LEARNING FROM PU DATA USING DISENTANGLED REPRESENTATIONS [SUPPLEMENTAL MATERIAL]

1. THEOREM 1

In this section we wish to justify the informal theorem. We investigate the minimization function mentioned in Equation 5 in the original manuscript. We note that this loss can be rewritten in the form

$$\hat{\mathcal{L}}(\theta) := \|f(\theta) - y\|_2^2$$

where

$$f(\theta) := \begin{bmatrix} \frac{1}{\sqrt{n_p}} v(x_1^p;\theta) \\ \frac{1}{\sqrt{n_p}} v(x_2^p;\theta) \\ \vdots \\ \frac{1}{\sqrt{n_p}} v(x_{n_p}^p;\theta) \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}} v(x_1^{up};\theta) \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}} v(x_2^{up};\theta) \\ \vdots \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}} v(x_{n_{up}}^{up};\theta) \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}} v(x_1^{un};\theta) \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}} v(x_2^{un};\theta) \\ \vdots \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}} v(x_{n_{un}}^{un};\theta) \end{bmatrix} \quad \text{and} \quad y := \begin{bmatrix} \frac{1}{\sqrt{n_p}} 1_{n_p} \otimes \mu_P \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}} 1_{n_{up}} \otimes \mu_U \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}} 1_{n_{un}} \otimes \mu_U \end{bmatrix}$$

where $\otimes$ denotes the Kronecker product. With this nonlinear least squares formulation one can use well established Neural Tangent Kernel (NTK) theory to show that the for sufficiently wide networks and sufficiently large scale of initialization the iterative updates and the output of the network remain close to that of the iterative updates on a linear problem of the form

$$\|J\theta - y\|_2^2 \tag{1}$$

where J denotes the Jacobian of the mapping f at random initialization. This is a direction consequence of the argument in Section 5.3 of [1] combined with NTK eigenvalue characterizations for deep convolutional networks in [2]. This argument is by now standard, and thus we omit unnecessary repetition given the informal/qualitative statement of our theorem and focus on the linearized form in (1). Without loss of generality we can focus on the case where μ_P and μ_U are scalar valued as the argument in the general case follows the exact same proof and can be thought of as repeating the scalar argument across the coordinates of μ_P/μ_U. The loss in this case can also be alternatively written in the form

$$\tilde{\mathcal{L}}(\theta) = \min_\theta \frac{1}{np}\|J_p\theta - \mu_P \mathbf{1}\|^2 + \frac{\alpha}{n_{up}}\|J_{up}\theta - \mu_U\mathbf{1}\|^2 + \frac{1-\alpha}{n_{un}}\|J_{up}\theta - \mu_{un}\mathbf{1}\|^2 \tag{2}$$

Where $J_p \in \mathbb{R}^{n_p \times d}$ is the Jacobian matrix corresponding to the positive labeled samples, similarly, the matrices $J_{up} \in \mathbb{R}^{n_{up} \times d}$ and $J_{un} \in \mathbb{R}^{n_{un} \times d}$ correspond to the unlabeled positive and negative samples, respectively, and $\theta \in \mathbb{R}^{d \times 1}$ is the linear model, and $\mathbf{1}$ is the all 1 vector.

For convenience, the three loss terms can be combined into a tall concatenated matrix as follows:

$$\tilde{\mathcal{L}}(\theta) = \left\| \begin{bmatrix} \frac{J_p}{\sqrt{n_p}}\theta - \frac{\mu_P}{\sqrt{n_p}}\mathbf{1} \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}J_{up}\theta - \frac{\sqrt{\alpha}\mu_U}{\sqrt{n_{up}}}\mathbf{1} \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}}J_{un}\theta - \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix} \right\|^2 \tag{3}$$

Thus in this case J corresponds to $\begin{bmatrix} \frac{J_p}{\sqrt{n_p}} \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}J_{up} \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}}J_{un} \end{bmatrix}$ and y to $\begin{bmatrix} \frac{\mu_P}{\sqrt{n_p}}\mathbf{1} \\ \frac{\sqrt{\alpha}\mu_U}{\sqrt{n_{up}}}\mathbf{1} \\ \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix}$

Applying gradient descent to minimize the loss function, the update rule for θ is :

$$\theta_{t+1} = \theta_t - \eta J^T(J\theta_t - y)$$

Where η is the learning rate. Defining the residual vector $r_t := J\theta_t - y$ after t iterations we have

$$\begin{aligned} r_t = J\theta_t - y &= J\theta_{t-1} - y - \eta JJ^T(J\theta_{t-1} - y) \\ &= (I - \eta JJ^T)(J\theta_{t-1} - y) \\ &= (I - \eta JJ^T)r_{t-1} \\ &= \left(I - \eta JJ^T\right)^t r_0 \end{aligned} \tag{4}$$

With sufficiently small or asymmetric initialization ([1]) we can ensure $\theta_0 \approx 0$ which implies that the initial residual is $r_0 = J\theta_0 - y \approx -y$, hence,

$$J\theta_t = y - \left(I - \eta JJ^T\right)^t y \tag{5}$$

Now consider the vector $\mathbf{w} = \begin{bmatrix} \frac{\sqrt{\alpha}\mathbf{1}}{\sqrt{n_p}} \\ \frac{-\mathbf{1}}{\sqrt{n_{up}}} \\ 0 \end{bmatrix}$. The critical observation is that this vector is approximately in the null space of J^T.

To see this note that

$$J^T w = \left(\frac{J_p}{\sqrt{n_p}}\right)^T \frac{\sqrt{\alpha}\mathbf{1}}{\sqrt{n_p}} - \left(\frac{\sqrt{\alpha}}{\sqrt{n_{up}}}J_{up}\right)^T \frac{\mathbf{1}}{\sqrt{n_{up}}} = \sqrt{\alpha}\left(\frac{J_p^T\mathbf{1}}{n_p} - \frac{J_{up}^T\mathbf{1}}{n_{up}}\right) \tag{6}$$

$\frac{J_p^T\mathbf{1}}{n_p}$ and $\frac{J_{up}^T\mathbf{1}}{n_{up}}$ are simply the empirical average of the NTK features over the labeled and unlabeled positive pairs. Since these two distributions are identical they converge to the same population mean. Let us denote this common mean by ϕ. Thus,

$$\|J^T w\| = \sqrt{\alpha}\left\|\frac{J_p^T\mathbf{1}}{n_p} - \phi - \left(\frac{J_{up}^T\mathbf{1}}{n_{up}} - \phi\right)\right\| \le \sqrt{\alpha}\left\|\frac{J_p^T\mathbf{1}}{n_p} - \phi\right\| + \sqrt{\alpha}\left\|\frac{J_{up}^T\mathbf{1}}{n_{up}} - \phi\right\| \le \sqrt{\alpha}\sqrt{\delta}$$

where the latter holds with high probability do to the concentration of the empirical mean around the true mean under mild technical assumptions about the NTK kernel and data distributions.Indeed, if the features are sub-Gaussian (e.g. bounded) one can show that $\sqrt{\delta}$ scales with $\max\left(1/\sqrt{n_p}, 1/sqrtn_{up}\right)$ and can thus be made arbitrarily small for a sufficiently large data set. To continue define the unit norm vector $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\sqrt{1+\alpha}}$ and note that

$$\hat{w}^T JJ^T \hat{w} \le \frac{\alpha}{\alpha+1}\delta \le \delta.$$

Now, we can decompose y into it's orthogonal projections onto $\hat{\mathbf{w}}$ where $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\sqrt{1+\alpha}}$: $y = y_\parallel + y_\perp = \hat{\mathbf{w}}\hat{\mathbf{w}}^T y + \left(I - \hat{\mathbf{w}}\hat{\mathbf{w}}^T\right) y$.

To continue note that

$$
\begin{aligned}
y_\perp := \left(I - \hat{\mathbf{w}}\hat{\mathbf{w}}^T\right) y &= y - \mathbf{w}\frac{\mathbf{w}^T y}{1+\alpha} \\
&= y - \mathbf{w}\frac{\sqrt{\alpha}\mu_P - \sqrt{\alpha}\mu_U}{1+\alpha} \\
&= y - \begin{bmatrix} \frac{\alpha}{1+\alpha}\frac{\mathbf{1}}{\sqrt{n_p}}(\mu_P - \mu_U) \\ -\frac{\sqrt{\alpha}}{1+\alpha}\frac{\mathbf{1}}{\sqrt{n_{up}}}(\mu_P - \mu_U) \\ \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\mu_P}{\sqrt{n_p}}\mathbf{1} \\ \frac{\sqrt{\alpha}\mu_U}{\sqrt{n_{up}}}\mathbf{1} \\ \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix} - \begin{bmatrix} \frac{\alpha}{1+\alpha}\frac{\mathbf{1}}{\sqrt{n_p}}(\mu_P - \mu_U) \\ -\frac{\sqrt{\alpha}}{1+\alpha}\frac{\mathbf{1}}{\sqrt{n_{up}}}(\mu_P - \mu_U) \\ \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{n_p}}\left[\left(1 - \frac{\alpha}{1+\alpha}\right)\mu_P + \frac{\mu_U}{1+\alpha}\right]\mathbf{1} \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}\left[\mu_P + \left(1 - \frac{1}{1+\alpha}\right)\mu_U\right]\mathbf{1} \\ \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{n_p}}\frac{\mu_P + \alpha\mu_U}{1+\alpha}\mathbf{1} \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}\frac{\mu_P + \alpha\mu_U}{1+\alpha}\mathbf{1} \\ \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix}
\end{aligned}
$$

Furthermore,

$$J^T y_\perp = \frac{1}{n_p}\frac{\mu_P + \alpha\mu_U}{1+\alpha}J_p^T\mathbf{1} + \frac{\alpha}{n_{up}}\frac{\mu_P + \alpha\mu_U}{1+\alpha}J_{up}^T\mathbf{1} + \frac{1-\alpha}{n_{un}}\mu_U J_{un}^T\mathbf{1}$$

Now using concentration of the rows of different J the above is approximately equal to the following with high probability

$$J^T y_\perp \approx (\mu_P + \alpha\mu_U)\,\phi + (1-\alpha)\mu_U\tilde{\phi}$$

where ϕ and $\tilde{\phi}$ are the average of the NTK features in the positive and unlabeled negative data. Thus, for $\hat{v} = y_\perp/\|y_\perp\|_2$ we have

$$v^T JJ^T v = \frac{1}{\|y_\perp\|_2^2} y_\perp^T JJ^T y_\perp \geq (1+\alpha)\frac{\|\left(\mu_P + \alpha\mu_U\right)\phi + (1-\alpha)\mu_U\tilde{\phi}\|_2^2}{\mu_P^2 + \mu_U^2 + 2\alpha\mu_P\mu_U} := \Delta$$

Thus in the direction of $\hat{w}$ the NTK kernel JJ^T is small where as in the direction $\hat{v}$ it is large. Intuitively, this implies that $(I - \eta JJ^T)^t y_\perp$ is small for a sufficiently large t where as $(I - \eta JJ^T)^t y_\parallel \approx y_\parallel$. Indeed, we can make this intuition precise and prove that

$$\|(I - \eta JJ^T)^t y_\perp\|_2 \leq (1 - \eta\Delta)^t\,\|y_\perp\|_2 \quad \text{and} \quad \|y_\parallel - (I - \eta JJ^T)^t y_\parallel\|_2 \leq \left(1 - (1 - \eta\delta)^t\right)\|y_\parallel\|_2$$

Since δ can be made arbitrarily small for a sufficiently large data set we have $\delta << \Delta$ therefore for a broad range of values of η one can find a stopping time T where both terms are very small. For instance for $\eta = \frac{1}{2\Delta}$ picking any stopping time obeying

$$\log\left(\frac{2}{\epsilon}\right) \leq T \leq \frac{\log\left(1 - \frac{\epsilon}{2}\right)}{\log\left(1 - \frac{\delta}{\Delta}\right)}$$

we have

$$\|(I - \eta JJ^T)^T y_\perp\|_2 \leq \frac{\epsilon}{2}\|y_\perp\|_2 \quad \text{and} \quad \|y_\parallel - (I - \eta JJ^T)^T y_\parallel\|_2 \leq \frac{\epsilon}{2}\|y_\parallel\|_2$$

using the above identities we conclude that

$$\|J\theta_T - y_\perp\| = \|y_\parallel - \left(I - \eta JJ^T\right)^T y_\parallel - \left(I - \eta JJ^T\right)^T y_\perp\| \leq \epsilon\|y\|$$

This formally proves that for an appropriate stopping time T

$$J\theta_T \approx y_\perp \tag{7}$$

Pulling back the definition of J:

$$J\theta_T = \begin{bmatrix} \frac{J_p}{\sqrt{n_p}}\theta_T \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}J_{up}\theta_T \\ \frac{\sqrt{1-\alpha}}{\sqrt{n_{un}}}J_{un}\theta_T \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{n_p}}\frac{\mu_P+\alpha\mu_U}{1+\alpha}\mathbf{1} \\ \frac{\sqrt{\alpha}}{\sqrt{n_{up}}}\frac{\mu_P+\alpha\mu_U}{1+\alpha}\mathbf{1} \\ \frac{\sqrt{1-\alpha}\mu_U}{\sqrt{n_{un}}}\mathbf{1} \end{bmatrix} \tag{8}$$

Resulting in:

$$J_p\theta_T = J_{up}\theta_T \approx \frac{\mu_P + \alpha\mu_U}{1+\alpha}\mathbf{1}, \quad \text{and} \quad J_{un}\theta_t \approx \mu_U\mathbf{1} \quad \square$$

2. ILLUSTRATING THE VALUE OF THE REPRESENTATION LEARNING METHOD

Here, we show in Fig. 1 a simple comparison between the t-SNE visualization of the unlabeled data in the representation space in our proposed method and in the representation space of a classical VQ-VAE [3]. The representation of the unlabeled data in the VQ-VAE shows the entanglement of the positive and negative samples, which makes it challenging to learn a binary classifier in the PU setting. In contrast, the figure highlights the effectiveness of the proposed method in learning a representation space in which the positive and negative samples are clearly concentrated in two clusters, making the problem much closer to the simple 1-dimensional scenario shown in Fig. 1 in the original manuscript. Furthermore, the figure also underscores the limitation of dimensionality reduction methods in achieving the separability attained by the proposed method.

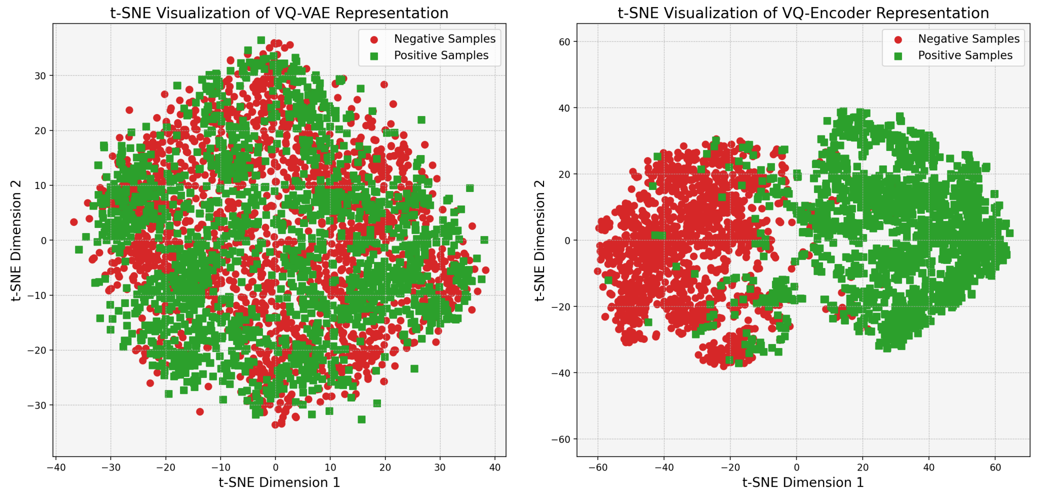

Fig. 1: The t-SNE visualization of the learned data representation (for the AFHQ dataset) in the proposed method compared to the data representation learned in a classical VQ-VAE showing how the proposed learned representation disentangles the positive and negative samples such that they can be easily told apart using simple clustering algorithms.

3. EXPERIMENTS

3.1. Stopping Criteria for Baseline Methods

We compare our method to five state-of-the-art PU learning methods that have shown good performance on image datasets. The first method is *Observer-GAN* [4], which uses a GAN-based setup to learn a binary classifier. The second is TED^n[5], which uses an alternating procedure between the problem of estimating the positive prior α, and the problem of learning a binary classifier. The third method, *D-GAN* [6], also uses a GAN-based setup to learn the distribution of negative samples and then learn a binary classifier. Finally, we compare against Robust-PU [7] and Dist-PU [8], which also show state-of-the-art performance on PU data.

Since the first two methods claim convergence to the correct solution, and there is no clear way to stop training at an early stage, we follow the same method of training each of the methods for 1000 epochs. We then look at the average performance of the last 50 and 100 epochs. For the third method, no specific criteria were presented for terminating the training of the second-stage classifier. Therefore, we train the classifier and apply early stopping based on a fully labeled validation set to prevent the second-stage classifier from overfitting.

For our proposed method, we look at the Euclidean distance between the two clusters identified by the K-means algorithm (applied to the training set), and stop training when this distance starts decreasing. Figure 2 shows that even though the accuracy on the validation set does not change dramatically after it reaches about 20 epochs, the point at which the distance between the two clusters found using the training set is largest also corresponds to the point of highest accuracy for the validation data. This behavior was evident in all experiments.

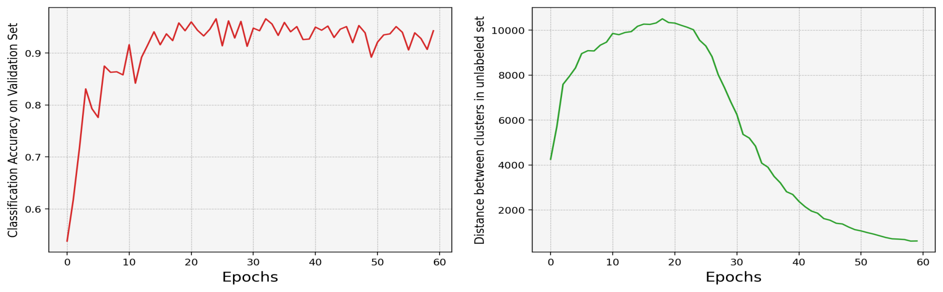

Fig. 2: Left: Accuracy curve as a function of the number of epochs on the test data. Right: The Euclidean distance between the centers of the two clusters identified by the K-means algorithm on the unlabeled (training) data.

4. ABLATION STUDIES

4.1. Adaptations of The Proposed Method

We empirically study various adaptations of the proposed method to evaluate the importance of each component. Initially, we study the implementation of the idea presented in section 3.2 in the original manuscript, where we project the input data to two constant vectors. Here one of the vectors is $\mu_U = \underline{0}$, where $\underline{0}$ is the all-zero vector, and the other vector is $\mu_P = \underline{a}$ where a is a scalar that takes value from the set $\{1, 5, 50, 100\}$ (we report the mean and standard deviation of the accuracy of all trials), and $\underline{a}$ is the vector of all a's. This experiment is referred to as "Constant encodings" in Table 1.
Next, we implement the idea while considering two normal distributions $U \sim \mathcal{N}(\underline{0}, I)$ and $P \sim \mathcal{N}(\underline{a}, I)$ instead of μ_U, and μ_P respectively, where $\mathcal{N}(\underline{a}, I)$ is the normal distribution that has a mean vector of all a's, and identity covariance matrix. Again, we let a to take values from the set $\{1, 5, 50, 100\}$ (we report the mean and standard deviation of accuracy of all trials). In this case, we penalize the KL divergence between the encoded vectors and the two normal distributions U and P. This variant is named "distributional encodings" in Table 1.
Thirdly, we assess the impact of the number of codebook vectors. We implement the proposed method using the minimal feasible number of codebook vectors, which is two vectors. The resulting accuracy is presented in Table 1 and referred to as "VQ (2 updated C.B. vectors).
Fourth, we explore the significance of updating the codebook vectors during the vector quantization of the representation space. We replicated our previous experiments but this time without any updates to the codebook vectors. In this case, because the idea of the method relies on having two distinct magnitudes of vectors in the representation space, we initialize the codebook vectors to have two modes, such that half of the codebook vectors are initialized from a normal distribution $\mathcal{N}_1(\underline{0}, I)$), and the other half is initialized from a normal distribution $\mathcal{N}_2(\underline{a}, I)$). Here a took values from the set $\{1, 5, 50, 100\}$. We refer to this method in Table 1 as "VQ (No updates)".
Lastly, we revisited the prior configuration but with a restriction to just two codebook vectors. The objective was to project both positive and negative samples onto one of these two vectors. This technique is denoted as "VQ (2 fixed C.B. vectors)" in Table 1.

The conducted experiments and ablation studies show the efficiency of learning a new representation from PU data and the significance introduced by the quantization of the representation space. Although the idea of learning a new representation space stems from the simple mathematical steps shown in section 3.2 in the original manuscript, the conducted ablation studies show that quantization helps achieve a clear separation between the positive and negative data in the unlabeled set. The ablation

Method	Accuracy% ($\pm$std)			
Constant Encodings	$73(\pm 4)$			
Distributional Encodings	$69(\pm 7)$			
VQ (2 updated C.B. vectors)	$97(\pm 0.5)$			
	Codebook Vectors Initialization			
	$\mathcal{N}(\underline{0}, I), \mathcal{N}(\underline{1}, I)$	$\mathcal{N}(\underline{0}, I), \mathcal{N}(\underline{5}, I)$	$\mathcal{N}(\underline{0}, I), \mathcal{N}(\underline{50}, I)$	$\mathcal{N}(\underline{0}, I), \mathcal{N}(\underline{100}, I)$
VQ (No updates)	$78.9(\pm 4.3)$	$95.1(\pm 0.7)$	$96.7(\pm 1)$	$88.9(\pm 3.2)$
VQ (2 fixed C.B. vectors)	$76.3(\pm 3)$	$97.3(\pm 0.3)$	$96.2(\pm 0.8)$	$91.4(\pm 2.7)$

Table 1: Ablation studies conducted on AFHQ dataset, illustrating the impact of different initializations of codebook vectors on the performance.

studies also demonstrate the importance of allowing updates of the codebook vectors when adopting a vector-quantized representation space. Since the choice of the means of the codebook vectors seems to impact the performance (as evident in the last two rows in Table 1), initializing all codebook vectors with zero means and updating them during training eliminates the need to fine-tune the means at initialization.

4.2. Testing the speed of K-means algorithm after each iteration

We test the speed of running the K-means algorithm when training on different datasets, and Table 2 shows the results. The table shows the average and standard deviation of the time elapsed by the K-means algorithm after each training epoch in seconds for each of three datasets, along with the dimensions of the learned representation the algorithm is run on and the number of samples. Although increasing data size will increase the time of running the algorithm, we expect the time to only increase linearly as the time complexity of K-means is linear in dimensions and number of samples. It should be pointed out that the K-means algorithm in the proposed method is run on the learned representation space, which is in lower dimensions than the input data.

Dataset	MNIST	Fashion MNIST	AFHQ
# of Samples, # of dimensions	$19000, 49$	$19000, 49$	$3300, 1024$
K-means Elapsed Time (s)	0.0207 ± 0.0038	0.0292 ± 0.0174	0.3681 ± 0.0071

Table 2: Time Elapsed in running K-means algorithm after each training epoch

4.3. Testing different α values

We conducted an experiment to assess the effectiveness of our method when the likelihood of positive samples was either less or greater than that of negative samples, characterized by different α values. We then compared the outcomes of this experiment with the results obtained using TED^n method in Table 3.

	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.8$
TED^n	86.9 ± 1	89.5 ± 2.5	89.9 ± 14.9	81.5 ± 3.7	71.1 ± 2
VQ K-means	97.6 ± 1.1	96.5 ± 0.7	95.4 ± 1.3	84.3 ± 1.1	70.2 ± 2.1

Table 3: Classification results with different α values

5. NETWORK ARCHITECTURES

5.1. Encoder in the proposed method

The architecture of the vector quantized encoder used in the proposed method consists of three initial convolutional layers followed by a stack of six residual layers. The first three layers sequentially process the input data, while each residual layer in the stack features a combination of a ReLU activation layer and two convolution layers whose output is added to their input and then used directly for quantization. We use a codebook that consists of 512 64-dimensional vectors that are used for the quantization of the output vector of the encoder network. For all methods, the same architecture was used by just adapting the input number of channels (3 for RGB images and 1 for grayscale images).

5.2. Classifier used for baseline methods

In the experiment whose results are presented in Table 2 in the original manuscript, we made a modification to the classifier architectures used by the baseline methods to accommodate for the change in the dimension of the input data (3 channel 2D images in the raw data and 1D vectors for the learned representations). The architecture we used for the classifier was a 4-layer (ReLU) fully connected network reducing the size of the input ($1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 1$) to 1 node that can be used for binary classification.

6. REFERENCES

[1] Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi, “Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian,” *arXiv preprint arXiv:1906.05392*, 2019.

[2] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai, “Gradient descent finds global minima of deep neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 1675–1685.

[3] Aaron Van Den Oord, Oriol Vinyals, et al., “Neural discrete representation learning,” *Advances in neural information processing systems*, vol. 30, 2017.

[4] Omar Zamzam, Haleh Akrami, and Richard Leahy, “Learning from positive and unlabeled data using observer-gan,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[5] Saurabh Garg, Yifan Wu, Alex Smola, Sivaraman Balakrishnan, and Zachary Lipton, “Mixture proportion estimation and PU learning: A modern approach,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Florent Chiaroni, Ghazaleh Khodabandelou, Mohamed-Cherif Rahal, Nicolas Hueber, and Frederic Dufaux, “Counter-examples generation from a positive unlabeled image dataset,” *Pattern Recognition*, vol. 107, pp. 107527, 2020.

[7] Zhangchi Zhu, Lu Wang, Pu Zhao, Chao Du, Wei Zhang, Hang Dong, Bo Qiao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang, “Robust positive-unlabeled learning via noise negative sample self-correction,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3663–3673.

[8] Yunrui Zhao, Qianqian Xu, Yangbangyan Jiang, Peisong Wen, and Qingming Huang, “Dist-pu: Positive-unlabeled learning from a label distribution perspective,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14461–14470.