Supplementary Material

A Model architecture

We elaborate on the network structure of CNN-13 [18] that we employ for a classifier, as well as of the basis learning autoencoder that we use for graph construction (also see *Remark* 1).

Layers	**Hyperparameters**
Conv1a	128 filters, 3×3 same padding
Conv1b	128 filters, 3×3 same padding
Conv1c	128 filters, 3×3 same padding
Pooling	Max pooling 2×2 kernel
Dropout	$p = 0.5$
Conv2a	256 filters, 3×3 same padding
Conv2b	256 filters, 3×3 same padding
Conv2c	256 filters, 3×3 same padding
Pooling	Max pooling 2×2 kernel
Dropout	$p = 0.5$
Conv3a	512 filters, 3×3 valid padding
Conv3b	256 filters, 3×3 same padding
Conv3c	128 filters, 3×3 same padding
Pooling	Global average pooling
Softmax	Fully-connected $128 \rightarrow 10$

Table 7: The network structure of CNN-13 [18] that we employ for a classifier. Batch normalization and leaky ReLU activation are applied for all hidden layers, although the description of such operations is omitted here for simple illustration.

Table 7 shows all the details of the layer type and hyperparameters that we adopt under CNN-13. As for perturbations that we apply to the student and teacher models, we consider the identical yet independent Gaussian distribution: $\xi \sim N(0, 0.15)$ and $\xi' \sim N(0, 0.15)$. For each hidden layer, we apply mean-only batch normalization [31] followed by leaky ReLU activation with parameter $\alpha = 0.1$. Our autoencoder is illustrated in Table 8. For MNIST, we combine the last conv layer feature of the teacher model (of size 128) with one-hot-coded label (of size 10) to construct an input vector of size 138. For CIFAR-10 and SVHN, we employ such features both from teacher and student models to construct a 266-dimensional input vector. As mentioned in *Remark* 1, we employ a special structure taking a non-linear (multi-layered) encoder followed by a linear (single-layer) decoder. We use leaky ReLU ($\alpha = 0.1$) for each hidden layer while taking tanh activation in the output layer. Notice that we do not apply any activation function for decoder for the purpose of encouraging it to linearly combine encoder vectors.

MNIST

Layers	Hyperparameters
Input	138×1 vector
Encoder	Layer1: Fully-connected, $138 \to 300$ leaky ReLU ($\alpha = 0.1$) Layer2: Fully-connected, $300 \to 300$ leaky ReLU ($\alpha = 0.1$) Layer3: Fully-connected, $300 \to 15$ tanh
Decoder	Fully-connected, $15 \to 138$

SVHN, CIFAR-10

Layers	Hyperparameters
Input	266×1 vector
Encoder	Layer 1: Fully-connected, $266 \to 500$ leaky ReLU ($\alpha = 0.1$) Layer 2: Fully-connected, $500 \to 100$ tanh
Decoder	Fully-connected, $100 \to 266$

Table 8: The network structure of the basis learning autoencoder that we use for graph construction.

B Basis learning via autoencoders with a linear decoder

While a mathematical analysis on our basis learning is not done yet, we have empirically shown that the bases of the data are indeed learned by the autoencoder which has a linear decoder. In what follows, we justify this observation via simulation utilizing synthetic data. To this end, we generate $n_1 \times n_2$ matrix $\mathbf{X}$, which has rank r by multiplying $n_1 \times r$ matrix and $n_2 \times r$ matrix consisting of Gaussian random variables. Basis learning can be confirmed by showing that

$$\text{col}(\mathbf{X}) = \text{col}(\mathbf{H}), \tag{11}$$

where $\mathbf{H}$ is the encoder's output, and $\text{col}(\mathbf{X})$ denotes the column space of $\mathbf{X}$. (11) can be proved by showing that

$$\text{col}(\mathbf{X}) \subseteq \text{col}(\mathbf{H}), \text{ and } \text{col}(\mathbf{X}) \supseteq \text{col}(\mathbf{H}). \tag{12}$$

The subspace relations in (12) can be proved by showing that

$$\|P_{\mathbf{X}}(\mathbf{h}_i)\|/\|\mathbf{h}_i\| = 1, \text{ and } \|P_{\mathbf{H}}(\mathbf{x}_i)\|/\|\mathbf{x}_i\| = 1 \tag{13}$$

for all i, where $\mathbf{x}_i$ is the i-th column of $\mathbf{X}$, and $P_{\mathbf{H}}(\mathbf{x})$ denotes the orthogonal projection of a vector $\mathbf{x}$ onto the space spanned by columns of $\mathbf{H}$. We identified

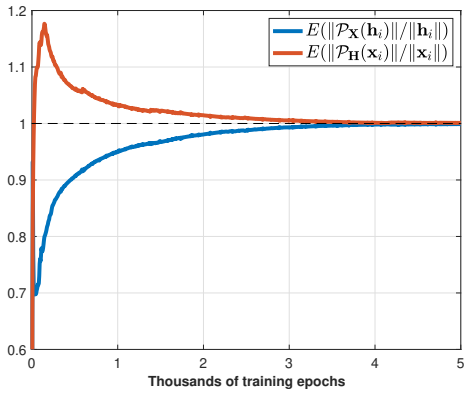

Fig. 3: Experimental results on synthetic data to show (13).

that (13) holds for all i on synthetic data in Figure 3. Furthermore, this type of structure is robust to the sparse inputs such as semi-supervised learning rather than a standard autoencoder.

C Experiment settings

We employ He initialization [14] and Xavier initialization [11] for the CNN classifier (θ) and autoencoder model (ϕ), respectively. We use Adam optimizer [16] with the following maximum learning rate: 0.0001 for MNIST; and 0.003 for SVHN and CIFAR-10. The number of epochs is 300. We apply the same ramp-up function $w(t)$ as in [24, 18, 37] (see (10)):

$$w(t) = \begin{cases} e^{-5(1-\frac{t}{80})^2} & t \leq 80 \\ 1 & 80 < t \leq 250 \\ e^{-12.5(1-\frac{300-t}{50})^2} & 250 < t \leq 300. \end{cases} \tag{14}$$

For the hyperparameter $\mu(t)$ that arises in the autoencoder loss $L_{AE}(\theta, \phi)$ in (8), we use the same functional with $w(t)$ except for the range of $t > 80$: $\mu(t) = 1$ in the range. As for input dropout that we apply to the autoencoder, we take the replacement probability of 0.9 (for MNIST) and 1 (for SVHN and CIFAR-10). The hyperparameter λ in (10) is searched over $\{0.2, 0.4, 0.6, 0.8, 1\}$. The margin in $L_g(\theta, \mathbf{W})$ is chosen from $\{0.5, 1\}$. Labeled and unlabeled data are randomly sampled while maintaining the constant ratio between them for every mini-batch; see below for details.

MNIST. Let the mini-batch size be $100 + \beta$ where $\beta \in \{20, 50, 100\}$ denotes the number of labeled data. The other 100 samples are randomly chosen from the entire dataset. Π model [18] is used for consistency loss.

Epochs	1	100	200	300
Proposed AE	19.51	98.78	99.08	99.38
Teacher model [24]	9.23	98.69	99.06	99.18

Table 9: Estimated Graph accuracy (%) on MNIST training data with 100 labels.

SVHN. We sample 50 labeled and 50 unlabeled examples to form a mini-batch. MT [37] is used for consistency loss.
CIFAR-10. Mini-batch size is 200. Each mini-batch consists of 100 labeled and 100 unlabeled samples. ICT [38] is used for consistency loss.
Programming. Our code is based on Python and TensorFlow. Our code and model are available at https://github.com/minkang23/BAE-GC.

D Additional experiment

Tables 9 shows the estimated graph accuracy on training data with the increase of epochs to address proposed method more convincing. Empirically, graph accuracy of basis learning autoencoder is indeed better than that of the teacher classifier [24]. We find that the matrix completion-based approach yields a more accurate graph. We expect this is because matrix completion simultaneously considers all the feature vectors when predicting labels, while a generic classifier makes an individual-sample-wise prediction. In particular, we see a large accuracy gap in the early epoch stage. We believe this was the key that leads our AE-based approach to yield a better performance. This suggests that it is more effective to predict labels with low-dimensional basis vectors, which are encoder outputs of basis learning autoencoder, than with weakly trained features in early epoch.