Advanced Metrics for Filtering Non-Learnable Data Points in Signal Processing*

Rishit Bhutra (210857)
Sameer Ahmad (210912)
Shubham Jangid (211022)
Bhavesh Shukla (210266)

Abstract

Modern deep learning models often waste resources on training with data points that are redundant or uninformative. In this project, we adapt and benchmark methods for identifying non-learnable data in the context of ECG signals using metrics from the Reducible Holdout Loss (RHO-Loss) framework, Autoencoders (AEs), and Variational Autoencoders (VAEs). Our experiments demonstrate the effectiveness of ELBO and RHO-Score in improving signal anomaly detection and training efficiency.

1 Problem Description and Motivation

Large datasets are often noisy and contain redundant information. Identifying and filtering out such non-learnable data can significantly improve training efficiency and model generalization. In this work, we extend the RHO-Loss framework to time-series signal data, focusing on ECG signals from the MIT-BIH Arrhythmia database. Our aim is to compare several metrics in their ability to identify outlier ECG segments.

2 Literature Review and Prior Work

Our work is motivated by the paper **Prioritized Training on Learnable, Worthwhile, and Untrained Data Points**. The authors propose techniques to **rank** and **prioritize data points** during training, showing that intelligently selected samples can significantly reduce training time without sacrificing performance.

Two prevalent methods for sample prioritization include:

- **Filtering out noisy data points**, which are often not worth learning.
- Focusing on **high-loss examples**, which are deemed more informative.

However, both methods come with limitations, particularly in handling varying data distributions.

3 Novelty of Our Work

Our work extends the foundational ideas from the original paper, applying them beyond **computer vision tasks** to address challenges in more complex domains, specifically **Natural Language Processing (NLP)** and **Signal Processing**. While computer vision tasks often allow for easier identification and removal of **noisy data points**, we aim to adapt this methodology to the aforementioned fields,

*Code available at `https://github.com/bhavesh932003/CS772-Project` where noisy data is often more difficult to isolate and manage. This will involve testing our approach on standard datasets from NLP and Signal Processing, with a particular focus on efficient **batching** and **load balancing strategies** to handle the **computational cost** effectively.

In addition to extending the approach to new domains, we propose exploring alternative metrics and loss functions to enhance the filtering of **non-learnable data points**. Beyond the standard **RHO Loss**, we plan to integrate **probabilistic autoencoders** to develop more sophisticated metrics that allow finer control over the ranking of data points, especially in **signal processing** tasks. As part of this, we introduce a dynamically scaled RHO-score, calculated as 1% of the maximum MSE, to improve the separability of the score between outliers and inliers. Furthermore, we plan to leverage **ELBO scores** from Variational Autoencoders (VAEs) for anomaly detection in ECG signals, comparing the effectiveness of these scores with traditional AE-MSE and RHO-scores using multiple evaluation metrics. This combination of techniques will allow for a more robust, context-sensitive method of anomaly detection.

4 Tools and Software Used

We used the following tools and frameworks throughout the project:

- **Python** for scripting and implementation.
- **PyTorch** for deep learning model development.
- **NumPy** and **Pandas** for data manipulation and preprocessing.
- **Matplotlib** for data visualization and plotting.
- **Scikit-learn** for precision-recall metrics and evaluation.
- **PyTorch Lightning** to enable efficient and GPU-accelerated training.
- ECG data was preprocessed using custom scripts written in Python.

5 Experimental Results

5.1 Adapting to Signal Processing

5.1.1 Data Description

- Dataset: Google Speech Commands v0.02 (35 spoken command classes)
- Sampling Rate: 16 kHz, mono-channel
- Clip Duration: 1 second (16,000 samples)
- Preprocessing: Padded or trimmed to 16,000 samples per clip
- Subsets: Training, Validation, and Testing (standard list-based splits)
- Normalization: Per-sample normalization (handled implicitly by waveform padding/trimming)
- Total Training Clips: $\sim$84,843 (excluding validation/testing)
- Labels: 35 keyword classes (e.g., “yes”, “no”, “up”, “down”, “left”, “right”, etc.)
- Class Imbalance: Mild imbalance among keyword frequencies

5.1.2 Model

We implemented a 1D Convolutional Neural Network (CNN) for speech command classification using raw audio waveforms:

- **AudioCNN:** A lightweight 1D CNN designed to process 1-second audio waveforms (sampled at 16kHz) from the SpeechCommands dataset. The architecture is as follows:
  - **Input:** 1×16000 mono waveform
  - **Conv1:** $1 \rightarrow 8$ channels, kernel size 9, stride 1, ReLU, followed by MaxPool1D (kernel size 4)

- **Conv2:** 8 → 16 channels, kernel size 9, stride 1, ReLU, followed by MaxPool1D (kernel size 4)
- **Flatten:** Dynamic flattening to feed forward layers
- **FC1:** Fully connected layer with 64 units, ReLU
- **FC2:** Output layer with units equal to the number of speech command classes

The model is trained using supervised classification to map input waveforms to their corresponding spoken labels.

5.1.3 Training Details

- Loss Function: Custom RHO-based loss[2], replacing standard Cross-Entropy (not shown in code snippet)
- Optimizer: Adam, learning rate 10^{-3}
- Batch size: 64, Epochs: 5

5.1.4 Result

Table 1: Training loss per epoch for AudioCNN model

Epoch	Average Loss
1	2.7192
2	1.9678
3	1.6517
4	1.4528
5	1.3069

The classification accuracy on the test set after training was **51.00%**.

5.2 Adapting to Natural Language Processing

5.2.1 Data Description

- Dataset: GLUE SST-2 (Stanford Sentiment Treebank v2) for binary sentiment classification
- Text Format: Movie review sentences (positive or negative sentiment)
- Tokenizer: `distilbert-base-uncased` from HuggingFace Transformers
- Max Sequence Length: 64 tokens
- Padding/Truncation: Applied to all sequences during tokenization
- Subsets: Training (split 80/20 for train and holdout), Validation (standard SST-2 validation set)
- Input Features: `input_ids`, `attention_mask`, and `label`
- Total Training Examples: 52,320 sentences (after 80/20 split from 65,000 original training set)
- Labels: Binary (0 = negative, 1 = positive)
- Imbalance: Balanced class distribution across splits

5.2.2 Model

We implemented a compact transformer architecture for binary text classification:

- **UltraCompactTransformer:** A lightweight transformer model based on a trimmed-down BERT configuration:

[2]The RHO loss is designed to scale the MSE to enhance anomaly sensitivity.

- Hidden Size: 72
- Layers: 2 Transformer encoder layers
- Attention Heads: 2 per layer
- Intermediate Size: 144 (for feedforward layers)
- Max Position Embeddings: 64
- Embedding Layer: Frozen to reduce parameter count
- Output: Classification head (Linear layer) maps [CLS] token to 2 logits

The model contains approximately 89,930 trainable parameters, making it efficient for rapid experimentation on resource-constrained hardware.

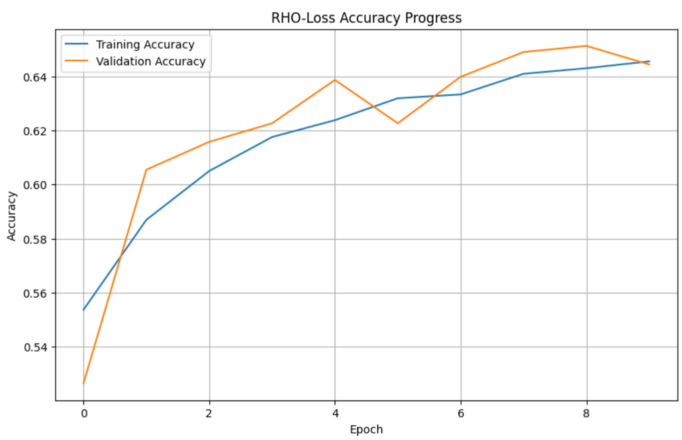
5.2.3 Training Details

- Phase 1 (IL Estimation): Trained on holdout split for 3 epochs using AdamW optimizer (learning rate 2×10^{-4}) and Cross-Entropy loss to estimate instance-level (IL) difficulty.
- Phase 2 (RHO-Based Training): Trained the full model on the training split using a curriculum derived from IL losses:
  - Loss Function: Cross-Entropy on samples with top-k *reducible loss* (difference between current and IL loss)
  - Optimizer: AdamW, learning rate 5×10^{-5}
  - Batch Size: 32
  - Epochs: 10
  - Gradient Clipping: 1.0
- Selection Strategy: At each batch, we selected examples with the highest reducible loss to prioritize learning on uncertain samples.
- Training Objective: Reduce overfitting and accelerate convergence by leveraging implicit instance difficulty.

5.2.4 Result

Table 2: Training and Validation Metrics per Epoch (SST-2)

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1	0.6857	0.6896	0.5536	0.5264
2	0.6697	0.6625	0.5869	0.6055
3	0.6581	0.6572	0.6050	0.6158
4	0.6512	0.6467	0.6176	0.6227
5	0.6449	0.6377	0.6238	0.6388
6	0.6404	0.6324	0.6320	0.6227
7	0.6382	0.6281	0.6334	0.6399
8	0.6320	0.6214	0.6410	0.6491
9	0.6295	0.6178	0.6431	0.6514
10	0.6266	0.6190	0.6456	0.6445



5.3 Alternative Metrics and Loss Functions

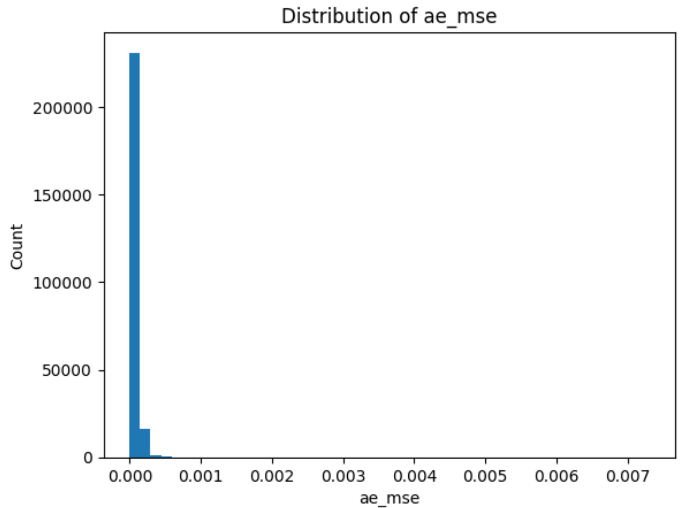
5.3.1 Data Description

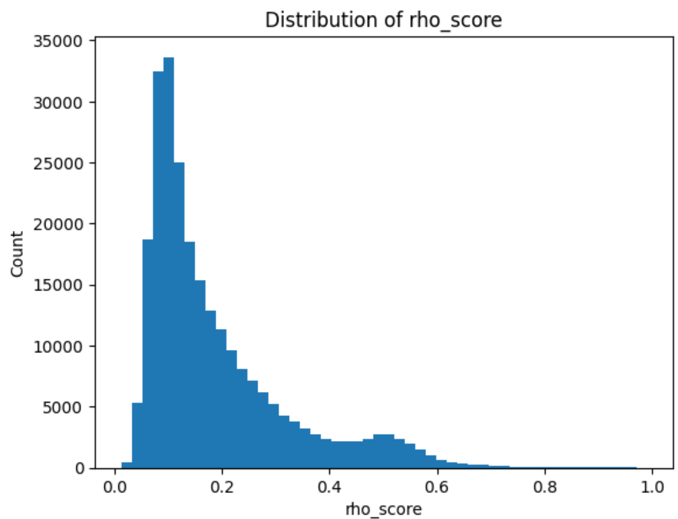
- Dataset: MIT-BIH Arrhythmia (48 recordings)
- Windowing: 250-sample windows, 125-sample stride
- Normalized to [0,1] per recording
- Total windows: 249,552
- Arrhythmic: 71,475 (28.6%), Normal: 178,077 (71.4%)

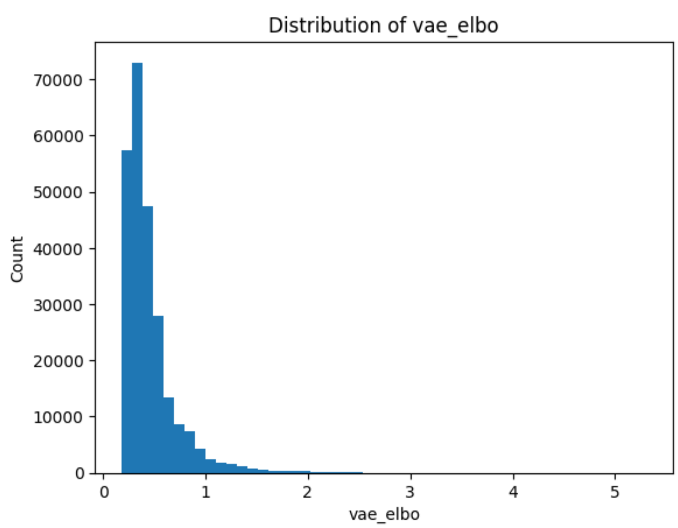
5.3.2 Descriptive Statistics

Table 3: Summary statistics for key metrics

Metric	AE-MSE	RHO-Score	VAE-ELBO
Mean	0.000051	0.1882	0.4474
Std Dev	0.000081	0.1338	0.2571
Min	0.000002	0.0132	0.1782
25%	0.000016	0.0944	0.2867
Median	0.000028	0.1397	0.3731
75%	0.000058	0.2375	0.5133
Max	0.007304	0.9901	5.3089


Distribution of ae_mse
Count
200000
150000
100000
50000
0
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
ae_mse


Distribution of rho_score
Count
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
rho_score


Distribution of vae_elbo
Count
70000
60000
50000
40000
30000
20000
10000
0
0
1
2
3
4
5
vae_elbo

5.3.3 Models

We implemented and evaluated three model architectures:

- **Autoencoder (AE):** A fully connected feedforward network with the architecture [$250 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 250$]. The AE is trained to minimize the Mean Squared Error (MSE) between input and reconstructed ECG windows, capturing reconstruction fidelity as a proxy for anomaly.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

  where x_i is the input value and $\hat{x}_i$ is the reconstructed value.

- **RHO-Autoencoder (RHO-AE):** Built on the same architecture as the standard AE, but trained using the RHO-score instead of raw MSE. The RHO-score is computed as

$$\rho_i = \frac{\text{MSE}_i}{\text{MSE}_i + \epsilon}$$

  where $\epsilon = 0.01 \times \max(\text{MSE})$. This formulation helps sharpen the separation between outliers and inliers by scaling the reconstruction error.

- **Variational Autoencoder (VAE):** A probabilistic autoencoder with encoder and decoder symmetric to the AE. The model maps input data to a latent Gaussian distribution (latent dimension = 32). It is trained to minimize the Evidence Lower Bound (ELBO), computed as the sum of reconstruction loss and weighted KL divergence between the approximate posterior and prior, i.e.,

$$\text{ELBO} = \text{Recon} + \beta \cdot \text{KL}$$

  with $\beta = 0.1$.

5.3.4 Training Details

- Epochs: 30, Batch size: 128
- Learning rate: 10^{-3}, VAE gradient clipping: 1.0

5.3.5 Results

Precision@5% Values: AE = 0.608, RHO = 0.606, VAE = 0.734

Table 4: Precision@k Scores

Top k%	AE-MSE	RHO-Score	VAE-ELBO
1	0.68	0.64	0.84
5	0.647	0.660	0.732
10	0.60	0.62	0.70
20	0.52	0.50	0.59

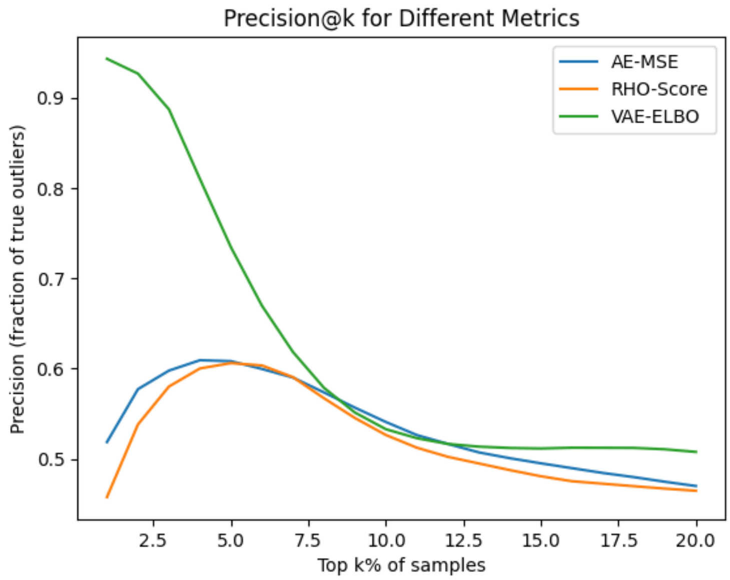
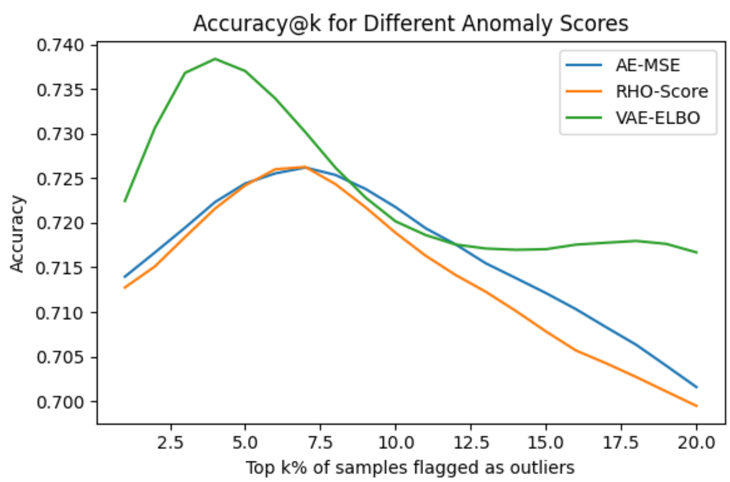


Table 5: PR-AUC Scores

Metric	PR-AUC
AE-MSE	0.71
RHO-Score	0.73
VAE-ELBO	0.82



6 Key Learnings

- RHO-Score helps distinguish learnable vs. non-learnable points.
- ELBO from VAEs gives a strong probabilistic basis for anomaly detection.
- Tailoring selection metrics to modality (e.g., ECG) matters.
- Visualization helps interpret model scoring.
- RHO-based loss boosts anomaly detection in ECG and speech but needs careful preprocessing.

- High reducible loss prioritization speeds NLP convergence, needing lightweight transformers.

7 Future Work

- Tune hyperparameters ρ and β
- Fuse ELBO and RHO-score using learned classifiers
- Apply to other signal domains like EEG and radar
- Build a live dashboard (e.g., using Streamlit)

8 Member Contributions

- **Rishit Bhutra** & **Bhavesh Shukla:** Alternative Metrics and Loss Functions
- **Sameer Ahmad** & **Shubham Jangid:** Adapting to New Domains (Natural Language Processing and Signal Processing)

Disclaimer

None of my teammates have worked on the project in the past. This is a new project.

References

[1] Kingma, D.P. & Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[2] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press. (Chapter on autoencoders)

[3] Baldi, P. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. In *Proceedings of the ICML Workshop on Unsupervised and Transfer Learning*.

[4] Toneva, M., Sordoni, A. & Courville, A. (2018). An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[5] Hu, R., Yang, Y. & Zhou, Z.-H. (2021). Prioritized Training on Learnable, Worthwhile, and Untrained Data Points. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[6] Charte, D., Rivera, A.J., del Jesus, M.J. & Herrera, F. (2019). Addressing Imbalance in Classification Problems with Deep Learning. *IEEE Access*, **7**: 71674–71685.

[7] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. & Bowman, S.R. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[8] Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.-K. & Stanley, H.E. (2000). PhysioNet: ECG5000 Dataset. *Circulation*, **101**(23): e215–e220.