIFAR-10 with $\epsilon = 0.5$. The top row is clean data, the second is adversarial images created as (11), the third is reconstructed images by our model, and the last is from a standard GS-VAE.

Table 3. Classification accuracy (in percent) under PGD attack on three datasets, where CNN is a basic pre-trained CNN. The values in bold indicate the best results. Percent symbols are omitted due to limited space.

MNIST						
Model / ϵ	0.5	0.4	0.3	0.2	0.1	0.0
Our model	**95.31**	**96.05**	**96.48**	**97.02**	**97.38**	99.19
CNN + GS-VAE	30.54	33.79	39.65	41.52	43.78	95.82
Madry et al. [31]	-	-	-	90.40	-	98.80
AEDPL-DL [5]	0.00	0.00	0.00	10.00	75.00	-
FGSM-AT [44]	-	-	-	85.86	-	**99.29**
Fashion-MNIST						
Model / ϵ	0.5	0.4	0.3	0.2	0.1	0.0
Our model	**87.63**	**88.45**	**89.34**	**90.01**	**90.75**	92.07
CNN + GS-VAE	29.69	33.77	36.24	38.55	39.73	89.54
StdCNN [23]	-	-	-	87.00	-	93.00
ResNet18 [33]	-	-	-	-	32.30	**93.60**
NRP [34]	-	-	-	-	86.15	90.56
CIFAR-10						
Model / ϵ	0.05	0.04	0.03	0.02	0.01	0.0
Our model	**59.38**	**64.06**	**67.19**	**70.31**	**71.88**	79.84
CNN + GS-VAE	12.36	19.39	32.18	47.28	55.11	76.31
CNN	0.00	0.14	0.15	1.36	1.72	87.30
VGG16 [22]	10.00	11.00	15.00	20.00	42.00	87.00
Guesmi et al. [15]	0.01	0.30	3.97	4.17	5.54	**99.80**

Figure 11 shows that perturbations added into images are imperceptible to human. It also indicates that our model effectively reconstructs data, whereas a standard GS-VAE fails this task.

We feed reconstructed data generated by our model to a basic pre-trained CNN to classify these reconstructed images. As in Table 3, the results from our model are consistently much better than others as ϵ increases. This shows that our model successfully sanitizes data compromised by PGD attack, thus preserving the performance of pre-trained classifiers.

7 Conclusion

This work proposed a parallel architecture for GS-VAE, significantly improving the model's adaptability to various datasets and tasks. The novelty lied in dynamically adjusting the Softmax temperature, coupled with a performance-based tuning algorithm. Our experiments demonstrated that, compared to standard GS-VAE models, ours showed substantial improvements in a wide range of applications. Moreover, our findings affirmed that maintaining a degree of continuity in sampling latent variables during training was crucial for the model to capture the core hidden features of the data effectively. This work also opened new avenues for studying anomaly detection and robustness in deep learning models.

Acknowledgments

The authors thank reviewers for improving the quality of this work. This research is supported by University of Dayton College of Arts and Sciences and Center for Cybersecurity and Data Intelligence, and the National Science Foundation (NSF) NSF-CRII-2245853.

References

[1] M. Adiban, M. Siniscalchi, K. Stefanov, and G. Salvi. Hierarchical residual learning based vector quantized variational autoencorder for image reconstruction and generation. In *33rd British Machine Vision Conference*, 2022.

[2] A. Agarwal, R. Singh, and M. Vatsa. The role of'sign'and'direction'of gradient on the performance of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 646–647, 2020.

[3] A. Agarwala, S. S. Schoenholz, J. Pennington, and Y. Dauphin. Temperature check: theory and practice for training models with softmax-cross-entropy losses. *Transactions on Machine Learning Research*, 2022.

[4] P. Agrawal, N. S. Punn, S. K. Sonbhadra, and S. Agarwal. Impact of attention on adversarial robustness of image classification models. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3013–3019. IEEE, 2021.

[5] M. N. Al-Andoli, S. C. Tan, K. S. Sim, P. Y. Goh, and C. P. Lim. A framework for robust deep learning models against adversarial attacks based on a protection layer approach. *IEEE Access*, 2024.

[6] H. E. Barkam, S. E. Jeon, S. Yun, C. Yeung, Z. Zou, X. Jiao, N. Srinivasa, and M. Imani. Hyperdimensional computing for resilient edge learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2023.

[7] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 77–82. IEEE, 1994.

[8] J. Chang, X. Zhang, Y. Guo, G. Meng, S. Xiang, and C. Pan. Differentiable architecture search with ensemble gumbel-softmax. *arXiv e-prints*, pages arXiv–1905, 2019.

[9] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[10] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11030–11039, 2020.

[11] R. Dabre and A. Fujita. Softmax tempering for training neural machine translation models. *arXiv e-prints*, pages arXiv–2009, 2020.

[12] T. Daniel and A. Tamar. Soft-introvae: Analyzing and improving the introspective variational autoencoder. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4391–4400, 2021.

[13] P. Ghosh, A. Losalka, and M. J. Black. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *AAAI conference on artificial intelligence*, volume 33, pages 541–548, 2019.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] A. Guesmi, K. N. Khasawneh, N. Abu-Ghazaleh, and I. Alouani. Room: Adversarial machine learning attacks under real-time constraints. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.

[16] Y.-L. He, X.-L. Zhang, W. Ao, and J. Z. Huang. Determining the optimal temperature parameter for softmax function in reinforcement learning. *Applied Soft Computing*, 70:80–85, 2018.

[17] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[18] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[19] U. Hwang, J. Park, H. Jang, S. Yoon, and N. I. Cho. Puvae: A variational autoencoder to purify adversarial examples. *IEEE Access*, 7:126582–126593, 2019.

[20] D. Im Im, S. Ahn, R. Memisevic, and Y. Bengio. Denoising criterion for variational auto-encoding framework. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[21] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

[22] M. Z. Joel, S. Umrao, E. Chang, R. Choi, D. Yang, J. Duncan, A. Omuro, R. Herbst, H. M. Krumholz, S. Aneja, et al. Adversarial attack vulnerability of deep learning models for oncologic images. *MedRxiv*, 2021.

[23] S. Kamath, A. Deshpande, and K. Subrahmanyam. How do sgd hyperparameters in natural training affect adversarial robustness? *arXiv preprint arXiv:2006.11604*, 2020.

[24] M. Khalooei, M. M. Homayounpour, and M. Amirmazlaghani. Layer-wise regularized adversarial training using layers sustainability analysis framework. *Neurocomputing*, 540:126182, 2023.

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv e-prints*, pages arXiv–1312, 2013.

[26] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[27] H. Li, H. Liu, X. Ji, G. Li, and L. Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

[28] X. Li and S. Ji. Defense-vae: A fast and accurate defense against adversarial attacks. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Part II*, pages 191–207. Springer, 2020.

[29] Z. Liu, P. Luo, X. Wang, and X. Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[30] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2016.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[32] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner. Minority reports defense: Defending against adversarial patches. In *International Conference on Applied Cryptography and Network Security*, pages 564–582. Springer, 2020.

[33] A. Muhammad, F. Shamshad, and S.-H. Bae. Adversarial attacks and batch normalization: A batch statistics perspective. *IEEE Access*, 2023.

[34] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 262–271, 2020.

[35] T. Osugi, D. Kim, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[36] G. Parmar, D. Li, K. Lee, and Z. Tu. Dual contradistinctive generative autoencoder. In *in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 823–832. IEEE Computer Society, 2021.

[37] M. B. Paulus, C. J. Maddison, and A. Krause. Rao-blackwellizing the straight-through gumbel-softmax gradient estimator. In *International Conference on Learning Representations*, 2020.

[38] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[39] G. G. Pihlgren, F. Sandin, and M. Liwicki. Improving image autoencoder embeddings with perceptual loss. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[40] J. Sen, A. Sen, and A. Chatterjee. Adversarial attacks on image classification models: Analysis and defense. *arXiv e-prints*, pages arXiv–2312, 2023.

[41] D. Shen, A. Celikyilmaz, Y. Zhang, L. Chen, X. Wang, J. Gao, and L. Carin. Towards generating long and coherent text with multi-level latent variable models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2079–2089, 2019.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[43] Y. Takida, W.-H. Liao, C.-H. Lai, T. Uesaka, S. Takahashi, and Y. Mitsufuji. Preventing oversmoothing in vae via generalized variance parameterization. *Neurocomputing*, 509:137–156, 2022.

[44] B. Vivek and R. V. Babu. Regularizers for single-step adversarial training. *arXiv preprint arXiv:2002.00614*, 2020.

[45] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.