IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 November 2023, accepted 18 December 2023, date of publication 25 December 2023, date of current version 3 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3346697

RESEARCH ARTICLE

Splitting of Composite Neural Networks via Proximal Operator With Information Bottleneck

SANG-IL HAN, KENSUKE NAKAMURA, AND BYUNG-WOO HONG
Department of Artificial Intelligence, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Byung-Woo Hong (hong@cau.ac.kr)

This work was supported in part by the Chung-Ang University Graduate Research Scholarship in 2022, in part by the Institute for Information and Communication Technology Planning and Evaluation (IITP) funded by the Korean Government [Ministry of Science and Information and Communication Technology (MSIT)] through the Chung-Ang University Artificial Intelligence Graduate School Program under Grant 2021-0-01341, and in part by the National Research Foundation of Korea under Grant NRF-RS-2023-00251366.

ABSTRACT Deep learning has achieved efficient success in the field of machine learning, made possible by the emergence of efficient optimization methods such as Stochastic Gradient Descent (SGD) and its variants. Simultaneously, the Information Bottleneck theory (IB) has been studied to train neural networks, aiming to enhance the performance of optimization methods. However, previous works have focused on their specific tasks, and the effect of the IB theory on general deep learning tasks is still unclear. In this study, we introduce a new method inspired by the proximal operator, which sequentially updates the neural network parameters based on the defined bottleneck features between the forward and backward networks. Unlike the conventional proximal-based methods, we consider the second-order gradients of the objective function to achieve better updates for the forward networks. In contrast to SGD-based methods, our approach involves accessing the network's black box, and incorporating the bottleneck feature update process into the parameter update process. This way, from the perspective of the IB theory, the data is well compressed up to the bottleneck feature, ensuring that the compressed information maintains sufficient mutual information up to the final output. To demonstrate the performance of the proposed approach, we applied the method to various optimizers with several tasks and analyzed the results by training on both the MNIST dataset and CIFAR-10 dataset. We also conducted several ablation studies by modifying the components of the proposed algorithm to further validate its performance.

INDEX TERMS Deep learning, information bottleneck, stochastic gradient descent, proximal algorithm.

I. INTRODUCTION

Deep learning methods have had a tremendous impact on machine learning and demonstrated successful performance in various fields [1], [2], [3], [4]. Deep learning consists of neural networks with large datasets, which requires efficient optimization methods. Stochastic Gradient Descent (SGD) [5] is a useful optimization method in deep learning, and several variants such as AdaGrad [6] and Adam [7] have been studied and developed based on it. These methods are widely used in the training of various models due to their efficiency and reliable performance. SGD and its variants select minibatches probabilistically to calculate gradients using backpropagation algorithms. They determine the update direction of network parameters and ensure that they do not deviate significantly from that direction through the use of adaptive learning rates and momentum. In this way, these methods update the parameters to expedite the convergence of the objective function towards its optimal point. Furthermore, studies have also been conducted to analyze deep neural networks from the perspective of the Information Bottleneck theory (IB) [8]. Based on the bottleneck features, the IB theory finds the optimal trade-off between reducing the complexity of the input to the bottleneck feature and

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

increasing the accuracy of predicting the output in the bottleneck. When applied to a deep neural network, each parameter can be optimized by selecting the bottleneck feature between the input and output of the network to adjust the mutual information between them.

SGD and its variants are efficient in finding an optimal solution for various tasks. However, since SGD and its variants update the direction based on stochastically divided data, convergence is not always guaranteed. To enhance the effectiveness of parameter update directions in SGD-based methods, it is worth considering the application of the IB perspective. First, define a bottleneck feature between the forward and backward network, and the parameters of the backward network are updated to increase the mutual information between the bottleneck feature and the output. Second, the parameters of the forward network are updated to decrease the mutual information between the input and the bottleneck feature. Updating these two sets of parameters sequentially with the same objective function yields no difference compared to updating them simultaneously. To create a distinction in this context, it is necessary to update each set of parameters sequentially using distinct objective functions. In other words, it is important to consider how to formulate the objective function to update the parameters in a manner that increases the mutual information between the bottleneck feature and the output or reduces the mutual information between the input and the bottleneck feature. However, directly optimizing each mutual information from the perspective of the IB theory is not easy in neural networks due to the difficulty of calculating them, presenting significant challenges.

Inspired by the proximal operator [9], we propose a sequential update method based on SGD and its variants. To simplify the problem, we address the task of fitting the parameters of the backward network first and subsequently update the parameters of the forward network, similar to the approach proposed by Shwartz-Ziv et al. [10]. The parameters of the backward network are updated using the basic objective function for SGD-based training. Additionally, the bottleneck feature is updated with this objective function to increase the mutual information between the output and the bottleneck feature. Next, we construct an objective function that updates the parameters of the forward network using the updated bottleneck feature. This objective function consists of the norm between the updated and existing bottleneck features. In most neural networks, the bottleneck feature can be positioned in the middle of the network. Therefore, the proposed method can be easily applied in an end-to-end learning environment that utilizes SGD or its variants as optimizers. Finally, we update the parameters of the network sequentially. Unlike the conventional proximal-based methods, we update the forward network by considering the second-order gradients of the objective function, resulting in a more accurate update direction. In our experiments, the proposed method is applied to various problems, including binary classification, image classification using convolutional networks, and Variational Autoencoder (VAE) [11]. Applying the proposed method to the binary classification problem resulted in faster convergence in the objective function and improved accuracy compared to the network that did not apply the method. In other tasks, we have observed improved convergence of the objective function and performance compared to using only SGD or its variants. Particularly, noteworthy results were achieved in the case of VAE. This demonstrates that the proposed method has a positive impact on objective function optimization about the IB theory.

In summary, we study the following in this study. (a) We consider the application of IB theory to existing SGD and its variants for sequential parameter update. (b) We propose optimization methods based on SGD and its variants, inspired by the proximal operator, that update the bottleneck feature and utilize it to update the parameters of the forward network. By adding a bottleneck feature updating step in the entire updating sequence, we can achieve the effect of optimizing the objective function of the IB theory. (c) We conduct experiments by applying the proposed method to models designed for various tasks. As a result, the proposed method demonstrated superior performance compared to existing models that solely employed SGD and its variants.

II. RELATED WORKS

A. SGD-BASED OPTIMIZATION METHODS

Optimization methods for objective functions in deep neural network learning have been extensively studied, and many of these methods have been developed based on SGD. SGD and its variants are utilized to update parameters in the majority of deep learning models, and these methods are known for their high efficiency and performance. The origin of these methods is the gradient descent algorithm, which is built upon the backpropagation algorithm. The concept of SGD was first introduced by Robbins et al. [5] and has since become a popular optimization method in deep learning models due to its efficiency and simplicity, achieved by stochastically selecting data. Bottou et al. [12] analyzed the training process under various conditions to consider when applying SGD to training large machine learning models. In practice, SGD has demonstrated good performance on complex convolutional models like VGGNet [13] and Inception networks [14].

However, SGD does not guarantee a constant convergence rate or direction because it relies on stochastic factors. To address this issue, SGD with the concept of momentum is introduced. Polyak et al. [15] introduced a momentum method that accelerates the convergence rate of SGD and reduces directional fluctuations. Nesterov et al. [16] introduced the Nesterov momentum method, which offers a more intuitive update approach. Sutskever et al. [17] conducted a study on Nesterov momentum and initialization methods in combination. In addition to the momentum method, variant optimization methods of SGD such as AdaGrad [6] and Adam [7], which offer adaptive learning rates, are popular. Various adaptive learning rate-based methods have evolved

in various ways. AdamW [18] provides better generalization performance to Adam through decoupled weight decay. AdamP [19] prevents an increase in weight norm through projection to help Adam find global optima. NAdam [20] replaces Adam's momentum with the Nesterov momentum method to find global optima more quickly. RAdam [21] computes the variance of the adaptive learning rate and uses it to provide stability in the training process through corrections in the learning formula. AdaInject [22] controls parameter updates to minimize oscillations closer to the minimum. Recently, there has been research on PNM [23] and AdaPNM [23], aiming to replace the traditional momentum with the Positive-Negative momentum approach. These methods maintain two independent momentum terms to control stochastic gradient noise. Adan [24] applies Nesterov momentum estimation in the adaptive gradient algorithm to accelerate convergence and estimate first and second-order moments. These various methods have demonstrated fast convergence and high performance.

B. IB THEORY IN DEEP LEARNING

IB theory has been studied extensively since its introduction is presented in Tishby et al. [8]. In deep learning, network training involves updating the network's parameters to predict Y that aligns with X in the joint probability distribution between X and Y, where X represents the input value and Y represents the output value. In IB theory, the focus is on the bottleneck representation feature Z within the network, aiming to control the trade-off between compressing the information from X into Z and preserving the information from Y within Z. From the perspective of mutual information, the amount of mutual information $I(X;Z)$ between X and Z should be minimized, and the amount of mutual information $I(Z;Y)$ between Z and Y should be maximized. Therefore, the objective function of the IB theory, which enables the training of deep learning networks, is represented as,

$$\inf_{p(z|x)} (I(X;Z) - \eta I(Z;Y)), \tag{1}$$

where η is a Lagrange multiplier and x and z are instances of X and Z, respectively.

Tishby et al. [8] demonstrated that during network learning, Z tries to compress information about X by minimizing its presence in X, while simultaneously maximizing the retention of information about Y. Based on the theory, Shwartz-Ziv et al. [10] divided network training into two main stages, demonstrating the fitting between Z and Y in the first stage and the compression between X and Z in the second stage.

Applying IB theory to deep learning is an intriguing approach. However, it comes with certain challenges. Typically, calculating the amount of mutual information in (1) is a highly challenging problem. Since obtaining the exact amount of mutual information directly from each data distribution is challenging, numerous estimators have been studied to approximate and calculate the mutual information in various ways. Belghazi et al. [25] introduce a method for approximating the lower bound of mutual information between data and training it in the neural network, enabling the calculation of mutual information. Goldfeld et al. [26] estimate the mutual information by considering the dependence between neurons in the network and focusing on the information flow within the neural network. Lorenzen et al. [27] analyzes information flow by accurately calculating the mutual information between quantized neurons in quantized neural networks, instead of continuous neurons. Through this estimation of mutual information, several studies have approached deep learning from various perspectives of the IB theory. Alemi et al. [28] train the network by estimating the lower bound using (1) through variational inference. Achille et al. [29] proposed information dropout as a regularization method to minimize $I(X;Z)$. Saxe et al. [30] analyzed the trade-off between $I(X;Z)$ and $I(Z;Y)$ by relating the training process of deep linear networks to IB theory. While many studies have applied the IB theory to neural networks and achieved some performance improvements, it is not easily applicable to models for all tasks. We aim to optimize (1) by simply incorporating an update of Z into the training process of neural networks.

C. PROXIMAL OPERATOR

The proximal operator is a field of research that aims to solve challenging, non-differentiable, and constrained problems in the realm of convex optimization. The proximal operator addresses the problem of minimizing $h(a)$ within the constrained environment of $a = v$, where $h(\cdot)$ is a convex function. The proximal operator scaled for λ, denoted as $\text{prox}_{h/\lambda}$, is defined by,

$$\text{prox}_{h/\lambda}(v) = \text{argmin}_a(h(a) + \lambda/2||a - v||_2^2). \tag{2}$$

The proximal algorithm [9] solves convex optimization problems using the proximal operator. Proximal minimization calculates a, which is the result of $\text{prox}_{h/\lambda}h(v_k)$ in the kth step, substitutes it for v_{k+1}, and repeats this process until v_k converges to obtain the optimal solution. The proximal gradient method [9] separates the function h into a differentiable function and a non-differentiable function to obtain an optimal solution. The accelerated proximal gradient descent [9] takes into account momentum to obtain the optimal solution, ensuring that the direction of the gradient in each iteration does not change rapidly. The alternating direction method of multipliers [31] adds a Lagrangian term for a function h with two variables to obtain the optimal solution. It achieves this by iteratively updating three variables in a sequential manner. Inspired by (2), we aim to express the update of the network as a proximal operator for intermediate features. By updating each variable sequentially, we achieve the effect of optimizing (1). Also, unlike conventional proximal-based methods, we consider the second-order gradients of the objective function to achieve better updates for the forward networks.

TABLE 1. Technical terms and their definitions.

Notations	Definition
x	The input data
y	The output data
$f(\cdot;\theta)$	The forward network with parameter θ
$g(\cdot;\phi)$	The backward network with parameter ϕ
z	The bottleneck feature between $f(\cdot;\theta)$ and $g(\cdot;\phi)$
$\mathcal{L}(g(z;\phi),y)$	The objective function

III. PROXIMAL-SEQUENTIAL NEURAL NETWORK UPDATE

In this section, we describe the whole method. In Section III-A, we introduce an optimization method that updates neural networks inspired by the definition of proximal operators in terms of IB theory. In Section III-B, we apply the proposed method to several tasks.

A. PROXIMAL OPERATOR-BASED SEQUENTIAL UPDATE

We define the intermediate feature of neural networks for input data as z. The location of z can be anywhere within the neural network. Based on z, we define a forward network $f(\cdot;\theta)$ with θ as the parameters, and a backward network $g(\cdot;\phi)$ with ϕ as the parameters. Furthermore, if a pair of dataset being learned is $(x,y) \in (X,Y)$, $z \in Z$ is the bottleneck feature that is forwarded from x to $f(\cdot;\theta)$, so the constraint $z = f(x;\theta)$ is established. The objective function is defined as the loss between $g(z;\phi)$ and y, so it is written as $\mathcal{L}(g(z;\phi),y)$. The notation of the entire neural network structure is in Table 1.

Applying IB theory to defined neural networks, training neural networks can be seen as a problem of finding optimal trade-offs that reduce $I(X;Z)$ and increase $I(Z;Y)$ for a given joint probability distribution $p(X,Y)$ of data, by finding θ, ϕ, and additionally z. To achieve this, we update ϕ and z first to fit $I(Z;Y)$, and then update θ to find the direction in which (1) is minimized.

We draw inspiration from the proximal operator for objective functions to update θ, ϕ, and z. Conventional proximal operators express the problem of minimizing $h(a)$ in constrained environments where $a = v$ for convex functions $h(\cdot)$. The proximal operator $\text{prox}_{h/\lambda}$, which is scaled by λ, is defined by (2). The constraint in neural networks is $z = f(x;\theta)$, and the equation to be minimized is $\mathcal{L}(g(z;\phi),y)$. Expressing it as a proximal operator is equivalent to,

$$\text{prox}_{\mathcal{L}/\lambda}(f(x;\theta)) = \text{argmin}_z(\mathcal{L}(g(z;\phi),y) + \lambda/2||z - f(x;\theta)||_2^2). \quad (3)$$

We present a method for updating θ, ϕ, and z using the objective function of (3). Here, we use SGD and its variants, which are commonly used for updating neural networks.

In the first attempt, the entire objective function of the proximal operator was to be used to update all parameters. Accordingly, ϕ and z were updated first through the objective function, and the objective function was recalculated using the updated z. However, it was determined that the proper optimization of the objective function was not achieved because the actual learning process did not proceed smoothly. To solve this problem, we break down the objective function into two parts and update ϕ and z using $\mathcal{L}(g(z;\phi),y)$ only to increase $I(Z;Y)$. Next, we substitute the updated z for $\tilde{z}$, replace $f(x;\theta)$ with $\tilde{z}$ in $\lambda/2||z - f(x;\theta)||_2^2$, and update θ to more directly guide the output of $\tilde{z}$ in the update direction of θ. The equations for updating ϕ, z, and θ using the basic gradient descent algorithm for each objective function are represented as,

$$\begin{aligned}\phi^{(n+1)} &:= \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}),y)\\ \tilde{z} &:= z - \beta \cdot \nabla_z \mathcal{L}(g(z;\phi^{(n)}),y)\\ \theta^{(n+1)} &:= \theta^{(n)} - \gamma \cdot \lambda/2 \cdot \nabla_\theta ||z - \tilde{z}||_2^2,\end{aligned} \quad (4)$$

where α, β, and γ represent the learning rates for ϕ, z, and θ, respectively. Also, z is initialized to $f(x;\theta)$ at the beginning of each iteration.

The update equation of θ in (4) is a first-order gradient method for updating $f(\cdot;\theta)$ that considers the gradient concerning θ to make $f(\cdot;\theta)$ output a fixed $\tilde{z}$. Here, we do not fix $\tilde{z}$, and instead, we substituted the update equation for $\tilde{z}$ in (4) into the update equation for θ. The equation is rearranged as,

$$\begin{aligned}\phi^{(n+1)} &:= \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}),y)\\ \theta^{(n+1)} &:= \theta^{(n)}\\ &\quad - \gamma \cdot \beta^2 \cdot \lambda/2 \cdot \nabla_\theta ||\nabla_z \mathcal{L}(g(z;\phi^{(n)}),y)||_2^2,\end{aligned} \quad (5)$$

where the update of θ becomes a second-order gradient method for updating $f(\cdot;\theta)$ that considers both the gradient with respect to z and the gradient with respect to θ in $\mathcal{L}(g(z;\phi),y)$.

The complete neural network update algorithm using the update equations from (5) is summarized as pseudo-code in Algorithm 1. Any gradient descent-based update methods can be used to update Algorithm 1, and by defining an appropriate bottleneck feature location within the network, Algorithm 1 can be applied to any network.

Algorithm 1 proximal Operator-Based Network Update

Require: $x, y, \alpha, \beta, \gamma, \lambda$
Ensure: $f(\cdot;\theta)$, $g(\cdot;\phi)$
Initialize $n = 1$
Initialize $\theta^{(n)}$, $\phi^{(n)}$ randomly
while $\mathcal{L}(g(z;\phi^{(n)}),y)$ not converged **do**
  $z \leftarrow f(x;\theta^{(n)})$
  $\phi^{(n+1)} \leftarrow \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}),y)$
  $\theta^{(n+1)} \leftarrow \theta^{(n)} - \gamma \cdot \beta^2 \cdot \lambda/2 \cdot \nabla_\theta ||\nabla_z \mathcal{L}(g(z;\phi^{(n)}),y)||_2^2$
  $n \leftarrow n + 1$
end while
Output θ, ϕ

B. EXTENSION TO MULTIPLE TASK MODELS

In this study, we applied the sequentially updated method inspired by the proximal operator to three task models: (a) binary classification, (b) image classification, and (c) VAE.

1) BINARY CLASSIFICATION AND IMAGE CLASSIFICATION

For binary classification and image classification, Algorithm 1 can be directly applied for training. To apply the method to binary classification and image classification models, it is necessary to determine the location of the feature z within the network as the bottleneck feature and update the parameters using Algorithm 1 according to the objective functions of each model. For binary classification, the objective function is mean square loss, while for image classification, the objective function is cross-entropy loss with the softmax function.

2) VARIATIONAL AUTOENCODER

To apply the method to the VAE, additional modifications are required. The VAE takes input data x and forwards it to the encoder $f(\cdot;\theta)$, which outputs the mean μ and the variance σ for the Gaussian distribution of the latent vector z, serving as the input to the decoder $g(\cdot;\phi)$. In the case of VAE, the question arises as to whether to consider the output of the encoder as the bottleneck feature or to consider the input of the decoder as the bottleneck feature. If the input of the decoder is considered the bottleneck feature, it implies that the input of the decoder is a randomly selected vector from a specific distribution. So the mutual information between this random value and the actual output data is not significant. Moreover, utilizing this mutual information to adjust the mutual information between the input and the output of the encoder is also not meaningful. Therefore, it is reasonable to consider the output of the encoder, which provides well-defined σ and μ values representing a specific distribution, as the bottleneck feature. The update of σ and μ is achieved by splitting the update of z into two parts within the update expression of (4). The loss used to update the encoder is expressed as the sum of L2 losses for both σ and μ. Update equation for VAE is represented as,

$$\begin{aligned} z &:= \mu + \sigma \cdot \epsilon \\ \phi^{(n+1)} &:= \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}), y) \\ \tilde{\mu} &:= \mu - \beta \cdot \nabla_\mu \mathcal{L}(g(z;\phi^{(n)}), y) \\ \tilde{\sigma} &:= \sigma - \beta \cdot \nabla_\sigma \mathcal{L}(g(z;\phi^{(n)}), y) \\ \theta^{(n+1)} &:= \theta^{(n)} - \gamma \cdot \lambda/2 \cdot \nabla_\theta(||\mu - \tilde{\mu}||_2^2 + ||\sigma - \tilde{\sigma}||_2^2), \end{aligned} \tag{6}$$

where ϵ is a random vector sampled from a Gaussian normal distribution $\mathcal{N}(0, I)$, z is the latent vector calculated based on σ and μ, and the objective function $\mathcal{L}$ corresponds to the mean square loss. α and γ are the learning rates for ϕ and θ, respectively, while β is the learning rate for μ and σ. At each iteration, we initialize (μ, σ) as $(\mu, \sigma) = f(x;\theta)$.

Like a second-order gradient method for updating $f(\cdot;\theta)$ in (5), we have substituted the update expressions of $\tilde{\mu}$ and $\tilde{\sigma}$ into the update expressions of θ in (6). When we simplify the equation, the gradient for the update of θ is expressed as the gradient of the sum of the L2 norm of the gradients of $\mathcal{L}(g(z;\phi), y)$ with respect to $\tilde{\mu}$ and the gradients of $\mathcal{L}(g(z;\phi), y)$ with respect to $\tilde{\sigma}$, represented as,

$$\begin{aligned} z &:= \mu + \sigma \cdot \epsilon \\ \phi^{(n+1)} &:= \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}), y) \\ \theta^{(n+1)} &:= \theta^{(n)} - \gamma \cdot \beta^2 \cdot \lambda/2 \cdot \nabla_\theta(||\nabla_\mu \mathcal{L}(g(z;\phi^{(n)}), y)||_2^2 \\ &\quad + ||\nabla_\sigma \mathcal{L}(g(z;\phi^{(n)}), y)||_2^2). \end{aligned} \tag{7}$$

The entire update algorithm of the VAE using the update equation of (7) is represented by pseudo-code as shown in Algorithm 2.

Algorithm 2 proximal Operator-Based VAE Update

Require $x, y, \alpha, \beta, \gamma, \lambda$
Ensure $f(\cdot;\theta), g(\cdot;\phi)$
Initialize $n = 1$
Initialize $\theta^{(n)}, \phi^{(n)}$ randomly
while $\mathcal{L}(g(z;\phi^{(n)}), y)$ not converged **do**
  Initialize ϵ randomly
  $\sigma, \mu \leftarrow f(x;\theta^{(n)})$
  $z \leftarrow \mu + \sigma \cdot \epsilon$
  $\phi^{(n+1)} \leftarrow \phi^{(n)} - \alpha \cdot \nabla_\phi \mathcal{L}(g(z;\phi^{(n)}), y)$
  $\theta^{(n+1)} \leftarrow \theta^{(n)} - \gamma \cdot \beta^2 \cdot \lambda/2 \cdot \nabla_\theta(||\nabla_\mu \mathcal{L}(g(z;\phi^{(n)}), y)||_2^2$
    $+||\nabla_\sigma \mathcal{L}(g(z;\phi^{(n)}), y)||_2^2)$
  $n \leftarrow n + 1$
end while
Output θ, ϕ

IV. EXPERIMENTS

In this section, to verify the effectiveness of the proposed method, we compare it with the baseline method on the three tasks described in Section III-B (binary classification, image classification, VAE).

A. IMPLEMENTATION DETAIL

1) DATASET

In the experiment, we use a dataset specific to each task. For binary classification, we use a dataset that consists of 600 randomly generated 2-dimensional input vectors from a Gaussian distribution $\mathcal{N}(0, I)$ classified into 2 classes. The dataset is shown in Fig. 3a. For image classification and VAE, we use the MNIST dataset and CIFAR-10 dataset. Specifically, for VAE, we trained the model on the task of adding noise to images of certain classes, then removing the noise and reconstructing the images. The MNIST dataset consists of hand-written digit images from 0 to 9, organized into 10 classes. It was introduced by LeCun et al. [32]. This dataset contains grayscale images of size 28×28 and is composed of 55,000 training samples and 10,000 test samples. In the actual experiments, images were resized to 32×32 for ease of configuration. The CIFAR-10 [33] dataset is widely used in the field of computer vision, consisting of 50,000 training samples and 10,000 test samples taken from 10 different classes. The classes include common objects such as cars, birds, and cats.

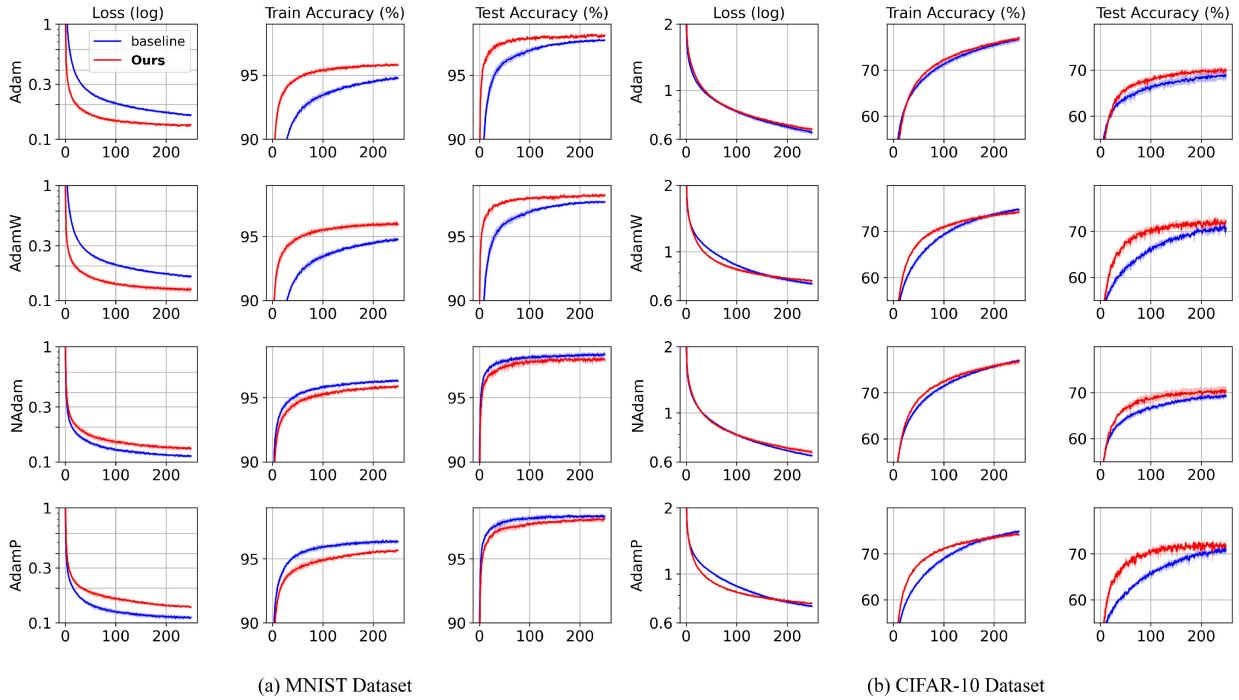


FIGURE 1. Empirical comparison of the proposed method (red line) with baseline (blue line) in image classification. (1st, 2nd, and 3rd columns in each dataset part) The training loss, training accuracy (%), and test accuracy (%) curves across (x-axis) iteration for (a) MNIST and (b) CIFAR-10, optimized using (1st, 2nd 3rd, and 4th rows) Adam, AdamW, NAdam, and AdamP respectively.

2) EXPERIMENT SETTING

For binary classification, we used 4 hidden layers with 8 features each, and the sigmoid activation function. The bottleneck feature was precisely positioned at the middle layer. We employed full-batch gradient descent as the base optimizer with learning rates $\alpha = 0.1$, $\beta = 10$, and $\gamma = 100$. For image classification, we used 4 convolutional layers with $f(\cdot;\theta)$ and ReLU as the activation function. For $g(\cdot;\phi)$, we used a classifier consisting of 2 hidden layers with ReLU as the activation function. For VAE, we used an encoder with 5 convolutional layers and 2 linear layers for $f(\cdot;\theta)$, with LeakyReLU as the activation function. For $g(\cdot;\phi)$, we used a decoder with 5 convolutional layers and Tanh as the activation function. The dimension of the latent vector is 128. Both $f(\cdot;\theta)$ and $g(\cdot;\phi)$ used batch normalization. In all experiments, we set $\lambda = 1$.

3) EVALUATION SETUP

For the evaluation metrics to compare the performance of the learning results, we compare the prediction accuracy for binary classification and image classification tasks. For VAE, we use the peak signal-to-noise ratio (PSNR) between the reconstructed images and the original images as the evaluation metric.

4) COMPARISON METHODS

We compare the performance of the proposed method by applying the proposed method to several state-of-the-art optimizers. We train image classification and denoising VAE using the following four optimizers and compare their performance:

Adam [7]: Adam is the most commonly considered optimizer when training deep learning networks. Adam estimates first and second-order moments to correct gradients, accelerating convergence speed. In the case of image classification, we used $\alpha = 0.001$, $\beta = 10$, and $\gamma = 0.001$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.01$, and $\gamma = 0.01$ for the CIFAR-10 dataset. For VAE, we used $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.01$ for the MNIST dataset, and $\alpha = 0.00001$, $\beta = 0.1$, and $\gamma = 0.0001$ for the CIFAR-10 dataset.

AdamW [18]: The conventional L2 regularization in Adam did not demonstrate adequate generalization performance. In contrast, AdamW decouples weight decay, improving the generalization performance of weight decay. In the case of image classification, we used $\alpha = 0.001$, $\beta = 0.1$, and $\gamma = 0.001$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.01$, and $\gamma = 0.01$ for the CIFAR-10 dataset. For VAE, we used $\alpha = 0.0001$, $\beta = 0.001$, and $\gamma = 0.001$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.001$ for the CIFAR-10 dataset.

NAdam [20]: NAdam replaces the traditional Adam's momentum method with the Nesterov accelerated gradient method to estimate momentum during training, which increases the convergence speed. In the case of image classification, we used $\alpha = 0.001$, $\beta = 0.01$, and $\gamma = 0.1$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.01$, and $\gamma = 0.01$ for the CIFAR-10 dataset. For VAE, we used $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.0001$ for the MNIST dataset,

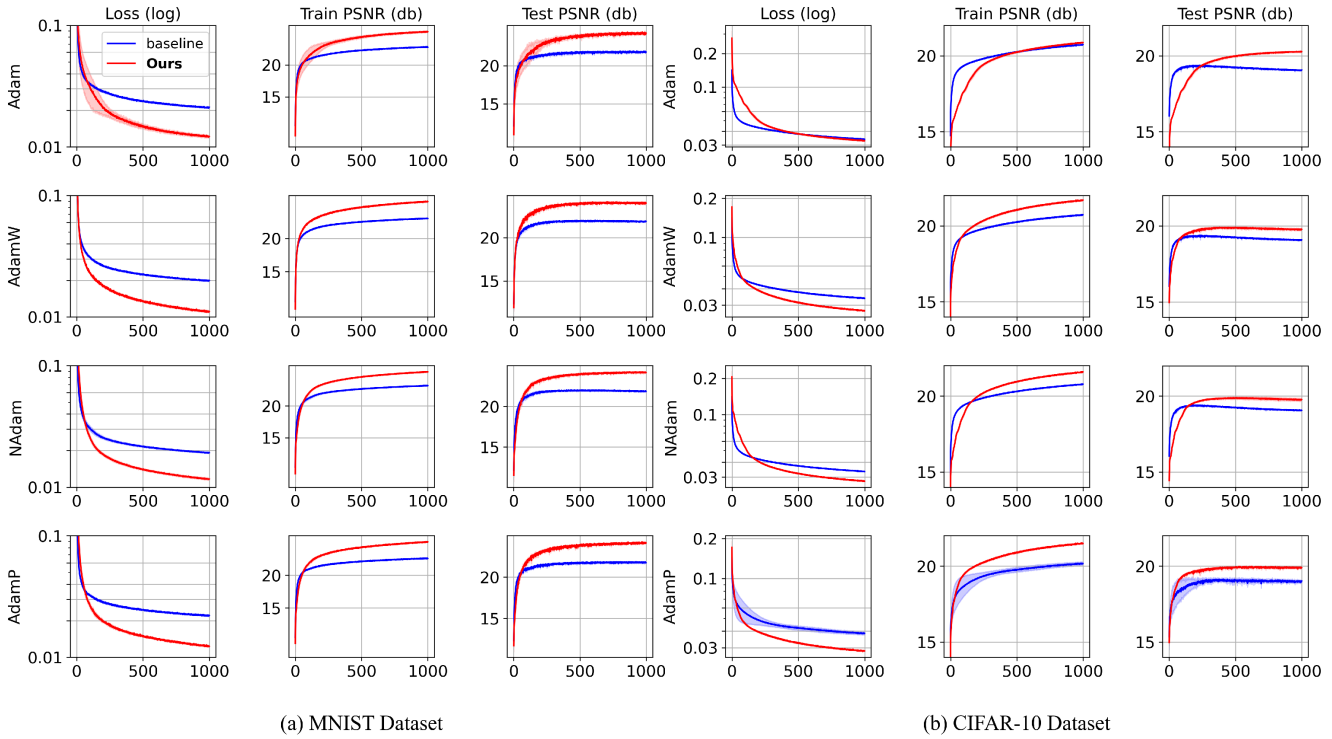


FIGURE 2. Empirical comparison of the proposed method (red line) with baseline (blue line) in VAE. (1st, 2nd, and 3rd columns in each dataset part) The training loss, training PSNR (db), and test PSNR (db) curves across (x-axis) iteration for (a) MNIST and (b) CIFAR-10, optimized using (1st, 2nd 3rd, and 4th rows) Adam, AdamW, NAdam, and AdamP respectively.

and $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.0001$ for the CIFAR-10 dataset.

AdamP [19]: When introducing momentum to gradient descent-based optimizers, there can be a problem of reduced update steps for scale-invariant weights. AdamP adjusts the update steps without significantly changing the convergence direction by removing the radial components or the norm-increasing direction. In the case of image classification, we used $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.001$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.01$, and $\gamma = 0.01$ for the CIFAR-10 dataset. For VAE, we used $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.0001$ for the MNIST dataset, and $\alpha = 0.0001$, $\beta = 0.1$, and $\gamma = 0.001$ for the CIFAR-10 dataset.

To compare the performance, for each optimizer and task, we have a baseline that uses only the base optimizer. We compare the results of the baseline and our proposed method, where the base optimizer is used with our approach applied. When comparing the objective values, we only consider the values of the objective function used to train the baseline. The hyperparameters for all optimizers and tasks' baselines are selected based on the experiment that yielded the highest performance.

B. EXPERIMENTAL RESULTS

1) BINARY CLASSIFICATION

In the binary classification experiment, we train our proposed method and the baseline using the given 2-dimensional data (Fig. 3a). We then compare the convergence and

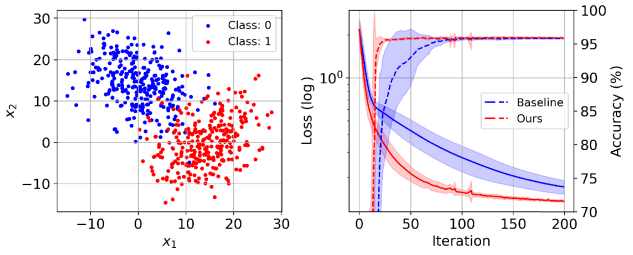


FIGURE 3. Experiment on binary classification. (a) The distribution of data, (b) (y-axis) the training loss (solid lines) and accuracy curve (dotted lines) over (x-axis) iteration.

convergence points of the loss values and accuracy graphs between the two methods to analyze the differences between our proposed method and the baseline. Fig. 3b is the graph of the objective values and accuracy for both methods during training, and it shows that the proposed method converges faster than the baseline. Hence, it can be inferred that updating the bottleneck feature in the middle after updating the backward network and utilizing it in the forward network update in the proposed method had a positive impact on the optimization of (1) in the IB theory.

2) IMAGE CLASSIFICATION

The linear regression task is a very simple problem. To apply our method to more complex problems, we trained our proposed method and the baseline on the image classification

TABLE 2. Training loss, training, and testing evaluation values for all image classification and VAE experiments using the trained models. The left table is the results of image classification training, and the right table is the results of VAE training. Each value is comprised of the mean (on the left) and the standard deviation (on the right).

Image classification			Train loss	Train Accuracy (%)	Test Accuracy (%)
MNIST	Baseline	Adam	0.16 ± 0.01	94.76 ± 0.03	97.74 ± 0.04
		AdamW	0.16 ± 0.01	94.77 ± 0.03	97.69 ± 0.10
		NAdam	0.11 ± 0.01	$\mathbf{96.33 \pm 0.09}$	$\mathbf{98.39 \pm 0.16}$
		AdamP	0.11 ± 0.01	$\mathbf{96.35 \pm 0.11}$	$\mathbf{98.35 \pm 0.14}$
	Ours	Adam	0.13 ± 0.01	$\mathbf{95.78 \pm 0.07}$	$\mathbf{98.12 \pm 0.02}$
		AdamW	0.13 ± 0.01	$\mathbf{95.94 \pm 0.08}$	$\mathbf{98.24 \pm 0.09}$
		NAdam	0.13 ± 0.01	95.91 ± 0.05	97.96 ± 0.16
		AdamP	0.14 ± 0.01	95.66 ± 0.04	98.16 ± 0.13
CIFAR-10	Baseline	Adam	0.64 ± 0.01	$\mathbf{76.99 \pm 0.28}$	68.95 ± 0.95
		AdamW	0.72 ± 0.01	$\mathbf{74.78 \pm 0.15}$	70.72 ± 0.42
		NAdam	0.64 ± 0.01	$\mathbf{76.97 \pm 0.20}$	69.20 ± 0.42
		AdamP	0.71 ± 0.01	$\mathbf{74.90 \pm 0.32}$	71.00 ± 0.52
	Ours	Adam	0.67 ± 0.01	76.62 ± 0.42	$\mathbf{69.69 \pm 0.51}$
		AdamW	0.74 ± 0.01	74.20 ± 0.21	$\mathbf{72.18 \pm 0.58}$
		NAdam	0.67 ± 0.01	76.72 ± 0.43	$\mathbf{70.42 \pm 0.86}$
		AdamP	0.74 ± 0.01	74.13 ± 0.09	$\mathbf{71.81 \pm 0.98}$

Denoising task using VAE			Train loss	Train PSNR (db)	Test PSNR (db)
MNIST	Baseline	Adam	0.02 ± 0.001	22.76 ± 0.03	21.78 ± 0.06
		AdamW	0.02 ± 0.001	23.03 ± 0.03	21.88 ± 0.05
		NAdam	0.02 ± 0.001	23.19 ± 0.03	21.87 ± 0.02
		AdamP	0.02 ± 0.001	22.58 ± 0.02	21.74 ± 0.08
	Ours	Adam	0.01 ± 0.001	$\mathbf{25.17 \pm 0.11}$	$\mathbf{24.24 \pm 0.08}$
		AdamW	0.01 ± 0.001	$\mathbf{25.58 \pm 0.04}$	$\mathbf{23.95 \pm 0.06}$
		NAdam	0.01 ± 0.001	$\mathbf{25.36 \pm 0.08}$	$\mathbf{24.19 \pm 0.12}$
		AdamP	0.01 ± 0.001	$\mathbf{25.12 \pm 0.02}$	$\mathbf{24.11 \pm 0.05}$
CIFAR-10	Baseline	Adam	0.03 ± 0.001	20.73 ± 0.02	19.06 ± 0.03
		AdamW	0.03 ± 0.001	20.73 ± 0.02	19.08 ± 0.03
		NAdam	0.03 ± 0.001	20.78 ± 0.01	19.06 ± 0.01
		AdamP	0.04 ± 0.001	20.17 ± 0.10	19.01 ± 0.10
	Ours	Adam	0.03 ± 0.001	$\mathbf{20.87 \pm 0.04}$	$\mathbf{20.27 \pm 0.04}$
		AdamW	0.03 ± 0.001	$\mathbf{21.69 \pm 0.03}$	$\mathbf{19.77 \pm 0.07}$
		NAdam	0.03 ± 0.001	$\mathbf{21.59 \pm 0.03}$	$\mathbf{19.76 \pm 0.09}$
		AdamP	0.03 ± 0.001	$\mathbf{21.49 \pm 0.02}$	$\mathbf{20.00 \pm 0.04}$

tasks of MNIST and CIFAR-10 datasets using four different optimizers. We then compared the convergence and convergence points of the objective values, training, and test accuracy graphs between the two methods. This analysis aims to show the differences between the proposed method and the baseline. Fig. 1a shows the training objective value, training accuracy, and test accuracy graphs for the two methods during training on MNIST. In the cases of Adam and AdamW, the proposed method converges faster than the baseline, and it also converges to a higher accuracy point. However, in the cases of NAdam and AdamP, the proposed method does not exhibit better performance than the baseline. Fig. 1b shows the training objective values, training accuracy, and test accuracy graphs for the two methods during training on CIFAR-10. The training objective values and training accuracy of the proposed method converge faster than the baseline, and we can also observe that the test accuracy of the proposed method converges to the maximum value faster than the baseline. In other words, when applying the proposed method to most optimizers in image classification that include convolutional layers, the addition of bottleneck updates between the encoder and decoder, through mutual information control, is presumed to have a positive impact on the optimization of (1). Specific numerical results for the image classification experiments can be found in Table 2.

3) VARIATIONAL AUTOENCODER

The autoencoder consists of an encoder and a decoder, making it easy to define the bottleneck between them and apply the proposed method. In the VAE experiment, we trained our proposed method and the baseline using four different optimizers on a task where Gaussian noise is added to images of specific classes from the MNIST and CIFAR-10 dataset and then reconstructed without noise. We then compared the convergence and convergence points of the objective values and the training and testing PSNR graphs between the two methods to analyze the differences between the proposed method and the baseline. Fig. 2 shows the training objective values, training PSNR, and test PSNR graphs for the two methods during training on MNIST and CIFAR-10. For the training objective values, in the early stages of training, the baseline converges faster than the proposed method. However, ultimately, for all optimizers, the proposed method converges to a lower value for the objective. In the case of PSNR, compared to the baseline, it can be observed that the proposed method converges better to a higher peak for all optimizers. In other words, applying the proposed method to VAE and adding the process of updating μ and σ seems to help with mutual information control, which likely has a positive impact on the optimization of (1). Specific numerical results for the VAE experiments can be found in Table 2.

C. ABLATION STUDIES

1) UPDATE METHODS OF THE FORWARD NETWORK

We examine the two updating methods: first-order gradient and second-order gradient approaches, as considered in Section III-A. The second-order gradient approach is our proposed method in the end. To compare the differences between the two methods, we apply the first-order gradient approach of (4) and the second-order gradient approach of (5) to the VAE task separately and train them. Then, we compare the results of the two methods. Fig. 4a is a graph comparing the loss values during training for the first-order gradient approach, second-order approach, and baseline. The first-order gradient approach converges similarly to the baseline and eventually converges to a larger value than the baseline. On the other hand, the second-order gradient approach converges to a lower value than the baseline.

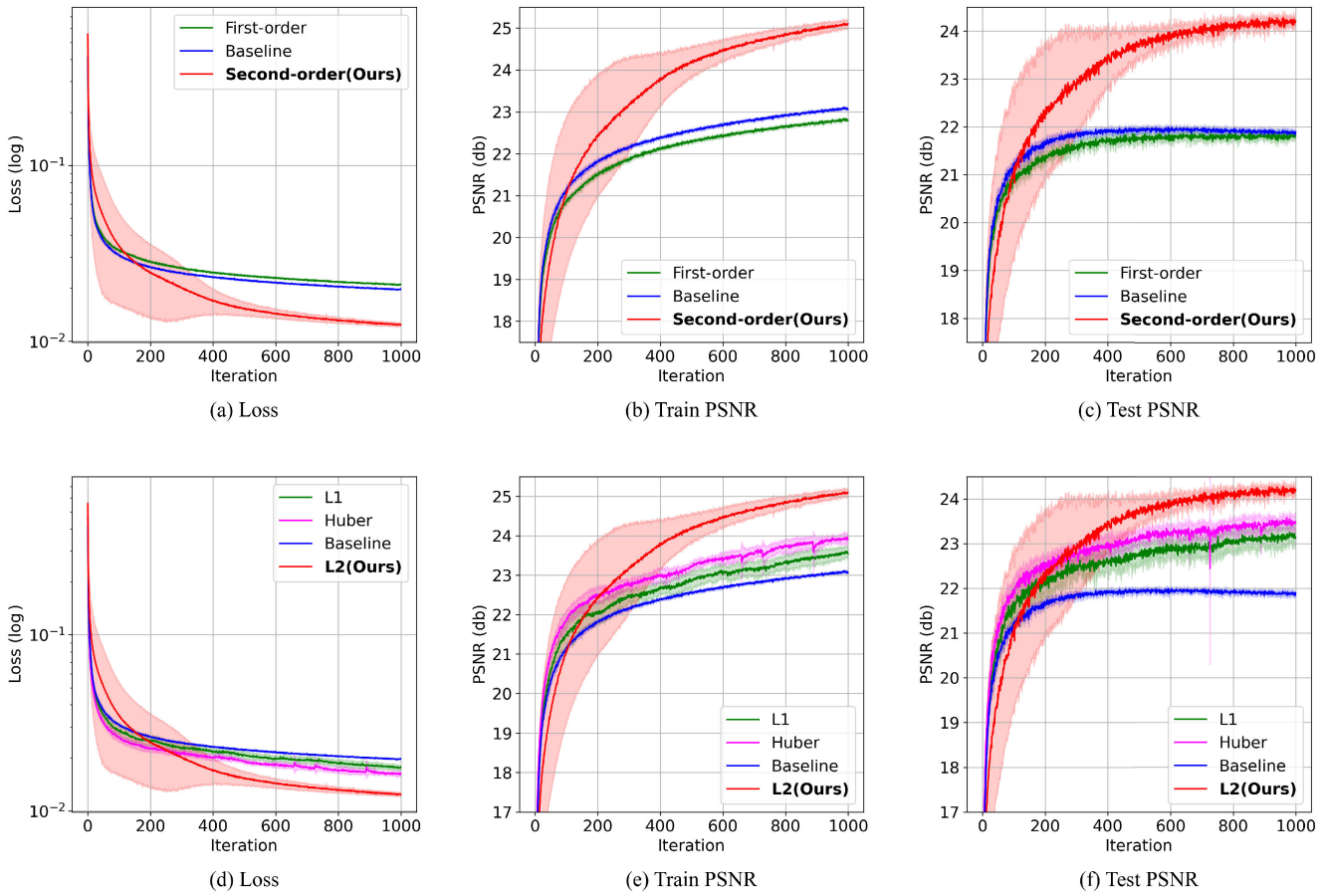


FIGURE 4. Experiments on update methods of $f(\cdot;\theta)$ and the constraint of proximal operator. (a) (y-axis) the training loss curve over (x-axis) iteration, (b) the training PSNR curve, and (c) the testing PSNR curve in update methods of $f(\cdot;\theta)$. (d) the training loss curve, (e) the training PSNR curve, and (f) the testing PSNR curve in the constraint of the proximal operator.

Fig. 4b and Fig. 4c compare the training and testing PSNR values of the three methods during training. The second-order gradient approach converges to higher PSNR values than the baseline, while the first-order gradient approach converges to slightly lower PSNR values than the baseline. This suggests that the first-order gradient approach may not optimize (1) as desired.

2) THE CONSTRAINTS OF PROXIMAL OPERATOR

As in (2), the conventional proximal operator adds the constraint as an L2 loss to the objective function. Therefore, when expressing the update equation as a proximal operator, as in (3), we used the L2 loss directly. To analyze the impact of different constraint losses, we applied the update algorithm to VAE using not only the proposed L2 loss but also L1 loss and Huber loss [34]. Then, we compared the results of each method to see how the different constraint losses affect the performance. In the case of L1 and Huber losses, when trained with the same γ value as L2, the loss values diverged. Therefore, to mitigate this issue, we adjusted the learning process with $\gamma = 0.01$. Fig. 4d is a graph comparing the loss values during training for L2, L1, Huber, and the baseline.

TABLE 3. Training and testing PSNR for all VAE Experiments including Ablations using the trained models. Each value is comprised of the mean (on the left) and the standard deviation (on the right).

		Train PSNR (db)	Test PSNR (db)
Baseline		23.08 ± 0.04	21.87 ± 0.06
First-order	L2	22.81 ± 0.04	21.81 ± 0.08
Second-order	L1	23.58 ± 0.15	23.18 ± 0.21
	Huber	23.93 ± 0.14	23.47 ± 0.17
	L2(Ours)	$\mathbf{25.09 \pm 1.10}$	$\mathbf{24.19 \pm 0.12}$

We have observed that the graphs for L1 and Huber converge to lower values than the baseline. Fig. 4e and Fig. 4f compare the training and testing PSNR values of the four methods during training. Both L1 and Huber show similar converging patterns to the baseline and converge to lower PSNR values than L2. Considering that they were trained with a smaller γ, it appears that L1 and Huber may also have potential like L2, but their stability seems to be more sensitive to the learning rate, indicating that L2 performs better in terms of stability. We can see the complete results of the ablation experiments in Table 3.

V. CONCLUSION

We introduced a method that adds bottleneck feature update to the existing neural network's SGD-based update algorithm, inspired by proximal operators. Additionally, we applied the proposed method to three tasks: binary classification, image classification, and VAE. We analyzed whether the proposed method improves performance and how it influences mutual information optimization in the IB theory. In practice, we have observed that the proposed method showed better performance than the existing methods in binary classification and image classification tasks. Particularly, when applied to VAE, the proposed method yielded much higher PSNR results compared to the existing method. Through these findings, we can infer that the proposed method provides more assistance to mutual information optimization compared to the existing methods.

However, there are also many areas for improvement in our work. In terms of performance metrics, the proposed method outperforms the existing methods. But it is not possible to directly examine the mutual information changes between the input, output, and bottleneck in the IB theory. Therefore, for more precise analysis, it is necessary to estimate the actual mutual information using mutual information estimation methods [25], [26], [27]. Furthermore, our study tested the proposed method on the most fundamental network architecture to verify its performance. However, it is also necessary to check the generalization performance on complex, state-of-the-art networks like residual networks [35], inception networks [14], Transformer [36], and others. Lastly, in the current proposed method, we are using a total of three learning rate hyperparameters: α, β, and γ. This makes the optimization of training more challenging, so it is worth considering the application of hyperparameter estimation methods that utilize suitable approaches such as Bayesian optimization [37], [38] for optimizing hyperparameters.

In future research, it will be possible to study the proposed method in combination with various fields. Beyond the three tasks applied in this paper, it would be beneficial to investigate whether there is a performance improvement when applying the proposed method to other tasks such as recent tasks like image segmentation [39], [40], [41], super-resolution [42], [43], [44], and generative models [45], [46], which can also have their networks separated into front and back layers. Considering the generalization of our proposed method to deep unfolding networks for super-resolution [47], which are more complex but structurally similar to the denoising VAE experimented with in our study, would be particularly worthwhile. Additionally, it would be necessary to investigate whether the proposed method improves performance when applied to various adaptive SGD-based methods [48] beyond the Adam-based algorithms [7], [18], [19], [20] experimented with in our study. Furthermore, when dealing with constrained environments involving multiple devices, such as in the case of federated learning [49], [50], where a certain amount of learning must be achieved across several devices, applying the proposed method could facilitate efficient learning. Particularly, in the context of decentralized federated averaging [51], where only neighboring devices communicate and synchronizing the training times of each device is crucial, the proposed method could enhance efficiency. Finally, in temporal difference learning in reinforcement learning, the process of computing gradients based on states and performing updates is similar to traditional SGD-based methods. Therefore, defining bottlenecks and applying the proposed method is feasible. This approach could also be applied to adaptive temporal difference methods [52], inspired by adaptive SGD-based techniques.

REFERENCES

[1] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Mach. Learn. With Appl.*, vol. 6, Dec. 2021, Art. no. 100134.
[2] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021.
[3] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Mach. Learn. Knowl. Extraction*, vol. 4, no. 1, pp. 276–315, Mar. 2022.
[4] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia Tools Appl.*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023.
[5] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.
[6] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, 2011.
[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[8] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," 2000, *arXiv:physics/0004057*.
[9] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Nov. 2014.
[10] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017, *arXiv:1703.00810*.
[11] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, Paris France. Cham, Switzerland: Springer, 2010, pp. 177–186.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[15] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, Jan. 1964.
[16] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate $O(\frac{1}{k^2})$," *Doklady Akademii Nauk*, vol. 269, no. 3, pp. 543–547, 1983.
[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
[18] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[19] B. Heo, S. Chun, S. Joon Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J.-W. Ha, "AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights," 2020, *arXiv:2006.08217*.
[20] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–4.

[21] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," 2019, *arXiv:1908.03265*.

[22] S. R. Dubey, S. H. S. Basha, S. K. Singh, and B. B. Chaudhuri, "AdaInject: Injection based adaptive gradient descent optimizers for convolutional neural networks," *IEEE Trans. Artif. Intell.*, 2022.

[23] Z. Xie, L. Yuan, Z. Zhu, and M. Sugiyama, "Positive-negative momentum: Manipulating stochastic gradient noise to improve generalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11448–11458.

[24] X. Xie, P. Zhou, H. Li, Z. Lin, and S. Yan, "Adan: Adaptive Nesterov momentum algorithm for faster optimizing deep models," 2022, *arXiv:2208.06677*.

[25] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "MINE: Mutual information neural estimation," 2018, *arXiv:1801.04062*.

[26] Z. Goldfeld, E. van den Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating information flow in deep neural networks," 2018, *arXiv:1810.05728*.

[27] S. S. Lorenzen, C. Igel, and M. Nielsen, "Information bottleneck: Exact analysis of (quantized) neural networks," 2021, *arXiv:2106.12912*.

[28] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," 2016, *arXiv:1612.00410*.

[29] A. Achille and S. Soatto, "Information dropout: Learning optimal representations through noisy computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2897–2905, Dec. 2018.

[30] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *J. Stat. Mech., Theory Exp.*, vol. 2019, no. 12, Dec. 2019, Art. no. 124020.

[31] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[33] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

[34] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 492–518.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[37] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1437–1446.

[38] A. Klein, L. C. Tiao, T. Lienart, C. Archambeau, and M. Seeger, "Model-based asynchronous hyperparameter and neural architecture search," 2020, *arXiv:2003.10865*.

[39] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 801–818.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.

[41] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.

[42] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[43] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 136–144.

[44] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2672–2680.

[46] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[47] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3217–3226.

[48] T. Sun, L. Qiao, Q. Liao, and D. Li, "Novel convergence results of adaptive stochastic gradient descents," *IEEE Trans. Image Process.*, vol. 30, pp. 1044–1056, 2021.

[49] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

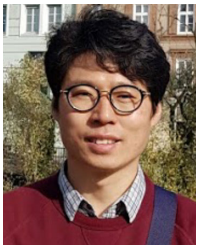
[50] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[51] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4289–4301, Apr. 2023.

[52] T. Sun, H. Shen, T. Chen, and D. Li, "Adaptive temporal difference learning with linear function approximation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8812–8824, Dec. 2022.

SANG-IL HAN received the bachelor's degree in software engineering from Chung-Ang University, in 2021, and the master's degree from the Image Laboratory, AI Department, Chung-Ang University, in 2022. He studied basic deep learning techniques and several generative models. His main research interests include generative models, optimization, and IB theory.

KENSUKE NAKAMURA received the M.Sc. and D.Phil. degrees in engineering from the Institute of Technology, Kyoto Institute of Technology, Japan, in 2004 and 2012, respectively. He is a Research Associate with Chung-Ang University. He has participated in research on models and applications of the 2-D/3-D human body shapes. His current research interests include computer vision and a support system for fashion design.



BYUNG-WOO HONG received the M.Sc. degree in computer vision from the Weizmann Institute of Science, in 2001, with Prof. Shimon Ullman, and the D.Phil. degree in computer vision from the University of Oxford, in 2005, with Prof. Michael Brady. In 2008, he joined the Computer Science Department, Chung-Ang University, South Korea, as a Faculty Member, after his postdoctoral research with the Computer Science Department, University of California, Los Angeles, with Prof. Stefano Soatto. He is interested in image processing, computer vision, machine learning, and medical image analysis.