Unsupervised Blind SNR Region Estimation Using Prototype-Based Multi-Stage Deep Neural Network

Charles Montes, Todd Morehouse, and Ruolin Zhou
Department of Electrical and Computer Engineering
University of Massachusetts, Dartmouth, MA, USA
{cmontes, tmorehouse, ruolin.zhou}@umassd.edu

2023 IEEE Future Networks World Forum (FNWF) | 979-8-3503-2458-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/FNWF58287.2023.10520631

***Abstract*—This paper presents a novel unsupervised learning approach for signal to noise ratio (SNR) region estimation combined with supervised modulation classification. Unsupervised learning consists of training a network while the input data labels are not provided. Previous work has shown that knowing the SNR or being within some range of SNR improves performance when performing modulation classification by using multiple networks trained separately. Existing methods are either supervised or have very specific requirements of a dataset that might not be possible to obtain in the implementation environment. Current modulation classification methods perform poorly at low or negative SNR values which previous works have shown is due to the difference in frames' SNR. Our proposed method is a frame-level SNR region estimator which uses a custom prototype-based objective function that is minimized using a regression deep neural network. The estimator network partitions a dataset by estimating SNR ranges and each range is trained on a separate network for modulation classification. We explore using a different number of clusters of a hierarchical clustering method to evaluate the separability of the SNR ranges to determine an upper bound and feasibility of a multi-network approach. The performance of our method is evaluated using a weighted dataset partition accuracy on the modulation classification dataset DeepSig RadioML2016, which consists of multiple modulation types and SNR values. Results show the ability to effectively estimate and separate multiple SNR ranges in a dataset.**

***Index Terms*—unsupervised learning, automatic modulation recognition, AMR, AMC, signal to noise ratio, CNN**

I. Introduction

Signal-to-noise ratio (SNR) is a critical parameter in evaluating the performance of communication systems. Accurate estimation of SNR plays a significant role in improving the efficiency of these systems. Recent approaches for SNR estimation predominantly rely on supervised deep learning algorithms, which assume the availability of SNR label information for training the regression networks [1] [2] [3]. However, in real-world applications or cognitive radio, the availability of SNR label information is not always guaranteed.

This method employs two primary networks to estimate the signal-to-noise ratio (SNR) and utilizes multiple secondary networks, each specializing in specific SNR regions. One of the key aspects of this method is the usage of two primary networks, where the SNR is initially estimated by the first network before being input into the other network for further processing, such as modulation classification [4] [5]. This technique raises questions about the reasons for implementing two networks and their effectiveness, as current literature lacks clarity on this matter [1] [2]. Furthermore, the issue of determining the optimal number of secondary networks also arises in this context.

Current methods for SNR estimation often employ a regression network to create a single output that corresponds to a continuous space representing the estimated SNR value [1] [2]. Nonetheless, these works assume that the SNR label information is available for training, which may not be the case in unsupervised settings. This limitation has driven the need for developing novel methods of SNR region estimation that do not rely on SNR labels or require any prior knowledge about the environment. An unsupervised approach would enable the system to determine SNR estimations without prior knowledge about the dataset during training.

To address this problem, we propose a hierarchical clustering approach, which organizes the data points into clusters having similar properties, thereby revealing the underlying structure of the data [6]. By conducting experiments to evaluate these clusters, we can determine the appropriate number of secondary networks to employ.

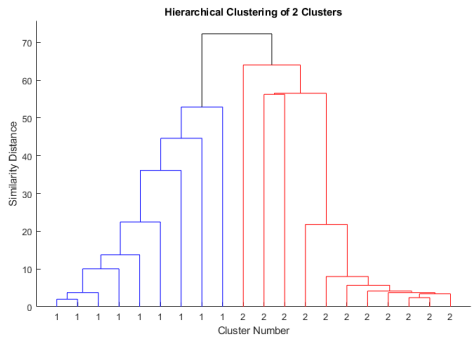


Fig. 1: Hierarchical Clustering to Determine the Number of Clusters.

A. Clustering

Previous works have used hierarchical clustering to determine a number of clusters to use for their methods [6]. For an examination into the hierarchical clustering analysis performed. This method was validated by training 20 different networks, each being trained on a small percentage of frames with various SNR values ranging from -20 dB to 18 dB in increments of 2, such as a network dedicated solely to

-20 dB SNR. These networks were then tested on all other SNR values, generating a 20×20 matrix that represents the relationship between each network when trained on one SNR and tested on others.

The resulting matrix revealed two distinct clusters Fig. (1), indicating that two distinct SNR regions were present. A subsequent experiment was conducted to test the efficacy of using three clusters; however, there was an imbalance between these clusters with one cluster having only 1 point and the other two having 9 and 10 points which proved inefficient in comparison to the two-cluster setup, which had a balanced distribution of 9 and 11 points.

Examining the hierarchical clustering plot in Fig. (1), it shows the similarity distance between points is smaller for higher SNR values on the right side, indicating greater similarity between them. On the left side, the similarity distance increased between lower SNR values, showing how they are less similar. By utilizing hierarchical clustering, these points were then partitioned into separate clusters based on their similarity distance and the pre-defined number of clusters indicated by cluster one (blue) and cluster two (red). The results of the experiments demonstrated that dividing the data into two clusters would provide a more effective solution, leading to the implementation of two separate primary networks, each focusing on different SNR regions.

B. Convolutional Prototype Learning

Convolutional Prototype Learning (CPL) is a method used to attempt to solve the open set recognition problem [7] where not all the classes are known at training time when using a neural network. The way CPL works is in using a regression network which outputs to a 2D feature space instead of a classification network that outputs to a softmax classification. To perform regression training in a CNN the output is minimized to the desired value or vector instead of classification training where the output is maximizing correct classification predictions. The regression in CPL minimizes the distance from the output to a prototype or cluster representing the correct class. This results in a cluster in feature space for each class where an unknown class can be represented by not being within some distance to any cluster. The goal of CPL is to transform classification with softmax into regression in a 2D output map for a nearest prototype classification with rejection and acceptance metrics.

C. New Contributions

To overcome the limitation of current methods, we propose an unsupervised blind SNR region estimation technique using a prototype-based multi-stage deep neural network. The proposed method focuses on the estimation of SNR without needing any prior knowledge of the underlying signal or noise characteristics. The core concept of this approach is to adapt the regression network to assume no known clusters, but create clusters based on the inherent clustering of SNR values observed from the hierarchical clustering analysis. The dataset is then split into two partitions (assuming it will divide into a high and low partition based on analysis), using only input data frames of SNR and modulation types which neither are not provided during training time.

II. Adapted Prototype Loss Formulation

This section describes the methods of prototype learning or distance based cross entropy loss (DCE) from [8] [7] and the adaptation to unsupervised prototype learning.

A. Prototype DCE Loss

CPL loss or Distance-based Cross Entropy loss (DCE) from [8] [7] is used to create clusters (or prototypes) of each class in the feature space while also pushing the clusters away from each other. This is done during training and is treated as the loss function to be minimized. The starting point of the feature space before training is shown in Fig. (2a) where the black dots are the mean of each class output or prototype (Eq. 1) and the red dots are the output of each individual input. Each input corresponds to two output values, an x and a y in 2D space.

$$\begin{aligned} &prototype_i = \text{mean}(class_i\ output), \\ &i \in \mathbb{Z}\{1, ..., \text{number of classes}\} \end{aligned} \tag{1}$$

The DCE loss is computed by first calculating the prototype of each class then calculating the distance from each output to each prototype in the 2D feature space given by (Eq. 2).

$$\begin{aligned} &D_{i,j} = \sqrt{(output_{x,j} - prototype_{x,i}) + (output_{y,j} - prototype_{y,i})}, \\ &i \in \mathbb{Z}\{1, ..., \text{number of classes}\},\ j \in \mathbb{Z}\{1, ..., \text{number of outputs}\} \end{aligned} \tag{2}$$

where $D_{i,j}$ is the distance between output j and prototype i for all outputs and all prototypes. Second, the distance between all outputs and prototypes is normalized $\in [0, 1]$ acting as a probability for each output belonging to each prototype proportional to the distance to each prototype (Eq. 3). For example, the closest prototype to an output will have the highest probability of the output belonging to that prototype's class (Eq. 4).

$$p_{i,j} = \frac{\mathrm{e}^{-D_{i,j}}}{\sum_i \mathrm{e}^{-Di,j}} \tag{3}$$

$$\text{classification} = \max_i(p_{i,j}) \tag{4}$$

where $p_{i,j}$ is the probability that output j belongs to prototype i and classification corresponds to the prototype's class which has the highest probability for that output. For (Eq. 3) the sum is over all i elements for each j output and in (Eq. 4) the max is over all i elements for each j output. Lastly each $p_{i,j}$ is used to calculate the DCE loss by taking the sum of the probabilities of each correct class i for each output j (Eq. 5) The loss is maximized when the correct class for each output is the highest probability (or the closest prototype) therefore the negated log is used.

$$\begin{aligned} &\text{DCE loss} = -\log(\sum_j p_{c,j}), \\ &c \in \mathbb{Z}\{1, ..., \text{number of classes}\} \end{aligned} \tag{5}$$

where $p_{c,j}$ corresponds to the probability that output j belongs to its correct prototype (or class) c and the sum is over all j outputs.

B. Adaptation to Unsupervised DCE Loss

This approach involves a series of steps and methods during the training phase, which are necessary in the proper functioning of the network. The DCE loss has been adapted for unsupervised learning application by replacing the labels during training with assumed labels during training where the assumed label is the cluster an input frame falls into when performing k-means clustering on the minibatch during training.

First, during the training phase, each frame is used as an input for training in minibatches. The frames are of size $1 \times 128 \times 2$, and they are used in the regression network Fig. 5a. The custom training procedure incorporates the k-means clustering algorithm, which assumes that there are two distinct clusters within the input data. This assumption forms the basis of the approach since it allows the algorithm to force the minibatches into two clusters and create two means to represent as the prototypes.

The k-means clustering algorithm calculates the mean of each cluster. These means are treated as prototypes during the computation of the DCE loss. In this context, the term ”prototype” refers to the centroid of a particular cluster, which serves as a reference point for minimizing the DCE loss. A visualization of the training process is provided in Fig. 2a-2d, which shows the black dots representing the means, with the blue and orange points corresponding to each of the two distinct clusters.

Fig. 2a shows the initial state of the two clusters, one blue, two orange, with their prototypes (means) as the black dots. The axis are unitless and is the feature space 2D output of the regression network and axis range from left to right represents the spread amount of the clusters. For example, the progress from Fig. 2c to Fig. 2d shows the x-axis range increase from 60 to 150 meaning the clusters are spreading apart along the horizontal dimension as the loss is minimizing.

The DCE loss computation focuses on the assumption that the k-means are the correct prototypes; in other words, the black dots. By using this assumption, the algorithm minimizes the DCE loss normally and creating two distinct clusters. In the optimization phase, the objective is to minimize the unsupervised DCE loss function. It is done by adjusting the clusters' blue and orange points, pulling them closer towards their respective prototypes, and simultaneously pushing the two prototypes further away from each other. This process ensures that the neural network effectively separates and creates two clusters. The utilization of assuming the k-means are the correct prototypes successfully learns complex data patterns. However, several challenges were observed during the regression training process, which have prompted the adoption of various techniques to improve the overall stability and performance during training.

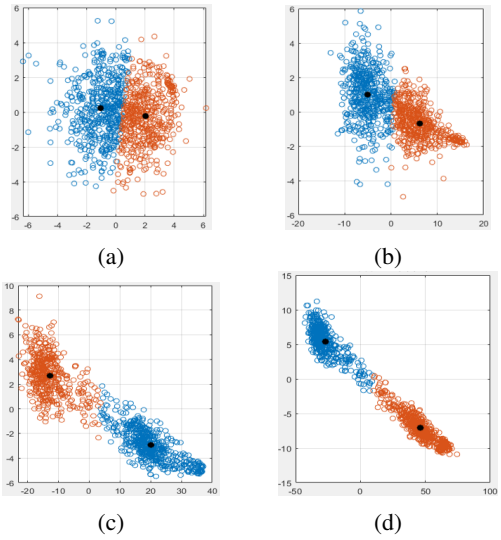


Fig. 2: Unsupervised Training Progress: (a) Initial Cluster State; (b) 3 Minutes of Training; (c) 6 Minutes of Training; (d) 8 Minutes of Training (end)

One of the primary issues encountered during regression training was the presence of NaN values, which were observed when training for long durations or using high learning rates. The appearance of NaN values in the training process may lead to unstable or inaccurate SNR region estimation. To mitigate this issue, the stochastic gradient descent with momentum (SGDM) optimization algorithm was employed instead of the widely used Adam optimizer that can lead to high learning rates and be unstable.

Another modification made to the DNN architecture was the use of exponential linear unit (ELU) activation layers in place of the popular rectified linear unit (ReLU) layers [9]. The choice of ELU was motivated by its ability to smoothly approximate the activation function by alowing negative values [9].

Lastly, to shorten the training time to further avoid NaN training values, only a small 15% portion of the dataset was utilized during the regression training process. It has been observed that the DNN was still capable of learning meaningful representations of the SNR regions and provided robust estimations, even when trained on a limited sample of the dataset. This suggests that in certain scenarios, adequate performance can be achieved using the proposed method without using the full training dataset.

III. Multi-Stage Estimation-Classification Process

This is the multistage process that during stage one, the training stage, trains the modulation classification networks using their respective partitioned dataset allocated by the unsupervised regression network. In the second stage, the testing stage, new frames are input to the multi-stage network which are first classified to a cluster determined previously by

k-means, and then classified by modulation type using one of two modulation classification networks.

A. Training Stage

The first step involves unsupervised regression training, in which the dataset used for training is fed into the regression network. As a result, the network processes the data and generates 2D points in the form of 1×2 coordinates, as depicted in Fig. 3. These 2D points serve as a foundation for the subsequent phase of training, as they are utilized to compute the final k-means of two assumed clusters. Once these clusters have been established, they represent the new partitioned datasets, which will subsequently be used to train the two modulation classification networks. Each network is trained using a training set that is a partition of the whole dataset, derived from one out of the two generated clusters. Cluster 1 is assumed to represent either the high or low SNR region, whereas Cluster 2 is assumed to represent the alternative SNR region. By training each network separately using distinct datasets, there is no overlap between the data used for one network and the other.

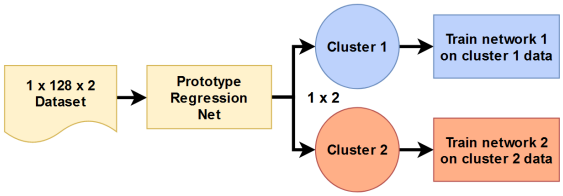


Fig. 3: Training Stage Flowchart.

After both networks have been trained independently, the entire system can be tested using both modulation classification networks which each have their own SNR region. This approach allows for an estimation of two different SNR regions.

B. Testing Stage

The testing stage consists of the following steps shown in Fig. 4. First, the input of a frame of size $1 \times 128 \times 2$ into the network. This frame represents the raw signal data that needs to be classified in terms of modulation type. Secondly, the input frame is then passed through a regression network, which results in a 2D feature space representation of the input frame as a 1×2 point, as shown in Fig. 4. This step is used for projecting the high-dimensional input data into a lower-dimensional space for clustering. Next, the 2D point generated in the previous step corresponds to a distance from each of the means determined during the training stage. These distances are used for finding the prototype, or mean, that is closest to the input frame in the 2D feature space.

Based on the calculated distances, the input frame is assigned to the closest cluster. For instance, if the input frame outputs a 2D point that is closer to the orange mean than the blue mean, the input frame is considered to belong to the

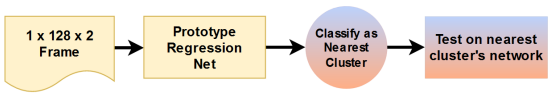


Fig. 4: Testing Stage Flowchart.

orange cluster, thus being processed by the orange network. Once the input frame has been assigned to a specific cluster, it is further passed to the corresponding deep neural network for classification. At this stage, the multi-stage deep neural network outputs the modulation type of the input frame, leading to the final modulation classification result.

IV. Results of Modulation Classification

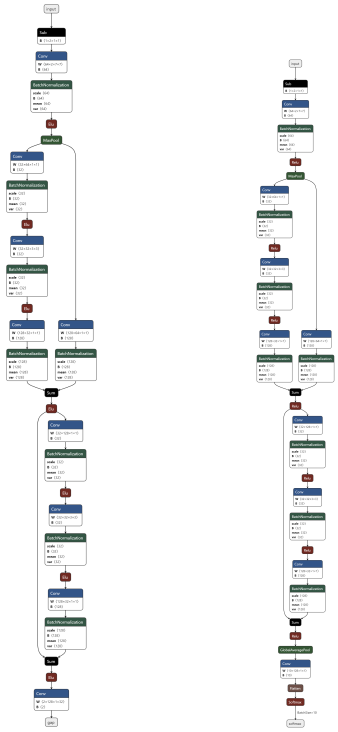
The proposed method's performance is evaluated using the DeepSig2016A dataset [10], which comprises ten classes and excludes AM-SSB for consistency purposes. This method was able to determine AM-SSB frames during testing with incorrect SNR labels, which were ultimately not included in the results due to this finding. The same ResNet network architecture Fig. 5b is used for all evaluations to ensure that the outcomes are based on the methods and not the network itself. Two-block ResNets adapted from [11] are used for regression and classification tasks.

For a comprehensive analysis, two scenarios cater to the potential applications of the methods. The first scenario is used for obtaining maximal accuracy in modulation classification specifically. In this case, the accuracy of both networks is combined in a weighted sum, accounting for the number of frames allocated to each network and their respective correct predictions. This is illustrated in Fig. 7. The second scenario preserves the SNR region information during modulation classification. Here, only the highest network accuracy is considered out of the two networks, as displayed in Fig 8. The weighting of accuracy is determined by the proportion of frames for each network at each SNR level, multiplied by the accuracy achieved at that SNR.

For a non-machine learning comparison to the proposed method, an entirely blind SNR estimator was needed due to the proposed methods ability to determine a range of SNR of a frame without any prior knowledge. A comparison was made to the MATLAB function $\text{snr}(x)$ where x corresponds to an input frame [12] and no other information is required. The MATLAB method attempts to detect the fundamental frequency of the signal and estimate the noise level. The noise is then subtracted from the signal and the SNR is estimated.

Weighting is used in calculating the results to evaluate the performance of the partitioning process. During the dataset partitioning, a weighting metric is employed to account for the fact that labels do exist in this case and can be tested Fig. 6. The weighting metric is used in both application plots to demonstrate the efficacy of ML and non-ML methods in partitioning the dataset into two distinct SNR regions.

The equation for weighting is given by (Eq. 6), emphasizing that the further the number of frames at any given SNR value



(a) (b)

Fig. 5: Network Architecture Visual Layers: (a) Regression Network; (b) Classification Network.

is from 50%, the better the partitioning. Ideally, the best partitioning scenario occurs when a cluster possesses either 100% or 0% of frames at a given SNR, signifying no overlap in frame clusters. For instance, if one cluster included all frames at -20 dB SNR and the other cluster contained 0 frames at -20 dB SNR, this would indicate an optimal case with clear separation. Conversely, a worst case scenario arises when one cluster has 50% of the frames at 0 dB SNR while the other cluster also contains 50% of the frames at 0 dB SNR.

$$\begin{aligned} &\text{weighting} = \text{sum}(0.50 - |0.50 - \text{percent of frames}|), \\ &\text{weighting} \in \mathbb{R}\{0, ..., 1\} \end{aligned} \tag{6}$$

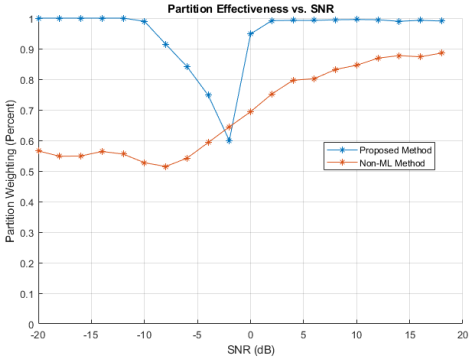


Fig. 6: Weighting used in Accuracy plots.

A. Weighted Combination of Accuracy

Considering only the performance of modulation classification and comparing results for the first evaluation type three methods are considered Fig. 7. The first being a single network with no partition of the dataset where a single network is trained on all training data and evaluated. The second method considers a partitioned dataset method with two networks, each trained on a separate dataset and evaluated, using the non-ML method $\text{snr}(x)$ and one with ML, the proposed method. The single network consistently performs well across all SNR values, the non-ML method was not able to effectively partition the datasets and has the lowest accuracy, and the proposed method has the highest accuracy at SNR values greater than 0 dB SNR due to almost entirely being able to estimate the SNR regions correctly. The two regions identifiable in Fig. 6 are approximately high SNR ≥ 0 dB and low SNR ≤ -10 dB. The SNR values where the proposed method has lower accuracy is due to ineffective partitioning shown in Fig. 6 between the two regions.

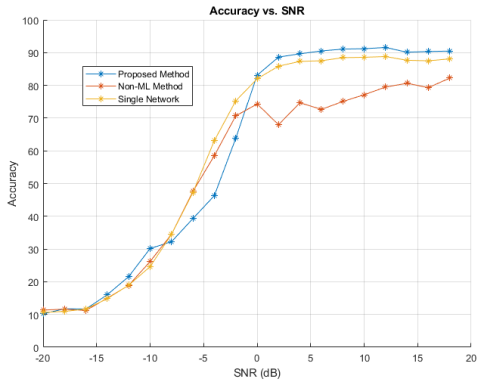


Fig. 7: Combined Weighted Modulation Classification Accuracy.

B. Accuracy with SNR Region Information

Considering the performance of modulation classification while maintaining the SNR region information for the second evaluation type only two methods are considered Fig. 8. The single network method is no longer considered due to all training data being used in one network which leads to no SNR region information, only modulation classification performance. These results are representative of both the methods ability to classify modulation type as well as effectively partition the dataset. This means that furthur processing can be done with the knowledge that the SNR of a frame is within a high or low region [5]. In this case, only one of the two networks contributes to the accuracy at each SNR value scaled by what percent of the frames at each SNR were given to the network. Fig. 6 shows the percent scaling for accuracy at each SNR value which translates well into the accuracy performance of the two methods, non-ML and the proposed. Due to the proposed method's ability to effectively partition nearly 100% of the dataset at low and high SNR, almost

all of the accuracy performance in modulation classification is maintained while the inability to effectively partition the dataset by the non-ML method shows the lesser accuracy. The single SNR value of -2 dB is where the proposed method has a lower accuracy value which is mirrored in the weighting plot where it is again below the non-ML method.

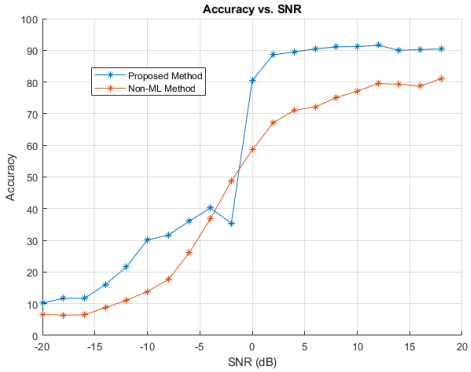


Fig. 8: Modulation Classification Accuracy While Maintaining SNR Region Information.

The plot in Fig. 9 demonstrates each of the modulation types versus SNR values, illustrating the effect of different modulation types on the estimation of SNR regions. Interestingly, WBFM modulation appears to have a notably lower accuracy than the rest of the modulation types. QAM16 and QAM64 modulation types tend to be slightly confused with each other which might result from the similarities in their structure, leading the DNN model to misclassify them. Despite this confusion, the performance of the DNN model in estimating SNR regions for QAM16 and QAM64 modulation types remains reasonably high.

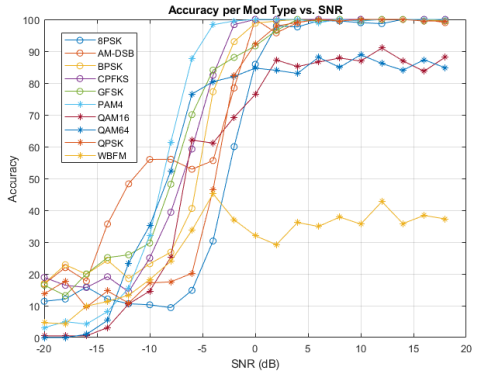


Fig. 9: Separated Modulation Types vs. SNR on ML Classification Network.

V. Conclusion

This approach has proven to be effective in estimating SNR regions for various modulation types. In both scenarios, the proposed method outperforms existing approaches in modulation classification when partitioning datasets and during modulation classification, even while maintaining SNR region information. The snr(x) function from [12] using the Kaiser window and harmonics serves as a comparison, providing an estimated SNR without any assumptions. The results demonstrate that the prototype-based multi-stage deep neural network can effectively partition a dataset without prior knowledge of the environment or frames. Consequently, two SNR regions are established, refining the ability to classify modulation types over existing methods.

Acknowledgment

This work was supported by the Office of Naval Research (ONR) Naval Engineering Education Consortium (NEEC) program under Grant No. N00174-22-1-0008, National Science Foundation (NSF) under Grant No. 2138898, and the University of Massachusetts Dartmouth's Marine and Undersea Technology (MUST) Research Program funded by the ONR under Grant No. N00014-20-1-2170.

References

[1] C. A. Harper, A. Sinha, M. A. Thornton, and E. C. Larson, "SNR-Boosted Automatic Modulation Classification," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, Oct. 2021, pp. 372–375, iSSN: 2576-2303.

[2] Y. Guo, Z. Zhang, Y. Huang, and P. Zhang, "DOA Estimation Method Based on Cascaded Neural Network for Two Closely Spaced Sources," *IEEE Signal Processing Letters*, vol. 27, pp. 570–574, 2020, conference Name: IEEE Signal Processing Letters.

[3] Y. Mao, M.-L. Zhu, T. Sun, Y.-Y. Dong, and C.-X. Dong, "Automatic Modulation Classification Based on SNR Estimation via Two-Stage Convolutional Neural Networks," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Apr. 2021, pp. 294–298.

[4] R. Zhou, F. Liu, and C. W. Gravelle, "Deep Learning for Modulation Recognition: A Survey With a Demonstration," *IEEE Access*, vol. 8, pp. 67 366–67 376, 2020, conference Name: IEEE Access.

[5] T. T. An and B. M. Lee, "Robust Automatic Modulation Classification in Low Signal to Noise Ratio," *IEEE Access*, vol. 11, pp. 7860–7872, 2023, conference Name: IEEE Access.

[6] H. Nouraei, H. Nouraei, and S. W. Rabkin, "Comparison of Unsupervised Machine Learning Approaches for Cluster Analysis to Define Subgroups of Heart Failure with Preserved Ejection Fraction with Different Outcomes," *Bioengineering*, vol. 9, no. 4, p. 175, Apr. 2022, number: 4 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2306-5354/9/4/175

[7] H.-M. Yang, X.-Y. Zhang, F. Yin, Q. Yang, and C.-L. Liu, "Convolutional Prototype Network for Open Set Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2358–2370, May 2022, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[8] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Robust Classification with Convolutional Prototype Learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 3474–3482, iSSN: 2575-7075.

[9] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," Feb. 2016, arXiv:1511.07289 [cs]. [Online]. Available: http://arxiv.org/abs/1511.07289

[10] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Engineering Applications of Neural Networks*, ser. Communications in Computer and Information Science, C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, 2016, pp. 213–226.

[11] "Create 2-D residual network - MATLAB resnetLayers." [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnetlayers.html

[12] "Signal-to-noise ratio - MATLAB snr." [Online]. Available: https://www.mathworks.com/help/signal/ref/snr.html