APPLYING DEEP-CLUSTERING APPROACHES TO CREATE MEANINGFUL REPRESENTATIONS OF ECG BEAT MORPHOLOGY

236781 - DEEP LEARNING ON COMPUTATIONAL ACCELERATORS
FINAL PROJECT

Amit Rotner
amitrotner@campus.technion.ac.il

Shaked Doron
shaked.doron@campus.technion.ac.il

September 10, 2020

ABSTRACT

In this project, we aim to research the effect of joint dimensionality reduction and clustering on an ECG diagnostic task. Our goal is to discover whether we can benefit from the latent space and clustering structures, to improve the model predictions. We compare supervised classification on the raw data with the approach of applying dimensionality reduction and clustering prior to classification. We experiment with both methods on the labeled PhysioNet MIT-BIH Arrhythmia and PTB Diagnostic ECG Databases [2, 10, 4]. According to the results, the suggested methods are able to make predictions with accuracies comparable to the baseline paper [6]. The code is available at `https://github.com/amitrotner/CS236781-Project`.

***K*eywords** Deep Clustering · Convolutional Autoencoders · Convolutional Neural Networks · ECG classification

1 Introduction

ECG is the main tool for monitoring cardiac health. Today, ECG data are analyzed manually by physicians for detecting abnormalities in the signal. This manual procedure is both time-consuming and prone to errors. Therefore, we are trying to figure out a suitable method to tackle this problem and to increase the accuracy of detection and analysis. Hopefully, machine learning methods will make this manual process automatic in the near future.

In the field of machine learning, there are two main types of tasks: supervised, and unsupervised. In supervised learning, we map an input to output labels or to continuous output. On the other hand, in unsupervised learning, we wish to learn the inherent structure of our data without using explicitly-provided labels. While intuitively, ECG heartbeat classification is a supervised learning task, we think that a dimensionality reduction to a learned clustered latent space can infer the natural structure of the ECG signals, leading to better classification and better cardiac health monitoring.

In this work, we use the DCEC algorithm [5] for a joint nonlinear embedding and clustering procedure before classification as a first approach and a convolutional neural network as a second approach. We compare those approaches, their drawbacks, advantages, results and performances on the PhysioNet MIT-BIH Arrhythmia and PTB Diagnostic ECG Databases.

2 Methods

2.1 Dataset

In this project, we use PhysioNet MIT-BIH Arrhythmia and PTB Diagnostic ECG Databases as a data source for labeled ECG records [10, 2, 4].

MIT-BIH. The MIT-BIH Arrhythmia dataset contains records from 47 subjects. Those samples were recorded at the sampling rate of 360Hz while two or more cardiologists independently annotated each record. Those annotations are

mapped into 5 different categories in accordance with Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard [3]. Table 1 presents this mapping.

PTB. The PTB Diagnostics dataset contains 549 records from 290 subjects. Within the header file of most of these ECG records, is a detailed clinical summary almost for every subject, except for 22 subjects whose clinical summary is not available. The diagnostic classes of the remaining 268 subjects are summarized in table 2. For comparison purposes, we have only worked with MI and healthy control categories, the same as the baseline paper [6].

Category	Annotations
N	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S	• Atrial premature • Aberrant atrial premature • Nodal premature • Supra-ventricular premature
V	• Premature ventricular contraction • Ventricular escape
F	• Fusion of ventricular and normal
Q	• Paced • Fusion of paced and normal • Unclassifiable

Table 1: Mapping between beat annotations and AAMI EC57 [3] categories.

Diagnostic class	Number of subjects
Myocardial infarction	148
Cardiomyopathy/Heart failure	18
Bundle branch block	15
Dysrhythmia	14
Myocardial hypertrophy	7
Valvular heart disease	6
Myocarditis	4
Miscellaneous	4
Healthy controls	52

Table 2: Summary of diagnostic classes in the PTB dataset [2].

2.2 Baselines

Our presented methods are compared to Kachuee et al. [6], in which the authors proposed a transferable method based on deep convolutional neural networks for the classification of heartbeats.

In the following we will lay the mathematical background for understanding both approaches we are presenting. For the CNN see subsection 2.3 and for the DCEC algorithm see subsection 2.4.

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is one of the most commonly used types of artificial neural networks. As traditional neural networks use matrix multiplication as a way to connect between the input and the output, CNNs introduce convolutional layers as a replace for the general matrix multiplication. Convolutional layers convolve the input using their kernel. Given an m-dimensional vector $x = (x_1, \ldots, x_m)$, the layer produces an n-dimensional vector $y = (y_1, \ldots, y_n)$ by the equation below:

$$y_j = \varphi\left(\sum_{i=1}^{m} w_{ij} * x_i\right) \tag{1}$$

where φ is an activation function and $*$ is the convolution operator.

Moreover, CNNs introduce pooling layers (or subsampling layers) which are a way to reduce the size of feature maps between the convolutional layers. For instance, the maximum pooling uses the maximum value in the adjacent area as an output and the average pooling uses the average.

For further readings about CNNs, see the relevant pages at the course website: 1 & 2.

2.4 Deep Clustering with Convolutional Autoencoders (DCEC) algorithm

(This subsection is heavily based on [5])

Let $X = [x_1, x_2, \ldots, x_N]$ be a set of points in $\mathbb{R}^D$ to be clustered. Clustering algorithms rely on inter-point distances. Hence, when D is high, these distances become less informative leading to a non-effective clustering [1]. To overcome this problem, we embed the data into a lower dimensional space $\mathbb{R}^d$. The embedding of the dataset into $\mathbb{R}^d$ is denoted by $Z = [z_1, z_2, \ldots, z_N]$. The function that performs the embedding is denoted by $f_\theta : \mathbb{R}^D \to \mathbb{R}^d$. Thus, we can summarize that $z_i = f_\theta(x_i)$ for all i.

To constrain f_θ to construct a faithful embedding of the original data, we require that the original data be reproducible from its low-dimensional image. Therefore, we consider a reverse mapping $g_\omega : \mathbb{R}^d \to \mathbb{R}^D$ and minimizing the following reconstruction loss, denoted as L_r:

$$\min_{\Omega} \|X - G_\omega(Z)\|_F^2 \tag{2}$$

where $\Omega = \{\theta, \omega\}$, $Z = F_\theta(X) = [f_\theta(x_1), \ldots, f_\theta(x_N)]$, $G_\omega(Z) = [g_\omega(z_1), \ldots, g_\omega(z_N)]$ and $\|\cdot\|_F$ is the Frobenius norm.

In order to do a clustering in tandem with continuous optimization of the embedding, the Deep Convolutional Embedded Clustering (DCEC) algorithm is built on the Convolutional AutoEncoders (CAE) formulation and connects the embedded layer of CAE to a clustering layer. The latter maps each embedded point z_i of input x_i into a soft label. Then the clustering loss L_c is defined as Kullback-Leibler divergence (KL divergence) between the distribution of soft labels and the predefined target distribution. CAE is used to learn embedded features and the clustering loss guides the embedded features to be prone to forming clusters.

The objective of DCEC is to minimize a loss function that is a weighted combination of these two losses:

$$L = L_r + \gamma L_c \tag{3}$$

where L_r and L_c are reconstruction loss and clustering loss respectively, and $\gamma > 0$ is a hyper-parameter that balances L_r and L_c.

2.4.1 Clustering Layer and Clustering Loss

The clustering layer and loss were first introduced in Xie et al. [14]. The clustering layer maintains cluster centers $\{\mu_j\}_1^K$ as trainable weights and maps each embedded point z_i into soft label q_i by Student's t-distribution [13]:

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2\right)^{-1}}{\sum_j \left(1 + \|z_i - \mu_j\|^2\right)^{-1}} \tag{4}$$

where q_{ij} is the j-th entry of q_i, representing the probability of z_i belonging to the cluster j. The clustering loss is defined as:

$$L_c = KL\left(P \parallel Q\right) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{5}$$

where P is the target distribution, defined as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j \left(q_{ij}^2 / \sum_i q_{ij}\right)} \tag{6}$$

The network architecture is illustrated in figure 1.

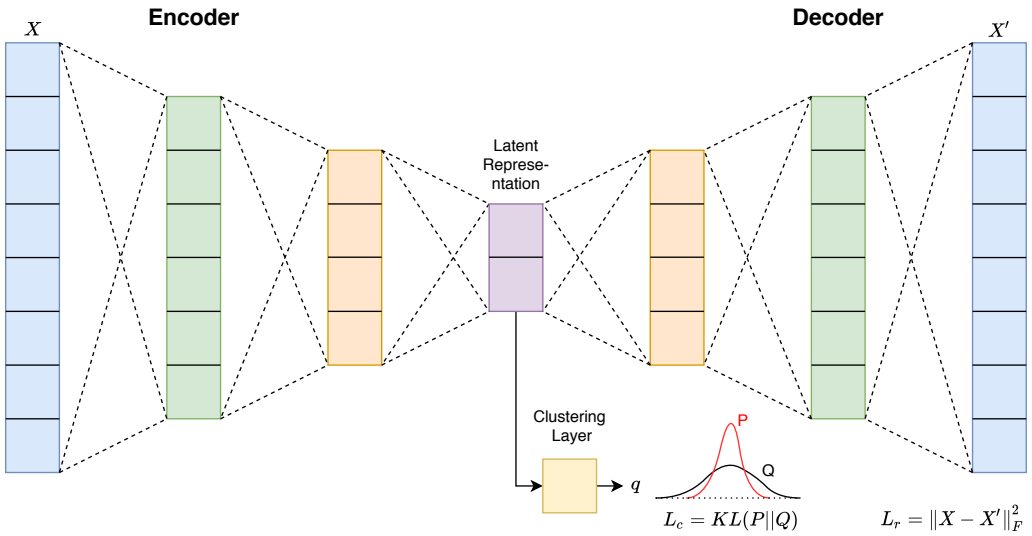


Figure 1: The structure DCEC. It is composed of a convolutional autoencoders and a clustering layer connected to embedded layer of autoencoders. Reconstruction loss (L_r) and the clustering loss (L_c) are given in the figure. Best viewed in color.

3 Implementation and experiments

3.1 Preprocessing

To extract beats from the ECG signals, the data were preprocessed using the technique presented in Kachuee et al. [6]. The extracted beats have the same dimension which is essential for training the models.

3.2 Training the CNN network

Figure 2 illustrates the network architecture proposed for both datasets. The model inputs, as explained in 3.1, are extracted beats with a dimension of 187. All convolution layers are applying 1-D convolution through time and each have a kernel of size 3, a stride of size 1 and a padding of size 1. After each 1-D convolution we used a LeakyReLU layer [8] with negative slope of 0.2 and a dropout layer [12] with a probability of 0.1. The last two 1-D convolutions function as a fully-connected layers followed by a Softmax layer to produce class probabilities. In total, the network consists 5 weight layers.

3.3 Training the DCEC network

Figure 1 illutrates the network architercture proposed for both datasets. The encoder network is a set of 3 1-D convolutional layers with dimensions D-128-64-d, where D is the dimension of input data (187 for both datasets) and d is the dimension of the latent space. Each convolutional layer has kernel of size 3 and a padding of size 1. After each 1-D convolution we used a LeakyReLU layer [8] with negative slope of 0.2. The decoder network is a mirror of the encoder, i.e. a set of 3 1-D transposed convolutional layers with dimensions d-64-128-D. We first pretrain the autoencoder only with the reconstruction loss (by setting $\gamma = 0$) to get consequential target distribution. After pretraining, the cluster centers are initialized by performing k-means on the embedded input. Then we set $\gamma = 0.1$, which is a commonly used value by other researchers (e.g [11]), and fine-tune the model. The autoencoder's weights and cluster centers are updated by using mini-batch Adam[7] optimization while the target distribution P is updated using all embedded points every T iterations. Once the model is fine-tuned, we perform heartbeats classification on the embedded points using the CNN network as explained in 3.2.

3.4 Implementation Details

We have used PyTorch computational library for training and evaluating the models. In the CNN models training, we have used Cross Entropy loss as the loss function, while during the DCEC model training, we have used the loss

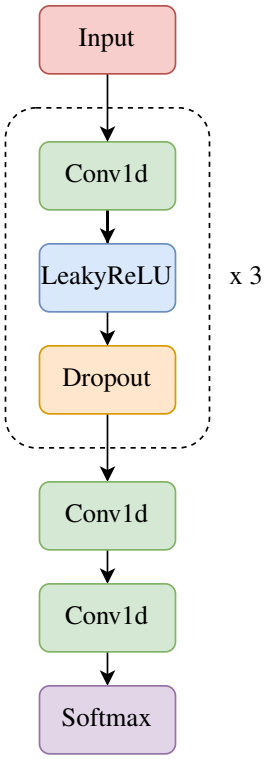


Figure 2: CNN Model Architecture.
Best viewed in color.

Work	Accuracy
Our CNN network	**92.7%**
Our DCEC network[a]	**89%**
Baseline paper [6]	93.4%

Table 3: Comparison of MIT-BIH dataset results

Work	Accuracy
Our CNN network	**96.7%**
Our DCEC network[b]	**95.1%**
Baseline paper [6]	95.9%

Table 4: Comparison of PTB dataset results

[a] Best model with $d = 30$
[b] Best model with $d = 40$

function developed in 2.4. For the CNN training, we have used Adam optimizer [7] with a learning rate, beta-1, beta-2 and epsilon of 0.001, 0.9, 0.999 and 1e-8, respectively. The learning rate is decayed exponentially with the decay factor of 0.75 every 500 iterations. We have trained the CNN model for 3000 epochs, with an early stopping of 5 epochs without improvement. Those hyperparameters were taken from the baseline paper [6], with some small modifications, for reliable comparison. For the DCEC training, we have pretrained the DCEC for 400 epochs using SGD optimizer with a momentum of 0.9 and a learning rate of 0.1. The latter is decayed exponentially with a decay factor of 0.1 every 200 iterations. For the DCEC fine-tuning, we have used Adam optimizer [7] with the default parameters for 600 epochs. Again, the learning rate is decayed exponentially with the decay factor of 0.1 every 200 iterations. We set γ to 0.1 and the update interval T to 180 in the fine-tuning step. Then we have trained the CNN network on the embedded features, using the same hyperparameters as above. We conducted our experiments iterating on d from 10 to 60 in steps of 10. We have trained all the networks in the Rishon server, on a GeForce RTX 2080 processor.

4 Results and discussion

4.1 Results

We have evaluated the methods on a test set contains 20% of the data which is not used in the training phase. Tables 3 and 4 show the results of PTB/MIT-BIH dataset, respectively. Our results are very close to the baseline paper and even surpass it using our CNN network on the PTB dataset. Figure 3 shows the DCEC accuracy as a function of latent space dimension d for both datasets. Figure 4 presents the confusion matrix of applying the classifier on the test set. As it can be seen from this figure, both models are able to make accurate predictions and distinguish different classes on both datasets.

4.2 Analysis

CNN and DCEC. Overall, the CNN method produces better results than the DCEC in both datasets, while our CNN network surpasses the baseline paper on the PTB dataset. This leads to the conclusion that the traditional method works better in the ECG heartbeat classification task. It is worth checking whether this conclusion is correct for every time series data. We will not cover other time-series datasets in this project but in future work.

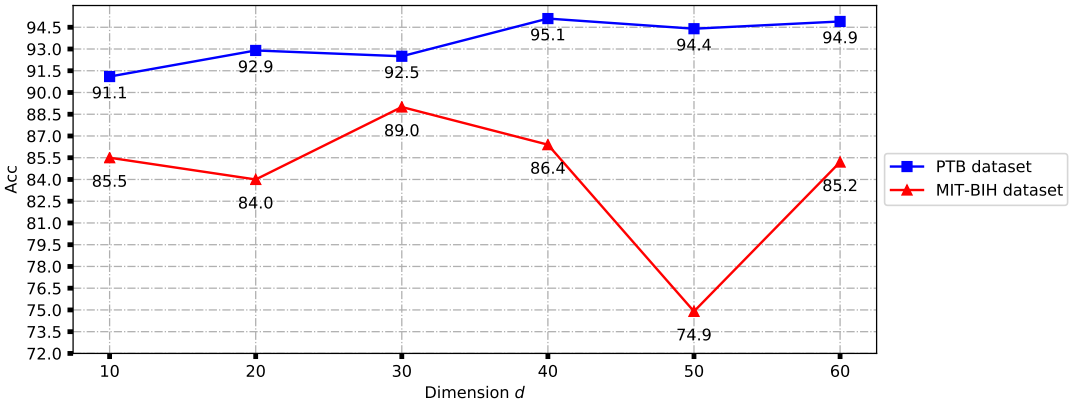


Figure 3: Classification accuracy as a function of the dimensionality d of the latent space. Best viewed in color.

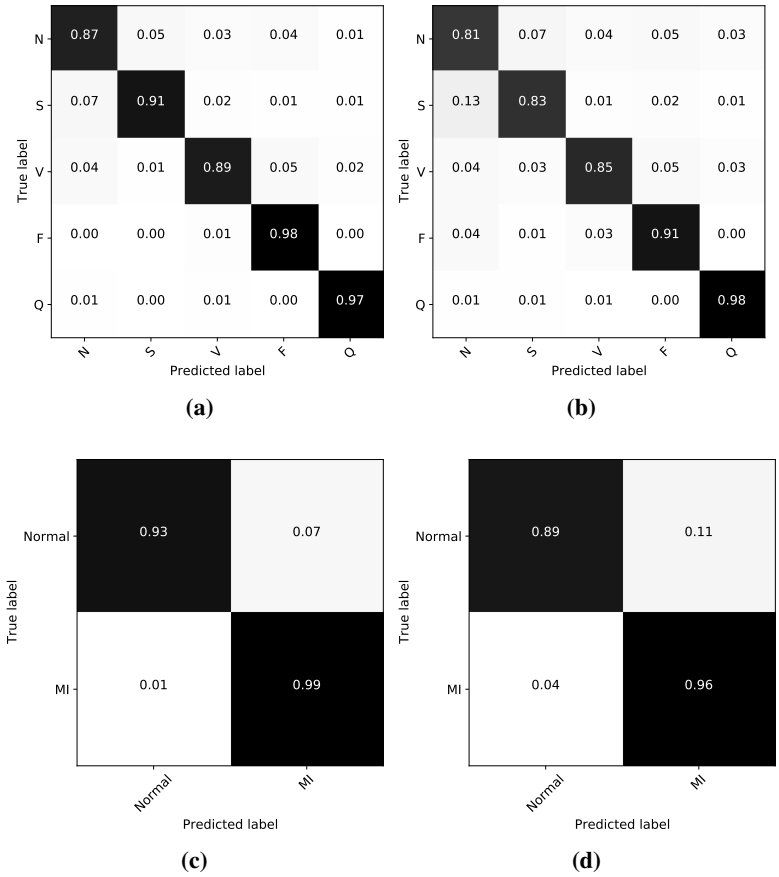


Figure 4: : Confusion matrix for heartbeat classification on the test set. Numbers inside blocks are number of samples classified in each category normalized by the total number of samples and rounded to two digits. (a), Confusion matrix of the CNN network for the MIT-BIH dataset. (b), Confusion matrix of the DCEC network with $d = 30$ for the MIT-BIH dataset. (c), Confusion matrix of the CNN network for the PTB dataset. (b), Confusion matrix of the DCEC network with $d = 40$ for the PTB dataset.

Dimensionality of the latent space. We study the robustness of our method to the dimensionality d of the latent space. For this experiment, we vary d between 10 and 60 and measure the accuracy on the PTB and MIT-BIH datasets. The results are shown in figure 3.

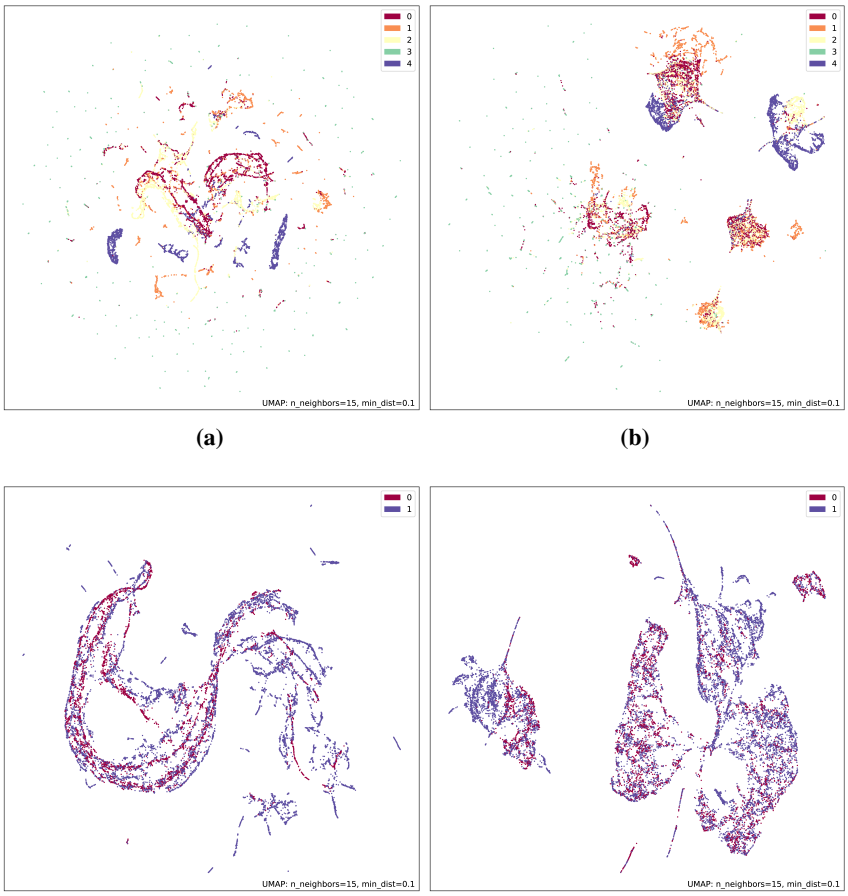


Figure 5: UMAP visualizations [9] of both datasets. (a), Raw MIT-BIH dataset. (originally in $\mathbb{R}^{187}$). (b), An embedding of the MIT-BIH dataset into $\mathbb{R}^{30}$ yielding highest classification accuracy. (c), Raw PTB dataset. (originally in $\mathbb{R}^{187}$). (b), An embedding of the PTB dataset into $\mathbb{R}^{40}$ yielding highest classification accuracy. Best viewed in color.

The results yield that the accuracy of DCEC gradually increases, reaches a maximum, and then decreases as the dimensionality d increases. Although one of the main functions of the DCEC method is automatic feature extraction and dimensionality reduction, when d is too small, the extracted features learned by the encoder are not sufficient for classification due to a higher reconstruction error. When d becomes higher, some of those extracted features are irrelevant yielding overfitting.

The choice of latent space dimensions depends on the complexity of the problem. Unfortunately, there is no rule of thumb for selecting d. A good approach would be starting with an overcomplete autoencoder and gradually lowering the latent space dimension until the accuracy starting to decrease.

Visualization. A UMAP [9] visualization of raw and embedded data for both datasets is provided in figure 5. We provided a visualization for both datasets showing the original dataset, the dataset embedded by DCEC into $\mathbb{R}^{10}, \mathbb{R}^{20}, \mathbb{R}^{30}, \mathbb{R}^{40}, \mathbb{R}^{50}$ and $\mathbb{R}^{60}$ in the appendix (figures 6 and 7).

As shown in the figure, both datasets are difficult to cluster while the DCEC produces slightly more clearly separated clusters for the MIT-BIH dataset.

4.3 Conclusion

In this project, we have compared the traditional CNN method with a DCEC method of dimensionality reduction and clustering prior to classification in the task of ECG heartbeat classification. According to the results, both methods are able to make predictions on both datasets with accuracies comparable to the baseline. However, the CNN

method produced better accuracy. Furthermore, we visualized the learned latent space using UMAP and illustrated the effectiveness of the proposed DCEC approach. The future work includes conducting more experiments on high dimensional time-series and image datasets and exploring more advanced networks.

References

[1] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? *ICDT 1999. LNCS*, 1540, 12 1997.

[2] Kreiseler D. Bousseljot R. and Schnabel A. Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet. *Biomedizinische Technik / Biomedical Engineering*, 40, 1995-01-01.

[3] Association for the Advancement of Medical Instrumentation and American National Standards Institute. *Testing and Reporting Performance Results of Cardiac Rhythm and ST-segment Measurement Algorithms*. ANSI/AAMI. The Association, 1999. ISBN 9781570201165. URL `https://books.google.co.il/books?id=gzPdtgAACAAJ`.

[4] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[5] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, pages 373–382, Cham, 2017. Springer International Publishing. ISBN 978-3-319-70096-0.

[6] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. ECG heartbeat classification: A deep transferable representation. *CoRR*, abs/1805.00794, 2018. URL `http://arxiv.org/abs/1805.00794`.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[8] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[9] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[10] G. B. Moody and R. G. Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

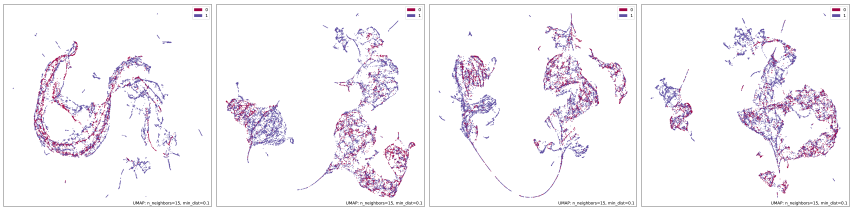
[11] S. M. Mousavi, W. Zhu, W. Ellsworth, and G. Beroza. Unsupervised clustering of seismic signals using deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 16(11):1693–1697, 2019.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

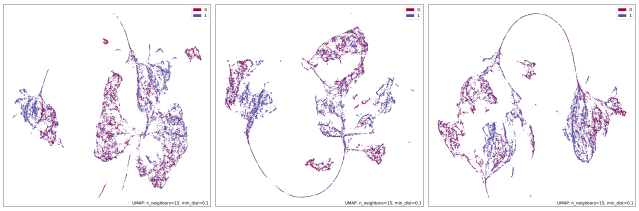
[13] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[14] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2015.

Appendix

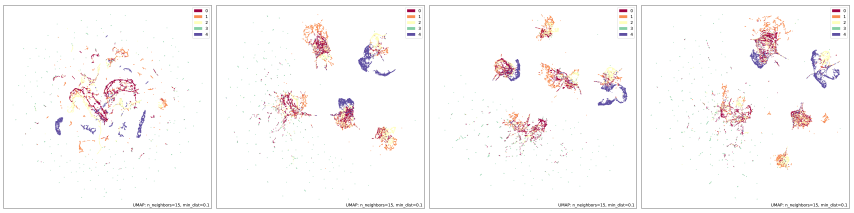


(a) Raw **(b)** $d = 10$ **(c)** $d = 20$ **(d)** $d = 30$

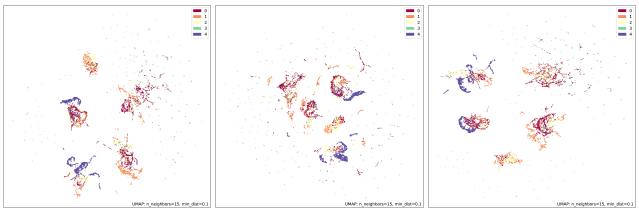


(e) $d = 40$ **(f)** $d = 50$ **(g)** $d = 60$

Figure 6: UMAP visualizations [9] to demonstrate the effect of joint dimensionality reduction and clustering on the embedding. (a), Raw PTB dataset. (originally in $\mathbb{R}^{187}$). (b), An embedding of the PTB dataset into $\mathbb{R}^{10}$. (c), An embedding of the PTB dataset into $\mathbb{R}^{20}$. (d), An embedding of the PTB dataset into $\mathbb{R}^{30}$. (e), An embedding of the PTB dataset into $\mathbb{R}^{40}$. (f), An embedding of the PTB dataset into $\mathbb{R}^{50}$. (g), An embedding of the PTB dataset into $\mathbb{R}^{60}$. Notice that this dataset is difficult to cluster, yet the data is most separable when $d = 40$ which is when the accuracy is highest, as well. Best viewed in color.



(a) Raw **(b)** $d = 10$ **(c)** $d = 20$ **(d)** $d = 30$



(e) $d = 40$ **(f)** $d = 50$ **(g)** $d = 60$

Figure 7: UMAP visualizations [9] to demonstrate the effect of joint dimensionality reduction and clustering on the embedding. (a), Raw MIT-BIH dataset. (originally in $\mathbb{R}^{187}$). (b), An embedding of the MIT-BIH dataset into $\mathbb{R}^{10}$. (c), An embedding of the MIT-BIH dataset into $\mathbb{R}^{20}$. (d), An embedding of the MIT-BIH dataset into $\mathbb{R}^{30}$. (e), An embedding of the MIT-BIH dataset into $\mathbb{R}^{40}$. (f), An embedding of the MIT-BIH dataset into $\mathbb{R}^{50}$. (g), An embedding of the MIT-BIH dataset into $\mathbb{R}^{60}$. Notice that this dataset has better clustering results. The data is most separable when $d = 30$. Again, this is when the accuracy is highest, as well. Best viewed in color.