Training Process of Unsupervised Learning Architecture for Gravity Spy Dataset

Yusuke Sakai[1,+], Yousuke Itoh[2, 3], Piljong Jung[4], Keiko Kokeyama[5], Chihiro Kozakai[6], Katsuko T. Nakahira[7], Shoichi Oshino[8], Yutaka Shikano[9, 10, 11], Hirotaka Takahashi[1, 12, 13, ++], Takashi Uchiyama[8], Gen Ueshima[7], Tatsuki Washimi[6], Takahiro Yamamoto[8], and Takaaki Yokozawa[8]

[1]Research Center for Space Science, Advanced Research Laboratories, Tokyo City University, Setagaya-ku, Tokyo 158-0082, Japan
[2]Graduate School of Science, Osaka Metropolitan University, Sumiyoshi-ku, Osaka City, Osaka 558-8585, Japan
[3]Nambu Yoichiro Institute of Theoretical and Experimental Physics (NITEP), Osaka Metropolitan University, Sumiyoshi-ku, Osaka City, Osaka 558-8585, Japan
[4]National Institute for Mathematical Sciences, Daejeon 34047, Republic of Korea
[5]School of Physics and Astronomy, Cardiff University, The Parade, Cardiff, CF24 3AA United Kingdom
[6]Gravitational Wave Science Project, Kamioka branch, National Astronomical Observatory of Japan, Hida City, Gifu 506-1205, Japan
[7]Department of Information and Management Systems Engineering, Nagaoka University of Technology, Nagaoka, Niigata 940-2188, Japan
[8]Institute for Cosmic Ray Research, KAGRA Observatory, The University of Tokyo, Hida City, Gifu 506-1205, Japan
[9]Graduate School of Science and Technology, Gunma University, Maebashi, Gunma 371-8510, Japan
[10]Institute for Quantum Studies, Chapman University, Orange, CA 92866, USA
[11]JST PRESTO, Kawaguchi, Saitama 332-0012, Japan
[12]Institute for Cosmic Ray Research, The University of Tokyo, Kashiwa City, Chiba 277-8582, Japan
[13]Earthquake Research Institute, The University of Tokyo, Bunkyo-ku, Tokyo 113-0032, Japan
[+]g2191402@tcu.ac.jp
[++]hirotaka@tcu.ac.jp

Transient noise appearing in the data from gravitational-wave detectors frequently causes problems, such as instability of the detectors and overlapping or mimicking gravitational-wave signals. Because transient noise is considered to be associated with the environment and instrument, its classification would help to understand its origin and improve the detector's performance. In a previous study, an architecture for classifying transient noise using a time–frequency 2D image (spectrogram) is proposed, which uses unsupervised deep learning combined with variational autoencoder and invariant information clustering. The proposed unsupervised-learning architecture is applied to the Gravity Spy dataset, which consists of Advanced Laser Interferometer Gravitational-Wave Observatory (Advanced LIGO) transient noises with their associated metadata to discuss the potential for online or offline data analysis. In this study, focused on the Gravity Spy dataset, the training process of unsupervised-learning architecture of the previous study is examined and reported.

Keywords: deep learning, training process, hyperparameter tuning, classification, transient noise

1 Introduction

Advanced Laser Interferometer Gravitational-Wave Observatory (Advanced LIGO) detector [1] located at Livingston and Hanford, USA, made its first observation of gravitational waves from the coalescence of a binary black hole in September 2015 [2]. Following that, Advanced LIGO and Advanced Virgo [3] in Pisa (Italy) have made three international joint obser-vations and observed as many as 90 events of gravitational waves emitted by the coalescence of the compact binary [4–7]. KAGRA [8–13] in Japan will join the next (fourth) observing run (O4) by Advanced LIGO and Advanced Virgo.

In the data analysis of gravitational waves, the technique of separating the gravitational waves from the noise in the observed data is essential because the signals of the gravitational waves are generally smaller than the detector noise. Because the gravitational-wave detector is sensitive to environmental and instrumental conditions, such as ground motions, air pressure, optics suspensions, laser fluctuations, vacuum, and mirror, the non-stationary and non-Gaussian noise called "transient noise" frequently appears in the detector's data [14]. The transient noise causes the detector to be unstable, it can also hide and imitate gravitational-wave signals [15]. During O3b [7], LIGO Scientific Collaboration, Virgo Collabora-tion, and KAGRA Collaboration (LVK) reported that the transient noise rate with a signal-to-noise ratio (SNR) of > 6.5 was 1.17 events per minute at LIGO Livingston.

Machine learning is increasingly being applied in the study of transient noise [16–18]. Transient noise has various time-frequency characteristics that are related to its causes. Classifying transient noise could provide a clue to explore its ori-gins and improve the performance of the detector [19]. Thus, the Gravity Spy project [20–23] attempted to classify the transient noise. In the Gravity Spy project, the Omicron software [24] was used to identify the signal of transient noise observed in the time-series data. Following that, using the Omega Scan [25], a time-frequency spectrogram was created around the identified transient noise as a 2D image. Based on a portion of these created 2D images, 22 types of labels as-sociated with the characteristics or causes of transient noise were annotated for the analysis using cloud resources in col-laboration with LIGO detector characterization experts and volunteer citizen scientists. Both the images and labels were recorded. Finally, using the pre-classified images and labels, they classified the transient noise in the remaining images us-ing supervised learning.

Recently, there are some reports on the clustering and/or classification of transient noise using unsupervised learning [26–30]. Sakai et al. [26] discussed an architecture for the classification of transient noise using unsupervised learning, which combines a variational autoencoder (VAE) [31] [32] and invariant information clustering (IIC) [33]. The consistency be-tween the label annotated by the Gravity Spy project and the class provided by the proposed unsupervised-learning archi-tecture was confirmed using the Gravity Spy dataset of LIGO O1, and the potential for the classification of transient noise using unsupervised learning was discussed. Unsupervised learning is expected to reduce annotation work for training data, increase classification objectivity, and even classify a new class, such as the transient noise because it does not require any pre-assigned labels for the training dataset. Moreover, the training process of unsupervised learning is essential and some detailed reports on the training process can be found in refs. [34, 35]. In this study, focused on the Gravity Spy dataset, the training process of unsupervised-learning architecture of ref. [26] is examined and reported.

The remainder of this study is organized as follows: In Section 2, we explain the outline of VAE and IIC, which are used in the proposed architecture. In Section 3, we review the architecture, which was proposed in our previous research [26]. In Section 4, we report on how the training process of the unsupervised learning architecture was conducted. We also give a summary of the evaluation and obtained results discussed in [26]. Section 5 presents a summary of the work.

2 Outline of Used Method for Proposed Architecture

The proposed architecture discussed in Section 3 to classify transient noise is combined VAE [31] [32] and IIC [33], both of which are known as unsupervised-learning methods.

2.1 Variational Autoencoder

We introduce VAE [31] [32] that forms the feature learning [36] [37] part of our proposed architecture. VAE is a Bayesian inference method that uses a predictive distribution for the parameters and was designed for unsupervised learning. VAE has an architecture that compresses an image to the latent variables before reconstructing the original image from the latent variables. Let $\mathcal{X} \subset \mathbb{R}^D$ be the input space and $\mathcal{Z} \subset \mathbb{R}^J$ be the latent space, where D, $J \in \mathbb{Z}$ and $J < D$. Suppose a true (but unknown) probability distribution is the parameterized model $P_{\boldsymbol{\theta}}$ of the variable $\boldsymbol{x} \in \mathcal{X}$, and the latent variable $\boldsymbol{z} \in \mathcal{Z}$ is represented instead of the parameters $\boldsymbol{\theta}$, which are not directly observed. The marginal likelihood $P_{\boldsymbol{\theta}}(\boldsymbol{x})$, which we are interested in, has the following relation (Bayes' theorem), $P_{\boldsymbol{\theta}}(\boldsymbol{x}) = P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})P_{\boldsymbol{\theta}}(\boldsymbol{z})/P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. Then, let the likelihood $P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ be the generative model, and a prior distribution $P_{\boldsymbol{\theta}}(\boldsymbol{z})$ be hypothesized to be a Gaussian distribution. Unfortu-nately, $P_{\boldsymbol{\theta}}(\boldsymbol{x})$ is *intractable*; therefore, we cannot evaluate it directly. However, considering the posterior distribution as the inference model $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \sim Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ that is parameterized by $\boldsymbol{\phi}$, we can evaluate $P_{\boldsymbol{\theta}}(\boldsymbol{x})$ indirectly. Then, the logarithm marginal likelihood can be expressed as

$$\ln P_{\boldsymbol{\theta}}(X) = D_{\mathrm{KL}}\left(Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) + L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}), \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence of two distributions, and $L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ is known as the variational lower bound, respectively. The first term on the right-hand side of Equation (1) gradually approaches zero as the accuracy is increased. Therefore, maximizing the log-likelihood can be replaced by the problem of maximizing the variational lower bound as follows, $\arg\max_{\boldsymbol{\theta}} \ln P_{\boldsymbol{\theta}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$. Then, the equation for the variational lower bound is expressed as

$$L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) \sim -D_{\mathrm{KL}}\left(Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||P_{\boldsymbol{\theta}}(\boldsymbol{z})\right) + \int Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \ln P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}, \tag{2}$$

where $\mathcal{X} = \{\boldsymbol{x} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) | \boldsymbol{z} \in \mathcal{Z}\}$ represents the input space and a map $f_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$ is known as a decoder. Similarly, the latent space is also represented by $\mathcal{Z} = \{\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{X}\}$ and a map $g_{\boldsymbol{\phi}} : \mathcal{X} \to \mathcal{Z}$ is is known as an encoder. Suppose the two multivariate Gaussian distributions in the latent space are expressed as follows, $Q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = N(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\boldsymbol{I})$ and $P_{\boldsymbol{\theta}}(\boldsymbol{z}) = N(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}$ is the variance, and $\boldsymbol{I}$ is the identity matrix. Then, using *the reparameterization trick* [31], Equation (2) can be expressed as

$$L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{J}(1 + \ln\sigma_j^2 - \mu_j^2 - \sigma_j^2) + \frac{1}{L}\sum_{l=1}^{L}\ln P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(l)}), \tag{3}$$

where μ_j and σ_j are the mean and variance of J-dimensional Gaussian distribution, respectively, and L is the number of samples per datapoint. L can be set to 1 when the minibatch size is large enough [31].

2.2 Invariant Information Clustering

We briefly explain IIC [33] that forms the classification part of our proposed architecture. Although classification using supervised learning generally requires numerous labels for its training, IIC can classify without these labels and achieve results comparable to supervised learning. The goal of IIC is to learn what is in common between the paired data. These paired data, for example, could be different images with the same characteristics. Let $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ be paired data following a joint probability distribution $P(\boldsymbol{x}, \boldsymbol{x}')$ and $C \in \mathbb{Z}$ be the number of output classes. IIC learns a classifier (representation) $\boldsymbol{\Phi} : \mathcal{X} \to \mathbb{R}^C$ that maximizes the mutual information I expressed by entropy H as follows

$$I(\boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}')) = H(\Phi(\boldsymbol{x})) - H(\Phi(\boldsymbol{x})|\Phi(\boldsymbol{x}')). \tag{4}$$

Moreover, the maximum value of Equation (4) such that $\boldsymbol{x} = \boldsymbol{x}'$ can be analytically obtained as

$$\text{follows: } \max(I(\boldsymbol{\Phi}(\boldsymbol{x})) = \ln C. \tag{5}$$

Calculating a criterion for data within the same class and data between different classes is useful for evaluating the performance of the classifier. Thus, let $P_{ij} = \boldsymbol{\Phi}(\boldsymbol{x}^{(i)}) \cdot \boldsymbol{\Phi}(\boldsymbol{x}'^{(j)})^T$ be the conditional joint distribution and $P_i = \sum_j \boldsymbol{\Phi}(\boldsymbol{x}^{(i)}) \cdot \boldsymbol{\Phi}(\boldsymbol{x}'^{(j)})^T$ be the marginal distribution, where $\boldsymbol{x}^{(i)} \in \mathcal{X}$ are data belonging to the ith class, and a notation T means a transpose and "$\cdot$" means an inner product. Then, the mutual information Equation (4) can be expressed as

$$I(\boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}')) = \sum_{i}^{C}\sum_{j}^{C} P_{ij} \ln \frac{P_{ij}}{P_i P_j}. \tag{6}$$

The objective of training IIC is to construct the classifier Φ that maximizes Equation (6).

3 Outline of Proposed Architecture

We review an unsupervised-learning architecture proposed in ref. [26], which has deep convolutional neural networks for the classification of transient noise. The proposed architecture consists of two processes: feature learning using VAE and classification using IIC. We input four 2D images of transient noise with different time durations to the proposed architecture, and its shape is (4, 224, 224) data having four square images (224, 224). More details of the input 2D images are shown in Section 4.1. Regarding the implementation of VAE, we used the architecture, as shown in **Figure 1**. Each block shows the components of neural networks and their shape. The purpose of VAE is to learn the latent variables, which are the compressed features from a large dataset. We used four convolutional neural networks in our proposed architecture be-cause convolutional layers are known to be effective in image processing. Additionally, we used a ReLU activation function to avoid the vanishing gradient problem in deep neural networks and used a batch normalization technique in all convolu-tional layers to stabilize training. For the encoder part, the first convolution layer outputs a shape of (M, 64, 112, 112) from the input 2D image having a shape of (M, 4, 224, 224), where M is a minibatch size. A max-pooling layer next to the con-volutional layer extracts the features and compresses them into a shape of (M, 112, 56, 56). Following that, further features are extracted with three convolutional layers and are averaged through an average-pooling layer. A fully connected layer connects to all variables in neural networks. After using *reparameterization trick* [31], the latent variables $\boldsymbol{z}$ are obtained (in the case of Figure 1, its shape is (512)). For the decoder part, the decoder constructs an image element whose shape is $(M, 512, 14, 14)$ from the latent variables $\boldsymbol{z}$ through a fully connected layer and an upsampling nearest layer. Because 2D nearest-neighbor upsampling simply doubles the shape of the input, this layer is combined with a convolutional layer to generate an image. The decoder has four upsampling and convolutional layers. Following that, a $(M, 4, 224, 224)$ image with the same shape as the input was obtained.

IIC for the classification of transient noise uses VAE's pre-trained encoder as shown in **Figure 2**. Because the features of transient noise have already been learned by the VAE's pre-trained encoder, the IIC architecture was simply configured

with VAE's pre-trained encoder and a fully connected layer. Additionally, overclustering [33] was applied to IIC to improve the performance of the architecture. Regarding the two inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ to IIC in training, we let $\boldsymbol{x}$ be the center-cropped 2D image of transient noise and $\boldsymbol{x}'$ be the perturbed 2D image (Section 4.1) in the time direction, whose shapes of images are the same as (4, 224, 224), and $g_{\text{train}} : \mathcal{X} \to \mathcal{Z}$ is the mapping of pre-trained encoder. Then, IIC trains to maximize the mutual information $I(\boldsymbol{\Phi}(g_{\text{train}}(\boldsymbol{x})), \boldsymbol{\Phi}(g_{\text{train}}(\boldsymbol{x}')))$ and overclustering $I(\boldsymbol{\Phi}_{\text{over}}(g_{\text{train}}(\boldsymbol{x})), \boldsymbol{\Phi}_{\text{over}}(g_{\text{train}}(\boldsymbol{x}')))$, where $\boldsymbol{\Phi} \in \mathbb{R}^C$ and $\boldsymbol{\Phi}_{\text{over}} \in \mathbb{R}^W$. We also used several classifiers (typically set as five) and backpropagated the ensemble average for this mutual information to reduce the initial value dependence of the neural networks. The details are shown in Section 4.3. Following IIC training, although we could obtain a classification result from one classifier, the softmax outputs of each classifier differ slightly due to the initial value of the neural networks. Therefore, this result should be averaged by multiple classifiers to obtain a uniform classification result. Regarding unsupervised learning, the classification labels are given ran-domly in training. For example, the first classifier classifies the data as class label "0," while the second classifier classifies the same data as class label "1." Therefore we could not calculate an ensemble average of multiple classification results. As an alternative, we used an approach that constructs one classification result from the features extracted by multiple clas-sification results using the spectral clustering [38]. Now, let $D \in \mathbb{Z}$ be the total number of datasets and $K \in \mathbb{Z}$ be the number of classifiers, $C \in \mathbb{Z}$ be the estimated number of classes, and $\mathcal{M}_{D\times C}(\mathbb{R})$ be the classification matrix output from one classifier. Based on the *hypermatrix* $\mathcal{H}_{D\times CK}(\mathbb{R})$, which is composed of the concatenated results of multiple classifiers, we calculated an affinity matrix $\mathcal{A}_{D\times D}(\mathbb{R})$ using the Gaussian similarity function $(A)_{i,j} = \exp(-||h_i - h_j||^2)$, where h_i is a row vector of the hypermatrix $\mathcal{H} = (h_1, \ldots, h_i, \ldots, h_D)^T$. After applying spectral clustering to the affinity matrix, we obtained the following new classification matrix: $\mathcal{M}_{D\times C}(\mathbb{R})$.

4 Training Process and Result

We report on the training process and results of our proposed architecture step by step.

4.1 Outline of Dataset

The target dataset is the Gravity Spy dataset of LIGO O1 [20] [21] which consists of 8535 transient noise images with 22 different labels (e.g., "Blip," "Power Line," and "Koi Fish"). All transient noise in the dataset has been selected with the SNR ≥ 7.5 by Omicron software [24]. Each transient noise image is represented as a time-frequency spectrogram 2D image. One transient noise is recorded in four time durations: 0.5, 1.0, 2.0, and 4.0 s. More details of the labels, the distribution of the dataset, and 2D images of the transient noise are explained in Figure 1 in ref. [26].
We also applied a preprocess that randomly shifts between ± 0-24 px from the center of the image to the transient noise 2D image. The 2D images of the transient noise in each time duration have the shape of 224 × 272 px. The preprocess-ing randomly shifts the time direction of the image between 0 and 24 px and crops it at 224 × 224 px (square image). The purpose of this preprocessing is to allow the proposed architecture to train transient noise features without relying on small-time shifts. Reducing the size of input images contributes to saving the training cost. More details of the preprocessing are shown in Figure 7 in ref. [26].
We used these four original and perturbed 2D images as the input data, as shown in Figures 1 and 2. More details of the dataset can be found in ref. [26].

4.2 Training of VAE

The objective of VAE training is to maximize the variational lower bound expressed by Equation (3). Regarding the log-likelihood expressed as the second term of Equation (3), the original study [31] proposed two equations: Bernoulli and Gaussian distributions. Moreover, we used the mean squared error (MSE) as a reconstruction error. The three equations are expressed as

$$\ln P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \begin{cases} \sum_i^D [x_i \ln y_i + (1 - x_i) \ln(1 - y_i)] & \text{(Bernoulli distribution)} \\ \sum_i^D (x_i - y_i)^2 & \text{(MSE)} \\ 1/2 \ln(2\pi) + \sum_i^D [\ln |\sigma_i| + (x_i - \mu_i)^2/(2\sigma_i^2)] & \text{(Gaussian distribution)} \end{cases}, \tag{7}$$

where y_i is each pixel of the reconstructed image, μ_i and σ_i are the mean and variance with respect to each pixel of x_i, respectively. Because the Gaussian distribution requires both μ_i and σ_i, we doubled the output layer of the decoder in Fig-ure 1. Generally, deep learning has hyperparameters for optimal training. We investigated the hyperparameters as shown in **Table 1** (top) and conducted an experiment using our proposed architecture as follows: the initial learning rate (used Adam [39], which is one of the stochastic gradient descent methods) is in the range of $[5 \times 10^{-7}, 5 \times 10^{-2}]$ in increments of one digit; the minibatch size is in the range of [32, 128] in increments of ≈ 32; the training size is in the range of [60%, 90%] in incre-ments of 10%; the dimensions of the latent variable $\boldsymbol{z}$ are 64, 128, 256, 512, and 1024; VAE has the evaluation phase every

Table 1: Hyperparameters on training of VAE (top) and IIC (bottom).

	Hyperparameter name	Range	Increment value
Training of VAE	Initial learning rate	$[5 \times 10^{-7}, 5 \times 10^{-2}]$	10^{-1}
	Mini-batch size	$[32, 128]$	≈ 32
	Training size	$[60\%, 90\%]$	10%
	Dimensions of $\boldsymbol{z}$	$64, 128, 256, 512, 1024$	-
Training of IIC	Initial learning rate	$[5 \times 10^{-7}, 5 \times 10^{-2}]$	10^{-1}
	Mini-batch size	$[64, 256]$	32
	Number of classes	$[22, 100]$	2
	Number of over clustering	$[50, 500]$	50
	Classifier number	$3, 5, 10, 20$	-

Table 2: Representative hyperparameters for the training process of the proposed architecture and its computational cost. All cases are adopted by Adam optimization [39].

		Initial learning rate	Batch size	Training size	Dimension of $\boldsymbol{z}$	Computational cost (500 epochs)
Bernoullie	Case1	5×10^{-5}	128	80%	512	5.0 h
	Case2	5×10^{-4}	128	80%	512	5.0 h
MSE	Case1	5×10^{-5}	64	60%	256	2.0 h
	Case2	5×10^{-4}	96	70%	512	3.5 h
Gaussian	Case1	5×10^{-4}	32	60%	128	1.8 h
	Case2	5×10^{-2}	82	80%	256	6.0 h

five epochs to quantify the performance of the model, and the size of the evaluation dataset is (1− Training size), respec-tively. In this study, the case of representative hyperparameters, as shown in **Table 2** was used to explain the training pro-cess.
The training curves using the Bernoulli distribution in Table 2 are shown in **Figure 3** (top). Because "Case 1" and "Case 2" of evaluation curves (plotted as dashed) are close to these training curves (plotted as solid), there is no overfitting in the training process. Although the training curve of "Case 1" (drawn in blue) progresses slowly, the training curve of "Case 2" (drawn in pink) sets a higher learning rate than "Case 1". It also progresses rapidly and is stabilized near 100 epochs. Considering the Bernoulli case, an optimizer with a small learning rate would be the cause of slow progress in training. The reconstructed images from the decoders of "Case 1" at 300 epochs and "Case 2" at 100 epochs are shown in Figure 3 (bottom). The "Case 1" image may not have been reconstructed appropriately. However, the "Case 2" image seems to ex-tract the feature of the input image appropriately. Therefore, "Case 2" is considered suitable in the case of the Bernoulli distribution.
The training curves using MSE in Table 2 are shown in **Figure 4** (top). The training curves converge in both cases, with the difference being that the "Case 2" curve (drawn in green) is close to zero, and the "Case 1" curve (drawn in red) is smaller than the "Case 2" curve. Therefore, we confirmed that "Case 2" is more optimal for the hyperparameters than "Case 1" in the MSE case. The reconstructed images of "Case 1" at 100 epochs and "Case 2" at 100 epochs are shown
in Figure 4 (Bottom). The "Case 1" image would only be roughly reconstructed because it is considered to be caused by insufficient optimization of the hyperparameters. The "Case 2" reconstructed image seems to be close to the input image, but this image is blurry. This is the reason why the decoder of "Case 2" reconstructs images along with background fea-tures. Therefore, the training curve might be close to zero.
The training curves using the Gaussian distribution are shown in **Figure 5** (top). The "Case 1" training curve (drawn in yellow) has a difference between the training and evaluation curves, and this training is considered overfitting. The "Case 2" training curve (drawn in black) converges at 150 epochs and the evaluation curve is close to the training curve. "Case 2" is stable. Therefore, "Case 2" is more appropriate than "Case 1" in the Gaussian case. The reconstructed images "Case 1" at 50 epochs and "Case 2" at 150 epochs, which are μ_i of the decoder output in Equation (7), is shown in Figure 5 (Bottom). Both images would be less clear than the input image. The reconstructed image is averaged because the mean μ_i of the Gaussian distribution was applied to the output.
We used two NVIDIA GeForce RTX 2080 Ti GPUs, Intel Xeon CPU E5-2637 v4 (eight cores), and with 125 GB of main memory. The computational cost of each training process is shown in Table 2. The computational cost increases as the batch size and training size increase. Bernoulli and MSE cases have almost the same computational cost because the neu-ral network layers are the same. However, the computational cost for the Gaussian case is higher than that for the Bernoulli and MSE cases. The reason for the higher computational cost in the Gaussian case is that the decoder optimizes the neu-ral network through the mean and variance of the output, which increases the neural network layers.
We also investigated the latent variables, which are outputs from the encoder. The auxiliary analysis is frequently ap-

plied to a training model to understand the learned features of the dataset. Latent variables are important for the representation of the dataset. However, because the latent space is generally the higher dimension, it is difficult to assess its utility. For this reason, visualization methods are frequently applied by embedding from a high-dimensional space to a lower-dimensional space. A typical approach is principal component analysis (PCA), which is a linear dimension reduction technique based on keeping the covariance of the data. Alternatively, the t-distributed stochastic neighbor embedding (t-SNE) [40] is a nonlinear technique that preserves the distance between the higher-dimensional and lower-dimensional features. PCA leads to a better understanding of the data's global structure. Alternately, t-SNE helps in the understanding of the local structure which is important for comprehending clustering visually. Therefore, we applied the t-SNE method for the visualization of the latent space. Let $p_{ij} \to \mathbb{R}$ be the joint distribution of the two latent variables $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathcal{Z}$ and $q_{ij} \to \mathbb{R}$ be the joint probability of the lower -dimensional variables $\boldsymbol{y}_i, \boldsymbol{y}_j \in \mathcal{Y}$, where $\mathcal{Y}$ is the lower -dimensional space. Suppose p_{ij} follows the Gaussian distribution and q_{ij} follows t-distribution, respectively, then, the t-SNE objective d is determined using the following equation

$$d = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{8}$$

Let $\mathcal{Y} \subset \mathbb{R}^3$ be the lower-dimensional space, and the learning parameters of t-SNE be set as below: perplexity, which is related to the number of nearest neighbors as 40, the maximum number of iterations for the optimization as 2000, and random seed as 10. We also used the Gravity Spy labels to visualize how the Gravity Spy dataset is clustered in the latent space. The t-SNE mapping was applied to each model: Bernoulli "Case 2" at 100 epochs, MSE "Case 2" at 100 epochs, and Gaussian "Case 2" at 150 epochs. The t-SNE results of the Bernoulli, MSE, and Gaussian cases are shown in **Figure 6** (top) , **Figure 7** (top), and **Figure 8** (top), respectively. Considering the Bernoulli case, the data in each class appear to be clustered in 3D space, and each cluster has a spatially expansive structure. However, considering the MSE case, the data are also clustered, and the difference is that the structure of each cluster seems to be smaller than that of the Bernoulli case. Similar aspects to the MSE were found in the Gaussian case.
Moreover, the t-SNE results are quantitatively examined using the silhouette coefficient. Let $\mathcal{C}_{\text{in}} \subset \mathcal{Y}$ be the intra-cluster dataset and $\mathcal{C}_{\text{near}} \subset \mathcal{Y}$ be the nearest-cluster dataset, where $\mathcal{C}_{\text{in}} \cap \mathcal{C}_{\text{near}} = \emptyset$. The mean intra-cluster distance of the ith data is defined as follows: $a^{(i)} = 1/(|\mathcal{C}_{\text{in}}| - 1) \sum_{\boldsymbol{y}^{(j)} \in \mathcal{C}_{\text{in}}} ||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||$, and the mean nearest-cluster distance of the ith data is defined as follows: $b^{(i)} = 1/|\mathcal{C}_{\text{near}}| \sum_{\boldsymbol{y}^{(j)} \in \mathcal{C}_{\text{near}}} ||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||$. The silhouette coefficient $s^{(i)}$ is defined as

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\left(b^{(i)}, a^{(i)}\right)}. \tag{9}$$

Note that the silhouette coefficient is a mapping $\mathcal{Y} \to [-1, 1]$ and it approaches 1 when the clustering is condensing and approaches 0 when the number of clusters is not appropriate. The negative value means that these data overlapped with each cluster.
The silhouette coefficients for the Bernoulli, MSE, and Gaussian cases are shown in Figure 6 (bottom), Figure 7 (bottom), and Figure 8 (bottom), respectively. The average values (drawn in red) for all classes are 0.22 in the Bernoulli case, 0.19 in the MSE case, and 0.19 in the Gaussian case. The average silhouette coefficient in the Bernoulli case is greater than that in the other two cases. The MSE and Gaussian cases overlapped with other classes because their clustering was more con-densed than the Bernoulli case. As a result, their silhouette coefficients decreased.
We summarize the results of VAE's training process. The training optimization was conducted in Equation (7) using three types of the reconstruction error. The Bernoulli "Case 2" decoder at 100 epochs was used to reconstruct the features of transient noise. We also confirmed the t-SNE results and silhouette coefficients. As the results show, it was also suggested that the Bernoulli "Case 2" encoder at 100 epochs was better for transient noise classification. Therefore, we used the Bernoulli "Case 2" encoder at 100 epochs as a pre-trained encoder for IIC.

4.3 Training of IIC

Following VAE training, IIC uses a pre-trained encoder to classify the transient noise 2D image. The objective of IIC train-ing is to maximize the mutual information expressed by Equation (6). The maximum mutual information expressed by Equation (5) increases monotonically with the number of classes. We divided Equation (6) by Equation (5), and obtained the normalized mutual information $I(\boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}'))/ \ln C$ for the number of classes C. Our proposed architecture maximizes the normalized mutual information in the training process. This normalized expression allows for mutual information com-parisons with the different numbers of classes.
The hyperparameters for IIC are also investigated as shown in Table 1 (bottom) and experimented using our proposed architecture as follows: the initial learning rate is in the range of $[5 \times 10^{-7}, 5 \times 10^{-2}]$ in increments of one digit (same as VAE training); the minibatch size is in the range of $[64, 256]$ in increments of 32; the number of classes is in the range of $[22, 100]$ in increments of 2; the number of over clustering is in the range of $[50, 500]$ in increments of 50; the classifier number is one of $3, 5, 10$, and 20.
Regarding the initial learning rate, when other hyperparameters are fixed, the normalized mutual information is not affected unless the value is far away, for example, 5×10^{-7} and 5×10^{-2}. Similar aspects to the number of overclustering,

the classifier number, and the minibatch size were found in the training process. Therefore, in this study, we set the initial learning rate to 5×10^{-4}, the number of classifiers to five, the number of over clustering to 250, and the minibatch size to 128. We investigated the range of [22, 100] to determine the number of classes that affect the normalized mutual informa-tion. Because the previous studies [22] [27] [28] presented new classes in addition to the 22 classes of the same dataset, the representative training curves for $C = 30, 40, 50$, and 100 are shown in **Figure 9**. Note that because the training curve was oscillating and covered by other training curves, we applied the exponential moving average to these training curves, and the average span was set to 10 epochs in Figure 9. All the cases converge and stabilize near 100 epochs, as shown in Figure 9. The training curves of over clustering were found to be independent of the number of classes C, and the normalized mutual information was high at $C = 30$ and $C = 40$. Therefore, from Figure 9, the number of classes C is considered to be suitable between 30 and 40 in our proposed architecture.

4.4 Correspondence to Result of Supervised Learning (Gravity Spy labels) and Summary of Obtained Result in Ref. [26]

After the IIC training discussed in Section 4.3, we investigated the correspondence between the results of supervised learn-ing (Gravity Spy labels) and our unsupervised learning. Although the details can be found in ref. [26], we briefly review the results. We used the accuracy in Equation (1) of ref. [26] defined as follows, $\sum_i^C \max(\boldsymbol{v}^{(i)})/|\sum_j^{C'} v^{(i)}|$, where C is the number of classes with unsupervised learning, $C' = 22$ is the number of classes with supervised learning (Gravity Spy la-bels), and $\boldsymbol{v}^{(i)}$ is the ith column in the confusion matrix (e.g., in Figure 4 of ref. [26]). By calculating the accuracy as a criterion, we determined the number of classes in unsupervised learning. Furthermore, we used the spectral clustering dis-cussed in Section 3 to compress the multiple results of classification into one result. We investigated the number of classes between 30 and 40, which yields high values for the normalized mutual information in Section 4.3. The results for the rep-resentative numbers $C =$ 30, 36, and 40 are shown in **Figure 10**. The dashed line represents the average accuracy from each classifier, which was set to five classifiers, and the solid line represents the accuracy using the spectral clustering for the results of five classifiers. In all cases, spectral clustering outperformed the average of the five classifiers, with the high-est accuracy of 90.9% achieved with the number of classes $C = 36$ at 200 epochs. Note that the Gravity Spy project [20] using supervised learning achieved 97.1% accuracy on the testing data using the same dataset used here. In terms of ac-curacy, the proposed architecture can classify with performance close to supervised learning, and also has the advantage that unsupervised learning does not require the data annotations. The proposed architecture shows the results of super-vised learning with a high level of consistency for classification. For example, the data of "1080Lines" were classified into one class for both supervised and unsupervised learning. Furthermore, our architecture indicates the existence of the un-revealed classes from the Gravity Spy dataset. For example, the "Blip" data that were classified as one class in supervised learning would be separated into more detailed subclasses in unsupervised learning.

We confirmed the consistency between the label annotated by the Gravity Spy project and the class provided by our pro-posed unsupervised-learning architecture and provided the potential for the existence of the unrevealed classes. More de-tails on the result and discussion of unsupervised classification are shown in ref. [26].

5 Summary

Transient noise appearing in the data from gravitational-wave detectors frequently causes problems. Because transient noise is considered to be associated with the environment and instrument, classifying it may help us understand its ori-gin and improve the detector's performance. In our previous study [26], an architecture of classifying the transient noise using a time-frequency 2D image (spectrogram), which uses the unsupervised deep learning combined with VAE and IIC was proposed. The training process of unsupervised learning is essential. In this study, using the Gravity Spy dataset, we reported on how the training process of unsupervised-learning architecture of ref. [26] was conducted. We used three types of objective functions in the training process for feature learning of transient noise using VAE. We investigated the three types of training curves, reconstructed images from the decoders, t-SNE mapping of the latent variables, and silhou-ette coefficient of t-SNE mapping. The normalized objective function was used in the training process using IIC for tran-sient noise classification, and the results were compared between the different classes to investigate the appropriate number of classes. We also confirmed that the classification result compressed from the multiple unsupervised classifications ap-proaches the accuracy in supervised learning.

A more detailed discussion of the results for the classification of transient noise using our proposed unsupervised architec-ture can be found in ref. [26].

We applied the unsupervised classification to the Gravity Spy dataset of LIGO O1 as a first step. In future work, we will apply our architecture to the recent observation run (O2 and O3) dataset. Furthermore, we will use our unsupervised clas-sification in KAGRA to develop a transient noise system. Additionally, we will extend our architecture to self-supervised learning [41] to improve classification accuracy, in which the architecture generates *pseudo labels* for a given dataset and re-trains it.

Acknowledgements
The authors grateful to the members of the Gravity Spy project for enlightening discussions. This study was supported in part by the Inter-University Research Program of the Institute for Cosmic Ray Research, University of Tokyo, Japan. It was also supported in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research on Innovative Areas, Grant No. 24103005 [JP17H06358, JP17H06361, and JP20H04731], by JSPS Core-to-Core Program A, Advanced Research Networks, and by JSPS KAKENHI [Grant No. 19H0190 (Y.I. and H.T.) and Nos. 19K14636 and 21H05599 (Y. Shikano.)], by JST, PRESTO [Grant No. JPMJPR20M4 (Y. Shikano.)].

Conflict of Interest
The authors declare no conflict of interest.

References

[1] LIGO Scientific Collaboration, *Classical and Quantum Gravity* **2015**, *32* 074001.

[2] LIGO Scientific Collaboration and Virgo Collaboration, *Phys. Rev. Lett.* **2016**, *116* 131103.

[3] Virgo Collaboration, *Classical and Quantum Gravity* **2015**, *32* 024001.

[4] LIGO Scientific Collaboration and Virgo Collaboration, *Phys. Rev. X* **2019**, *9* 031040.

[5] LIGO Scientific Collaboration and Virgo Collaboration, *Phys. Rev. X* **2021**, *11* 021053.

[6] LIGO Scientific Collaboration and Virgo Collaboration, *arXiv preprint arXiv:2108.01045* **2021**.

[7] LIGO Scientific Collaboration and Virgo Collaboration and KAGRA Collaboration, *arXiv preprint arXiv:2111.03606* **2021**.

[8] KAGRA Collaboration, *Nature Astronomy* **2019**, *3* 35–40.

[9] KAGRA Collaboration, *Progress of Theoretical and Experimental Physics* **2021**, *2021*, 5 05A101.

[10] KAGRA Collaboration, *Progress of Theoretical and Experimental Physics* **2021**, *2021*, 5 05A103.

[11] KAGRA Collaboration, *Progress of Theoretical and Experimental Physics* **2021**, *2021*, 5 05A102.

[12] KAGRA Collaboration, *arXiv preprint arXiv:2203.07011* **2022**.

[13] LIGO Scientific Collaboration and Virgo Collaboration and KAGRA Collaboration, *Progress of Theoretical and Experimental Physics* **2022**, *2022*, 6 063F01.

[14] P. Nguyen, *et al.*, *Classical and Quantum Gravity* **2021**, *38*, 14 145001.

[15] D. Davis, *et al.*, *Classical and Quantum Gravity* **2021**, *38*, 13 135014.

[16] R. Biswas, L. Blackburn, J. Cao, R. Essick, K. A. Hodge, E. Katsavounidis, K. Kim, Y.-M. Kim, E.-O. Le Bigot, C.-H. Lee, J. J. Oh, S. H. Oh, E. J. Son, Y. Tao, R. Vaulin, X. Wang, *Phys. Rev. D* **2013**, *88* 062003.

[17] J. Powell, D. Trifirò, E. Cuoco, I. S. Heng, M. Cavaglià, *Classical and Quantum Gravity* **2015**, *32*, 21 215012.

[18] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, N. S. Philip, *Phys. Rev. D* **2017**, *95* 104059.

[19] LIGO Scientific Collaboration, *Classical and Quantum Gravity* **2021**, *38*, 2 025016.

[20] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsaggelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. Østerlund, J. R. Smith, L. Trouille, V. Kalogera, *Classical and Quantum Gravity* **2017**, *34*, 6 064003.

[21] S. Bahaadini, V. Noroozi, N. Rohani, S. Coughlin, M. Zevin, J. Smith, V. Kalogera, A. Katsaggelos, *Information Sciences* **2018**, *444* 172.

[22] S. Soni, C. P. L. Berry, S. B. Coughlin, M. Harandi, C. B. Jackson, K. Crowston, C. Østerlund, O. Patane, A. K. Katsaggelos, L. Trouille, V.-G. Baranowski, W. F. Domainko, K. Kaminski, M. A. L. Rodriguez, U. Marciniak, P. Nauta, G. Niklasch, R. R. Rote, B. Téglás, C. Unsworth, C. Zhang, *Classical and Quantum Gravity* **2021**, *38*, 19 195016.

[23] S. Bahaadini, N. Rohani, S. Coughlin, M. Zevin, V. Kalogera, A. K. Katsaggelos, In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, IEEE, New Orleans, Louisiana, USA, **2017** 2931–2935.

[24] F. Robinet, N. Arnaud, N. Leroy, A. Lundgren, D. Macleod, J. McIver, *SoftwareX* **2020**, *12* 100620.

[25] S. Chatterji, L. Blackburn, G. Martin, E. Katsavounidis, *Classical and Quantum Gravity* **2004**, *21* S1809.

[26] Y. Sakai, Y. Itoh, P. Jung, K. Kokeyama, C. Kozakai, K. T. Nakahira, S. Oshino, Y. Shikano, H. Takahashi, T. Uchiyama, G. Ueshima, T. Washimi, T. Yamamoto, T. Yokozawa, *Scientific reports* **2022**, *12*, 1 1.

[27] D. George, H. Shen, E. A. Huerta, *Physical Review D* **2018**, *97*, 10 101501.

[28] S. Bahaadini, N. Rohani, A. K. Katsaggelos, V. Noroozi, S. Coughlin, M. Zevin, In *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, Athens, Greece, **2018** 748–752.

[29] S. Bini, Unsupervised classification of short transient noise to improve gravitational wave detection, `https://etd.adm.unipi.it/t/etd-08302020-184201/` (accessed: 23 March **2022**).

[30] Z. Ramezani, A. Pourdarvish, *Physica A: Statistical Mechanics and its Applications* **2021**, *561* 125273.

[31] D. P. Kingma, M. Welling, *arXiv preprint arXiv:1312.6114* **2013**.

[32] D. P. Kingma, M. Welling, *Foundations and Trends in Machine Learning* **2019**, *12* 307.

[33] X. Ji, A. Vedaldi, J. Henriques, In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October. IEEE, Seoul, Korea, ISSN 15505499, **2019** 9865–9874.

[34] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, G. E. Dahl, *arXiv preprint arXiv:1910.05446* **2019**.

[35] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht, In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, ISBN 9781510860964, **2017** 4151–4161.

[36] G. Zhong, L.-N. Wang, X. Ling, J. Dong, *The Journal of Finance and Data Science* **2016**, *2* 265.

[37] Y. Bengio, A. Courville, P. Vincent, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2013**, *35* 1798.

[38] U. von Luxburg, *Statistics and computing* **2007**, *17*, 4 395.

[39] D. P. Kingma, J. Ba, *arXiv preprint arXiv:1412.6980* **2014**.

[40] S. T. Roweis, L. K. Saul, *Science* **2000**, *290*, 5500 2323.

[41] D.-H. Lee, In *Workshop on challenges in representation learning, ICML*, volume 3. **2013** 896.

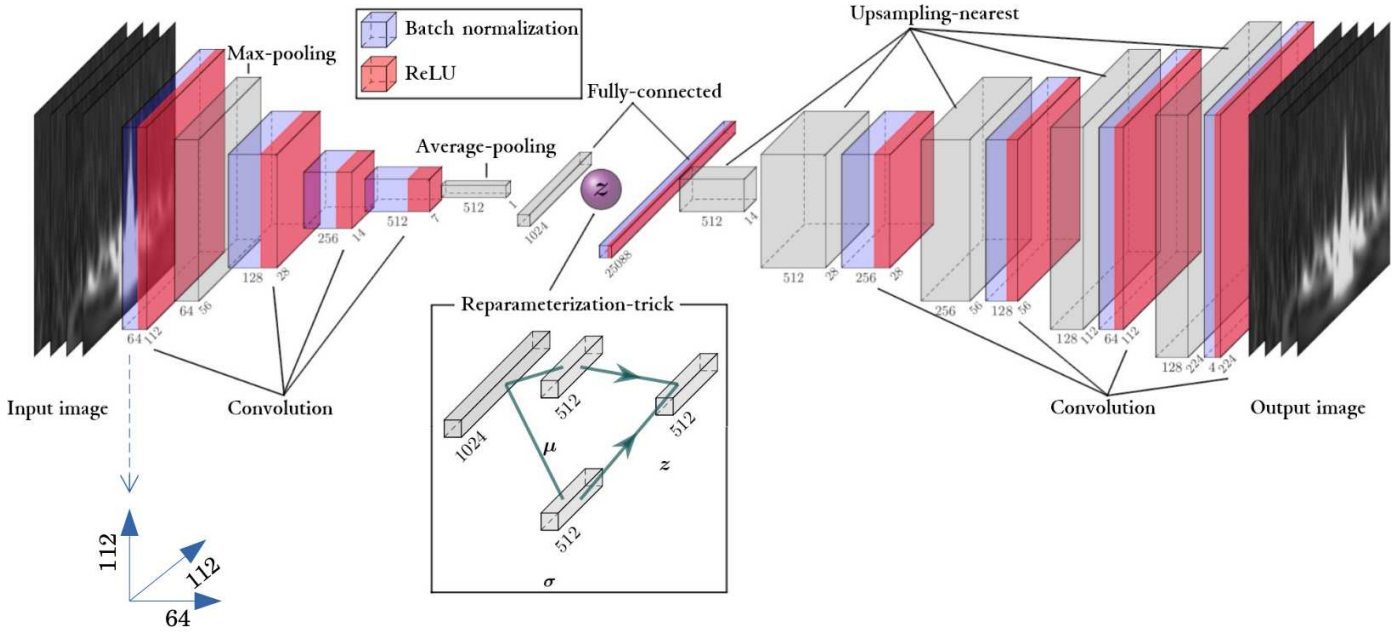

Figure 1: Schematic view of the proposed VAE architecture for training the latent variables from input images of transient noise. The encoder part is from the input layer (at the left end) to the latent variables $\boldsymbol{z}$ (at the center), and the decoder part is from $\boldsymbol{z}$ to the output layer (at the right end). The blue block represents batch normalization, whereas the red block represents the ReLU activation function. A solid line in reparameterization-trick blocks simply means splitting the component in half, and lines with an arrow mean the sampling from the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}$.

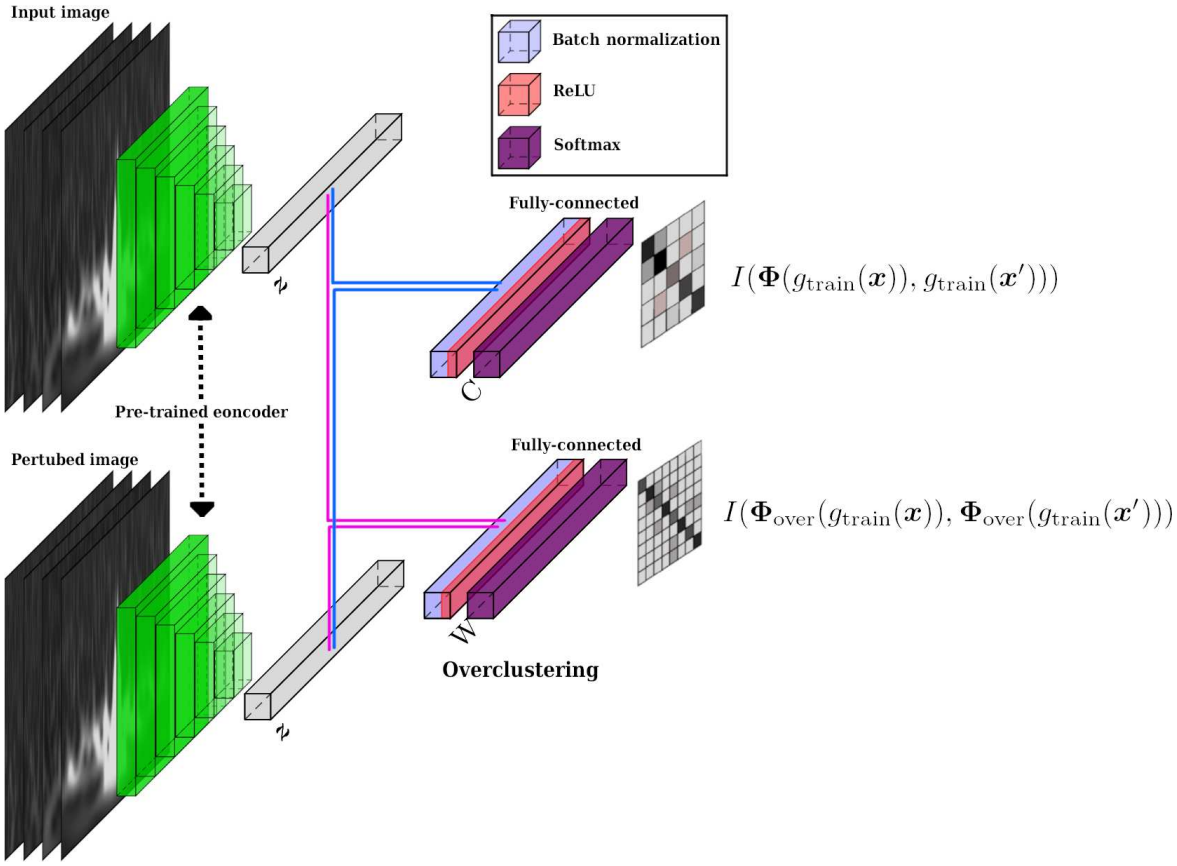

Figure 2: Schematic view of the proposed IIC architecture for image classification of transient noise. $C \in \mathbb{Z}$ is the estimated number of classes of the transient noise and $W \in \mathbb{Z}$ is the number of classes for overclustering, where $C < W$. IIC uses the VAE's pre-trained encoder and classifies transient noise from the latent variables $\boldsymbol{z}$, and the softmax (also called normalized exponential function) layer outputs the probabilities of classification of transient noise in the range of $[0, 1]$. The result of the mutual information, including the clustering of C and the overclustering of W are added (at the right end), and then IIC trains to maximize this mutual information. The two input images to IIC are the center-cropped image of transient noise and its perturbed image.

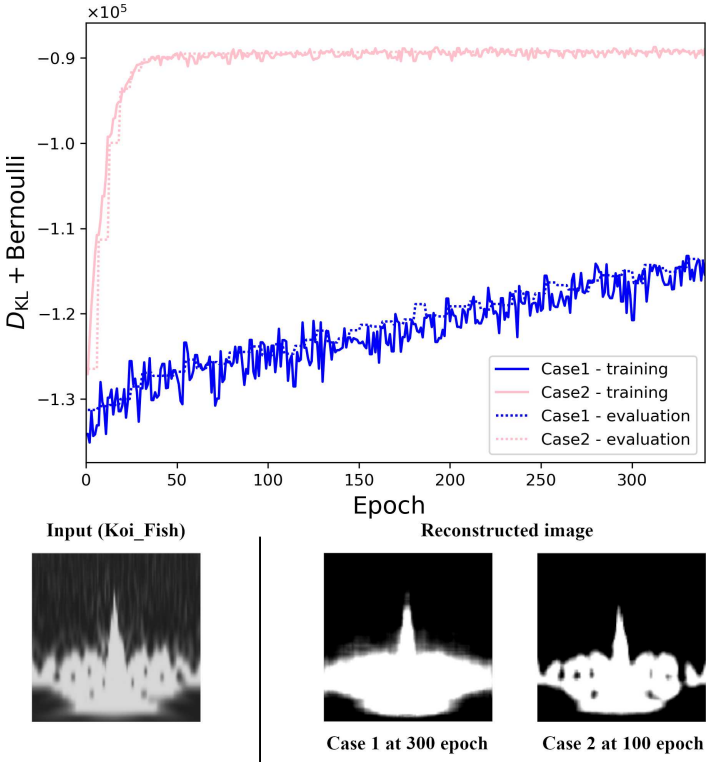

Figure 3: VAE training curves using the Bernoulli distribution in Table 2 **(Top)**. The decoder's output used the model at the specific epoch **(Bottom)**. An input image with the Gravity Spy label of "Koi_Fish" (bottom left). The output image by "Case 1" at 300 epochs does not appropriately reconstruct the input image (bottom center). The output image by "Case 2" at 100 epochs appropriately reconstructs the input image (bottom right).

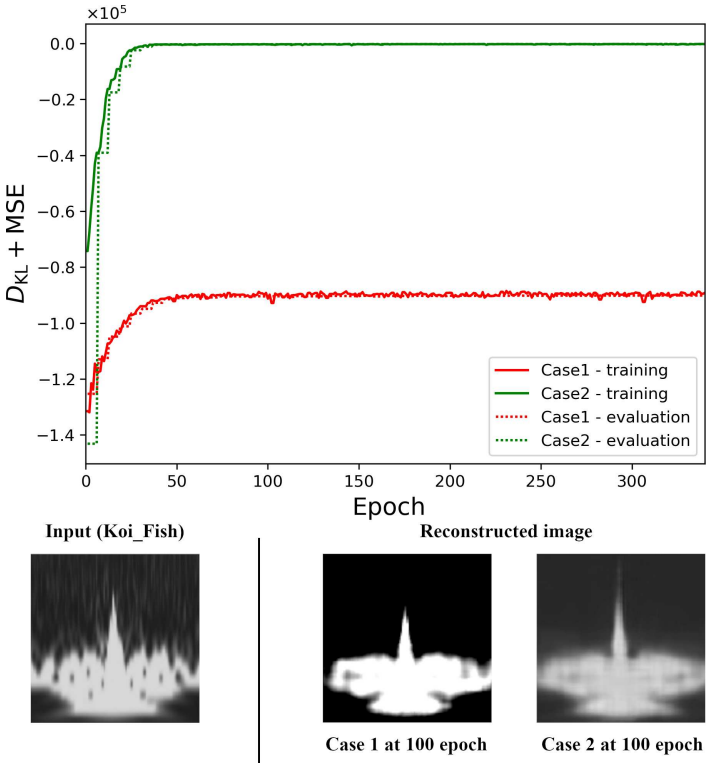

Figure 4: Same representation of Figure 3 but for MSE. The output image by "Case 1" at 100 epochs does not appropriately reconstruct the input image (bottom center). The output image by "Case 2" at 100 epochs appropriately reconstructs the input image (bottom right). The training curve of "Case 1" is worse than "Case 2". Regarding "Case 2", the training curve is small, but it reconstructs the transient noise along with the background. As a result, the reconstructed image seems to be blurred. Therefore the MSE architecture is not suitable to reconstruct the transient noise.

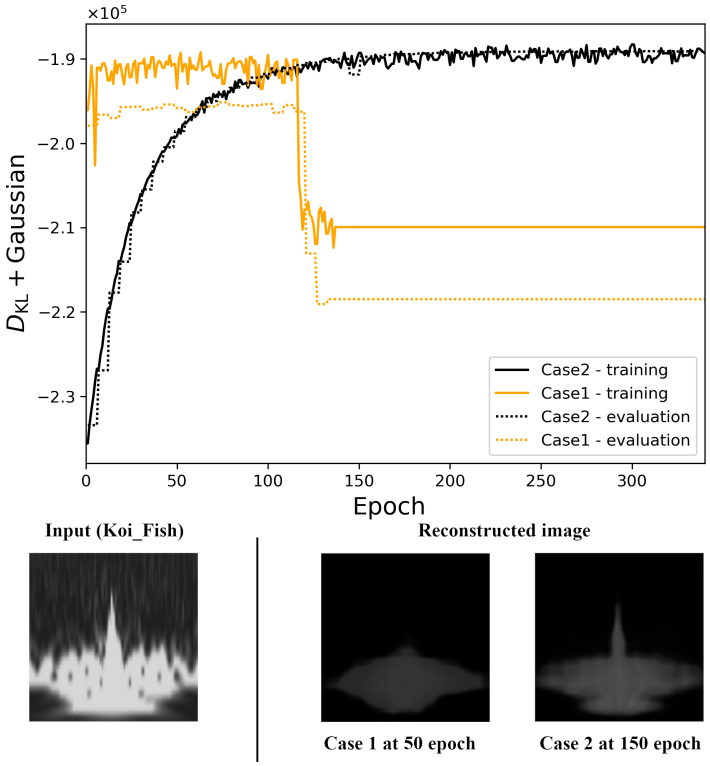

Figure 5: Same representation of Figure 3 but for the Gaussian distribution. An output image by "Case 1" at 50 epochs does not appropriately reconstruct the input image (bottom center). An output image by "Case 2" at 150 epochs approximately reconstructs the input image (bottom right).

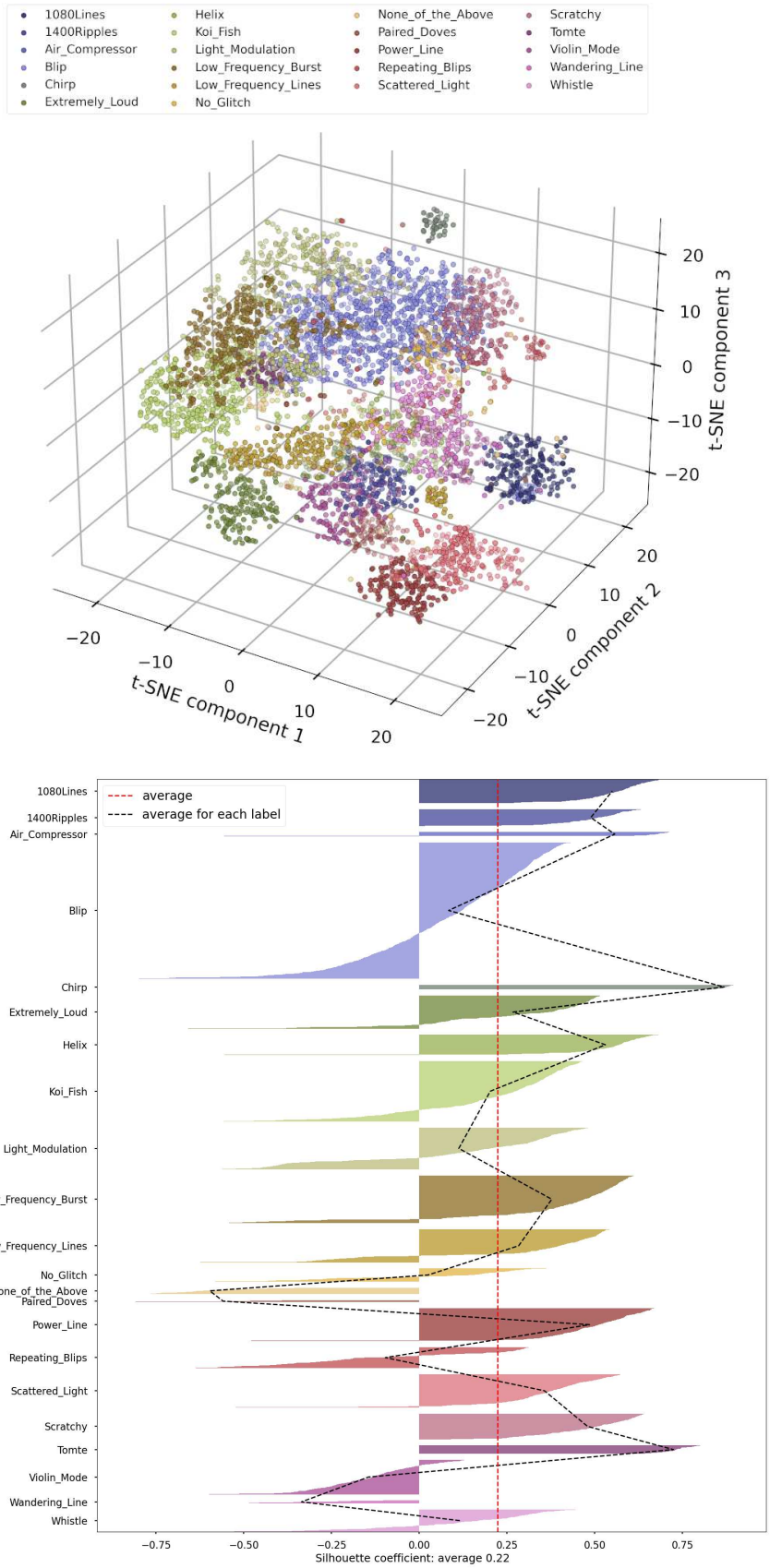

Figure 6: Colors represents 22 types of Gravity Spy label. The t-SNE mapping with Bernoulli "Case 2" at 100 epochs **(Top)**. The silhouette coefficient of the above t-SNE results **(Bottom)**. A dashed line drawn in black represents the average silhouette coefficient of each label, and the dashed red line represents an average for all labels.

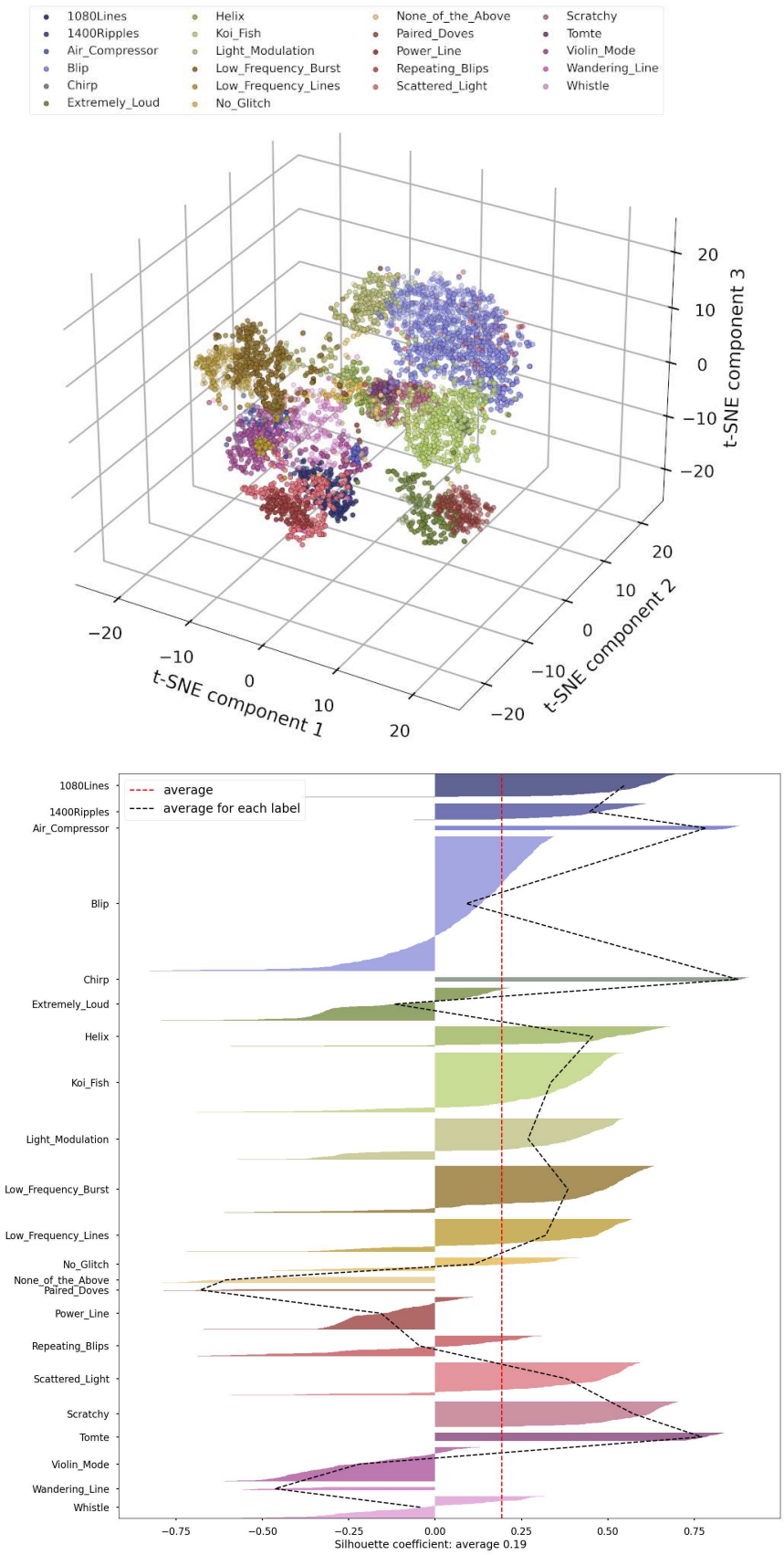

Figure 7: Same representation of Figure 6 but for the MSE "Case 2" at 100 epochs.

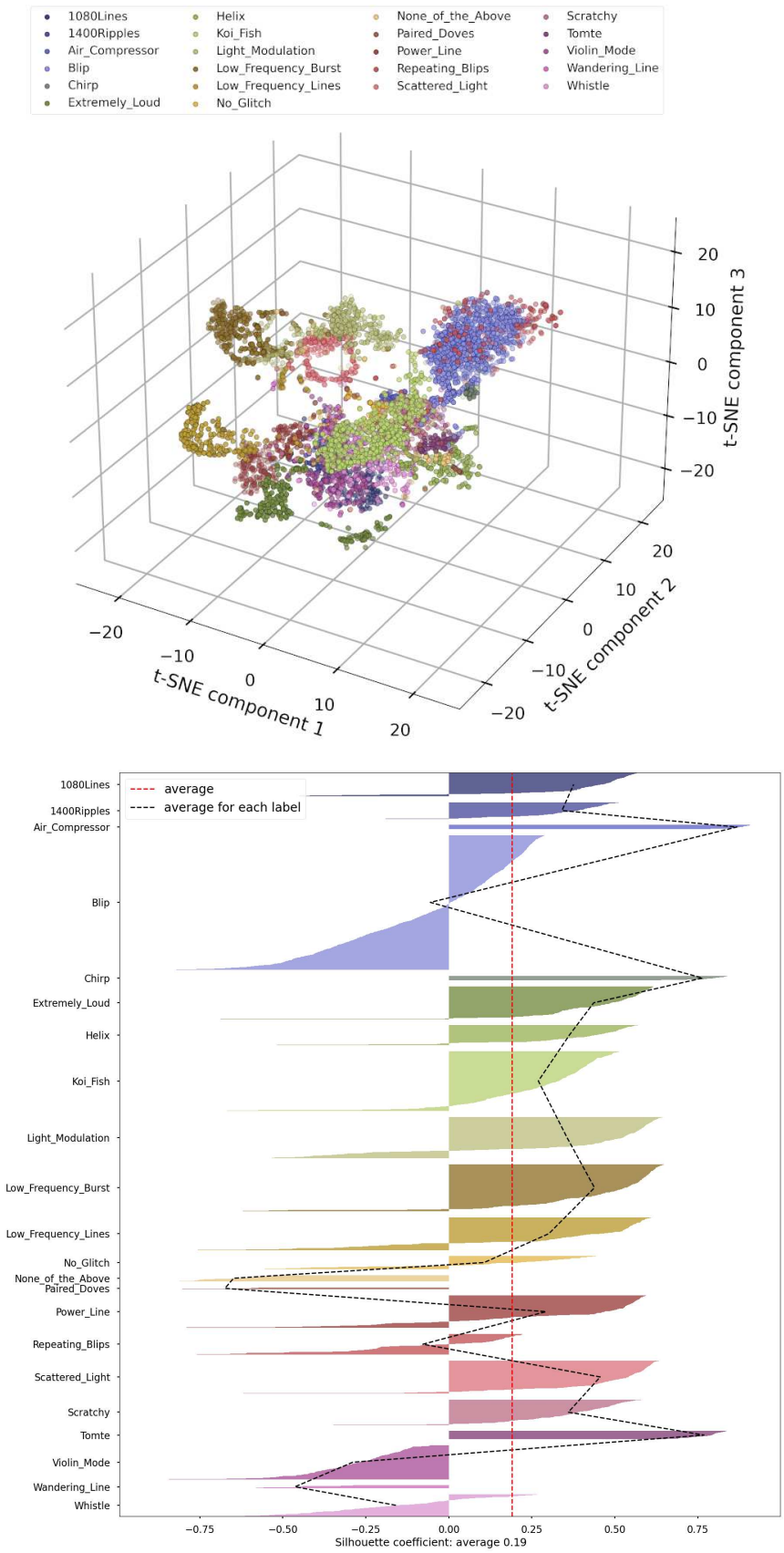

Figure 8: Same representation of Figure 6 but for the Gaussian "Case 2" at 150 epochs.

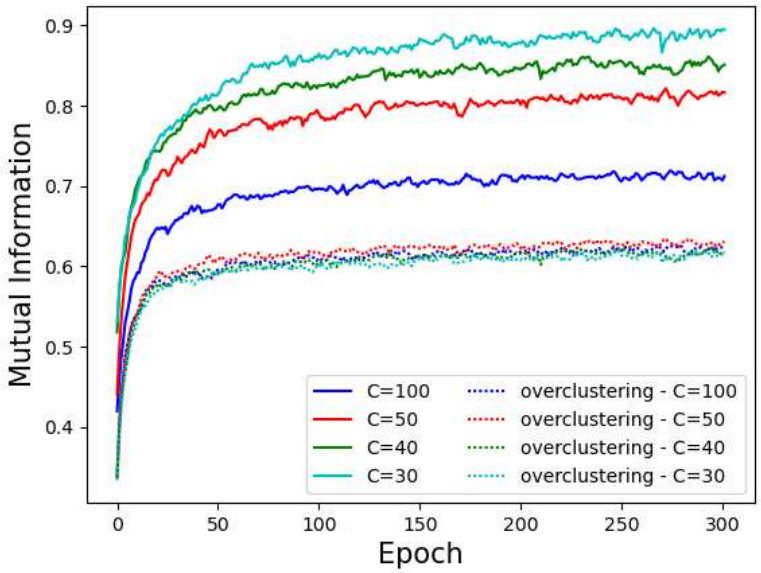

Figure 9: Training curves of IIC. The solid lines represent the average mutual information for each number of classes C. The dashed lines represent the average mutual information with overclustering corresponding to the number of classes C, and the classes of all overclustering are set to $W = 250$.

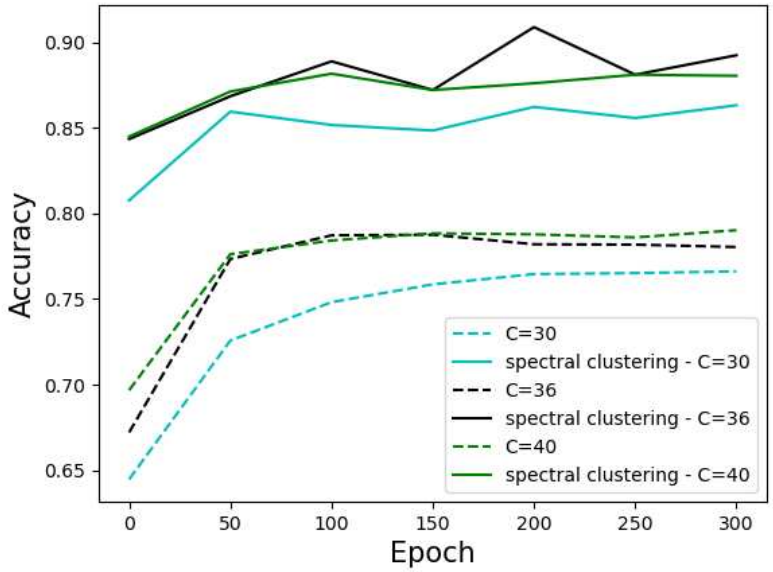

Figure 10: Accuracy of the proposed architecture (unsupervised learning) using the Gravity Spy label (supervised learning). The colored solid lines show the results of the cases where the number of classes is 30, 36, and 40 (denoted by C in the legend) with the spectral clustering of five classifiers. The dashed line represents the average accuracy of five classifiers.