MULTILABEL 12-LEAD ELECTROCARDIOGRAM CLASSIFICATION USING BEAT TO SEQUENCE AUTOENCODERS

Alexander William Wong, Amir Salimi, Abram Hindle

University of Alberta
Department of Computing Science
Edmonton, Alberta, Canada

Sunil Vasu Kalmady, Padma Kaul

University of Alberta
Canadian VIGOUR Centre
Edmonton, Alberta, Canada

ABSTRACT

The 12-lead electrocardiogram (ECG) measures the electrical activity of the heart for physicians to use in diagnosing cardiac disorders. This paper investigates the multi-label, multi-class classification of ECG records into one or more of 27 possible medical diagnoses. Our multi-step approach uses conventional physiological algorithms for segmentation of heartbeats from the baseline signals. We stack a heartbeat autoencoder over heartbeat windows to make embeddings, then we encode this sequence of embeddings to make an ECG embedding which we then classify on. We utilize the public dataset of 43,101 available ECG records provided by the *PhysioNet/CinC 2020 challenge*, performing repeated random subsampling and splitting the available records into 80% training, 10% validation, and 10% test splits, 20 times. We attain a mean test split challenge score of 0.248 with an overall macro F_1 score of 0.260 across the 27 labels.

***Index Terms*—** electrocardiogram, signal autoencoder, signal embedding, multi-label classification, PhysioNet/CinC

1. INTRODUCTION

Although the electrocardiogram (ECG) is an effective tool for detecting cardiac diseases, the analysis of the ECG is a specialized skill requiring training and human over-reading of computerized interpretations. Our work extends the *PhysioNet/CinC 2020 Challenge* [1] involving multi-label classification of ECGs, which are 12 channel 500-1000Hz signals with 27 labels indicating cardiologist diagnoses. Our novel component is the use of sequentially windowed embeddings to classify the ECGs. We train a two component signal autoencoder algorithm, encoding the heartbeat windows, then the sequence of window embeddings, before using the sequence bottleneck embedding for classification. We extend and compare against our prior work by learning autoencoded features rather than using manually engineered features [2].

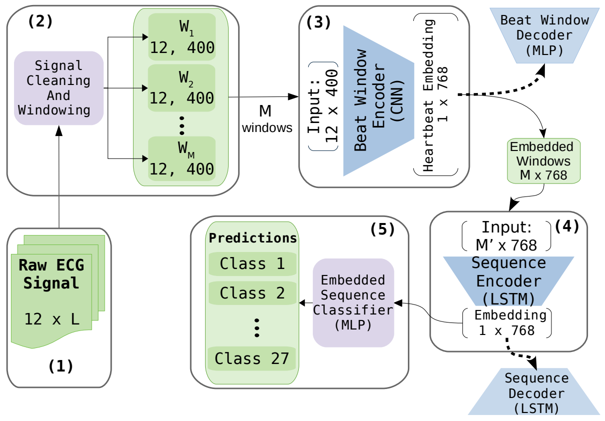

Fig. 1: Methodology overview. From (1) $\mathcal{L}$-sized ECG signal, we (2) extract $\mathcal{M}$ heartbeat windows prior to (3) heartbeat & (4) sequence autoencoding. Finally we (5) pass our sequence embedding to the classifier to output our 27 predictions.

2. RELATED WORK

We are inspired by prior work that uses autoencoders to generate features for signal classification [3, 4]. Recent advancements in machine learning and available data have heralded an influx of multi-lead ECG classification algorithms [5, 6, 7, 8, 9, 2]. We extend our prior work by using neural networks over feature engineering with gradient boosted tree classifiers [2]. Despite the large improvements in automated ECG classification, trained human over-reading and cardiologist confirmation is still mandated during use in the clinical setting [10, 11].

2.1. Challenge Dataset and Task Specification

Refer to Perez Alday *et al.* [1] for the ECG sources and competition rules. The challenge provides 43,101 ECG records where each record is labelled as one or more of 111 possible diagnoses. The evaluated 27 label subset is shown in Table 1.

We reuse the scoring function in preparation for the 2021 challenge, which extends this task and adds a 2-lead classification variant. We want to maximize the following scoring

Table 1: Evaluated labels, count and percentage in dataset.

Abbr.	Diagnosis	Count (%)
IAVB	1st degree av block	2394 (5.6%)
AF	atrial fibrillation	3475 (8.0%)
AFL	atrial flutter	314 (0.7%)
Brady	bradycardia	288 (0.7%)
CRBBB	complete right bundle branch block	683 (1.6%)
IRBBB	incomplete right bundle branch block	1611 (3.7%)
LAnFB	left anterior fascicular block	1806 (4.2%)
LAD	left axis deviation	6086 (14.1%)
LBBB	left bundle branch block	1041 (2.4%)
LQRSV	low QRS voltages	556 (1.3%)
NSIVCB	nonspecific intraventricular conduction	997 (2.3%)
PR	pacing rhythm	299 (0.7%)
PAC	premature atrial contraction	1729 (4.0%)
PVC	premature ventricular contractions	188 (0.4%)
LPR	prolonged PR interval	340 (0.7%)
LQT	prolonged QT interval	1513 (3.5%)
QAb	Q wave abnormal	1013 (2.4%)
RAD	right axis deviation	427 (1.0%)
RBBB	right bundle branch block	2402 (5.6%)
SA	sinus arrhythmia	1240 (2.9%)
SB	sinus bradycardia	2359 (5.5%)
SNR	sinus rhythm	20846 (48.4%)
STach	sinus tachycardia	2402 (5.6%)
SVPB	supraventricular premature beats	215 (0.5%)
TAb	T wave abnormal	4673 (10.8%)
TInv	T wave inversion	1112 (2.6%)
VPB	ventricular premature beats	365 (0.8%)

function: $\sum_{ij} w_{ij} a_{ij}$. Provided predictions $C = \{c_i\}$, we create a confusion matrix $A = [a_{ij}]$ where a_{ij} indicates a record classified as class c_i belongs to class c_j. The weights $W = [w_{ij}]$, shown in Figure 2, are challenge defined to provide partial reward for incorrect predictions.

3. METHODOLOGY

We propose a staged neural network architecture for autoencoding the extracted heartbeats, autoencoding the sequence of heartbeat embeddings, and training a multi layer perceptron classifier. An overview is shown in Figure 1. Using 20 repeated random subsampling, we split our 43,101 available ECG records into 80% training, 10% validation, and 10% test splits. No label proportion stratification of the splits occurred.

3.1. Signal Preprocessing

We use the *NeuroKit2* (`v0.0.40`) neurophysiological signal processing library [12] to annotate our ECG signals and the *SciPy* (`v1.5.2`) family of Python packages for signal filtering and statistical tests [13]. The ECG cleaning approach removes slow drift and DC offset using a Butterworth highpass filter (5Hz, $Q = 0.5$) then smooths the signal using a moving average kernel of 0.02 seconds. The R-peaks, or heartbeat locations, are annotated for each of our 12 signals.

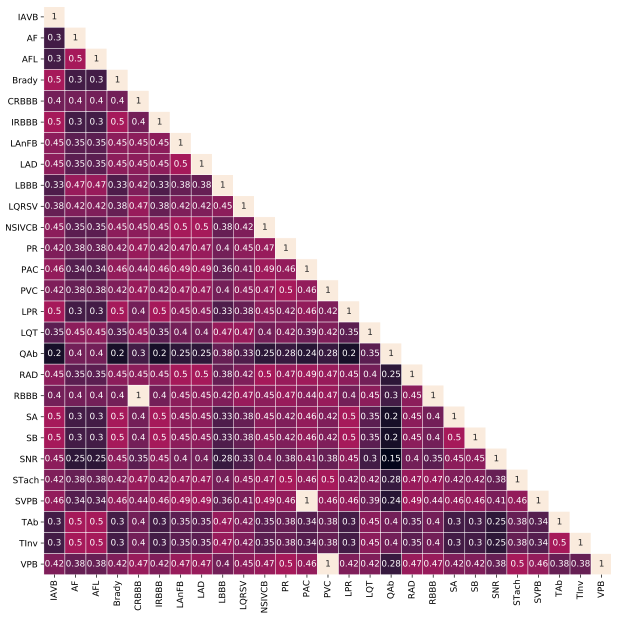

Fig. 2: Evaluation scoring function weights per label.

Due to variable quality of sensor placements or noise artifacts caused by patient movements, the independent heartbeat annotations per channel may not be congruent within the ECG record. We address this limitation with a kernel density estimation function fitted to the indices of all the R-peak annotations. The bandwidth is the mean channel-wise heart rate multiplied by a $\frac{1}{4}$ scaling factor. Next, we determine the peaks in the R-peak probability densities by finding all local maxima relative to their neighboring values. We apply a cutoff threshold, dropping peaks that are over two standard deviations away from the mean peak value.

Given the overall R-peak indices for our 12-channel signal, we resample the entire signal such that the mean distance between each R-peak is 400 samples. We slice windows of size 400, positioning the R-peak to occur at one-third of the length, and ignoring windows that do not contain 400 samples. An l_2 normalization step is applied on all remaining windows. We use the Breunig *et al.* local outlier factor algorithm [14] to find the most abnormal heartbeat in the ECG.

Our signal processing steps allow us to extract $\mathcal{M}$ normalized, fixed size heart beat windows from arbitrary length 12-channel ECG records. A full example of the entire signal processing and windowing procedure is provided in Figure 3.

3.2. Heartbeat Autoencoder

We rely on the dimensionality reducing properties of autoencoders [15] to encode our heartbeat windows into an embed-

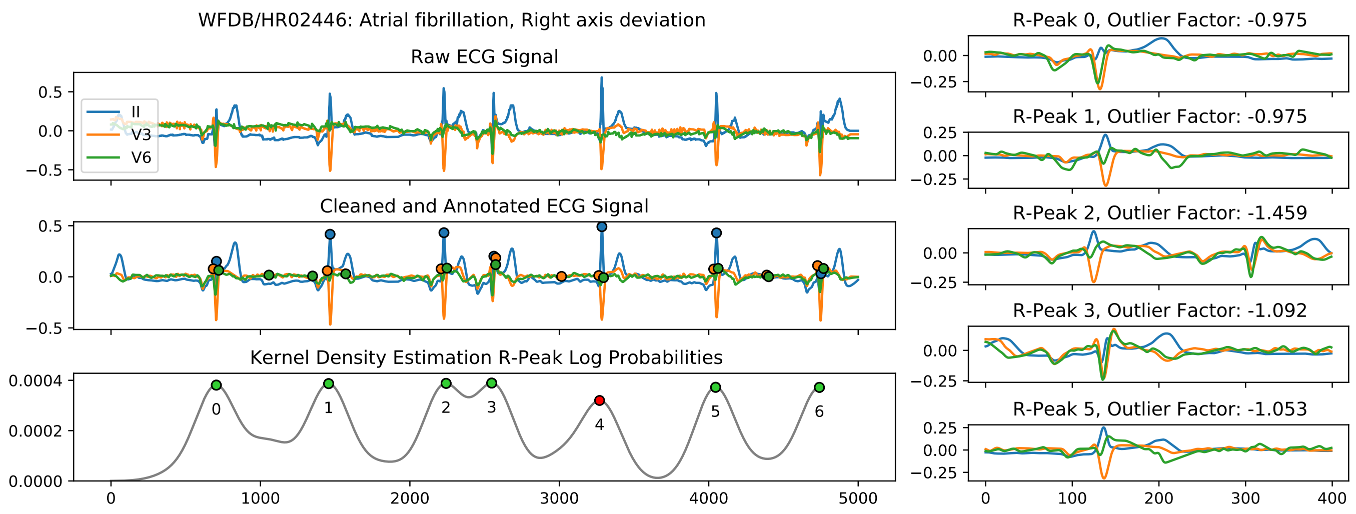

Fig. 3: Signal processing from the raw signal to the annotated intermediary signal, to output heartbeat windows. Window 4 is dropped due to the cutoff threshold. Window 6 is dropped due to insufficient window size. Window 2 is the most abnormal heartbeat with the minimum outlier factor of the given windows. Only 3 of the 12 leads (`II`, `V3`, and `V6`) are shown for clarity.

ding, then concatenate our embeddings to get a representation of the overall ECG. Our heartbeat autoencoder converts our 12×400 heartbeat windows into embeddings of size 768.

Table 2: Heartbeat autoencoder neural network architecture.

Block	Modules	Output Shape
Enc1	Conv1d(12, 16, 164), BatchNorm1d(16), ReLU(), Dropout(p=0.1)	$[\mathcal{M}, 16, 237]$
Enc2	Conv1d(16, 20, 128), BatchNorm1d(20), ReLU(), Dropout(p=0.1)	$[\mathcal{M}, 20, 110]$
Enc3	Conv1d(20, 24, 64), BatchNorm1d(24), ReLU(), Dropout(p=0.1)	$[\mathcal{M}, 24, 47]$
Enc4	Flatten(), Linear(1128, 768)	$[\mathcal{M}, 768]$
Dec1	Linear(768, 1024), ReLU(), Dropout(p=0.1)	$[\mathcal{M}, 1024]$
Dec2	Linear(1024, 4800), View(400, 12), Tanh()	$[\mathcal{M}, 400, 12]$

The encoder has 970,420 trainable parameters among three convolutional blocks followed by a linear layer to generate the embedding. Each block contains a convolutional layer followed by batch normalization, a ReLU activation, and a dropout normalization layer. The decoder architecture, with 5,707,456 trainable parameters, contains two linear layers separated by a ReLU activation and a dropout normalization layer with a Tanh nonlinearity applied to the outputs. See Table 2 for the heartbeat autoencoder architecture.

We use stochastic gradient descent (SGD) with 0.9 momentum, to optimize our mean square error (MSE) objective. We cyclically oscillate our learning rate between 1.0×10^{-3} and 1.0×10^{-5}. Training stops if the validation loss fails to attain a new minimum value after 3 epochs or after 100 epochs.

3.3. Embedding Sequence Autoencoder and Classifier

The number of heartbeats extracted from an ECG record varies between 2 beats up to over 3,000 beats. We limit this sequence length, capping the number of heartbeat embeddings used $\mathcal{M}'$ to 20. Beginning with the abnormal heartbeat, we iteratively pick the rest of the candidate heartbeats by prepending the left neighbors and appending the right neighbors. We stop when the neighbors are exhausted or 20 heartbeats are chosen. For records with fewer than 20 beats, empty positions are masked and do not contribute to the loss.

Table 3: Sequence autoencoder and classifier architecture.

Block	Modules	Output Shape
Encoder	LSTM(768, 768, num_layers=2, dropout=0.1)	Hidden [768]
Decoder	LSTM(768, 768, num_layers=2, dropout=0.1)	Sequence $[\mathcal{M}', 768]$
Classifier	Linear(768, 256), ReLU() Dropout(p=0.1), Linear(256, 27)	Predictions [27]

The sequence autoencoder is symmetrical, with identical encoder and decoder architectures. A two layer LSTM module with input and hidden sizes set to 768 and dropout of 0.1 is used, containing 9,449,472 parameters. It encodes our $\mathcal{M}' \times 768$ heartbeat embeddings into a bottleneck of size 768. A multilayer perceptron consisting of two linear layers,

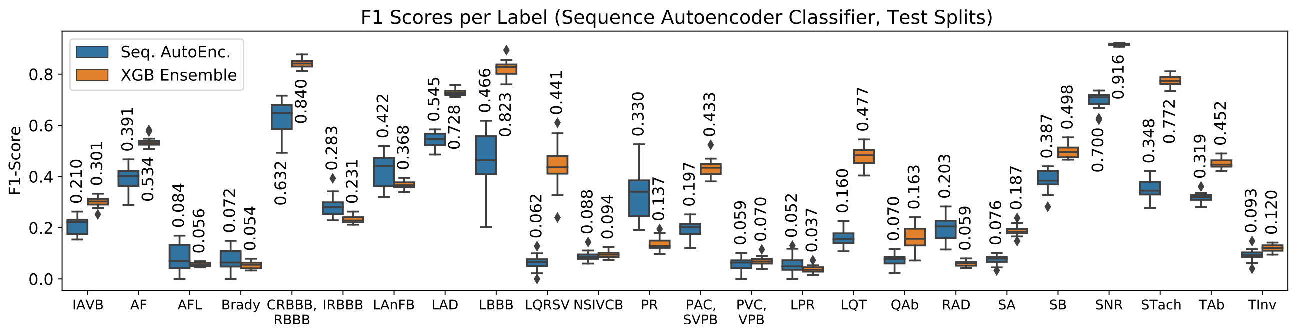

Fig. 4: Test set F_1 scores of all 27 labels compared with our prior XGBoost ensemble method [2]. Mean values annotated.

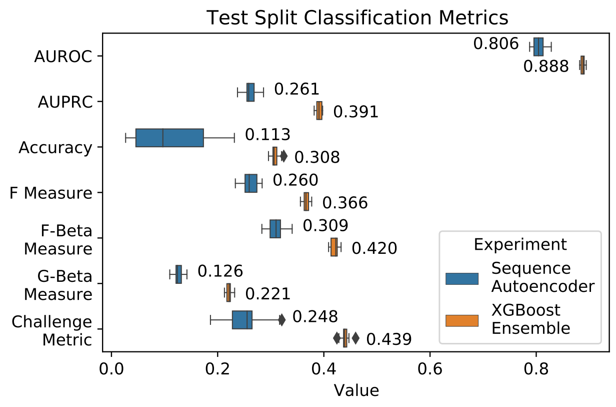

Fig. 5: Classification metric summary of 20 experiments compared to XGBoost ensembles [2]. Mean values annotated.

separated by a ReLU and dropout layer ($p = 0.1$) takes the sequence embedding of 768, computes a hidden representation of 256, and outputs 27 label probabilities. Our classifier has 203,803 parameters. See Table 3 for architecture design.

To mitigate internal validity risk the training, validation, and test splits are reused from the heartbeat autoencoder experiment. Our pre-trained heartbeat encoder is frozen and does not update during the training of the sequence autoencoder. We train the sequence autoencoder and classifier simultaneously using SGD and a cyclic learning rate. Our overall loss function is the autoencoder MSE loss added to the binary cross entropy (BCE) classifier loss. We scale the BCE weights to the count of negative samples over the positive samples in the training set split.

Training stops if the validation set challenge score fails to improve after 30 epochs or 200 epochs pass. We use the highest validation set scoring model from all epochs and calculate the challenge metrics on the test set split, setting thresholds to maximize the training data receiver operating characteristic.

4. RESULTS AND DISCUSSION

We compare our results with our prior XGBoost ensemble classifier [2]. Label-wise test F_1 scores can be found in Figure 4. Using the Wilcoxon signed rank test, our autoencoder F_1 means statistically outperform our prior work in detecting incomplete right bundle branch block (IRBBB, $p = 1.9 \times 10^{-6}$), left anterior fascicular block (LAnFB, $p = 9.4 \times 10^{-3}$), pacing rhythm (PR, $p = 1.9 \times 10^{-6}$), and right axis deviation (RAD, $p = 1.9 \times 10^{-6}$).

Overall test classification metrics is shown in Figure 5. Our methodology achieves a test split mean *PhysioNet/CinC 2020 Challenge* score of 0.248, AUROC of 0.806, AUPRC of 0.261, accuracy of 0.113, macro F_1 score of 0.260, F_β of 0.309, and G_β of 0.126 using $\beta = 2$. Our autoencoder is worse than our shallow classifier on all summary metrics. Our results cannot be compared with official rankings because the challenge evaluates the algorithms on secret hold-out test sets.

Our methodology trains a neural network using the general shapes of heartbeat windows to indirectly model the overall signal as consecutive heartbeats. Because of variable distances of the R-peaks within an ECG record, portions of the ECG signal not bounded within heartbeat windows are dropped. Due to resampling the signal to ensure heartbeat windows are 400 samples long, we also lose heart rate information like average heart rate and changes in heart rate over time. This loss of ECG temporal information is a contributing factor to the worse results when compared to our prior work. Additionally, because of the l_2 normalization of the heartbeat windows, we also do not capture the original signal amplitudes and voltage changes. Future work should expand on our findings to incorporate temporal heart rate features, raw amplitudes, and continuous full signal characteristics.

5. CONCLUSION

Using a signal processing and heartbeat window extraction preprocessing step, we train heartbeat autoencoders to be fed into ECG sequence autoencoders before training a multi-label perceptron to classify 27 heart conditions. We run 20 independent experiments, randomly sampling our available dataset into 80% training, 10% validation, and 10% test set splits. Our methodology achieves a mean unofficial test challenge score of 0.248 with an overall macro F_1 score of 0.260.

6. REFERENCES

[1] Erick A. Perez Alday, Annie Gu, Amit Shah, Chad Robichaux, An-Kwok Ian Wong, Chengyu Liu, Feifei Liu, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Qiao Li, Ashish Sharma, Gari D. Clifford, and Matthew A Reyna, "Classification of 12-lead ECGs: the PhysioNet/Computing in Cardiology Challenge 2020," *Physiological Measurement*, 2020, In Press.

[2] Alexander W Wong, Weijie Sun, Sunil V Kalmady, Padma Kaul, and Abram Hindle, "Multilabel 12-lead electrocardiogram classification using gradient boosting tree ensemble," in *2020 Computing in Cardiology (CinC) PhysioNet Challenge*, 2020, pp. 1–4.

[3] B. Hou, J. Yang, P. Wang, and R. Yan, "Lstm-based auto-encoder model for ecg arrhythmias classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1232–1240, 2020.

[4] A. Gogna, A. Majumdar, and R. Ward, "Semi-supervised stacked label consistent autoencoder for reconstruction and analysis of biomedical signals," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2196–2205, 2017.

[5] Antônio H. Ribeiro, Manoel Horta Ribeiro, Gabriela M. M. Paixão, Derick M. Oliveira, Paulo R. Gomes, Jéssica A. Canazart, Milton P. S. Ferreira, Carl R. Andersson, Peter W. Macfarlane, Wagner Meira Jr., Thomas B. Schön, and Antonio Luiz P. Ribeiro, "Automatic diagnosis of the 12-lead ECG using a deep neural network," *Nature Communications*, vol. 11, no. 1760, April 2020.

[6] Tsai-Min Chen, Chih-Han Huang, Edward S.C. Shih, Yu-Feng Hu, and Ming-Jing Hwang, "Detection and classification of cardiac arrhythmias by a challenge-best deep learning neural network model," *iScience*, vol. 23, no. 3, pp. 100886, 2020.

[7] M. Wasimuddin, K. Elleithy, A. S. Abuzneid, M. Faezipour, and O. Abuzaghleh, "Stages-based ecg signal analysis from traditional signal processing to machine learning approaches: A survey," *IEEE Access*, vol. 8, pp. 177782–177803, 2020.

[8] Ulas Baran Baloglu, Muhammed Talo, Ozal Yildirim, Ru San Tan, and U Rajendra Acharya, "Classification of myocardial infarction with multi-lead ecg signals and deep cnn," *Pattern Recognition Letters*, vol. 122, pp. 23 – 30, 2019.

[9] J. Niu, Y. Tang, Z. Sun, and W. Zhang, "Inter-patient ecg classification with symbolic representations and multi-perspective convolutional neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 5, pp. 1321–1332, 2020.

[10] Stephen W. Smith, Brooks Walsh, Ken Grauer, Kyuhyun Wang, Jeremy Rapin, Jia Li, William Fennell, and Pierre Taboulet, "A deep neural network learning algorithm outperforms a conventional algorithm for emergency department electrocardiogram interpretation," *Journal of Electrocardiology*, vol. 52, pp. 88 – 95, 2019.

[11] John E. Madias, "Computerized interpretation of electrocardiograms: Taking stock and implementing new knowledge," *Journal of Electrocardiology*, vol. 51, no. 3, pp. 413 – 415, 2018.

[12] Dominique Makowski, Tam Pham, Zen J. Lau, Jan C. Brammer, François Lespinasse, Hung Pham, Christopher Schölzel, and Annabel S H Chen, "NeuroKit2: A python toolbox for neurophysiological signal processing," .

[13] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[14] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander, "LOF: Identifying Density-Based Local Outliers," *SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, May 2000.

[15] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.