

# AUS Repository

## Unsupervised Deep Learning for Classification Of Bats Calls Using Acoustic Data

Item Type	Thesis
Authors	Arshad, Muhammad Arbab
Download date	2026-05-20 17:27:40
Link to Item	<a href="http://hdl.handle.net/11073/21556">http://hdl.handle.net/11073/21556</a>

UNSUPERVISED DEEP LEARNING FOR CLASSIFICATION OF  
BATS CALLS USING ACOUSTIC DATA

by  
Muhammad Arbab Arshad

A Thesis presented to the Faculty of the  
American University of Sharjah  
College of Engineering  
In Partial Fulfillment  
of the Requirements  
for the Degree of  
  
Master of Science in  
Computer Engineering

Sharjah, United Arab Emirates  
August 2021

## **Declaration of Authorship**

I declare that this thesis is my own work and, to the best of my knowledge and belief, it does not contain material published or written by a third party, except where permission has been obtained and/or appropriately cited through full and accurate referencing.

Signed            Muhammad Arbab Arshad

Date              24<sup>th</sup> August 2021

The Author controls copyright for this report.

Material should not be reused without the consent of the author. Due acknowledgement should be made where appropriate.

© Year 2021

Muhammad Arbab Arshad

ALL RIGHTS RESERVE

## Approval Signatures

We, the undersigned, approve the Master's Thesis of Muhammad Arbab Arshad

Thesis Title: Unsupervised Deep Learning for Classification Of Bats Calls Using Acoustic Data

Date of Defense: 26<sup>th</sup> August 2021

Name, Title and Affiliation	Signature
Dr. Imran Zualkernan Professor, Department of Computer Science and Engineering Thesis Advisor	
Dr. Michel Pasquier Associate Professor, Department of Computer Science and Engineering Thesis Committee Member	
Dr. Usman Tariq Associate Professor, Department of Electrical Engineering Thesis Committee Member	
Dr. Fadi Aloul Head Department of Computer Science and Engineering	
Dr. Lotfi Romdhane Associate Dean of Graduate Affairs and Research College of Engineering	
Dr. Sameer Al- Asheh Interim Dean College of Engineering	
Dr. Mohamed El-Tarhuni Vice Provost for Research and Graduate Studies Office of Graduate Studies	

## **Acknowledgment**

I would like to thank my advisor Dr. Imran Zualkernan for providing knowledge, guidance, support, and motivation throughout my research stages. I am deeply beholden for his great assistance, worthy discussion, and suggestions. And I am thankful to Dr. Jacky Judas and Emirates Nature - World Wildlife Foundation (WWF) for providing data and valuable feedback for the completion of this research work.

I would also like to thank the American University of Sharjah, especially the department of Computer Science and Engineering, for providing me with the graduate teaching assistantship. Finally, I am thankful to the authors of [1] [2] [3] [4] [5] for letting me use their model diagrams in this work.

## Abstract

Analysis and understanding of bat behaviors have taken on an increased importance post-Covid 19. Manual analysis of echolocation calls in bats to deduce behavior is cumbersome, time-consuming and costly. Previous attempts to automate this process have relied on labeled data which is expensive and difficult to collect. This thesis explored the use of state-of-the-art unsupervised learning algorithms like IMSAT, IIC, SCAN, JULE and DeepCluster to determine if interesting bat behaviors can be automatically determined based on unlabeled bat echolocation data which is readily available. The algorithms originally developed for image classification were adapted to work with audio data. One small labeled echolocation data set from the UAE Al-Hajar mountains and a large unlabeled dataset from an urban space in Dubai from the Emirates Nature - World Wildlife Foundation (WWF) were utilized. A coding scheme for interpreting bats' behavior was also developed. The results are that different algorithms capture different behavior. For example, IIC and IMSAT identified the presence of multiple bats, DeepCluster was better able to identify prey capture attempts, SCAN could distinguish bat calls in a close habitat and JULE could capture different species types. Based on Mutual Information (MI) the most similar pairs of algorithms were IIC and IMSAT (0.429), IIC and DeepCluster (0.374), and IMSAT and DeepCluster (0.266). On the small labeled data set, IIC performed the best with an accuracy of 48.28% followed by IMSAT (43.59%), JULE (43.13%), DeepCluster (39.84%) and SCAN (29.38%). A baseline K-Medoid algorithm only had an accuracy of 23.75%. For future work, better audio augmentation techniques can be explored and other unsupervised learning algorithms like DAC, DEC and K-Autoencoders can be investigated as well.

**Keywords: Unsupervised Deep Learning; Audio Classification; Wildlife Monitoring.**

## Contents

Abstract.....	5
Contents.....	6
List of Figures.....	10
List of Tables.....	11
Chapter 1. Introduction.....	14
1.1    Overview.....	14
1.2    Thesis Objectives.....	15
1.3    Research Contribution.....	15
1.4    Thesis Organization.....	15
Chapter 2. Background and Literature Review.....	17
2.1    Core Concepts.....	17
2.1.1. Cross-entropy.....	17
2.1.2. Kullback-leibler ( <i>KL</i> ) divergence.....	17
2.1.3. Mutual information.....	18
2.2    Related Work.....	18
2.2.1. Augmentation-independent algorithms.....	18
2.2.2. Augmentation-dependent algorithms.....	19
2.2.3. Shortlisting algorithms for the experiment.....	19
Chapter 3. Methodology.....	21
3.1    Problem Formulation.....	21
3.2    Data Collection.....	22
3.3    Data Pre-processing.....	22
3.3.1. Selection of bat calls from the recording.....	22
3.3.2. Normalization of audio levels of each recording.....	22
3.3.3. Generation of spectrograms.....	23
3.4    Data Augmentation Techniques:.....	25

3.4.1.	Temporal perturbation.....	25
3.4.2.	Speed perturbation. ....	25
3.5	Framework for Classification.....	26
3.6	Framework for Interpretation of Spectrograms Clusters .....	27
3.6.1.	Multiple bats presence. ....	27
3.6.2.	Prey capture attempt.....	28
3.6.3.	Habitat type. ....	28
3.6.4.	Sound pressure level (SPL).....	28
3.6.5.	Noise.....	29
3.7	Metrics for Quantitative Comparison .....	30
3.7.1.	Mutual information score.....	30
3.7.2.	Rand score.....	31
3.7.3.	Fowlkes mallows score. ....	31
3.7.4.	Accuracy.....	31
3.8	Distance Metrics for K-Medoid Clustering .....	32
3.8.1.	Structural similarity index metric (SSIM). ....	32
3.8.2.	Oriented FAST and rotated BRIEF (ORB).....	32
3.8.3.	Root mean squared error (RMSE). ....	33
3.9	Finding an Approximate Number of Clusters Using K-Medoids .....	33
Chapter 4.	Establishing the Baseline using K-Medoid Clustering .....	35
4.1	Theoretical Explanation and Training of K-Medoid.....	35
4.2	Results.....	35
4.2.1.	Vertical/characteristic analysis. ....	35
4.2.2.	Horizontal/cluster analysis.....	35
Chapter 5.	Learning Discrete Representations via Information Maximizing Self-Augmented Training (IMSAT) .....	37
5.1	Theoretical Explanation of IMSAT .....	37

5.2	Training on Our Dataset.....	38
5.3	Results.....	39
5.3.1.	Vertical/characteristic analysis. ....	40
5.3.2.	Horizontal/cluster analysis.....	40
Chapter 6.	Invariant Information Clustering (IIC).....	43
6.1	Theoretical Explanation of IIC .....	43
6.2	Training on Our Dataset.....	44
6.3	Results.....	45
6.3.1.	Vertical/characteristic analysis. ....	46
6.3.2.	Horizontal/cluster analysis.....	46
Chapter 7.	Deep Clustering for Unsupervised Learning of Visual Features .....	48
7.1	Theoretical Explanation of DeepCluster.....	48
7.2	Training on Our Dataset.....	49
7.3	Results.....	50
7.3.1.	Vertical/characteristic analysis. ....	50
7.3.2.	Horizontal/cluster analysis.....	51
Chapter 8.	Semantic Clustering by Adopting Nearest neighbors (SCAN).....	53
8.1	Theoretical Explanation of SCAN .....	53
8.2	Training on Our Dataset.....	53
8.3	Results.....	54
8.3.1.	Vertical/characteristic analysis. ....	54
8.3.2.	Horizontal/cluster analysis.....	55
Chapter 9.	Joint Unsupervised Learning (JULE).....	58
9.1	Theoretical Explanation of JULE .....	58
9.2	Training on Our Dataset.....	59
9.3	Results.....	59
9.3.1.	Vertical/characteristic analysis. ....	60

9.3.2. Horizontal/cluster analysis.....	60
Chapter 10. Inter-Algorithm Quantitative Analysis.....	63
Chapter 11. Specie Clustering using Unsupervised Deep Learning .....	72
11.1    Data Description and Pre-Processing:.....	72
11.2    Training Setup:.....	74
11.3    Results:.....	76
Chapter 12. Other Unsupervised Clustering Algorithms .....	78
12.1    Deep Adaptive Clustering (DAC).....	78
12.2    Deep Embedded Clustering (DEC).....	80
12.3    K-Autoencoders .....	81
Chapter 13. Conclusion and Future Work .....	84
References.....	86
Vita.....	92

## List of Figures

Figure 1: Flow Chart for the Pre-processing of Single Audio File.....	23
Figure 2 Spectrograms Before Normalization .....	24
Figure 3: Spectrograms After Normalization .....	24
Figure 4: Histogram of Length of Audio Files - With 5 Bins.....	25
Figure 5: Example of Original and Augmented Spectrograms.....	26
Figure 6: Pipeline for Classification .....	27
Figure 7: Examples of Different Behaviors .....	29
Figure 8: Example of Miscellaneous Signals with Bat Calls.....	29
Figure 9: Inertia Based On ORB, SSIM, And RMSE Vs Target Clusters.....	34
Figure 10: Basic Idea of IMSAT [1].....	38
Figure 11: Learning Curve of IMSAT for Bats Calls Clustering .....	39
Figure 12: Representative Samples from the Cluster Assignment of IMSAT.....	41
Figure 13: Flowchart of IIC [2] .....	44
Figure 14: Learning Curve of IIC for Bats Calls Clustering .....	45
Figure 15: Representative Samples from the Cluster Assignment of IIC.....	47
Figure 16. Illustration of DeepCluster[5].....	49
Figure 17: Learning Curve of DeepCluster for Bats Calls Clustering.....	49
Figure 18: Representative Samples from the Cluster Assignment of DeepCluster.....	51
Figure 19: Learning Curve of SCAN for Bats Calls Clustering .....	54
Figure 20: Representative Samples from the Cluster Assignment of SCAN .....	56
Figure 21: Framework for JULE [4].....	59
Figure 22: Representative Samples from the Cluster Assignment of JULE.....	61
Figure 23: Example of a Spectrogram Generated from Labeled Dataset. ....	73
Figure 24: Call Signatures of Different Bat Species from the Dataset .....	75
Figure 25: Learning Curves for Unsupervised Clustering of Bats' Calls .....	75
Figure 26: Flowchart of DAC [19] .....	78
Figure 27: Network Structure of DEC[20] .....	81
Figure 28: Block Diagram of K-Autoencoders.....	82

## List of Tables

Table 1: Augmentation-Independent Clustering Algorithms .....	19
Table 2: Augmentation-Dependent Clustering Algorithms .....	20
Table 3: Number of Shortlisted/Total Samples for all Techniques. ....	30
Table 4: Numerical Distribution of Bats calls' Characteristics for K-Medoid.....	36
Table 5: Numerical Distribution of Bats calls' Characteristics for IMSAT .....	39
Table 6: Comments on the Interpretation of IMSAT Results by a Field Expert .....	42
Table 7: Numerical Distribution of Bats calls' Characteristics for IIC .....	45
Table 8: Comments on the Interpretation of IIC Results by a Field Expert .....	47
Table 9: Numerical Distribution of Bats calls' Characteristics for DeepCluster.....	50
Table 10: Comments on the Interpretation of DeepCluster Results by an Expert.....	52
Table 11: Numerical Distribution of Bats calls' Characteristics for SCAN .....	55
Table 12: Comments on the Interpretation of SCAN Results by a Field Expert.....	57
Table 13: Numerical Distribution of Bats calls' Characteristics for JULE .....	60
Table 14: Comments on the Interpretation of JULE Results by a Field Expert .....	62
Table 15: Mutual Information Score Between All Algorithms .....	64
Table 16: Normalized Mutual Information Score Between All Algorithms .....	64
Table 17: Common characteristics for Defining Cluster Between Two Algorithms...65	
Table 18: Rand Score Between Different Algorithms .....	65
Table 19: Adjusted Rand Score Between Different Algorithms .....	66
Table 20: Cross Tabulation between IMSAT and IIC .....	67
Table 21: Cross Tabulation between IIC and DeepCluster .....	67
Table 22: Cross Tabulation between IMSAT and DeepCluster .....	67
Table 23: Cross Tabulation between JULE and IIC .....	68
Table 24: Cross Tabulation between JULE and IMSAT.....	68
Table 25: Cross Tabulation between JULE and DeepCluster .....	68
Table 26: Cross Tabulation between SCAN and IIC.....	69
Table 27: Cross Tabulation between SCAN and IMSAT.....	69
Table 28: Cross Tabulation between SCAN and DeepCluster .....	69
Table 29: Cross Tabulation between SCAN and JULE.....	70
Table 30: Cross Tabulation between SCAN and K-Medoid .....	70
Table 31: Cross Tabulation between K-Medoid and IIC.....	70

Table 32: Cross Tabulation between K-Medoid and IMSAT .....	71
Table 33: Cross Tabulation between K-Medoid and DeepCluster .....	71
Table 34: Cross Tabulation between K-Medoid and JULE.....	71
Table 35: Count of Bat calls from different Specie .....	73
Table 36: Comparision Between Parameters for Spectrogram Generation .....	74
Table 37: Results from Unsupervised Clustering of Labelled bats' calls .....	76
Table 38: Evolution of Unsupervised Clustering of Labelled Datasets[3] .....	77

## List of Abbreviations

CIFAR	Canadian Institute for Advanced Research
DAC	Deep Adaptive Clustering
DCGAN	Deep Convolution Generative Neural Adversarial Networks
DEC	Deep Embedded Clustering
FMS	Fowlkes Mallows Score
IIC	Invariant information clustering
IMSAT	Information Maximizing Self-Augmented Training
JULE	Joint Unsupervised Learning
<i>KL</i> -divergence	Kullback–Leibler divergence
<i>K</i> -NN	<i>k</i> -Nearest Neighbours
ORB	Oriented FAST and Rotated BRIEF
SCAN	Semantic Clustering by Adopting Nearest Neighbors
SPL	Sound Pressure Level
SSIM	Structural Similarity Index Metric
STFT	Short-time Fourier transform

## Chapter 1. Introduction

### 1.1 Overview

Bats are an integral part of our ecosystem, and they play their roles in reducing forest restoration [6], Tropical Forest Succession [7], and pollination [8]. These factors ensure the stability of the food chain in any ecosystem, and any decline in the population or disturbance in the behavior of bats could affect it. Therefore, it is crucial to keep a close check on them. There are around 1300 different species of bats [9], each having its characteristics. For example, some bats eat soft fruits such as avocados and mangoes and discard their seeds, which ultimately grow into mature trees. So, they play a significant role in the preservation of fruits in the wild [10]. Unfortunately, few species of bats have started to become endangered [11], e.g., *Eumops floridanus* is one of the nine species which has been categorized as endangered by the U.S. Endangered Species Act [12]. Therefore, it is important from the ecological perspective to monitor the different species of bats in any ecosystem.

Another perspective is that the viruses are present in bats, and their interaction with humans could lead to the transmission of such viruses. Various studies have claimed that COVID-19 has also been transferred from bats to humans [13] [14]. The behavior of species in a site with a virus outbreak could help us narrow down the origin of such viruses because there are hundreds of species of bats and shortlisting the likely aspects (such as behavior) could be helpful.

One laborious way to get the data about the behavior of different species is for an expert to manually inspect the reasonable number of bats in an ecosystem and draw the statistical projections (or use an annotated dataset to develop automated techniques [15]). It is a costly and time-consuming process. To this end, we propose to collect just the acoustic data of bats via recording instruments and then classify that via unsupervised deep learning techniques. This groups the bats' recordings with similar acoustic signatures, which means that the grouping of species is automated. The next step could be for an expert to identify a few acoustic signatures from every group, which is identified with the deep learning models. Essentially, this work is aimed at eliminating the step of analyzing every audio recording of bats and then manually labeling it to get the overall picture of an ecosystem. Once we have the classification,

examining only a few tapes (by an expert) could give us the required information. Using acoustic signatures to identify the species of bats has already been demonstrated for labeled data using supervised learning [16]. However, as claimed earlier, the collection of such labeled data is expensive and time-consuming. Thus, better automation is required, which makes use of just the unlabeled data for classifying bat calls (based on the behaviors beings exhibited). A detailed framework for the interoperation of these behaviors from bat calls is explained in section 3.6.

## **1.2 Thesis Objectives**

The objective of this study is to advance the unsupervised image clustering algorithms for the applications of audio clustering and determine which of the various unsupervised learning techniques work best for audio signal clustering of bats' acoustic data.

## **1.3 Research Contribution**

The contributions are:

- A novel application of unsupervised deep learning algorithms to find semantic behavior of bats from acoustic calls, explored for the first time.
- Empirical data on the relative performance of the various well-known unsupervised learning algorithms to cluster bats echolocation calls, and direction for future work based on the experimental results.
- Demonstrated the results using two audio augmentations techniques.
- Development of a coding scheme for interpreting bats' behavior.

## **1.4 Thesis Organization**

Chapter 2 describes the core concepts and state-of-the-art literature work. Chapter 3 describes data collection and pre-processing along with formally defining the problem. It proposes a framework for clustering and interpreting the bat calls. Initial experiments used to decide the input parameters for clustering algorithms are also described along with the explanation of various metrics used for evaluation. Chapter 4-9 describes the clustering algorithms and conducted experiments in detail. Chapter 10 gives a quantitative comparison between the results from different algorithms. Chapter 11 explains the application of unsupervised clustering for species classification. Chapter 12 describes the theoretical explanation of additional

unsupervised learning algorithms. Finally, chapter 13 provides a conclusion and direction for future work.

This Chapter explains the need for bats' calls clustering and identifies the key objectives which motivated this thesis. Research contributions and the organization of the overall thesis are also explained.

## Chapter 2. Background and Literature Review

In this Chapter, various fundamental concepts, terminologies are discussed, which are essential for the understanding of previous work in the literature. Then, multiple techniques for unsupervised learning are discussed in detail. For each method, its description, objective/loss functions used, performance on various datasets, and block diagrams/algorithms are explained in their respective Chapters.

### 2.1 Core Concepts

Let us say,  $X_l$  and  $X_u$  represents labeled and unlabeled data. The focus of this work will remain on the unsupervised classification of data in which the labels will not be available. The nomenclature and symbolic representations are adapted from [17].

**2.1.1. Cross-entropy.** The concept of cross-entropy (CE) originates from the information theory and it is the most common loss function used for classifying images. The cross-entropy is measured between two probability distributions. Consider the neural network  $f$  with input  $x$ ,  $f(x)$  as the output. To quantify the difference between  $f(x)$  and the corresponding label  $z_x$  (for simplicity, we are assuming labeled data here), Equation (1) is used:

$$CE(f(x), z) = \sum_{c=1}^C P_{f(x)}(c) \log (P_z(c)) \quad (1)$$

where the probability distribution (chances of belonging to a cluster), its entropy (purity of clusters) are denoted by  $P$  and  $H$ . The total number of classes is indicated by  $C$ .

**2.1.2. Kullback-leibler (KL) divergence.**  $KL$  divergence is another loss function used during the classification of images, which measures the difference between distributions (between the output  $f(x)$  and the underlying discrete probability distribution ( $P$ ) over all the classes ( $C$ )) – as provided in Equation (2).

$$KL(f(x), P) = \sum_{c=1}^C P_{f(x)}(c) \log \left( \frac{P_{f(x)}(c)}{P(c)} \right) \quad (2)$$

Subtracting the entropy over  $z$  from  $CE$  gives the  $KL$ -divergence. This relationship is given in Equation (3).

$$CE(f(x), z) = H(P_z) + KL(P_z | P_{f(x)}) \quad (3)$$

**2.1.3. Mutual information.** The information we get regarding a random variable by observing the other one is quantified using MI. MI is more prevalent in literature in the case of unsupervised clustering (over CE). For two different outputs  $x$  and  $y$  with their probability distributions  $P_{f(x)}$  and  $P_{f(y)}$ , MI is the KL-divergence between  $P_{f(x),f(y)}$  (Joint distribution) and  $P_{f(x)} * P_{f(y)}$  (Marginal distribution) – as provided in Equation (4).

$$-IM(P_{f(x)}, P_{f(y)}) = -KL(P_{f(x),f(y)} | P_{f(x)} * P_{f(y)}) \quad (4)$$

Conceptually, just as correlation measures the distance between random variables, MI does the same for probability distributions.

## 2.2 Related Work

The acoustic signatures of bat calls will be converted to images, and thus we need image clustering algorithms that could make different clusters. In our context, each cluster will represent different behavior. The framework for interpretation is also provided. Different frameworks have been proposed in the literature to divide deep unsupervised learning into various categories. Network architecture can be used as a basis for the classification of these algorithms [18]. This thesis proposes using the ‘learning strategy’ as the basis for classification, which classifies the techniques into the following two categories:

**2.2.1. Augmentation-independent algorithms.** In this category, the algorithm utilizes only the feature representations (from the output of CNNs) as the basis of clustering [3]. In literature, this approach is also referred to as *end-to-end learning*. The clusters, in this approach, are formed by the regulatory signals from either confident samples [19] or through the cluster-reassignment [5]. Table 1 lists the latest techniques from the literature that utilized this approach, and the general idea behind each algorithm. These include DAC [19], JULE [4], DEC [20], DeepCluster [5] and K-Autoencoders [21]. While the results from these algorithms are reasonable on the standard datasets, it is important to note that the learning from these algorithms is limited. It is because these algorithms do not utilize the transformation of the original data to carry out more diverse and robust learning. This is why their performance does not constitute state-of-the-art results.

Table 1: Augmentation-Independent Clustering Algorithms

Technique	Year	General Idea	Accuracy
DAC [19]	2017	Uses pairwise classification to Combine the clustering algorithm along with the representation/feature learning from the images.	52% on CIFAR-10
JULE [4]	2016	Worked on the principle of agglomerative clustering and used a recurrent framework to implement the consecutive tasks in the clustering phase.	78.55% on CIFAR-10
DEC [20]	2016	Iteratively computes the auxiliary/target distribution and minimizes the loss (in the form of KL-divergence) between this and the output of stacked auto-encoder (which learned the embedding).	84.3% on MNIST
DeepCluster [5]	2019	Outputs of CNN are clustered using k-means, and these assignments are used as pseudo-labels for training the parameters of the network.	73.7% on PASCAL VOC
K-Autoencoders [21]	2020	Uses multiple auto-encoders (equivalent to the number of clusters required) and assigns each data point/Image to the autoencoder, which gives us the minimum reconstruction error.	88% on MNIST

**2.2.2. Augmentation-dependent algorithms.** These algorithms enforce the consistency/similarity between the augmentation of an image and the original image itself to disentangle the different clusters [3]. Table 2 lists the latest techniques from the literature that used augmentation, and the general idea behind each algorithm.

**2.2.3. Shortlisting algorithms for the experiment.** From the algorithms/techniques mentioned in Tables 1 and 2, a few were shortlisted considering the scalability of the algorithm, backbone/architecture used, its category, and few common ideas such as over-clustering and pretext tasks. IMSAT gives the baseline performance for our evaluation, and it had also been evaluated on bigger datasets such as CIFAR-10. IIC is one of the recent developments in this area, which also utilized the pretext tasks and over-clustering. SCAN, which is the latest, and gave the state of the art accuracy of 87% on CIFAR-10 was also considered. The direction of utilizing

the invariant information by introducing augmentations is a promising direction that is why the majority of the algorithms are shortlisted from this category.

Table 2: Augmentation-Dependent Clustering Algorithms

Technique	Year	General Idea	Accuracy
IMSAT [1]	2017	This maximizes the MI between input and the cluster assigned and enforces invariance using data augmentation.	45.6% on CIFAR-10
IIC [2]	2019	Images are paired (with one of them transformed) such that the essence of the image does not change, then the invariant information is learned, and neural network output is used as a cluster.	88% on STL-10
SCAN [3]	2020	A self-supervised task from representation learning is used to achieve useful features for clustering. After which semantic clustering is performed on these features.	87.6% on CIFAR-10

From Augmentation-Independent approaches, DeepCluster is shortlisted because of its unique approach to utilize k-mean to generate the pseudo-labels and ultimately relying on the CNN training in an end-to-end fashion. The clustering of high-resolution images was also demonstrated with this approach. JULE was shortlisted because, unlike others, it used bottom-up/agglomerative learning and gave an accuracy of 78.5% on the CIFAR-10. The theoretical explanation of the rest of the algorithms is also provided in Chapter 12. In conclusion, the selection is varied across a spectrum of design choices to evaluate the technique which could work best in the (different) domain of spectrograms' clustering.

Various key concepts and terminologies are covered in this Chapter, which is necessary for comprehending past work in the literature. Then, a variety of unsupervised learning strategies are briefly examined. The justification for choosing strategies for our experimentation is also described in detail.

## Chapter 3. Methodology

### 3.1 Problem Formulation

Given the domain of input  $X$  with data points  $\{x_1, x_2, x_3, \dots, x_N\}$  with no labels and clusters  $\{C_1, C_2, C_3, \dots, C_M\}$  given  $M < N$ , our goal is to assign each data point  $x_i$  to a cluster  $C_i$ , such that all the rest of the data points in a particular cluster are similar to each other. Our final task defines how we consider the data points to be similar [1]. This is a complex problem, especially for bats' calls clustering. Some of the reasons relate to the general issues in the field of unsupervised learning while others relate to the complexity in formulation and interpretation of bat calls datasets. These reasons are given below:

- It is difficult to define an objective function for unsupervised learning algorithms, which generalizes well across domains and that is why massive modifications are required to use them on the bats calls dataset. For example, coming up with custom augmentation functions - as explained in section 3.4.
- Standard datasets for bat calls do not exist and it is required to build and pre-process them from scratch.
- A single bat call cannot be labeled into a single category of behavior and a more comprehensive procedure is required to interpret and compare the outputs – as explained in section 3.6.
- There is no systematic way to find the hyper-parameters – which are used to train unsupervised learning models. Choosing the sub-optimized hyper-parameters could result in the models not converging or producing degenerate clusters. Therefore the selection of hyperparameters, in our experiments, was guided by empirical evidence, suggestions by the authors of the respective algorithms, and computational constraints.
- To visually distinguish between various characteristics of individual bat calls, it is required to keep an individual spectrogram at the dimensions of at least 128x128. This factor, along with the massive size of the dataset and computationally expensive nature of unsupervised deep learning algorithms, results in a lot of time and effort being required.

### 3.2 Data Collection

The site of the collection is the Hajar Mountains, which are in three states (Sharjah, Fujairah, and Ras Al Khaimah) of the United Arab Emirates during the early three months of 2018. The instrument used was the Pettersson DX1000 Ultrasound detector. It has the capability of collecting signals with frequencies between 5 and 235 kHz. The instrument was fixed at a position, and it is capable of recording bat calls up to a distance of 20 meters.

### 3.3 Data Pre-processing

The pre-processing involved the following major steps:

**3.3.1. Selection of bat calls from the recording.** The random selection of audio segments leads to instances of no signal(s) being captured in the spectrogram. To solve this issue, each recording was first processed through batdetect [22]. Batdetect is a Convolutional Neural Network (CNN) based model for recognizing bat echolocation calls in noisy audio recordings. It outputs the detected locations of the bat signal in each recording. Audio signal chunks spanning 500ms, around the detected location in audio recording, were drawn. This chunk after normalization (as explained in the next section) was used for drawing an individual spectrogram. As an alternative to open-source software, another tool called Kaleidoscope, a commercial alternative, could also be used to identify the position of bat calls in a recording. However, in our experiments, the training of the algorithms was based on the open-source non-commercial program: batdetect.

**3.3.2. Normalization of audio levels of each recording.** Due to differences in audio levels, spectrograms generate different color tones. Thus, the clustering becomes dependent on the color tone of the generated spectrogram (instead of the characteristics of the signal). To solve this issue, each recording was first normalized to the same decibel level 0 *dB* and then the spectrograms were generated. This led to the same color tone across all the spectrograms. Figure 2 and Figure 3 compare this (based on the same two files).

The Amplitude/loudness of the recorded audio signals was maintained at 0 *dB*. The difference in *dB* level of the audio segment was calculated with reference to the target *dB* level. The resulting *dB* value was then converted to an equivalent number

(factor), using Equation (6), and then multiplied with the values of the original audio segment using Equation (5).

$$\text{Audio Sample}_i = \text{Factor} \times \text{Audio Sample}_i, \quad (5)$$

$$i = 1, \dots, \text{Total Samples in an Audio}$$

$$\text{Factor} = 10^{\frac{\text{Target dB level} - \text{Audio Segment dB level}}{20}} \quad (6)$$

A flow chart of exactly what is done to a single sample of raw audio data is given in Figure 1. An overall pipeline is explained in section 3.5 and illustrated in Figure 6.

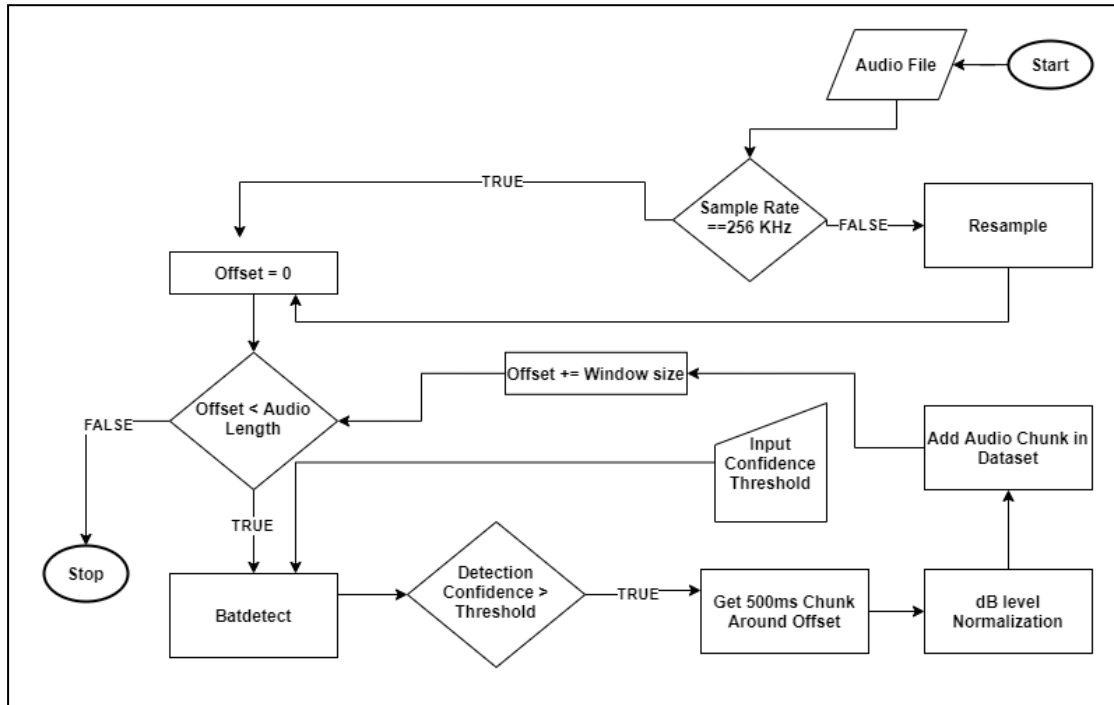


Figure 1: Flow Chart for the Pre-processing of Single Audio File

**3.3.3. Generation of spectrograms.** For generating each spectrogram, the length of the signal was taken to be 0.5 seconds. In terms of configuration, the length of the windowed signal after padding with zeros was set to 5120 samples and the hop length was set to 512. The y-axis was set to linear, and hann-window was used for STFT computation. The frequencies to be displayed were limited from 20KHz to 92KHz (which is the usual range of bat calls frequencies) [23]. Further processing was done to

remove axis labels, axis ticks, and titles from the image (before giving this as input for training) with a final resolution of 128x128 for each spectrogram.

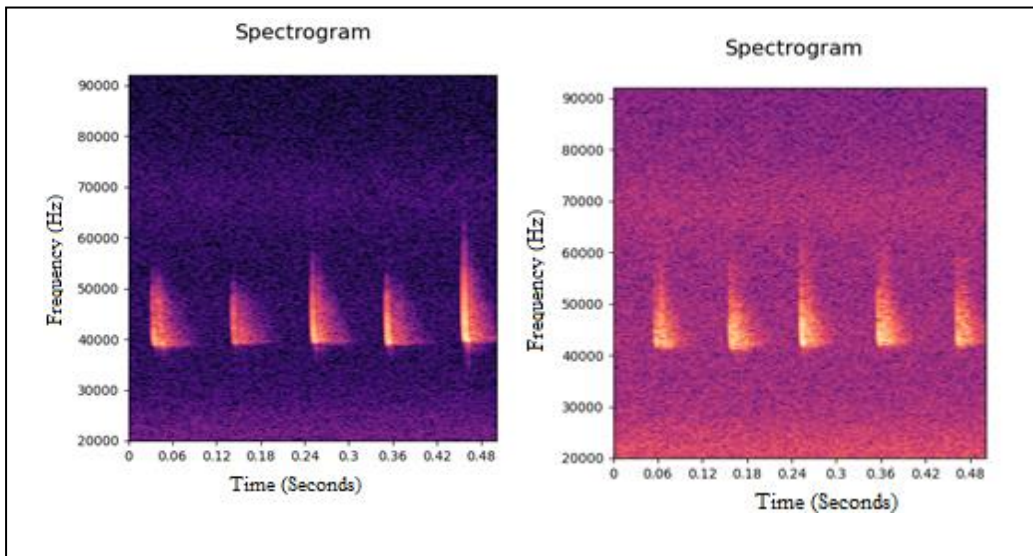


Figure 2 Spectrograms Before Normalization

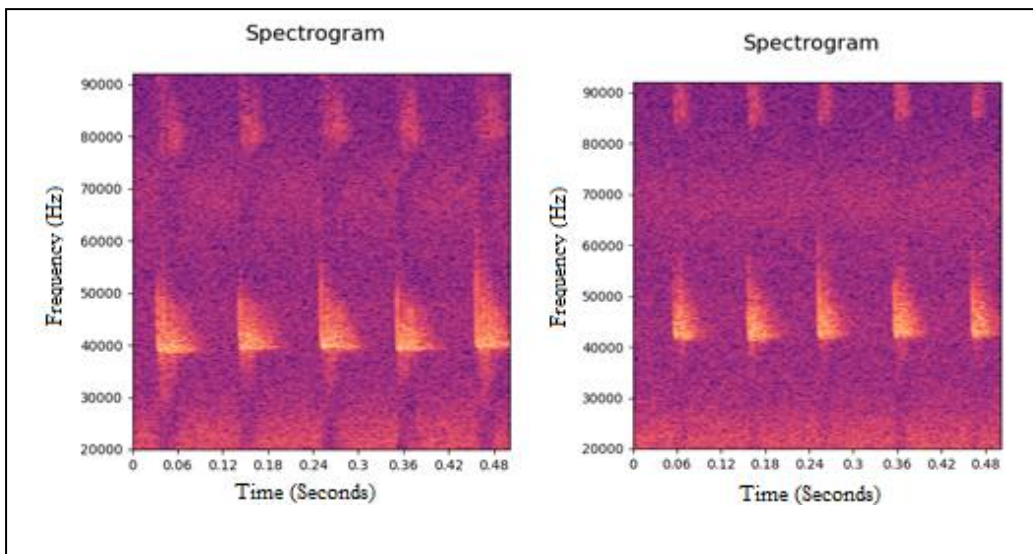


Figure 3: Spectrograms After Normalization

To experiment, a total of 48951 audio recordings were used. The average length consisted of 10.19 seconds (with a standard deviation of 4.4 seconds). The histogram of length is added in Figure 4. The sampling rate had two unique values of 384,000 Hz and 256000 Hz. All the recordings were resampled to 256000 Hz. The detection probability threshold, for batdetect, of 95% was considered and one bat call was

arbitrarily drawn from each recording. The reason for not choosing the location with maximum detection probability was to introduce diversity in the bat calls and avoid monotonic bat calls. After applying these steps, a total of 24610 (non-augmented) spectrograms were generated which were used as input to the clustering algorithms.

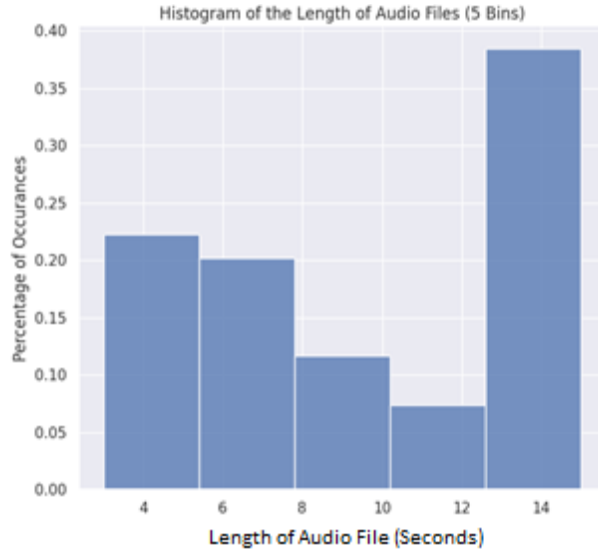


Figure 4: Histogram of Length of Audio Files - With 5 Bins

### 3.4 Data Augmentation Techniques:

Most of the best-performing clustering algorithms rely on the augmentation of signal/input to compute the clusters. In our experiments, such algorithms include SCAN, IMSAT, and IIC. In their default configuration, the augmentations of images are calculated based on scaling, rotating, flipping, and blurring [24]. However, in the context of spectrograms, this does not make sense. Thus, two separate augmentation methods are devised (based on augmenting the raw audio signal). Their explanation and visual examples (in Figure 5) are given below:

**3.4.1. Temporal perturbation.** For this technique, the starting point of the signal is varied (in either direction) and then the augmented signal is padded with noise. The value of shift is chosen randomly between -20% and +20% of the sampling rate. This value was chosen in [25] for improving the singing voice detection using neural networks.

**3.4.2. Speed perturbation.** For this technique, the speed is varied randomly between 0.7x and 1.3x of the original speed. Then to match the new signal length (with

the original signal's length) the audio signal is either sliced (if speed was less than 1x) or padded (if speed was greater than 1x). A variety of choices exist, in literature, for choosing the speed factor. For example, [26] used the factor between 0.5 and 1.5 while [27] used the speed factor between 0.9 and 1.1. For our experiment, a moderate approach of selecting a speed factor between the range of 0.7 and 1.3 was used. In order from left to right, the examples of right shift, left shift, speed decrease and speed increase are provided in Figure 5.

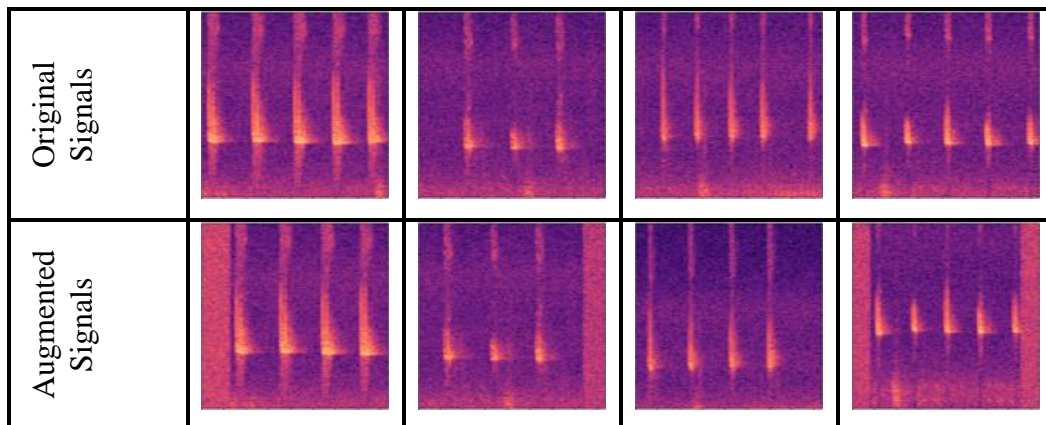


Figure 5: Example of Original and Augmented Spectrograms.

### 3.5 Framework for Classification

Classification of the labeled bats' acoustic dataset through supervised learning has already been demonstrated with an accuracy of around 97% [16]. Three different features (Mel spectrograms, Mel-frequency cepstral coefficients, and Short-time Fourier transform) were used to convert the acoustic data into images. Essentially, these image-based features are trained along with their labels being given as a supervisory signal. This thesis proposes an extension of a similar approach in the domain of unsupervised clustering. Other classification systems (for acoustic data) also exist which utilize image-based features with CNNs [28]. And the work in the field of unsupervised image clustering has too matured during the last 4 to 5 years [29] [30]. For these reasons, our problem has been converted into utilizing the image-based features for classifying the bat calls.

The algorithms shortlisted, given in Table 2 and Table 1, do not make use of transfer learning rather learn from scratch, and that is why they are expected to work equally well on the spectrogram-based features as they do on the standard image's

datasets. The images/spectrograms generated after pre-processing are used as input to the clustering algorithms. They are compatible as input to the algorithms described above because all of them take as input the images.

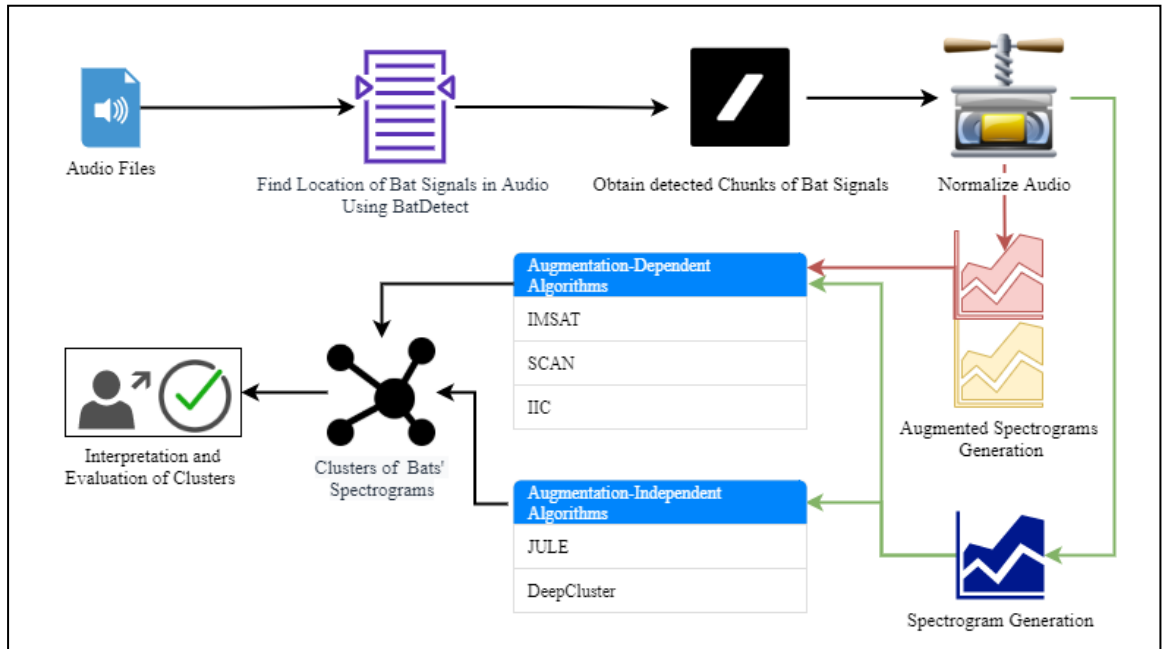


Figure 6: Pipeline for Classification

### 3.6 Framework for Interpretation of Spectrograms Clusters

One-to-One mapping of spectrogram clusters to characterize bats' echolocation calls is very restrictive and nearly impossible because of the uncontrolled environment during the recording of these calls. For example, multiple bats may be present while foraging in different types of habitats. So, there are multiple combinations of factors and trying to isolate one behavior makes the comparison very restrictive. Thus, to make the comparison more comprehensive, each cluster could be interpreted based on multiple factors such as prey capture attempt, presence of multiple bats, habitat type, background noise, and proximity of bat to the prey (via sound pressure level [31] and variations of pulses frequencies). Most spectrograms in each cluster are expected to have a particular signature of these characteristics. The following coding scheme is used for identifying the presence of a particular behavior in the detected bat calls:

**3.6.1. Multiple bats presence.** This could be detected by observing multiple and distinct frequency bands [32]. It confirms the presence of two or more bats who have changed their frequency bands to distinguish their own calls' echoes from the

other bats. Its purpose is the detection, classification, and localization of their prey. The Upper left spectrogram in Figure 7 represents bat calls at two different frequency spectrums. The calls of the first and second bat are annotated by yellow and white color, respectively. There are approximately 5 to 6 calls from each bat and the frequency range of the calls, emitted by different bats, differs by only a few kHz. However, the presence of multiple spectrums of frequency is clear in the mentioned figure.

**3.6.2. Prey capture attempt.** This behavior is characterized by rapid variations of inter-pulses intervals and a decrease of signal frequency [33] [34]. This indicates the shifting of bat behavior from search (also called foraging) to approach phase. The dotted line in the lower left spectrogram inside Figure 7 highlights the prey capture attempt. It could be observed that a sudden drop in the frequency range of bat calls occurs while the number of such calls increases suddenly. This behavior depicts the mentioned definition of prey capture attempt.

**3.6.3. Habitat type.** Bat calls have a faster pulse rate when they are in a close habitat as compared to when they are in an open area. Changes in pulse rate were observed at four different observation sites and bats at the smallest site (20 x 55 m) showed a higher bat call repetition rate [35]. Our spectrograms were generated at a 500ms window and empirically, if there is a presence of five or more bat calls then it indicates a close habitat and vice versa. The upper right spectrogram in Figure 7 has more than six bat calls which is an example involving a close habitat. On contrary, if less than five bat calls are present then it could be classified into an open habitat type.

**3.6.4. Sound pressure level (SPL).** A proxy measure for SPL [31] is signal strength as indicated by the thickness of the bat call. It means that the bats are closer to the ground, or the instrument being used for recording [31]. A thicker bat call represents a high sound pressure level. Given the current parameters for generating spectrogram, spanning 500ms, a bat call is considered to belong to the category of high SPL if it spans more than 10ms horizontally and 15.4Khz vertically. The lower right spectrogram in Figure 7 shows an example of high SPL. Additional sources of sound attenuation include sound scattering by atmospheric turbulences or sound energy dissipation into heat [36]. This does not directly relate to any specific behavior of bat but has a huge impact on the clustering results.

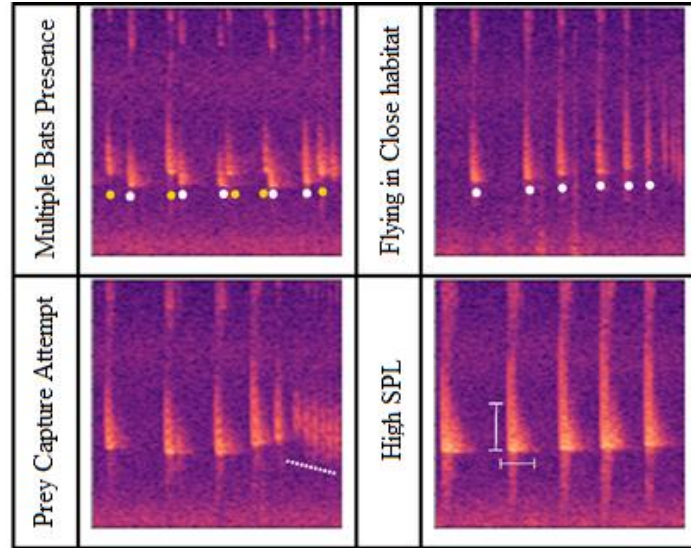


Figure 7: Examples of Different Behaviors

**3.6.5. Noise.** Anthropogenic and background noise (from insects) could introduce artifacts in the generated spectrograms – especially, at lower frequency scales. This also alters the echolocation pattern and activity of bats [37]. Various examples are given in Figure 8.

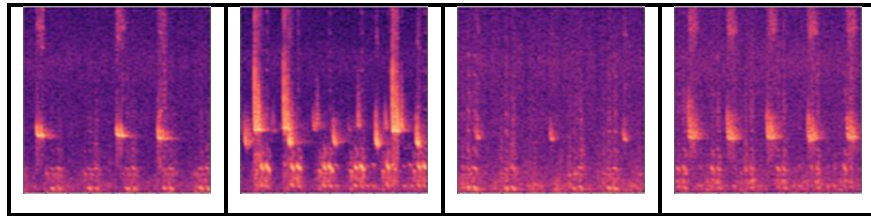


Figure 8: Example of Miscellaneous Signals with Bat Calls

For analysis, the above factors are considered binary variables. A spectrogram is manually labeled across these five dimensions to test the results from clustering algorithms. After clustering is performed by an algorithm, a reasonable number of samples need to be analyzed manually. Thus, to shortlist samples for analysis from each cluster, the following power calculation formulae, given in Equation (7), was used.

$$\text{Sample Size from each cluster} = \frac{\frac{z^2 \times p(1-p)}{e^2}}{1 + \left(\frac{z^2 \times p(1-p)}{e^2 N}\right)} \quad (7)$$

where  $z$  is the z-score (1.96),  $e$  is the margin of error (5%), and  $p$  is the standard of deviation (0.5). Since we don't know how much variance is to be expected, the standard deviation value of 0.5 was used – to make sure that the group is large enough. The distribution of the resulting number of samples, which are required to be manually inspected, is provided in Table 3. It represents the number of samples shortlisted, after applying (7), divided by the total number of samples in a particular cluster. Please note that the justification for selecting five clusters, as represented in Table 3, is given in section 3.9.

Table 3: Number of Shortlisted/Total Samples for all Techniques.

Technique/ Cluster ID	IMSAT	IIC	DeepCluster	SCAN	JULE	K- Medoid
Cluster 0	357/5095	364/7124	339/2866	157/265	378/24577	350/3953
Cluster 1	360/5937	340/2987	343/3169	378/23214	3/3	341/2997
Cluster 2	343/3251	358/5464	376/16857	261/811	14/15	365/7379
Cluster 3	359/5616	303/1445	234/602	165/290	5/5	346/3444
Cluster 4	355/4711	365/7590	286/1116	28/30	10/10	367/6837

### 3.7 Metrics for Quantitative Comparison

The metrics used for comparison of the results from different clustering algorithms and to judge the performance of species clustering are as follows.

**3.7.1. Mutual information score.** The mutual information score [38] between clustering (as explained in section 3.1)  $U$  and  $V$  (for  $N$  datapoints) is given in Equation (8).

$$MI(U, V) = \sum_{i=1}^R \sum_{j=1}^C \frac{|U_i \cap V_j|}{N} \log \frac{P_{UV}(i, j)}{P_U(i)P_V(j)} \quad (8)$$

This metric is symmetric. In the absence of ground truth, this is a good metric to find the clustering output of two different independent algorithms. The output is a non-negative value. This could be normalized, between 0 and 1, by the arithmetic average of entropy of  $U$  and entropy of  $V$ . That is termed as normalized mutual information score. Intuitively, it tells us that how much information could be deduced about one random variable by looking at the second one.

**3.7.2. Rand score.** Rand index (RI) also calculates the similarity between two label assignments. First, all the pairs of samples are computed and then Equation (9) is calculated.

$$RI = \frac{\text{Correct Similar Pairs} + \text{Correct Dissimilar Pairs}}{\text{Total number of pairs}} \quad (9)$$

If clustering does not match,  $RI$  has a value close to 0. Otherwise, a higher non-negative value indicates a better match. It could be adjusted for chance using Equation (10).

$$ARI = \frac{RI - E\{RI\}}{\max\{RI\} - E\{RI\}} \quad (10)$$

Where  $E\{RI\}$  is the expected value of  $RI$ . If the  $RI$  exists because of the chance alone then  $ARI$  has a value close to 0 (could be negative). Otherwise, if both partitions match exactly, the value is 1.

**3.7.3. Fowlkes mallows score.** This metric could only be used if ground truth is available. This metric is used to evaluate species clustering in Chapter 11. Fowlkes Mallows Score (FMS) between the output of two clustering algorithms using Equation (11).

$$FMI = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \quad (11)$$

where TP means the number of true positives, FP means the number of false positives and FN means the number of false negatives. The output value ranges from 0 to 1. A perfect match between two clustering outputs gives the value of 1.

**3.7.4. Accuracy.** This metric is also applicable only if ground truth is available - as is the case in Chapter 11. It is adapted from [1]. The number of clusters is set to the categories in the ground truth and Equation (12) is calculated.

$$ACC = \max_m \frac{\sum_{n=1}^N \mathbf{1}\{l_n = m(c_n)\}}{N} \quad (12)$$

where  $c_n$  and  $l_n$  are the assignment of clusters and the ground truth, respectively.  $m$  represent all the one-to-one mappings between ground truth labels and clusters. It is computed using the Hungarian algorithm [39].

### 3.8 Distance Metrics for K-Medoid Clustering

To get an estimate for the target number of clusters ( $k$ ) to be specified for unsupervised deep learning algorithms, the conventional clustering technique of K-medoid [40] was applied. Experiments were performed using K-medoid with three different distance metrics. The following image comparison metrics were used:

**3.8.1. Structural similarity index metric (SSIM).** In contrast to the pixel-to-pixel image comparison metrics, SSIM [41] was built on the premise that human perception is highly sensitive to the overall structure of images. For two images  $x$  and  $y$  of equal dimensions, SSIM is calculated using Equation (13).

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + (k_1L)^2)(2\sigma_{xy} + (k_2L)^2)}{(\mu_x^2 + \mu_y^2 + (k_1L)^2)(\sigma_x^2 + \sigma_y^2 + (k_2L)^2)} \quad (13)$$

where  $\mu_x$  is the average of  $x$ ,  $\mu_y$  is the average of  $y$ ,  $\sigma_x^2$  is the variance of  $x$ ,  $\sigma_y^2$  is the variance of  $y$ ,  $\sigma_{xy}$  is the covariance of  $x$  and  $y$ ,  $k_1$  and  $k_2$  are constants,  $L$  is the range of pixels. In our experimental setup,  $k_1$  was set to 0.01,  $k_2$  was set to 0.03, and  $L$  was based on maximum and minimum pixel value for each image. Implementation from scikit-image [42] library was used. The above equation is evaluated on a window of image and the mean is taken for all the individual measurements to calculate the overall index. We used the window size of 9x9 pixels (as suggested by the authors [41]). This implementation also supports the multichannel images and calculations were performed independently for each channel and then averaged. The range of this metric is between 0 and 1. For K-Medoid,  $1 - \text{SSIM}$  was used as a distance metric.

**3.8.2. Oriented FAST and rotated BRIEF (ORB).** ORB is built on top of the two image features descriptors FAST (Features from Accelerated and Segments Test) [43] and BRIEF (Binary robust independent elementary feature) [44]. This technique adds rotational invariance and is also resistant to noise. ORB is an improved open-source alternative to SIFT and SURF algorithms. Its implementation [45] from OpenCV was used. This algorithm computes the descriptors of the identified key points for each image. This step is repeated for both images and a brute force matcher is used to get the best match between two images (using Hamming distance [46]). Now, we have a list of matches, and a threshold (50 in our case) was used to obtain a second list. The ratio of the number of matches (after threshold) to the total number of matches

gives the ORB similarity metric. The range of this metric is between 0 and 1. For K-Medoid, 1 - ORB was used as a distance metric.

**3.8.3. Root mean squared error (RMSE).** For a pixel-to-pixel comparison, RMSE was also used as a metric in K-medoid clustering. For two images of size  $M * N$ , it could be calculated using Equation (14).

$$RMSE = \sqrt{\frac{1}{M * N} \sum_{i=0, j=0}^{M-1, N-1} [I(i, j) - K(i, j)]^2} \quad (14)$$

where  $I(i, j)$  and  $K(i, j)$  are the pixel value at the position  $(i, j)$  of two images. The implementation from [47] was used.

### 3.9 Finding an Approximate Number of Clusters Using K-Medoids

The implementation of K-Medoids [48] from scikit-learn was used. The number of target clusters was varied from 1 to 9 (single digits). Alternate implementation which optimizes PAM (Partition Around Medoids) algorithm was used for K-Medoids. Medoid initialization was done by picking the points (equivalent to the number of clusters) with the smallest sum distance to every other point. Maximum iterations were set to 50,000 and the training runs always converged.

K-Medoid has polynomial space complexity of  $O(\text{Number of samples})^2$ . This algorithm is also computationally expensive and does not scale well to the large size of data [49]. For example, in our dataset, computing k-medoid on 500 samples takes 5 minutes but computation on 5000 samples takes 5.80 hours while using the ORB metric. For SSIM, the computation time for clustering 5000 samples was 16.58 hours. The machine had Tesla P100 – 16GB GPU, 24GB memory, 200GB disk space, and 2vCPU at 2.2GHz.

This motivated finding a number for a reasonable number of samples that will be representative of the dataset and provide a good estimation for  $k$ . To estimate the percentage of data that could represent the overall dataset, empirical results from mini-batch active learning (clustering) were considered [50]. On the CIFAR-10 dataset, only 10,000 (out of a total of 50,000) samples were used to get 60% accuracy, and for MNIST, only 1,000 (out of a total of 60,000) samples were used to get 95% accuracy. For this K-Medoids, 20% samples (5,000 from a total of 24,610) were chosen randomly

from the overall dataset. Figure 9 suggests that elbow analysis (from each metric) concentrates around the 5/6 (as the target number of the cluster). Inertia is the sum of distances of samples to their closest cluster center. The slight deviation exists because of the difference in the nature of each metric. To assist with evaluation, the number of target clusters was set to 5 since the number of dimensions to evaluate it against, as described in section 3.6 is also 5.

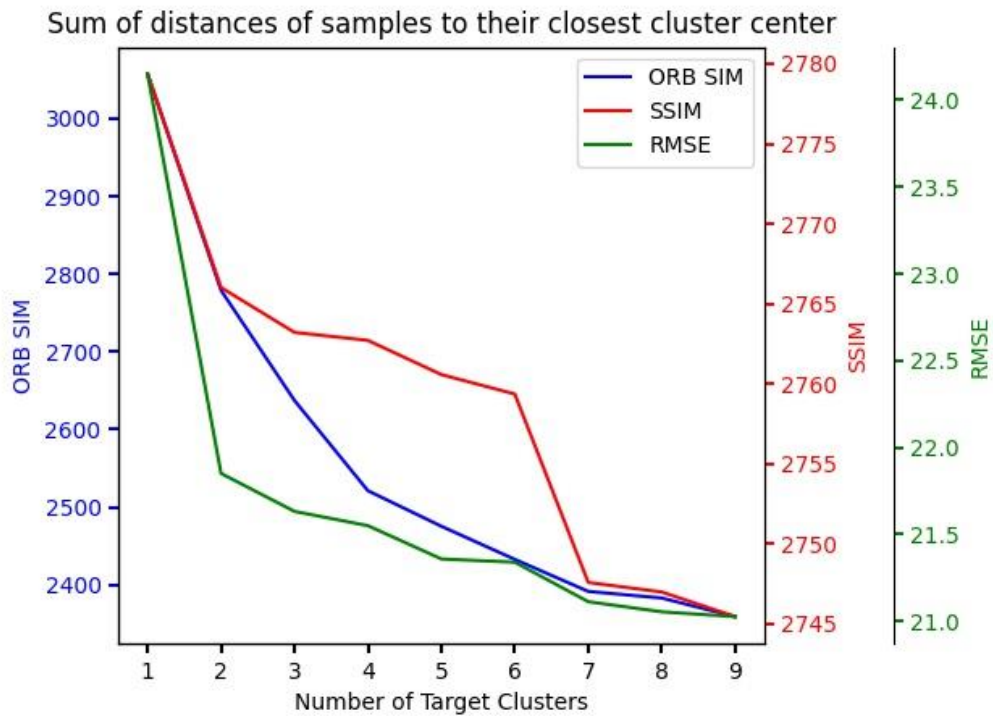


Figure 9: Inertia Based On ORB, SSIM, And RMSE Vs Target Clusters

This Chapter covers data collection and pre-processing as well as a formal definition of the clustering problem. It presents the frameworks for grouping and understanding bat calls. The initial experiments that were used to determine the input parameters for clustering algorithms are also explained, as well as the numerous metrics that were used to evaluate the results.

## Chapter 4. Establishing the Baseline using K-Medoid Clustering

### 4.1 Theoretical Explanation and Training of K-Medoid

To highlight the significance of utilizing the selected image clustering algorithm and judge their performance, it is important to set the baseline results using the conventional clustering technique (K-Medoid). Multiple variations of this algorithm exist in the literature [51] however the implementation from [52] is used for this work. In comparison to the popular K-Means algorithm, K-Medoid is more robust to outliers and chooses the actual datapoints as the cluster center. Another advantage of K-medoid is that custom distance metrics could also be used easily with this algorithm - as demonstrated in section 3.8. Euclidean distance metric was used to perform clustering on our dataset. The target number of clusters was set to 5.

### 4.2 Results

The resulting clusters are analyzed across two dimensions i.e., vertical, and horizontal. The vertical view describes the distribution of a single characteristic across the clusters. On the other hand, the horizontal view describes the actual attribution of each cluster, across all characteristics, in comparison to other clusters. The same method is used to analyze the rest of the algorithms as well.

Based on the power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 4, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters.

**4.2.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals that the presence of multiple bats occurs in all the clusters range from the proportion of 3.84% to 27.47%. A similar trend (with different proportions) occurs with the rest of the characteristics as well. So, no characteristic is dominant in an individual cluster.

**4.2.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, revealed that no significant pattern is visible. The resulting clusters do not show much deviation from each other. Based on the framework defined in section 3.6, no conclusion could be

made. It suggests that the algorithm failed to cluster based on the semantics of spectrograms. This was the motivating factor behind modifying and utilizing the image clustering algorithm for the application of spectrograms clustering.

Table 4: Numerical Distribution of Bats calls' Characteristics for K-Medoid

<b>Characteristic /Cluster ID</b>	<b>Multiple Bats Presence</b>	<b>Prey Capture Attempt</b>	<b>Flying in Close habitat</b>	<b>High SPL</b>	<b>Noise</b>	<b>Percentage</b>
Cluster 0	14.00%	0.29%	44.00%	56.86%	14.86%	16.06%
Cluster 1	25.00%	2.35%	47.35%	70.29%	53.53%	12.18%
Cluster 2	3.84%	0.27%	20.55%	15.34%	22.47%	29.98%
Cluster 3	10.98%	1.16%	56.94%	47.69%	10.40%	13.99%
Cluster 4	27.47%	1.37%	65.11%	85.16%	28.30%	27.78%

All the clusters show mix distribution of behavior, and no conclusion could be drawn. As per the feedback from the expert, after looking at the formed clusters, it was hypothesized that clusters formed might be influenced more by the specie of bat, as compared to the behavior.

## Chapter 5. Learning Discrete Representations via Information Maximizing Self-Augmented Training (IMSAT)

### 5.1 Theoretical Explanation of IMSAT

Building on the concept of getting lower-dimension representations of the data which capture its non-linearity and could be used for the application of interest, i.e., clustering, **IMSAT** was proposed [1]. Intuitively, this approach encouraged the output of the network to remain the same, even when the input image was transformed. Secondly, the dependency between the embedding and input data was also maximized. In precise terms, this approach maximized the MI between input  $X$  and the cluster it was assigned to,  $Y$  (shown by the blue arrow in Figure 10) and regularized using the self-augmented training (shown by the red arrow in Figure and explained below). The overall objective of the overall model was to minimize the objective, given in Equation (15).

$$\mathcal{R}_{\text{SAT}}(\theta; T) - \lambda[H(Y) - H(Y | X)] \quad (15)$$

where  $X$  is the input,  $Y$  is the clustering output, and  $T$  is the augmentation function.  $\lambda$  is the trade-off hyper-parameter. The mutual information part (on the right side), which maximized the MI, could be expanded – as provided in Equations (16) and (17):

$$H(Y) \equiv h(p_{\theta}(y)) = h\left(\frac{1}{N} \sum_{i=1}^N p_{\theta}(y | x_i)\right) \quad (16)$$

$$H(Y | X) \equiv \frac{1}{N} \sum_{i=1}^N h(p_{\theta}(y | x_i)) \quad (17)$$

where  $P_{\theta}(y|x)$  represents the neural network used, and  $N$  is the number of training examples. And the  $\mathcal{R}_{\text{SAT}}$  is provided in Equations (18) and (19).

$$\mathcal{R}_{\text{SAT}}(\theta; T) = \frac{1}{N} \sum_{n=1}^N \mathcal{R}_{\text{SAT}}(\theta; x_n, T(x_n)) \quad (18)$$

$$\begin{aligned} & \mathcal{R}_{\text{SAT}}(\theta; x, T(x)) \quad (19) \\ &= - \sum_{m=1}^M \sum_{y_m=0}^{V_m-1} p_{\theta}(y_m | x) \log p_{\theta}(y_m | T(x)) \end{aligned}$$

The goal was to get the multi-output probabilistic classifier whose dimension was  $M$ . After applying the augmentation on the input image  $x$  using the function  $T$

(VAT [53] was used). Equation (19) was used to enforce the invariance, i.e., give a similar output representation on the augmented image as it was on the original image. On CIFAR-10, IMSAT achieved an accuracy of 45.6%.

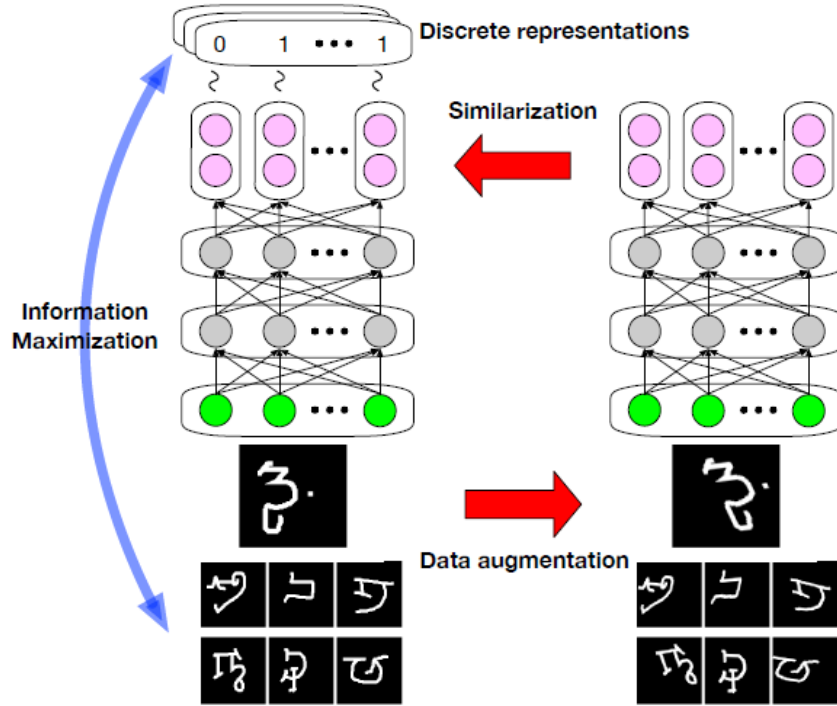


Figure 10: Basic Idea of IMSAT [1]

## 5.2 Training on Our Dataset

The augmentation function  $T$  in Eq. (15), (18), and (19) was replaced with the temporal and speed perturbation (as explained in section 3.4). The learning rate was set to 0.02 with a batch size of 200. The convergence took place after around 100 epochs. The value of  $\lambda$  in Eq. (15) was set to the default value of 0.1. Based on the results from section 3.9, the target number of clusters was set to 5. The updated Pytorch implementation with selected hyperparameters could be found here [54]. The Learning Curve is provided in Figure 11. The loss of the model decreased over time until it converged. This indicates that the model learned according to the defined objective function and clustered the bat calls into different groups. To interpret these clusters, the investigation of the bat calls grouped in each cluster is provided in detail in the next section.

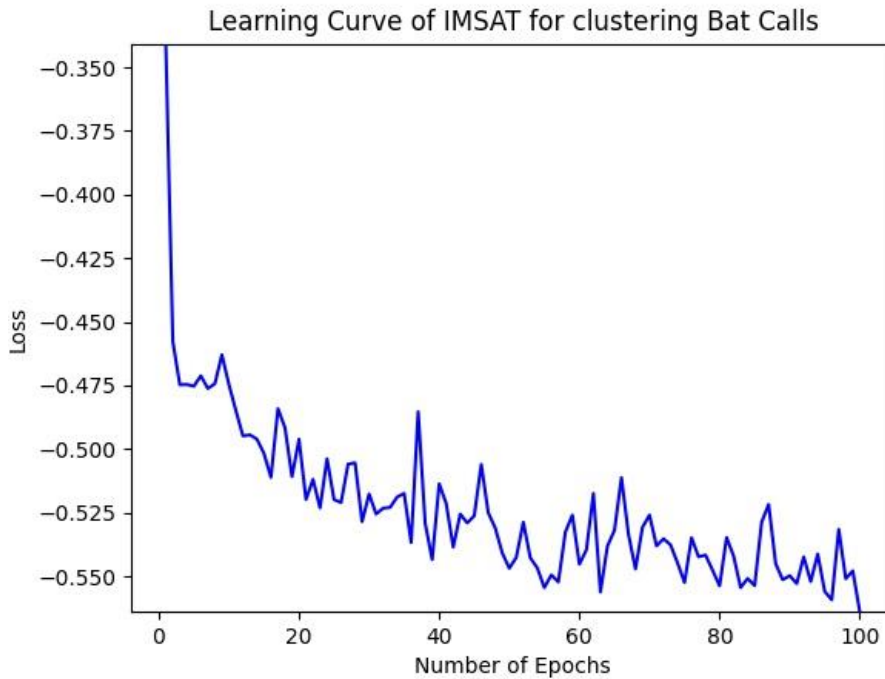


Figure 11: Learning Curve of IMSAT for Bats Calls Clustering

### 5.3 Results

Based on the power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 5, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters. Figure 12 shows 5 representative samples from each cluster of IMSAT.

Table 5: Numerical Distribution of Bats calls' Characteristics for IMSAT

Characteristic /Cluster ID	Multiple Bats Presence	Prey Capture Attempt	Flying in Close habitat	High SPL	Noise	Percentage
Cluster 0	2.52%	0.00%	22.67%	24.65%	80.11%	20.70%
Cluster 1	55.56%	4.44%	70.28%	97.78%	5.83%	24.12%
Cluster 2	0.29%	0.00%	7.29%	4.37%	5.54%	13.21%
Cluster 3	0.56%	0.00%	25.35%	38.44%	8.64%	22.82%
Cluster 4	3.94%	1.69%	89.86%	89.58%	4.79%	19.14%

**5.3.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals the following observations:

- The presence of multiple bats is captured by cluster 1 with a proportion of 55.6%. The rest of the clusters do not show any significant social activity.
- A major Proportion of prey capture attempts happens in cluster 1. The second set of observations for this behavior happens in cluster 4. This cluster also shows a minor presence of multiple bats (3.9%) suggesting the correlation between prey capture behavior and the presence of multiple bats.
- The close habitat is distributed across different clusters – each cluster showing a different proportion of this characteristic. The highest proportion (89.6%) of the close habitat is present in cluster 4.
- A major proportion (80.1%) of noise is present in Cluster 0. The rest of the clusters are not significantly affected by the noise.

**5.3.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, could reveal the defining characteristics/attributions of those clusters.

- In cluster 0, the defining characteristic is a high Noise level. In terms of other characteristics, this cluster is closest to cluster 3. However, cluster 3 has only 8.6% proportion of noise.
- In cluster 1, the defining characteristic is the presence of multiple bats. Apart from this factor, this cluster is the same as cluster 4.
- In cluster 2, the defining characteristics are open habitat and low SPL.
- In cluster 3, the defining characteristic is the low level of noise.
- In cluster 4, the low presence of multiple bats is the defining characteristic.

Based on the above two analyses and feedback from the expert (provided in Table 6), it could be concluded that cluster 0 captures bat calls with high background noise. Cluster 1 captures bat calls where multiple bats are present, and they are flying in close habitat. Cluster 2 captures bats flying in an open environment at a distant place from the microphone. Cluster 3 capture bat calls indicate normal foraging. Cluster 4 indicates normal foraging in a close environment, close to the microphone.

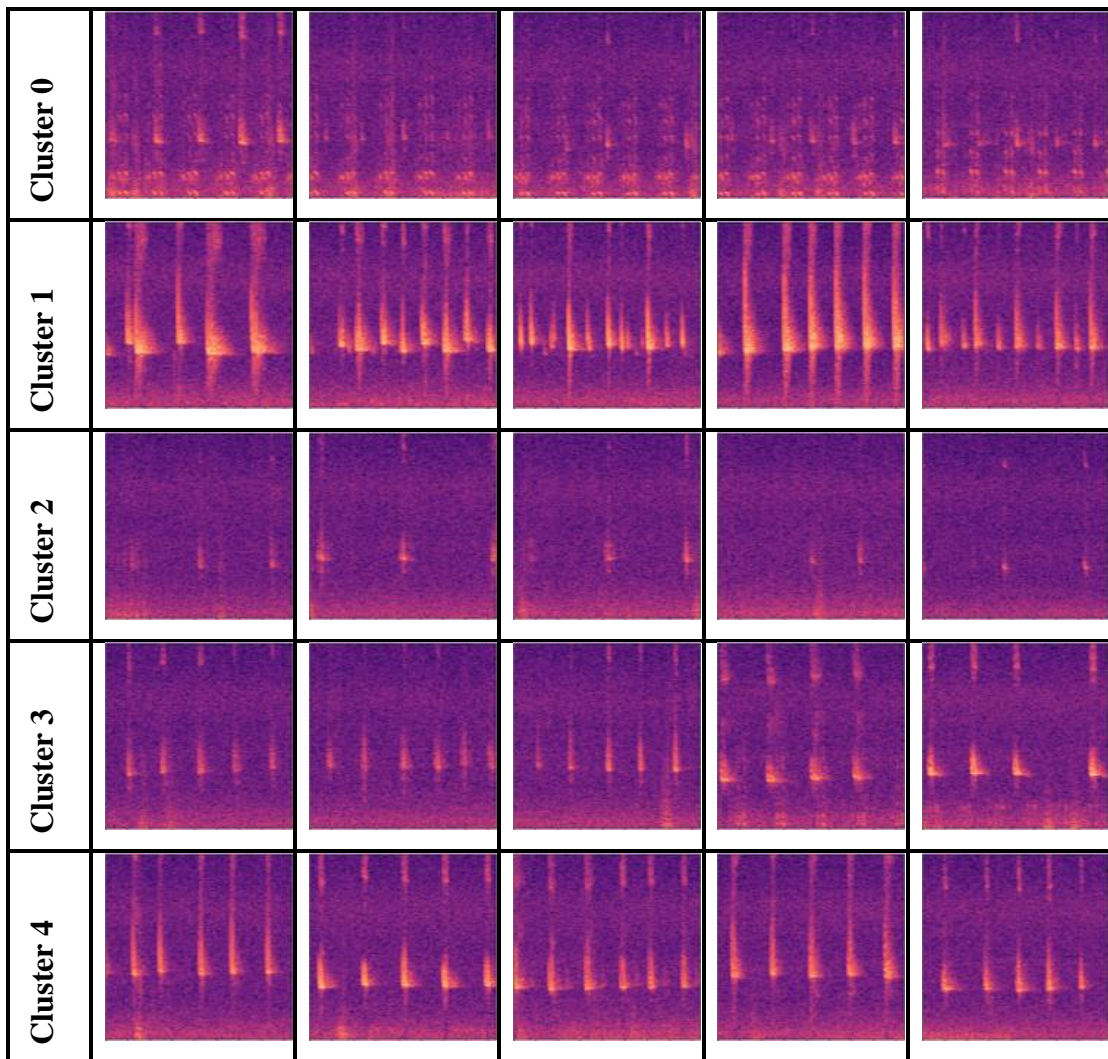


Figure 12: Representative Samples from the Cluster Assignment of IMSAT

It could be observed that every cluster has a unique signature of characteristics and this interpretation, based on the coding scheme developed in section 3.6, was validated and complemented by the opinion of an expert. Cluster 1 has a dominantly close environment and it was pointed out that various external factors might also affect the bat calls frequencies and inter-pulse rates (such as bat flying over the bond or close to the microphone). However, these factors have a negligible effect and thus the interpretation obtained through coding scheme could be considered true with a high level of confidence. For cluster 2, the estimate of height (at which bats might be flying) is estimated by the expert to be higher than a typical tree. For cluster 3, various terminological issues were fixed – as suggested by the expert. The detailed comments from the terrestrial biodiversity expert on the interpretation of clusters formed through IMSAT are given in Table 6.

Table 6: Comments on the Interpretation of IMSAT Results by a Field Expert

Cluster ID	Comments
Cluster 1	Variations in inter-pulse rates and frequencies might be related to other factors than close/ open habitats. Slight variations might however potentially occur if the bats flew close from the tree where the microphone was deployed, or over the pond (more open habitat) nearby. I would not expect these differences to be very significant.
Cluster 2	Important inter-pulses intervals would suggest a bat passing by in rather open area, possibly at higher height than the tree where the microphone was deployed.
Cluster 3	I would see these spectrograms with regular frequencies and inter-pulses intervals as “foraging”, which is the search for preys, to distinguish from what you call “foraging” which is “prey capture attempt”, after having detected a prey while foraging. (This comment was addressed through renaming the terminology)
Cluster 4	Should be foraging at close distance (so high sound pressure level).

## Chapter 6. Invariant Information Clustering (IIC)

### 6.1 Theoretical Explanation of IIC

IIC [2] was based on the concept of co-clustering [55] [56] (which are also in general single-stage networks). The intuition behind was the same as IMSAT, i.e., invariant/preserved information could be learned (from the original image and the transformation) to be used for clustering. Formally, this approach also worked on the principle of maximizing the MI between the class allocation of each pair, i.e., the original image and the transformation of the original image (through scaling, rotation, skewing, flipping, varying saturation, and contrast) such that the essence of the image did not change then the invariant/preserved information could be learned and used for clustering (shown in Figure 13). This method is generalizable on any paired data (not limited to Images). This does not output the embeddings/representations (unlike the other algorithms [57] [19] [58] [59] ) and directly outputs the cluster/semantic labels.

One advantage of this method was that it avoided the degenerate solutions, which meant it usually did not could give less than the required/desired number of clusters [60]. The accuracy of 88.8% was achieved on the STL10 dataset [61] for classification. At the time of publishing, this led to the improvement of 9.5 percentage points on CIFAR-10 image classification.

If  $x$  belong to the unlabeled dataset,  $x'$  represents its transformation, and both are fed to the neural network  $\phi$  whose output is given as  $\phi(x)$  (whose output has softmax activation and then the applies is one-hot encoded) such that the common features are preserved between  $x$  and  $x'$ , then our objective function is given in Equation (20) follows:

$$\max_{\phi} I(\phi(x), \phi(x')) \quad (20)$$

where the representation space, i.e., the cluster (represented by  $z$ ) after passing through the neural network is  $\{1, \dots, C\}$ . i.e.,  $\phi(x) \in [0,1]^C$ . The value of  $C$  is fixed. The final objective function for the IIC model is simplified to give Equation (21).

$$I(z, z') = \sum_{c=1}^C \sum_{c'=1}^C P_{cc'} \ln \frac{P_{cc'}}{P_c \cdot P_{c'}} \quad (21)$$

where  $P_{cc}$  represents the joint probability. Also, the activation used was soft-max.

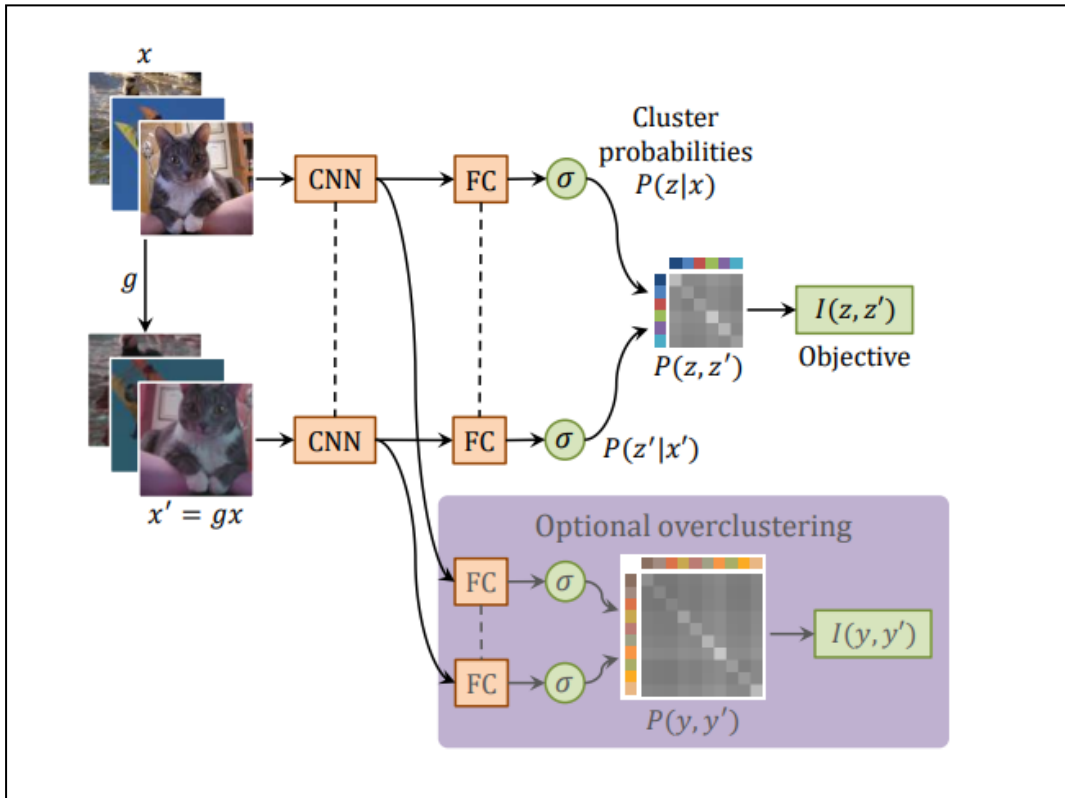


Figure 13: Flowchart of IIC [2]

Figure 13 illustrates the transformation function  $g$  converting the original image  $x$  to  $x'$  and both being fed to the network. The dashed line means that the parameters are shared between them. After training  $\phi(x)$ , which is one-hot encoded, gives the cluster.

## 6.2 Training on Our Dataset

The learning rate was set to 0.001 with a batch size of 200. The convergence took place after around 200 epochs. The target number of clusters was set to 5. The updated TensorFlow implementation with selected hyperparameters could be found here [62]. The Learning Curve (of two heads) is provided in Figure 14. The loss of the model decreased over time until it converged. This indicates that the model learned according to the defined objective function and clustered the bat calls into different groups. To interpret these clusters, the investigation of the bat calls grouped in each cluster is provided in detail in the next section.

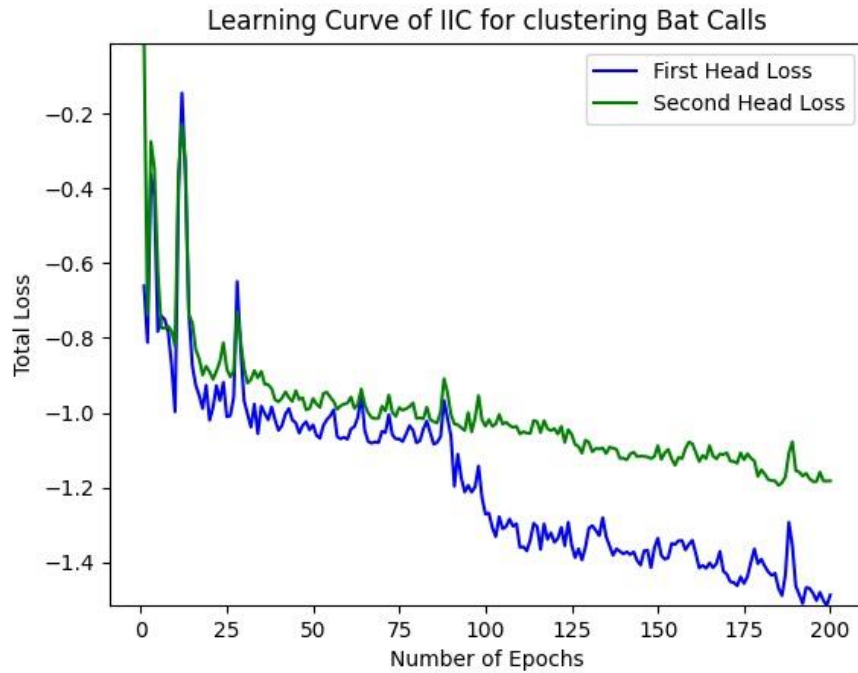


Figure 14: Learning Curve of IIC for Bats Calls Clustering

### 6.3 Results

Based on the power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 7, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters. Figure 15 shows 5 representative samples from each cluster of IIC.

Table 7: Numerical Distribution of Bats calls' Characteristics for IIC

Characteristic /Cluster ID	Multiple Bats Presence	Prey Capture Attempt	Flying in Close habitat	High SPL	Noise	Percentage
Cluster 0	62.36%	3.30%	67.58%	95.05%	7.69%	28.95%
Cluster 1	3.82%	0.00%	43.24%	12.94%	82.35%	12.14%
Cluster 2	2.23%	0.00%	13.97%	5.31%	10.89%	22.20%
Cluster 3	12.54%	0.00%	32.67%	69.97%	95.05%	5.87%
Cluster 4	7.95%	1.37%	68.77%	86.85%	6.03%	30.84%

**6.3.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals the following observations:

- The presence of multiple bats is captured by cluster 0 with a proportion of 62.36%. Unlike IMSAT, other clusters also indicate this behavior.
- The distribution for prey capture attempt is almost the same as in IMSAT. The highest proportion (3.3%) is found in cluster 0, followed by cluster 4.
- The proportion of close habitat is the same in both clusters 4 and 5. The rest of the clusters show a dominantly open habitat.
- Each cluster has a different proportion of SPL. The highest SPL is shown by clusters 0 and 4, while the lowest SPL is exhibited by cluster 2.
- The highest proportion of background noise is present in clusters 1 (82.35%) and 3 (95.05%). For the rest of the clusters, this number is less than 11%.

**6.3.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, revealed the defining characteristics of those clusters.

- In cluster 0, the defining characteristic is the presence of multiple bats, with a proportion of 62.36%. Cluster 4 has almost the same characteristics as cluster 0, except that it has a low proportion (7.95%) of the mentioned behavior.
- In cluster 1, the defining characteristic is SPL level, with a proportion of 12.94%. For the rest of the characteristics, cluster 3 shows almost the same proportion as cluster 1 but the difference (of 57.03%) occurs in SPL level.
- In Cluster 2, a mix of noise level (10.89%) and SPL (5.31%) is the defining characteristic.

Based on the above two analyses and feedback from the expert (provided in Table 8), it could be concluded that cluster 0 captures bat calls with multiple bats presence, with a majority of them flying in a close habitat and some of them making prey capture attempt. Cluster 1 captures normal foraging bat calls with noise in the background. Cluster 2 captures bat calls that are in open habitat and away from the microphone. Cluster 3 captures bat calls that are close to the ground/microphone and have noise in the background. Cluster 4 captures normal foraging happening close to the microphone/ground.

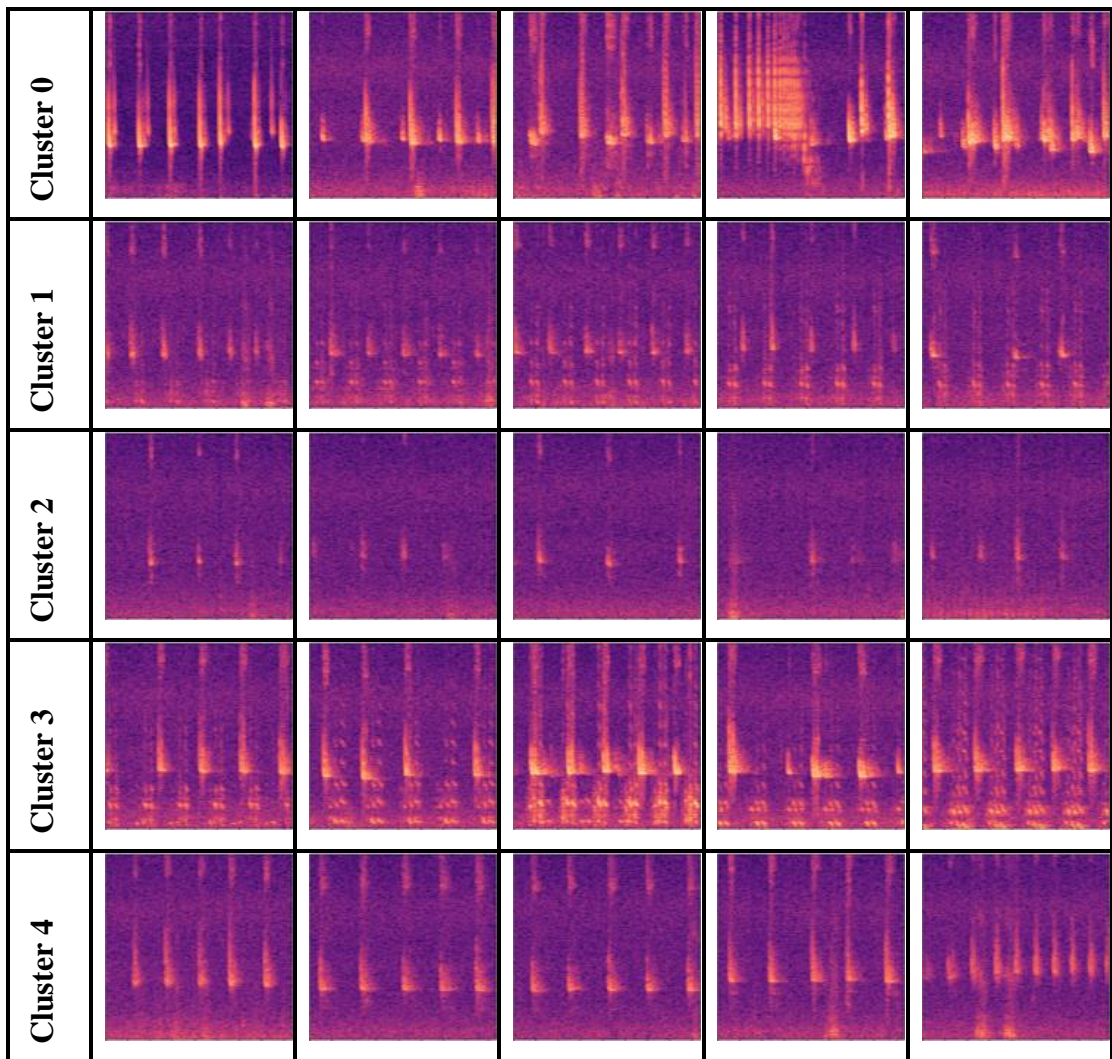


Figure 15: Representative Samples from the Cluster Assignment of IIC

The comments from a terrestrial biodiversity expert on the interpretation of clusters formed through IIC are given in Table 8.

Table 8: Comments on the Interpretation of IIC Results by a Field Expert

Cluster ID	Comments
Cluster 0	It indicates clearly simultaneous presence of 2 or more individuals
Cluster 1	Looks like normal foraging
Cluster 2	Open environment
Cluster 3	Foraging at close distance (high SPL)
Cluster 4	Normal foraging

## Chapter 7. Deep Clustering for Unsupervised Learning of Visual Features

### 7.1 Theoretical Explanation of DeepCluster

DeepCluster [5] was an end-to-end training model for large-scale datasets. The intuition behind this approach was the same as Deep Embedding Clustering (DEC), i.e., using standard clustering algorithms to learn the initial clusters (after obtaining the embeddings) and then iteratively improving on it using the neural networks, except that it used different components to implement it. Specifically, this combined the training of convolutional neural networks (for the extraction of features) with the clustering process (using k-means) so that a complete end-to-end clustering model was built for a large amount of data (in the form of images) where we did not have the labels available. As illustrated in figure 7, the set of features from all images/data points were clustered using k-means, and these label assignments were used as pseudo-labels for training. Essentially, it iterated between updating the parameters/weights of the CNN and clustering the features obtained using k-means (after every epoch).

The two components of the network are classifier  $g_W$  (which performed k-mean and had parameter  $W$ ) and CNN (with parameter  $\theta$ ). They were learned together using the objective, given in Equation (22), with  $N$  training examples.

$$\min_{\theta, W} \frac{1}{N} \sum_{n=1}^N \ell(g_W(f_\theta(x_n)), y_n) \quad (22)$$

The loss function used was multinomial logistic loss and the cluster assignment/pseudo-labels ( $y_n$ ) was found using Equation (23).

$$\min_{C \in \mathbb{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^N \min_{y_n \in \{0,1\}^k} \|f_\theta(x_n) - C y_n\|_2^2 \quad (23)$$

where  $k$  is the number of distinct groups (fixed), and  $C$  is the centroid matrix. In other words, Equation (23) was used to generate the pseudo-labels, and Equation (22) was used to predict the pseudo-labels for updating the parameters of the CNN. This gave the accuracy of 73.7% on the PASCAL VOC dataset. One advantage of this approach was that this is easily generalizable and does not assume a lot of knowledge about the specific domain. Also, the authors claimed that in the specific case of unsupervised

learning where we have limited knowledge about the domain and no labels, this approach offers good performance.

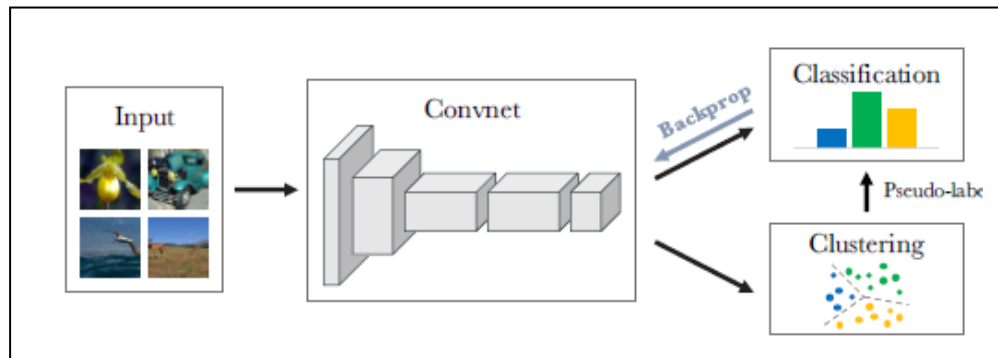


Figure 16. Illustration of DeepCluster[5]

## 7.2 Training on Our Dataset

The learning rate was set to 0.05 with a batch size of 100. The target number of clusters was set to 5 and the convergence took place after 30 epochs. For Convnet VGG-16 was used. The updated Pytorch implementation with selected hyperparameters could be found here [63]. The Learning Curve is provided in Figure 17. The loss of the model decreased over time until it converged.

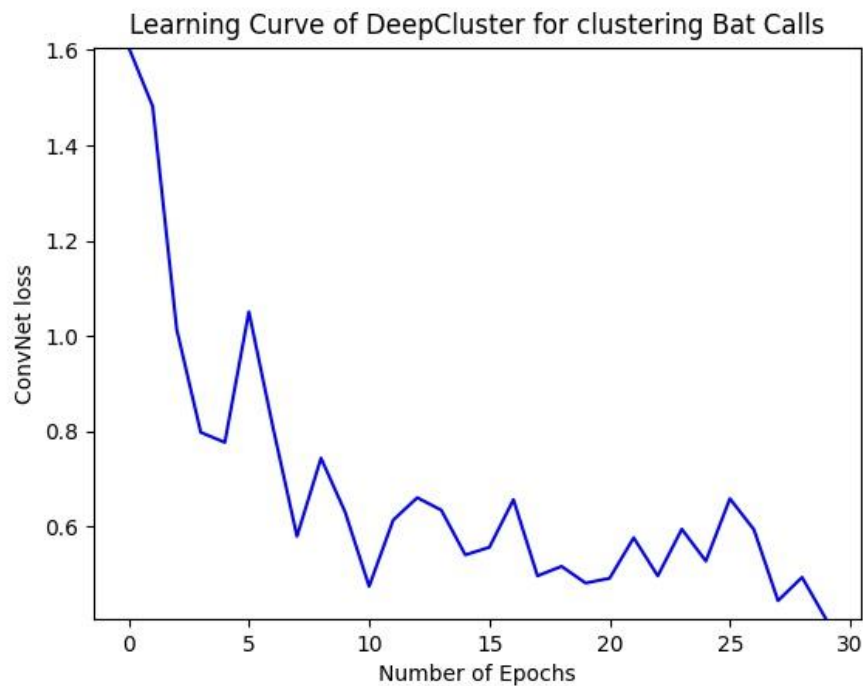


Figure 17: Learning Curve of DeepCluster for Bats Calls Clustering

### 7.3 Results

Based on the power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 9, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters. Figure 18 shows 5 representative samples from each cluster of DeepCluster.

Table 9: Numerical Distribution of Bats calls' Characteristics for DeepCluster

Characteristic /Cluster ID	Multiple Bats Presence	Prey Capture Attempt	Flying in Close habitat	High SPL	Noise	Percentage
Cluster 0	9.44%	0.0%	60.77%	58.11%	4.42%	11.65%
Cluster 1	2.92%	0.0%	79.88%	4.66%	4.96%	12.88%
Cluster 2	16.76%	3.99%	52.93%	93.35%	13.30%	68.50%
Cluster 3	1.71%	0.0%	2.14%	0.85%	11.54%	2.45%
Cluster 4	1.05%	0.0%	11.89%	0.00%	1.05%	4.53%

**7.3.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals the following observations:

- The presence of multiple bats does not have a significant effect on clusters formation. Cluster 2 has the highest proportion (16.76%) of prey capture attempts.
- Prey capture attempt is captured by cluster 2. Unlike IMSAT and IIC, no other cluster has any instances of this behavior.
- The highest proportion (79.88%) of close habitat occurs in cluster 1. Cluster 0 and 2 also dominantly show a close habitat, whereas the rest of the clusters indicate an open habitat.
- The proportion of high SPL varies significantly across the clusters. The highest proportion (93.35%) occurs in cluster 2 while the lowest proportion (0%) occurs in cluster 4. This indicates SPL's significant effect on clusters formation with DeepCluster.
- Just like the presence of multiple bats, noise also does not have a considerable effect on clusters formation.

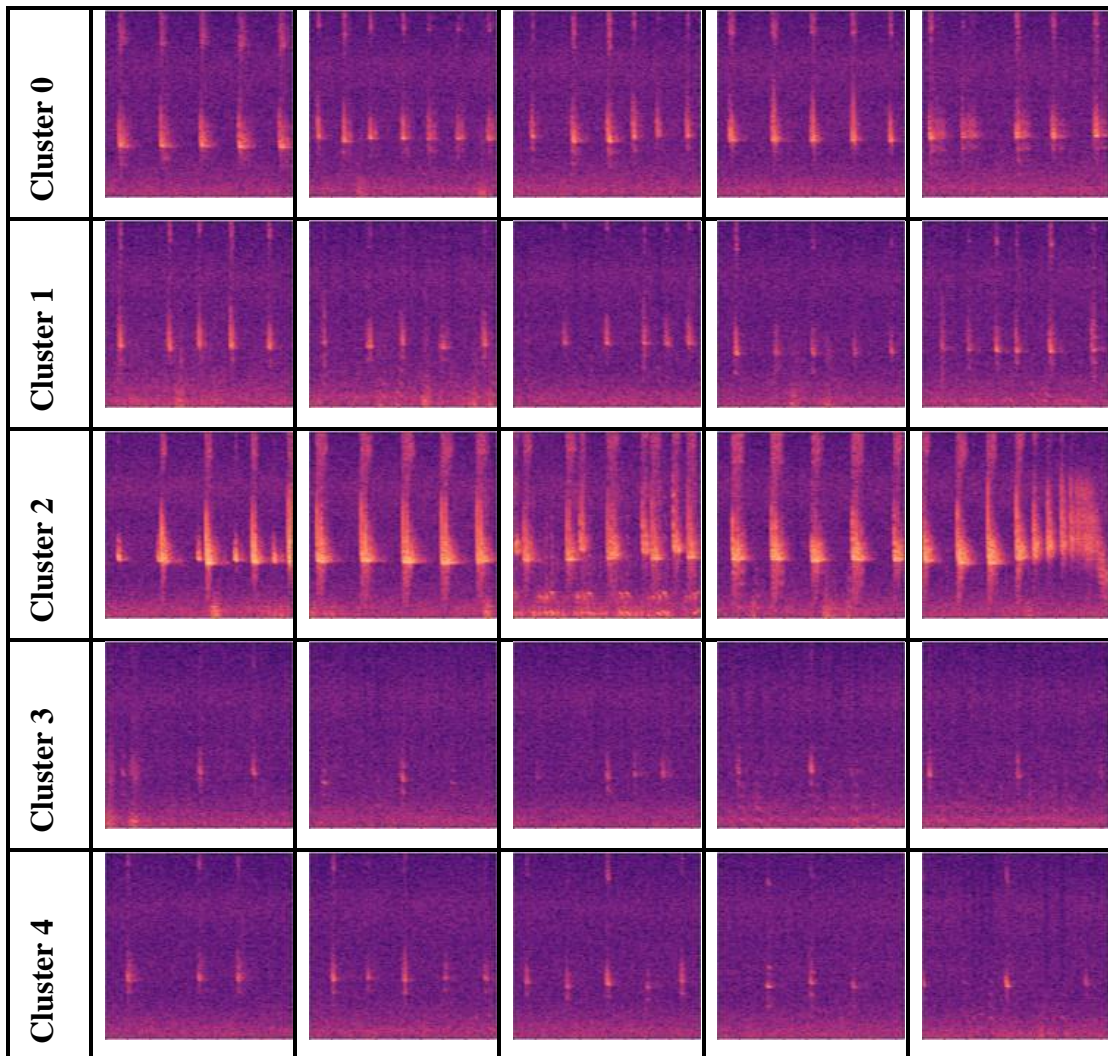


Figure 18: Representative Samples from the Cluster Assignment of DeepCluster

**7.3.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, revealed the defining characteristics of those clusters.

- In cluster 0, the defining characteristic is SPL. The closest cluster to cluster 0 is cluster 1 apart from the difference in SPL. The former has an SPL proportion of 58.11% whereas the latter has an SPL proportion of 4.66%.
- In cluster 2, the prey capture attempt is the dominant characteristic – with a proportion of 3.99%. In terms of other characteristics, cluster 0 is closest to cluster 2.

- Cluster 3 and 4 could be characterized in terms of the mix of noise and habitat type. Cluster 3 has a relatively higher proportion (11.54%) of noise and a lower proportion (2.14%) of close habitat. Cluster 4, on the other hand, has a relatively lower proportion (1.05%) of noise and a higher proportion (11.89%) of close habitat.

Based on the above two analyses and feedback from the expert (provided in Table 10), it could be concluded that cluster 0 captures mixed behavior with a majority of them flying in close habitats. Cluster 1 captures bat calls flying in close habitats away from the microphone/ground. Cluster 2 captures prey capture attempts in bats that are flying close to the ground/microphone. Cluster 3 captures bat calls in an open habitat exhibiting normal foraging. Almost all captures in cluster 4 include the bat calls that are away from the ground/microphone and exhibiting normal foraging. The comments from a terrestrial biodiversity expert on the interpretation of clusters formed through DeepCluster are given in Table 10.

Table 10: Comments on the Interpretation of DeepCluster Results by an Expert

<b>Cluster ID</b>	<b>Comments</b>
Cluster 0	Normal foraging
Cluster 1	Normal foraging a bit distant from the mic (low SPL)
Cluster 2	There is a mix of different category here. Presence of 2 individuals, at close distance (high SPL), the spectrogram on right is a prey capture attempt
Cluster 3	Foraging or direct flight in open environment
Cluster 4	Normal foraging a bit distant from the mic

## Chapter 8. Semantic Clustering by Adopting Nearest neighbors (SCAN)

### 8.1 Theoretical Explanation of SCAN

In contrast to [20], a multi-step approach was presented in which the clustering and learning were separated/decoupled from each other. This approach is termed SCAN (Semantic Clustering by Adopting Nearest neighbors) [3]. Intuitively, this approach could be thought of as an extension to the IIC and IMSAT, because in addition to considering the invariance due to transformation, this also considered the invariance due to the other semantically similar images (termed as the nearest neighbors). First, a pretext task is used to minimize the distance between the original image  $X_i$  and its transformation/augmentation  $T[X_i]$  – as given in Equation (24).

$$\mathit{mind}_{\theta}(\Phi_{\theta}(X_i), \Phi_{\theta}(T[X_i])) \quad (24)$$

where  $\phi_{\theta}$  is the embedding function, which was implemented using a neural network with weights  $\theta$ . After the embedding of images was obtained, K-nearest neighbors are shortlisted for each of them.

Another neural network  $\phi_{\eta}$  was defined for clustering.  $\phi_{\eta}$  which was responsible for classifying the datapoint  $X_i$  and the nearest neighbors ( $\mathcal{N}_{X_i}$ ) calculated in the previous step. The output of  $\phi_{\eta}$  had the soft-max activation, and it gave the soft assignment over  $C$  clusters (like  $\{1, \dots, C\}$ ). Also,  $C$  was fixed. Given the datapoint  $X_i$ , its output belonged to  $[0,1]^C$  – as represented in Equation (25).

$$\Lambda = -\frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \sum_{k \in \mathcal{N}_X} \log \langle \Phi_{\eta}(X), \Phi_{\eta}(k) \rangle + \lambda \sum_{c \in \mathcal{C}} \Phi_{\eta}^c \log \Phi_{\eta}^c \quad (25)$$

where  $\phi_{\eta}^c(X_i)$  means the probability of assigning  $X_i$  to the  $c$  cluster. The second term (entropy) in the equation was used to distribute the cluster assignment uniformly. In the final state, self-labeling was employed. It means the model learned again from the high-confidence labels (which were above a threshold).

### 8.2 Training on Our Dataset

The learning rate was set to 0.4 (for pretext task) and  $10^{-3}$  (for SCAN step). The batch size was set to 50 (in both cases).  $\mathcal{N}$  was set to 20. The target number of clusters was set to 5 and the convergence took place after around 50 epochs (for pretext

task) and 33 epochs (with early stopping for SCAN’s consistency training). As a network backbone, ResnNet50 was used. The updated Pytorch implementation with selected hyperparameters could be found here [64]. The Learning Curve is provided in Figure 19. The loss of the model decreased over time until it converged. The loss of the model decreased over time until it converged. This indicates that the model learned according to the defined objective function and clustered the bat calls into different groups. To interpret these clusters, the investigation of the bat calls grouped in each cluster is provided in detail in the next section.

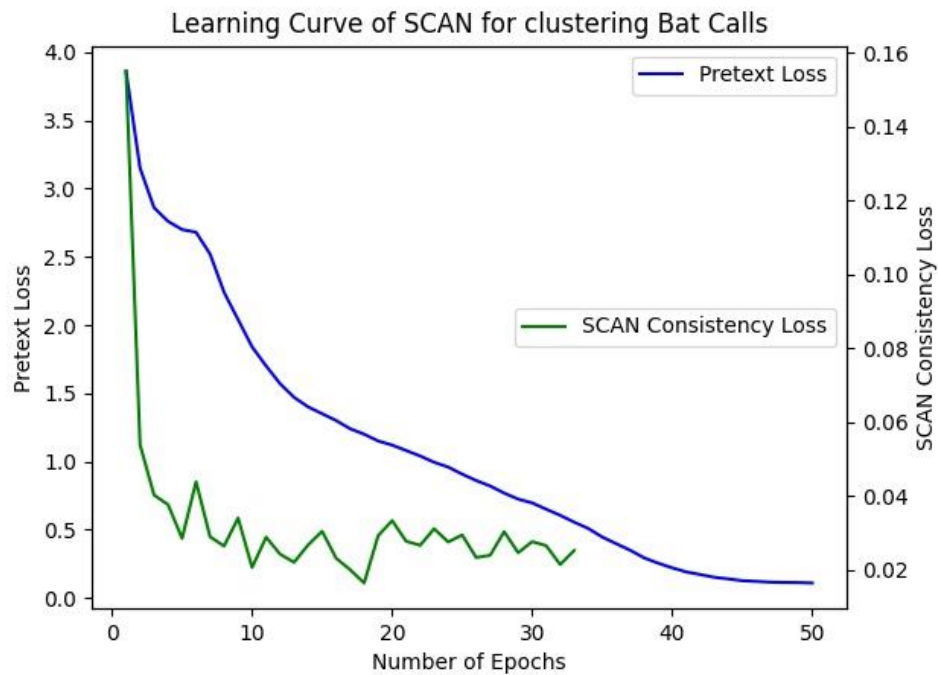


Figure 19: Learning Curve of SCAN for Bats Calls Clustering

### 8.3 Results

Based on the power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 11, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters. Figure 20 shows 5 representative samples from each cluster of SCAN.

**8.3.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals the following observations:

- The presence of multiple bats is captured in clusters 1 and 3 with an almost equal proportion of 27.51% and 25.45% respectively.
- Unlike IMSAT, IIC, and DeepCluster, prey capture attempt does not affect clustering by SCAN.
- Cluster 1 and 2 have dominantly higher proportions, 85.71% and 70.5% respectively, of close habitat. The rest of the clusters indicate an open habitat.
- The highest SPL proportion (87.9%) is shown by cluster 0. SPL is dominantly high in cluster 1 and low in clusters 3 and 4. Cluster 2 indicate mix proportion (51.34%).
- The noise level in clusters 2 and 4 have a high proportion value of 60.15% and 57.14%. The rest of the clusters indicate a noise proportion of less than 20%.
- The distribution of the percentage of samples in each cluster is skewed towards cluster 1.

Table 11: Numerical Distribution of Bats calls' Characteristics for SCAN

Characteristic /Cluster ID	Multiple Bats Presence	Prey Capture Attempt	Flying in Close habitat	High SPL	Noise	Percentage
Cluster 0	7.64%	1.27%	20.38%	87.90%	19.11%	1.08%
Cluster 1	27.51%	1.32%	85.71%	66.93%	10.05%	94.33%
Cluster 2	8.81%	1.15%	70.50%	51.34%	60.15%	3.30%
Cluster 3	25.45%	0.00%	14.55%	10.91%	19.39%	1.18%
Cluster 4	0.00%	0.00%	14.29%	7.14%	57.14%	0.12%

**8.3.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, revealed the defining characteristics of those clusters.

- Between clusters 1 and 2, the defining factor is noise with the proportion of 10.05% and 60.15% respectively.
- Between clusters 0 and 3, the primary defining factor is SPL with the proportion of 87.90% and 10.19% respectively. The secondary factor is the presence of multiple bats.

- Between clusters 1 and 3, the primary defining factor is a mix of habitat type and SPL. The proportion of close habitat is 85.71% and 14.55% while the proportion of SPL is 66.93% and 10.91% for clusters 1 and 3 respectively.
- Cluster 4 does not indicate any characteristics.

Based on the above two analyses and feedback from the expert (provided in Table 12), it could be concluded that cluster 0 captures bat calls, from bats flying in open habitat, close to microphone/ground. Cluster 1 captures the presence of multiple bats that are flying in a close habitat. Cluster 2 captures normal foraging in a close habitat with background noise. Cluster 3 captures the presence of multiple bats that are flying far away from the microphone/ground. Cluster 4 captures bat calls that are flying in open habitat, away from the ground and with some noise in the background.

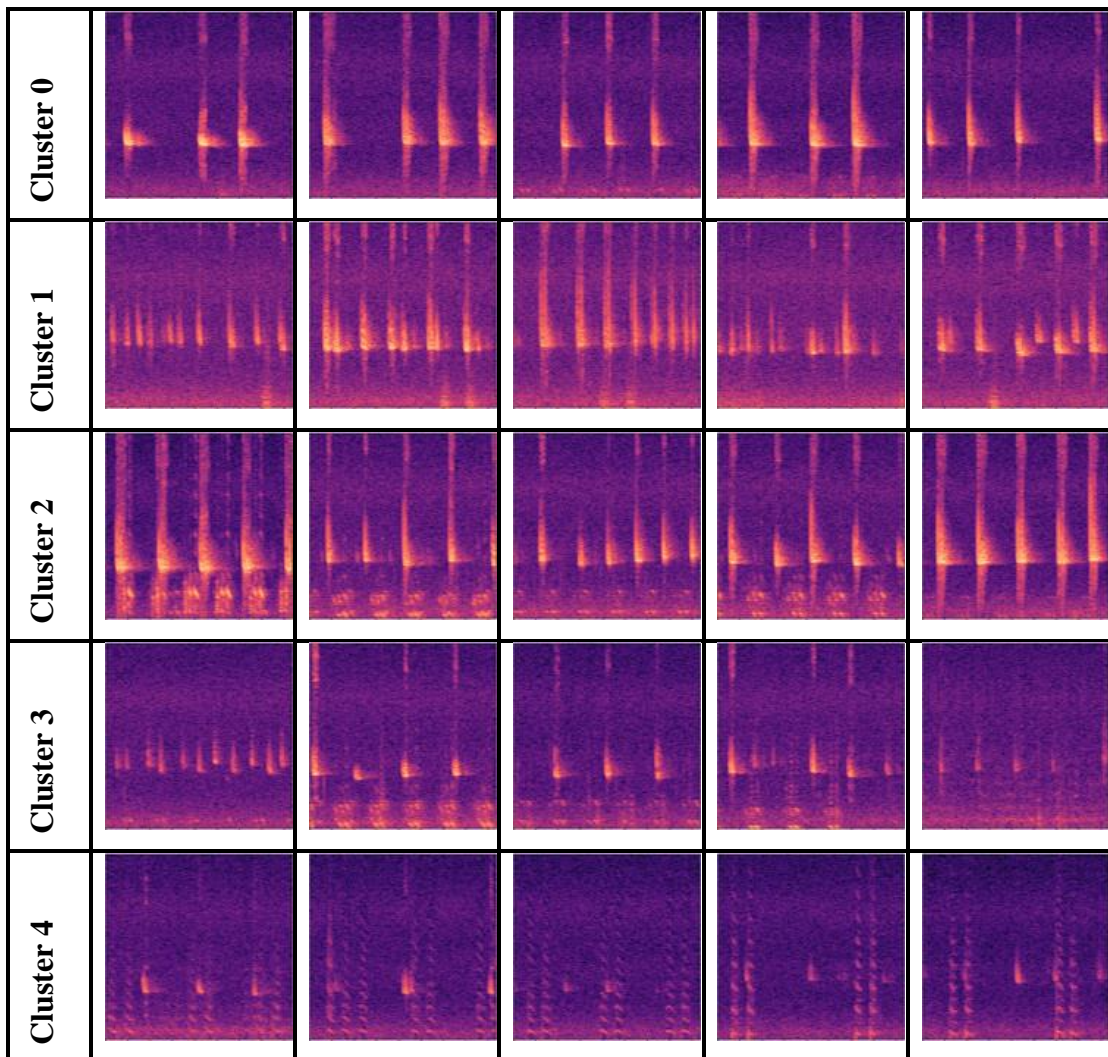


Figure 20: Representative Samples from the Cluster Assignment of SCAN

The comments from a terrestrial biodiversity expert on the interpretation of clusters formed through SCAN are given in Table 12.

Table 12: Comments on the Interpretation of SCAN Results by a Field Expert

<b>Cluster ID</b>	<b>Comments</b>
Cluster 0	Normal foraging at close distance (high SPL)
Cluster 1	Normal foraging by 2 individuals
Cluster 2	Normal foraging at close distance
Cluster 3	Normal foraging of 2 individuals at some distance
Cluster 4	Foraging in open environment at some distance

## Chapter 9. Joint Unsupervised Learning (JULE)

### 9.1 Theoretical Explanation of JULE

Another single-stage clustering algorithm called Joint Unsupervised Learning (JULE) [4] was proposed. Intuitively, JULE used a bottom-up clustering approach in contrast to the most mentioned algorithms, which used the bottom-up partitioning/clustering approach. Starting from every image having its own cluster, they were merged iteratively, and the output at each stage was used to train the CNN. In technical terms, this was based on agglomerative clustering, and a recurrent framework was used to implement the consecutive tasks in the clustering phase. First, embeddings (generated from images) were used to cluster, and then these clusters served to train CNN, making embeddings better over time (because clusters were merged at each step).

Figure 5 illustrates the proposed recurrent framework. Formally, during the time  $t$ , the CNN gave representation  $X^t$  when the image  $I$  was fed into it. Then this representation was used along with the output (of clustering)  $h^{t-1}$  from the previous timestamp to predict the labels for current timestamp  $t$  ( $Y^t$ ). The CNN parameters were represented by  $\theta^t$ . An optimization technique termed as ‘partial unrolling’ was also used, which meant the training of CNN occurs after a defined number of timestamps ( $T$ ). The loss function, given in Equation (26) is unified for both clustering and training the CNN.

$$\mathcal{L}(\{\mathbf{y}^1, \dots, \mathbf{y}^T\}, \{\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^T\} | I) = \sum_{t=1}^T \mathcal{L}^t(\mathbf{y}^t, \boldsymbol{\theta}^t | \mathbf{y}^{t-1}, I) \quad (26)$$

Due to partial unrolling, the loss was added over the timestamps  $T$ , and the loss for the individual timestamp is given in Equation (27).

$$\begin{aligned} \mathcal{L}^t(\mathbf{y}^t, \boldsymbol{\theta}^t | \mathbf{y}^{t-1}, I) = & -A\left(\mathcal{C}_i^t, \mathcal{N}_{\mathcal{C}_i^t}^{K_c}[1]\right) \\ & - \frac{\lambda}{(K_c - 1)} \sum_{k=2}^{K_c} \left( \mathcal{A}\left(\mathcal{C}_i^t, \mathcal{N}_{\mathcal{C}_i^t}^{K_c}[1]\right) - \mathcal{A}\left(\mathcal{C}_i^t, \mathcal{N}_{\mathcal{C}_i^t}^{K_c}[k]\right) \right) \end{aligned} \quad (27)$$

where  $\lambda$  is a constant,  $\mathcal{N}_{\mathcal{C}_i^t}^{K_c}$  means  $K_c$  number of nearest neighbors of the cluster  $\mathcal{C}_i$ .  $\mathcal{A}$  is the affinity/similarity metric (adapted from [65]). The first term calculates the affinity between the cluster  $\mathcal{C}_i$  and the nearest neighbor (it picked the first term from

the neighbor's vector because that was sorted in descending order). The second term (which was not part of standard agglomerative clustering) calculated the difference between the affinity of cluster  $C_i$  and the neighbor clusters of  $C_i$ . Intuitively, this term could be thought of considering the local structure around the clusters (where affinity is maximum).

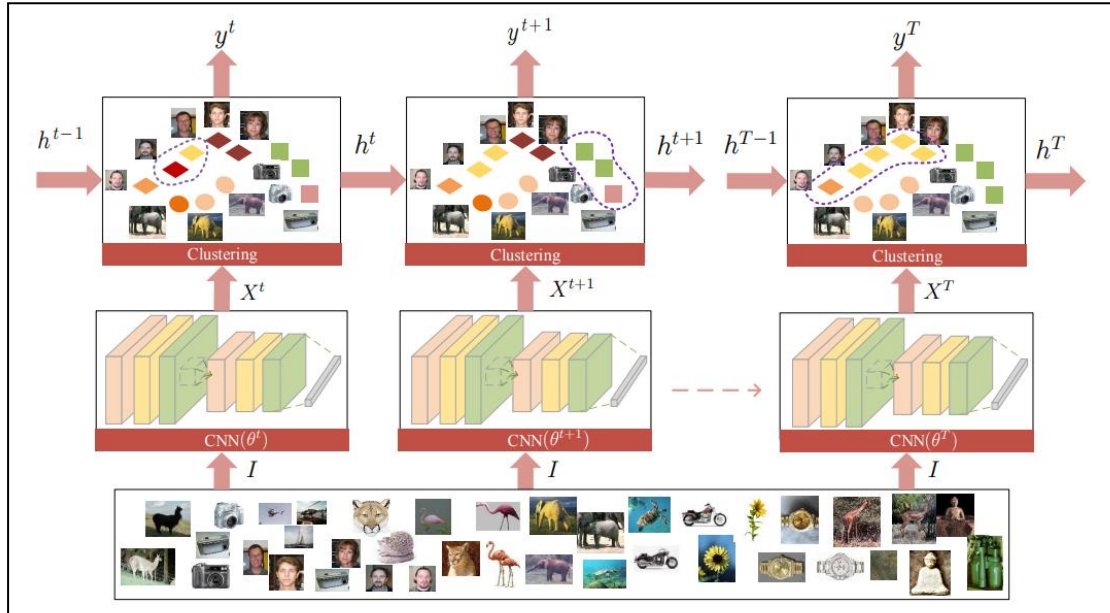


Figure 21: Framework for JULE [4]

## 9.2 Training on Our Dataset

The learning rate was set to 0.01 with a batch size of 50. Value of  $\lambda$  and  $K_c$  in Eq. (27) was set to the default value of 1 and 5 respectively. The target number of clusters was set to 5. The samples were downsized to 28x28 to manage the tradeoff between the computational requirement of JULE and its learning cycles. The updated TensorFlow implementation with selected hyperparameters could be found here [66]. The investigation of the clustered bat calls is provided in detail in the next section.

## 9.3 Results

Based on power calculation, as given in Table 3, few spectrograms were shortlisted and then analyzed according to the procedure given in section 3.6. This outputs, as given in Table 13, the percentage of spectrograms satisfying the bat behavior characteristic in different clusters. Figure 22 shows 5 representative samples from each cluster of SCAN.

Table 13: Numerical Distribution of Bats calls' Characteristics for JULE

Characteristic /Cluster ID	Multiple Bats Presence	Prey Capture Attempt	Flying in Close habitat	High SPL	Noise	Percentage
Cluster 0	28.31%	1.06%	44.44%	56.88%	10.32%	99.87%
Cluster 1	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%
Cluster 2	14.29%	0.00%	14.29%	0.00%	100.00%	0.06%
Cluster 3	20.00%	0.00%	0.00%	100.00%	20.00%	0.02%
Cluster 4	0.00%	0.0%	10.00%	0.00%	100.00%	0.04%

**9.3.1. Vertical/characteristic analysis.** The vertical analysis across five characteristics reveals the following observations:

- Presence of multiple bats is dominantly distributed across the cluster 0, 2 and 3 with a proportion of 28.31%, 14.29%, and 20.00% respectively. Apart from DeepCluster, this factor showed more significance in all other algorithms.
- It does not catch up on the prey capture attempt behavior. A very low proportion (1.06%) is picked by cluster 0 only.
- The highest proportion (44.44%) of the close habitat is present in cluster 0. The rest of the clusters dominantly indicate an open habitat.
- Dominant proportion (100%) of high SPL is present in cluster 3. On the other hand, clusters 1,2 and 4 show no instances of high SPL.
- All the instances in clusters 2 and 4 have background noise. Cluster 0 on the other hand has no such instance.
- The distribution of the percentage of samples in each cluster is highly skewed towards cluster 0.

**9.3.2. Horizontal/cluster analysis.** Analyzing each cluster individually and comparing it with other clusters, in terms of all the characteristics, revealed the defining characteristics of those clusters. These characteristics include:

- The primary attribute of cluster 4 is noise with a proportion of 100%. Cluster 1 is the closest cluster to cluster 4, in terms of other factors, but it has a noise proportion of 0%.

- The primary factor for cluster 0 is the mix of SPL and habitat type. In other factors, this cluster is closest to cluster 3.
- In cluster 2, the defining characteristic is the presence of multiple bats with a proportion of 20%. Cluster 2 is closest to cluster 4, in terms of other factors, but it does not show any presence of multiple bats.

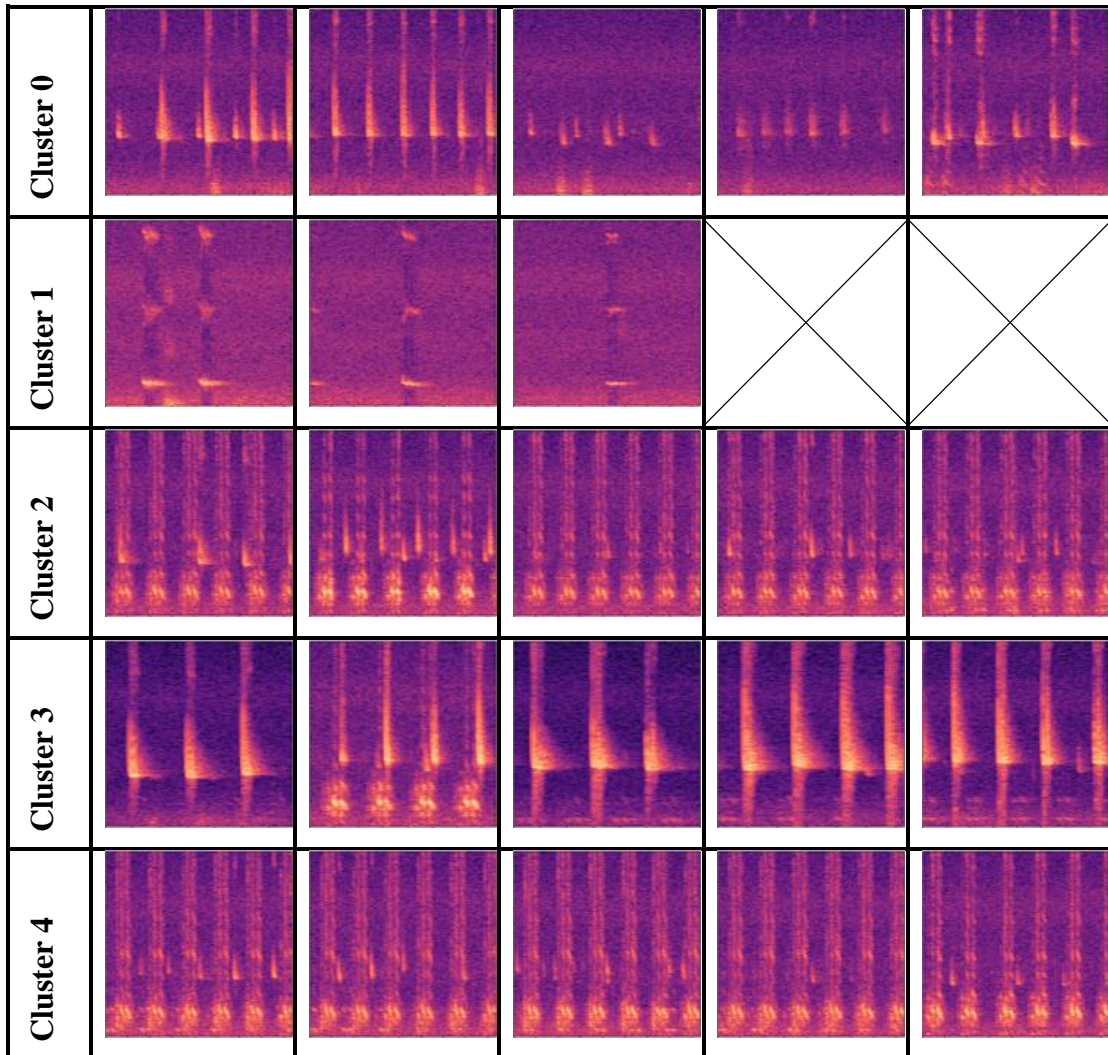


Figure 22: Representative Samples from the Cluster Assignment of JULE

Based on the above two analyses and feedback from the expert (provided in Table 12), it could be concluded that cluster 0 captures the mix of behaviors. It also includes few instances of bat calls where multiple bats are present. Cluster 1 indicates a bat flying in open habitat, but as per the expert, it could be a different species of bat. Cluster 2 indicates normal foraging, with background noise. Cluster 3 indicates bat calls that are flying close to the microphone/ground, in an open environment with the seldom

presence of multiple bats. Cluster 4, like cluster 2, indicates background noise and all the bat calls exhibit normal foraging. The comments from a terrestrial biodiversity expert on the interpretation of clusters formed through JULE are given in Table 14.

Table 14: Comments on the Interpretation of JULE Results by a Field Expert

<b>Cluster ID</b>	<b>Comments</b>
Cluster 0	Normal foraging of 2 individuals
Cluster 1	Presumably a different species than most others in other spectrograms
Cluster 2	Mainly background noise (possibly an insect) with 1 bat passing at some distance
Cluster 3	Normal foraging at close distance
Cluster 4	Looks more like background noise

## Chapter 10. Inter-Algorithm Quantitative Analysis

External validations for each algorithm are described in their respective Chapters. To compare the similarity in cluster assignment across different algorithms, quantitative metrics of Mutual Information (MI), Normalized Mutual Information (NMI), Rand Score (RS), and Adjusted Rand Score (ARS) were calculated. It is important to note that the relative values between metrics are more important than the absolute value of a single metric. The intuition behind this is that the results should be consistent across the metrics if they are measuring the same property. In short, similar metrics should agree in preference (not necessarily in numbers). This Chapter features multiple metrics, but it could be desirable to use a particular metric depending upon the situation. For example, ARI is preferred when the clusters are equal in size, but the number of clusters is large while AMI could be preferred when the resulting clustering is asymmetric, and the number of clusters is small [67].

Based on MI, in Table 15, DeepCluster and IIC show the highest similarity. JULE and SCAN show the lowest similarity. This observation is also validated by the fact that JULE and SCAN show the asymmetric assignment of clusters in comparison to other algorithms. In SCAN and JULE, the proportion of spectrograms that fall in the largest cluster are 94.33% and 99.87% respectively. While the algorithms gave valid semantic clustering, it was based on the different weightage of characteristics - as described in sections 8.3 and 9.3. Thus, due to the skewed distribution of clusters, these metrics show negligible similarity between these algorithms. NMI is given in Table 16.

Analyzing the cluster/horizontal analysis from each algorithm could further validate the resulting measures of MI. For IMSAT, mainly, the distribution of clusters could be attributed to noise, the presence of multiple bats, and a mixture of habitat type and SPL (as described in section 5.3). For DeepCluster, it could be attributed to SPL, prey capture attempts, and a mixture of noise and habitat type (as described in section 7.3). And for IIC, it could be attributed to the presence of multiple bats, SPL, and a mixture of noise and SPL (as described in section 6.3). The resulting common characteristics between each of these are given in Table 17. *Characteristic*<sup>1</sup> indicate that it occurs in the mix of attribute activity of one cluster. *Characteristic*<sup>2</sup> indicate that it occurs in the mix of attribute activity of both clusters.

Table 15: Mutual Information Score Between All Algorithms

Algo-1 \ Algo-2	IIC	IMSAT	DeepCluster	JULE	SCAN	K-Medoid
IIC	-	<b>0.429</b>	<b>0.374</b>	0.003	0.028	0.000
IMSAT	0.429	-	<b>0.266</b>	0.002	0.017	0.000
DEEPCUSTER	0.374	0.266	-	0.001	0.019	0.001
JULE	0.003	0.002	0.001	-	0.000	0.000
SCAN	<b>0.028</b>	<b>0.017</b>	0.019	0.000	-	0.000
K-Medoid	0.000	0.000	0.001	0.000	0.000	-

Table 16: Normalized Mutual Information Score Between All Algorithms

Algo-1 \ Algo-2	IIC	IMSAT	DeepCluster	JULE	SCAN	K-Medoid
IIC	-	<b>0.279</b>	<b>0.301</b>	0.004	0.032	0.000
IMSAT	0.279	-	<b>0.205</b>	0.002	0.018	0.000
DEEPCUSTER	0.301	0.205	-	0.001	0.030	0.000
JULE	0.004	0.002	0.001	-	0.001	0.000
SCAN	<b>0.032</b>	<b>0.018</b>	0.030	0.001	-	0.000
K-Medoid	0.000	0.000	0.000	0.000	0.000	-

Four attributes are common between IMSAT and IIC, hence the highest MI score of 0.429. This score is followed by the MI between IIC and DeepCluster (0.374), which has three attributes in common. IMSAT and DeepCluster also have three attributes in common, but all three of them are secondary (i.e., *Characteristic<sup>x</sup>*), resulting in MI score of 0.266. The difference in MI between IMSAT with IIC and IIC with DeepCluster is minor though. The fact that 68.50% of samples occur in a single cluster of DeepCluster, makes the MI metric a bit skewed. Hence to adjust for that factor, Normalized Mutual Information (NMI) was also calculated - given in Table 16. The order of values between MI and NMI remains almost the same, apart from a single order switch (with a minor difference of just 0.02).

Table 17: Common characteristics for Defining Cluster Between Two Algorithms.

<b>Algo-1 \ Algo-2</b>	<b>IIC</b>	<b>IMSAT</b>	<b>DEEPCUSTER</b>
IIC	-	<i>Multiple Bats , SPL<sup>1</sup>, SPL<sup>2</sup>, Noise<sup>1</sup></i>	<i>SPL, SPL<sup>1</sup>, Noise<sup>2</sup></i>
IMSAT	<i>Multiple Bats , SPL<sup>1</sup>, SPL<sup>2</sup>, Noise<sup>1</sup></i>	-	<i>Noise<sup>1</sup>, Habitat<sup>2</sup>, SPL<sup>1</sup></i>
DEEPCUSTER	<i>SPL, SPL<sup>1</sup>, Noise<sup>2</sup></i>	<i>Noise<sup>1</sup>, Habitat<sup>2</sup>, SPL<sup>1</sup></i>	-

Rand score between different clustering is provided in Table 18. This metric seems to be affected, the most, by the asymmetric distribution of clusters. For example, it is highest between SCAN and JULE where the number of samples that reside in the largest cluster is 94.33% and 99.87% respectively. Hence, it is based on chance. To adjust these anomalies, ARI was also calculated – given in Table 19. This also results in the same preference of similarity between different algorithms as MI – given in Table 15. In summary, different metrics indicate consistency between the comparison of multiple algorithms. Hence it could be concluded that the semantic features of bat calls are captured by the unsupervised deep learning approaches. However, each algorithm gives different weightage to different features of bat calls to cluster them. This results in semantically meaningful clusters from each algorithm.

Table 18: Rand Score Between Different Algorithms

<b>Algo-1 \ Algo-2</b>	<b>IIC</b>	<b>IMSAT</b>	<b>DeepCluster</b>	<b>JULE</b>	<b>SCAN</b>	<b>K-Medoid</b>
IIC	-	0.736	0.565	0.249	0.310	0.638
IMSAT	0.736	-	0.550	0.209	0.274	0.660
DEEPCUSTER	0.565	0.550	-	0.501	0.466	0.500
JULE	0.249	0.209	0.501	-	0.889	0.229
SCAN	0.310	0.274	0.466	0.889	-	0.287
K-Medoid	0.638	0.660	0.500	0.229	0.287	-

Table 19: Adjusted Rand Score Between Different Algorithms

<b>Algo-1 \ Algo-2</b>	<b>IIC</b>	<b>IMSAT</b>	<b>DeepCluster</b>	<b>JULE</b>	<b>SCAN</b>	<b>K-Medoid</b>
IIC	-	0.250	0.131	0.001	0.012	0.000
IMSAT	0.250	-	0.101	0.000	0.004	0.000
DEEPCUSTER	0.131	0.101	-	-0.002	-0.071	0.001
JULE	0.001	0.000	-0.002	-	-0.003	0.000
SCAN	0.012	0.004	-0.071	-0.003	-	0.001
K-Medoid	0.000	0.000	0.001	0.000	0.001	-

These results are expected to generalize well over an even larger population of bats because the emitted bat calls do not show a lot of variation. For example, the state-of-the-art supervised deep learning model, batdetect [22], used only 4,782 labeled echolocation calls to achieve the precision of 0.89 for detection of bat call [68] – as explained in more detail in section 3.3. On the other hand, our dataset consisted of 24,610 training instances. So, it is safe to consider that it will generalize well over a large population of bats.

The metrics, given above, provide a global estimate of the similarity between two clusterings. To look deeper and evaluate the local relationship between two labeling outputs, it is helpful to look at the cross-tabulation among each combination. For 6 algorithms, there are 15 combinations ( ${}^6C_2$ ). The resulting tabulations are provided in Table 20 to Table 34. These calculations could help us in further evaluation, interpretation, and improvement of results. For example, in terms of interpretation, it could be observed that in Table 20, cluster 0 of IMSAT corresponds closely to cluster 1 of IIC. Both show the presence of only single bats, no prey capture attempt, open habitat, low SPL and presence of noise.

It could also be helpful to utilize a combination of algorithms because each algorithm picks up different characteristics. For example, SCAN does not pick up on the prey capture attempt characteristic while DeepCluster does. Now, if we want to extract prey capture attempts from cluster 0 of SCAN (having open habitat and high SPL without noise) then we can take intersection with cluster 2 of DeepCluster.

Similarly, the capabilities of different algorithms could be combined. All the possible combinations are given to assist with combining the capability of multiple algorithms.

Table 20: Cross Tabulation between IMSAT and IIC

<b>IIC \ IMSAT</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>Cluster 0</b>	2.57%	8.01%	3.68%	2.11%	4.33%
<b>Cluster 1</b>	19.13%	0.74%	0.02%	1.15%	3.09%
<b>Cluster 2</b>	0.24%	1.13%	10.62%	0.46%	0.78%
<b>Cluster 3</b>	1.65%	1.75%	7.19%	2.01%	10.22%
<b>Cluster 4</b>	5.37%	0.51%	0.69%	0.14%	12.43%

Table 21: Cross Tabulation between IIC and DeepCluster

<b>IIC \ DeepCluster</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>Cluster 0</b>	0.14%	0.00%	28.81%	0.00%	0.00%
<b>Cluster 1</b>	0.03%	0.30%	11.80%	0.00%	0.00%
<b>Cluster 2</b>	1.64%	9.50%	4.16%	2.45%	4.45%
<b>Cluster 3</b>	0.00%	0.00%	5.86%	0.00%	0.00%
<b>Cluster 4</b>	9.83%	3.06%	17.87%	0.00%	0.09%

Table 22: Cross Tabulation between IMSAT and DeepCluster

<b>IMSAT \ DeepCluster</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>Cluster 0</b>	1.37%	1.42%	17.58%	0.10%	0.23%
<b>Cluster 1</b>	0.63%	0.00%	23.49%	0.00%	0.00%
<b>Cluster 2</b>	0.08%	3.93%	3.44%	2.30%	3.46%
<b>Cluster 3</b>	3.64%	6.84%	11.46%	0.04%	0.84%
<b>Cluster 4</b>	5.93%	0.67%	12.52%	0.00%	0.01%

Table 23: Cross Tabulation between JULE and IIC

<b>IIC</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>JULE</b>					
Cluster 0	28.93%	12.13%	22.20%	5.77%	30.84%
Cluster 1	0.00%	0.01%	0.00%	0.00%	0.00%
Cluster 2	0.00%	0.00%	0.00%	0.06%	0.00%
Cluster 3	0.02%	0.00%	0.00%	0.00%	0.00%
Cluster 4	0.00%	0.00%	0.00%	0.04%	0.00%

Table 24: Cross Tabulation between JULE and IMSAT

<b>IMSAT</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>JULE</b>					
Cluster 0	20.59%	24.10%	13.21%	22.82%	19.14%
Cluster 1	0.01%	0.00%	0.00%	0.00%	0.00%
Cluster 2	0.06%	0.00%	0.00%	0.00%	0.00%
Cluster 3	0.00%	0.02%	0.00%	0.00%	0.00%
Cluster 4	0.04%	0.00%	0.00%	0.00%	0.00%

Table 25: Cross Tabulation between JULE and DeepCluster

<b>DeepCluster</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
<b>JULE</b>					
Cluster 0	11.65%	12.88%	68.36%	2.45%	4.53%
Cluster 1	0.00%	0.00%	0.01%	0.00%	0.00%
Cluster 2	0.00%	0.00%	0.06%	0.00%	0.00%
Cluster 3	0.00%	0.00%	0.02%	0.00%	0.00%
Cluster 4	0.00%	0.00%	0.04%	0.00%	0.00%

Table 26: Cross Tabulation between SCAN and IIC

<b>IIC</b> <b>SCAN</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	0.78%	0.18%	0.01%	0.08%	0.03%
Cluster 1	26.08%	10.67%	22.00%	5.12%	30.46%
Cluster 2	1.76%	0.81%	0.13%	0.36%	0.23%
Cluster 3	0.29%	0.44%	0.06%	0.27%	0.12%
Cluster 4	0.03%	0.04%	0.01%	0.04%	0.00%

Table 27: Cross Tabulation between SCAN and IMSAT

<b>IMSAT</b> <b>SCAN</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	0.02%	0.38%	0.22%	0.28%	0.18%
Cluster 1	20.39%	22.58%	11.49%	21.74%	18.13%
Cluster 2	0.26%	1.08%	0.60%	0.66%	0.70%
Cluster 3	0.03%	0.08%	0.78%	0.15%	0.13%
Cluster 4	0.00%	0.00%	0.12%	0.00%	0.00%

Table 28: Cross Tabulation between SCAN and DeepCluster

<b>DeepCluster</b> <b>SCAN</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	0.00%	0.00%	1.08%	0.00%	0.00%
Cluster 1	11.64%	12.84%	62.91%	2.42%	4.52%
Cluster 2	0.00%	0.04%	3.21%	0.03%	0.01%
Cluster 3	0.00%	0.00%	1.17%	0.00%	0.00%
Cluster 4	0.00%	0.00%	0.12%	0.00%	0.00%

Table 29: Cross Tabulation between SCAN and JULE

<b>JULE</b> <b>SCAN</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	1.08%	0.00%	0.00%	0.00%	0.00%
Cluster 1	94.19%	0.01%	0.06%	0.02%	0.04%
Cluster 2	3.30%	0.00%	0.00%	0.00%	0.00%
Cluster 3	1.18%	0.00%	0.00%	0.00%	0.00%
Cluster 4	0.12%	0.00%	0.00%	0.00%	0.00%

Table 30: Cross Tabulation between SCAN and K-Medoid

<b>K-Medoid</b> <b>SCAN</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	0.17%	0.13%	0.31%	0.17%	0.29%
Cluster 1	15.16%	11.44%	28.39%	13.12%	26.22%
Cluster 2	0.54%	0.43%	0.92%	0.52%	0.89%
Cluster 3	0.20%	0.17%	0.32%	0.17%	0.33%
Cluster 4	0.01%	0.01%	0.04%	0.01%	0.05%

Table 31: Cross Tabulation between K-Medoid and IIC

<b>IIC</b> <b>k-Medoid</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	4.70%	1.86%	3.70%	0.89%	4.92%
Cluster 1	3.60%	1.40%	2.67%	0.81%	3.70%
Cluster 2	8.61%	3.58%	6.64%	1.65%	9.50%
Cluster 3	3.90%	1.76%	3.12%	0.82%	4.38%
Cluster 4	8.13%	3.54%	6.07%	1.69%	8.34%

Table 32: Cross Tabulation between K-Medoid and IMSAT

<b>IMSAT</b> <b>K-Medoid</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	3.21%	3.89%	2.10%	3.64%	3.22%
Cluster 1	2.58%	2.90%	1.56%	2.88%	2.26%
Cluster 2	6.17%	7.16%	3.89%	6.92%	5.84%
Cluster 3	2.93%	3.31%	1.98%	3.16%	2.61%
Cluster 4	5.82%	6.87%	3.67%	6.22%	5.21%

Table 33: Cross Tabulation between K-Medoid and DeepCluster

<b>DeepCluster</b> <b>K-Medoid</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	1.92%	2.04%	10.84%	0.43%	0.84%
Cluster 1	1.41%	1.59%	8.39%	0.26%	0.53%
Cluster 2	3.72%	3.81%	20.44%	0.74%	1.27%
Cluster 3	1.68%	1.75%	9.47%	0.37%	0.73%
Cluster 4	2.92%	3.69%	19.36%	0.65%	1.17%

Table 34: Cross Tabulation between K-Medoid and JULE

<b>JULE</b> <b>K-Medoid</b>	<b>Cluster 0</b>	<b>Cluster 1</b>	<b>Cluster 2</b>	<b>Cluster 3</b>	<b>Cluster 4</b>
Cluster 0	16.04%	0.00%	0.01%	0.00%	0.01%
Cluster 1	12.15%	0.00%	0.01%	0.00%	0.01%
Cluster 2	29.94%	0.00%	0.02%	0.00%	0.01%
Cluster 3	13.98%	0.00%	0.00%	0.00%	0.00%
Cluster 4	27.75%	0.00%	0.02%	0.00%	0.01%

## **Chapter 11. Specie Clustering using Unsupervised Deep Learning**

In previous experiments, bat calls were clustered based on the way a bat was interacting with its surroundings/environment. This included various characteristics such as if it was flying in an open/close habitat, attempting to capture its prey, or flying in presence of other bats. It also included other external factors such as its distance to the microphone and the presence of background noise. Overall this was termed as its ‘behavior’. However, another dimension to look at bat calls could be to cluster them based on the species i.e. if they belong to the same group of the organism in taxonomic rank.

As mentioned, previous experiments were performed to investigate the behavior of different clustered groups. For that case, it was important to look at the distance between individual bat calls, the intensity of each bat call, the presence of calls at multiple frequency bands, and background noise. That is why the window length for the generation of each spectrogram was set at 500ms. However, to cluster according to different species types, spectrograms needed to be generated in a way that individual bat call signature is visible. For that purpose, a second dataset was used, where each audio file was already labeled by the species name – by an expert. The procedure for that is explained in section 11.1.

This application has been demonstrated to work well with the labeled data [16] and an extension of this is explored which does not use any labels for training. Labels are only used for testing while calculating the accuracy and other evaluation metrics. It is important because more than 1000 species of bats are present, and their types vary significantly depending on the region. This makes it difficult to annotate the datasets and automation is required. The process below illustrates how it could be used to cluster the bat calls into different species groups without getting dependent on any labeled data or manual work.

### **11.1 Data Description and Pre-Processing:**

The details about the data collection are given in the prior Chapter, Methodology. Few of those collected recordings were manually labeled by the experts during the collection phase. Considering skewness in the number of recorded bats calls for different species, five top species, according to the total number of labeled cells,

were selected. The distribution of calls according to species is provided in Table 35. The recordings had a sample rate of 256 kHz.

Table 35: Count of Bat calls from different Specie

<b>Specie Name</b>	Eptesicus Bottae (EPTBOT)	Pipistrellus Kuhli (PIPKUH)	Rhinopoma Muscatellum (RHIMUS)	Rhynptesicus Nasutus (RHYNAS)	Taphozous Perforatus (TAPPER)
<b>Count</b>	128	352	928	275	225

To get these bat calls and generate spectrograms, a sample pipeline, as described in the methodology Chapter was used. The detailed procedure to process a single audio file and extract audio chunks (containing bat calls) is given in Figure 1 and the processing pipeline is illustrated in Figure 6. Configurations such as augmentation techniques, spectrogram cropping, and y-axis frequency clipping were kept the same as before. Since the focus was to get a detail look at the shape of signals (as given in Figure 23) to cluster them, different parameters were chosen for generating spectrograms - in contrast to Chapter 3.

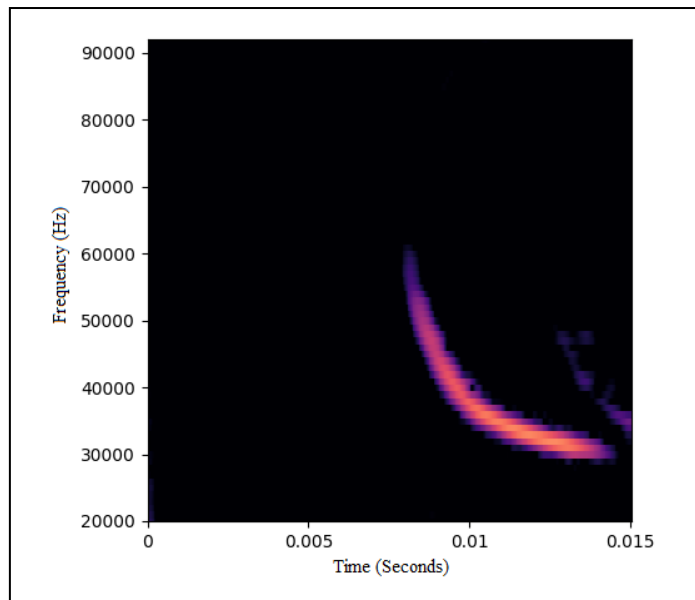


Figure 23: Example of a Spectrogram Generated from Labeled Dataset.

The summary of parameters used for both labeled and unlabeled datasets is given in Table 36. Important differences are:

- For batdetect, the confidence threshold was set at 80%, as compared to 95% used in section 3.3. It was done to extract a higher number of bat calls from the limited amount of labeled data.
- While generating the spectrograms, the target signal length of 15ms was used.
- The length of the windowed signal after padding with zeros was set to 256 and the hop length was set to 8. This ensured high-quality spectrograms with minimum artifacts.
- Instead of doing the target amplitude matching, value clipping between -45 and 0 was used for the color bar range during the pseudo-color plot for spectrogram generation. This value range is empirical and based on the experimental evaluation. The resulting spectrograms from different species are given in Figure 24.
- The dataset was balanced according to the number of lowest bats' calls in any species category i.e., Eptesicus Bottae.

Table 36: Comparison Between Parameters for Spectrogram Generation

Parameter /Dataset	Batdetect Threshold	Audio Signal Length	STFT window length	Hop Length	Sample Rate	Normalization Technique
Unlabeled	95%	500ms	5120	512	256 KHz	Loudness Level
Labeled	80%	15ms	256	8	256 KHz	Value Clipping

## 11.2 Training Setup:

Same configurations from unsupervised clustering were utilized here. The only difference was the pre-processing stage – for reasons described in the first paragraph of this Chapter. This highlights the generic nature of these algorithms that can be applied to any formation of the spectrogram. The target number of clusters was set to 5. Each algorithm was trained for a different number of epochs and sometimes early stopping was applied. Such details and learning curves are given in Figure 25. The loss of all the models decreased over time until they converged. This indicates that the models learned according to the defined objective function and clustered the bat calls into different species groups. The evaluation of these clusters is provided in detail in the next section.

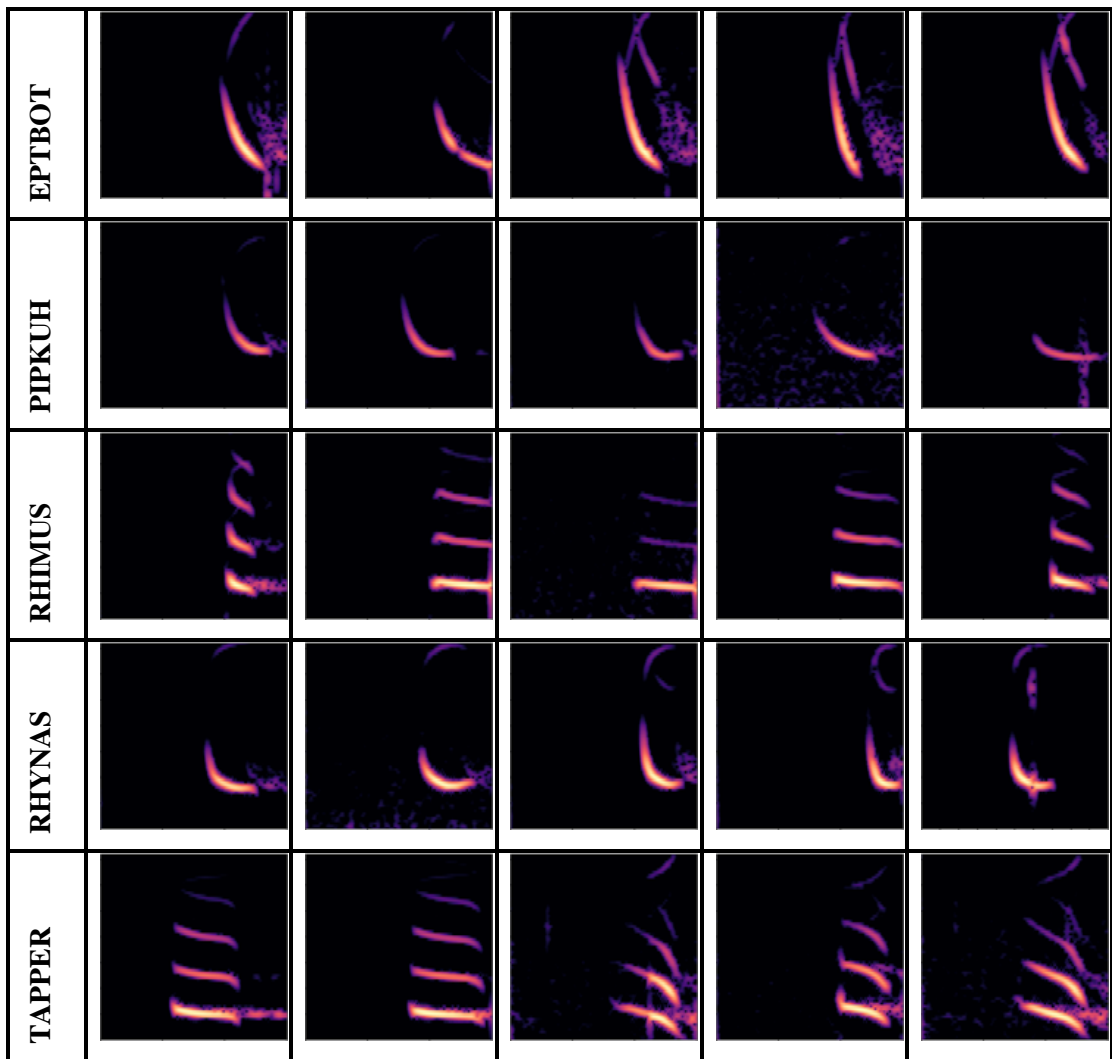


Figure 24: Call Signatures of Different Bat Species from the Dataset

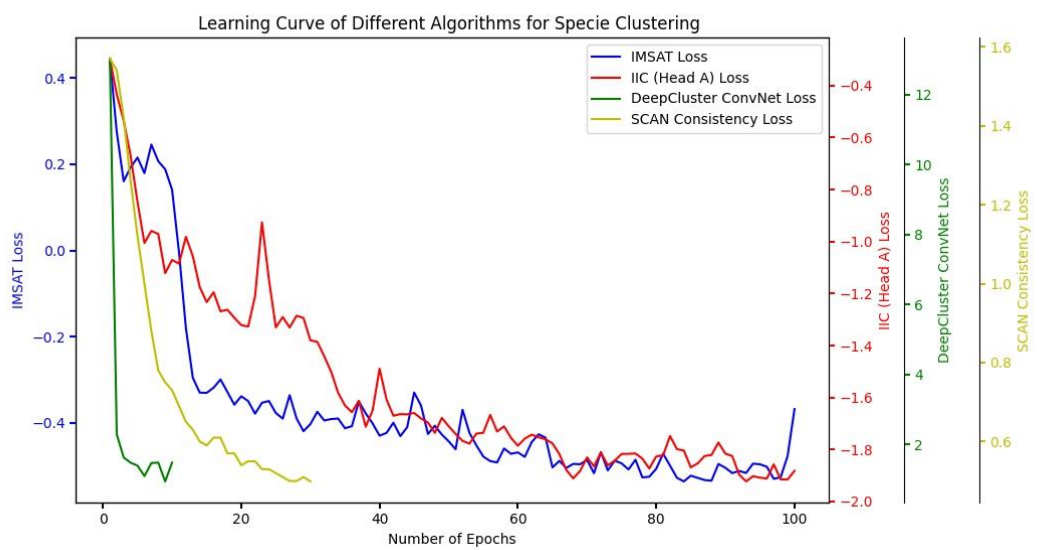


Figure 25: Learning Curves for Unsupervised Clustering of Bats' Calls

### 11.3 Results:

The comparison of clustered bats' calls against the labeled ones is provided in Table 37. The results consist of various metrics defined in section 3.7. IIC provides the best results with an accuracy of 0.48 and an MI score of 0.54. The results of the rest of the algorithms are arranged in descending order.

Table 37: Results from Unsupervised Clustering of Labelled bats' calls

<b>Metric</b> <b>Technique</b>	<b>Accuracy</b>	<b>Mutual Information Score</b>	<b>Rand Score</b>	<b>Fowlkes Mallows Score</b>
IIC	48.28%	54.46%	75.92%	40.82%
IMSAT	43.59%	43.11%	70.73%	37.09%
JULE	43.13%	46.51%	68.61%	37.30%
DeepCluster	39.84%	36.29%	72.51%	33.51%
SCAN	29.38%	7.44%	59.00%	29.17%
K-Medoid	23.75%	1.24%	68.12%	19.88%

For perspective, it is important to note that till only a couple of years ago the results from unsupervised clustering of standard labeled datasets were also in a similar accuracy bracket. The detailed evolution of few unsupervised learning techniques used for image classification is provided in Table 38. For instance, for CIFAR-10, while the supervised classification accuracy stood at 93.8% [3], the unsupervised clustering accuracy of only 27.2% was achieved using JULE [3] [4]. It was even lower for conventional clustering techniques. For example, K-means gave an accuracy of only 22.9% [3]. Contrary to initial indications, this field has provided promising results of more than 80% accuracy without using any ground labels for training, during the last one year. We have similar aspirations for the application of unsupervised learning in wildlife monitoring. This work demonstrated reasonable results in this novel area of research and provided a promising direction for future work. The low results from JULE are as per expectation but SCAN is an anomaly in our case. The reasons for its low performance and methods to improve it, are given in Chapter 13. The direction for future work and further explanation on making the SCAN better are also provided in that Chapter.

Table 38: Evolution of Unsupervised Clustering of Labelled Datasets[3]

<b>Dataset</b>				
<b>Technique</b>	<b>Year</b>	<b>CIFAR10</b>	<b>CIFAR100-20</b>	<b>STL10</b>
Triplets [69]	2004	20.50%	9.94%	24.40%
AE [70]	2006	31.40%	16.50%	30.30%
K-means [71]	2015	22.90%	13.00%	19.20%
JULE [4]	2016	27.20%	13.70%	27.70%
GAN [72]	2016	31.50%	15.10%	29.80%
DEC [20]	2016	30.10%	18.50%	35.90%
DeepCluster [5]	2019	37.40%	18.90%	33.40%
IIC [2]	2019	61.70%	25.70%	59.60%
SCAN [3]	2020	88.30%	50.70%	80.90%

## Chapter 12. Other Unsupervised Clustering Algorithms

The theoretical explanation of few other algorithms which utilize different learning paradigms is explained from sections 12.1 to 12.3.

### 12.1 Deep Adaptive Clustering (DAC)

DAC combines a clustering algorithm along with the representation/feature learning from images [19] [73]. Intuitively, the model starts by learning from the pair of images that are like/unlike each other (using a threshold), and after each iteration when the model has trained on such identified instances, it identifies a greater number of such pairs of images. It continues until all the training examples are included. In technical terms, DAC basically reframes the task of clustering into a binary pairwise classification framework [74]. This is done to evaluate if the two images are from the same or different clusters. A deep convolution neural network is first used to generate features from images, and subsequently, clusters are formed by calculating the pairwise cosine distance between the generated features – as illustrated in Figure 26. This is purely unsupervised, and the labels of the image are not used at all during the training phase. This approach is categorized as a single-stage because it does not use any external/non-deep learning algorithm for clustering.

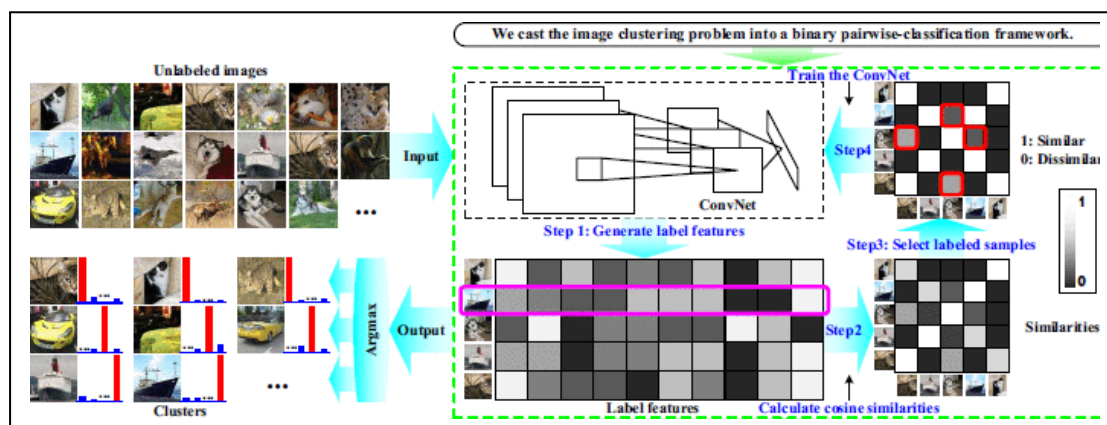


Figure 26: Flowchart of DAC [19]

The steps for DAC are as follows:

1. Pass the images through the CNN to calculate features.
2. Use cosine similarity to find the similarity between each pair.

3. Select a subset of samples based on the cosine similarity.
4. Train the CNN with shortlisted samples (cluster is found by locating the largest response of label features)
5. Repeat the above steps until all the training examples are included.

The objective is given in Equation (28), for the whole DAC model including CNN.

$$\min_{w, \lambda} E(w, \lambda) = \sum_{i,j} v_{ij} L(r_{ij}, l_i \cdot l_j) + u(\lambda) - l(\lambda) \quad (28)$$

where  $w$  is the weights of neural network in the CNN,  $v$  is the indicative term used for selecting the sample for training and  $l_i \cdot l_j$  is the cosine similarity between the output labels of two inputs (with  $i$  and  $j$  indicating row and column number respectively). The loss function  $L$  is the binary cross-entropy between  $r_{ij}$  (defined in the next equation) and the cosine similarity (mentioned before).  $\lambda$  is the parameter of the model, which is ultimately used to control the selection of data points for during each iteration and  $l(\lambda)$  and  $u(\lambda)$  are the threshold values that indicate if the samples are similar or not. Such that:

$$l(\lambda) \leq u(\lambda) \text{ and}$$

$$v_{ij} \in \{0, 1\}, i, j = 1, \dots, n$$

$$\forall i, \|l_i\|_2, \text{ and } l_{ih} \geq 0, h = 1, \dots, k$$

where  $k$  is the size of the label feature, and  $n$  is the number of training examples.  $r_{ij}$  is expanded in Equation (29) and the indicator co-efficient ( $v_{ij}$ ) is provided in Equation (30).

$$r_{ij} = \begin{cases} 1, & \text{if } l_i \cdot l_j \geq u(\lambda) \\ 0, & \text{if } l_i \cdot l_j \leq l(\lambda) \\ \text{None}, & \text{Otherwise} \end{cases}, i, j = 1, \dots, n \quad (29)$$

$$v_{ij} = \begin{cases} 1, & \text{if } r_{ij} \in \{0, 1\} \\ 0, & \text{Otherwise} \end{cases}, i, j = 1, \dots, n \quad (30)$$

Further details and applications on binary pairwise classification are found in [75] [76]. This model achieved an accuracy of 52% on the CIFAR-10 dataset.

## 12.2 Deep Embedded Clustering (DEC)

DEC [20] also uses a deep neural network and learns the clustering and data embedding (i.e., converting data to low-dimension) in parallel. Intuitively, this model can be thought of as an extension of semi-supervised self-training [77] into the domain of unsupervised learning. After learning the lower dimension/embedding of data, this is directly clustered using a standard clustering algorithm, and then a deep learning model is trained iteratively such that it optimizes the loss function. Technically, a clustering objective based on KL-divergence is optimized at each step. DEC has two main steps terms as initialization and optimization/clustering.

During initialization, a stacked autoencoder is trained with the unlabeled data, and the initial weights for the encoder which give minimum reconstruction error are obtained. The embedding is referenced as  $z_i$ . Another initialization is done for the cluster centers (represented by  $\mu_j$ ). This is performed by the k-mean clustering [78] on the embedding obtained after the training (say k clusters). This is shown in Figure 27.

Before diving into the loss function, let us look at the two probability distributions. To quantify the similarity between the embedded datapoint  $z_i$  and centroid  $\mu_j$  soft assignments Equation (31) is used.

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'} \left(1 + \|z_i - \mu_{j'}\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}} \quad (31)$$

where  $z_i$  is  $f_\theta(x_i)$ , i.e., the embedding after passing through the encoder.  $\alpha$  is a constant set to 1.  $\mu_j$  is the cluster centroid such that j ranges from 1 to the number of clusters. The other term  $P$  (auxiliary/target distribution) is obtained using Equation (32).

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}} \quad (32)$$

where  $f_j$  is  $\sum_i q_{ij}$  and this is used for normalizing by the total occurrences per cluster so that the loss weightage contributed by each centroid could be normalized. The loss function is defined as the KL-divergence between the soft assignments  $q_{ij}$  (i.e., probability of assigning the image  $i$  to cluster  $j$ ) and an auxiliary distribution  $p_j$  – as provided in Equation (33).

$$L = \text{KL}(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (33)$$

This loss could be interpreted as learning from the high-confidence predictions and then continues to improve iteratively using the loss function. Also, this is important to note that k-means by itself does hard clustering, but in this case, only the cluster centroids are used in the equation for soft-assignment. The accuracy of 84.3% is obtained on the MNIST dataset. This algorithm is also easily scalable because of its linear computational complexity. Over time various variations of the DEC algorithm have evolved. For example, using data augmentation along with DEC [79].

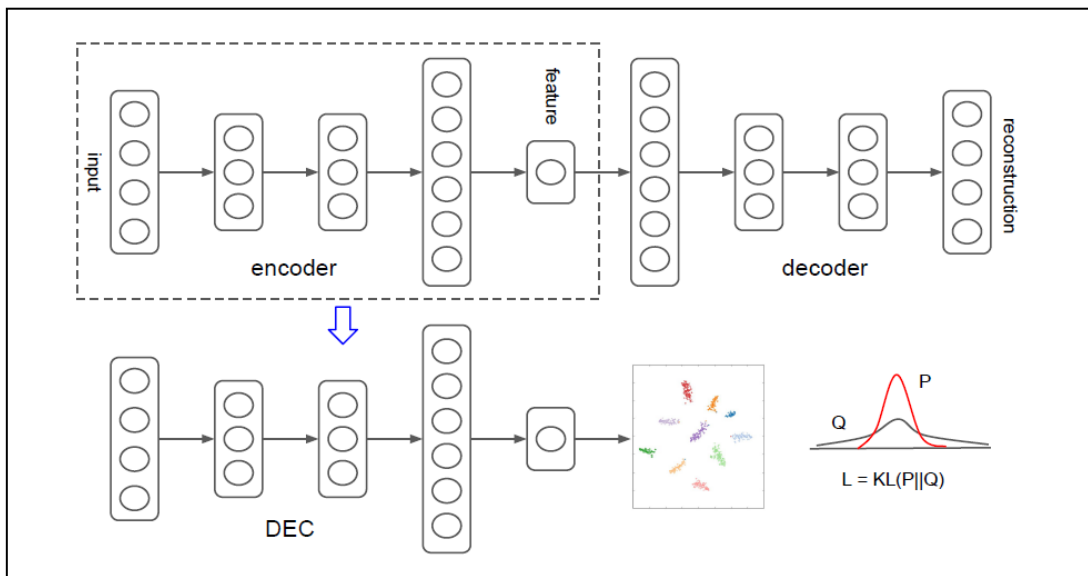


Figure 27: Network Structure of DEC[20]

### 12.3 K-Autoencoders

K-Autoencoders [21] uses multiple auto-encoders equal to the number of clusters required and assigns each sample to the autoencoder that gives the minimum reconstruction error (also called the mixture of local experts [80]). The intuition behind this is that, instead of generating the embeddings from a single auto-encoder[81], multiple autoencoders could be used, with each one of them specializing in rebuilding the images from one cluster – as illustrated in Figure 28. Technically, the overall mean-squared error loss is minimized by selecting/learning the set of autoencoders. A similar concept has also been used in [82] [83]. Auto-Encoding Variational Bayes (with an accuracy of 29.1%) [58] and Greedy Layer-wise training of deep neural networks

(with an accuracy of 31.4%) [84] also fall under this category. It gave an accuracy of 88% on the MNIST dataset.

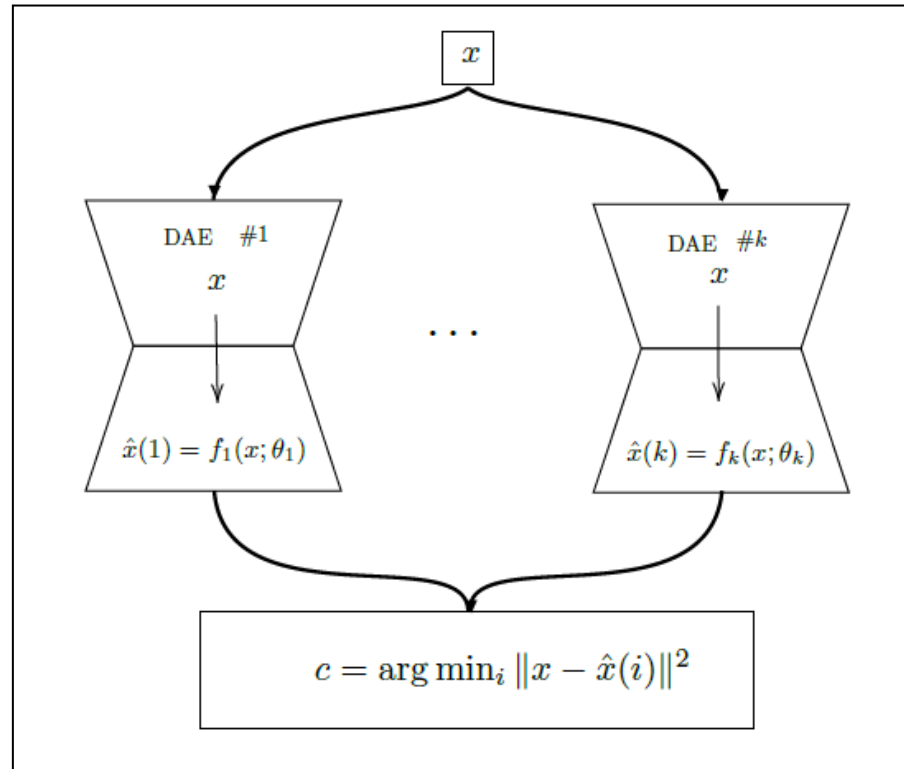


Figure 28: Block Diagram of K-Autoencoders

The steps involved are as follows:

1. Train one auto-encoder on the complete data.
2. Apply k-mean clustering to the embedding and get the assigned labels.
3. Now, using the samples that were assigned the  $i^{th}$  label, pre-train the corresponding autoencoder.
4. Now use the objective function to train the autoencoder - provided in Equation(34).

$$L(\theta_1, \dots, \theta_k) = \sum_{t=1}^n \min_{i=1}^k d(x_t, \hat{x}_t(i)) \quad (34)$$

There are  $k$  auto-encoders (equivalent to the number of clusters), and the above objective is minimizing the reconstruction error over all the training examples ( $n$ ).  $\hat{x}_t(i)$  is the reconstructed datapoint from the  $i^{th}$  autoencoder. This is illustrated in Figure 28.

5. Finally, the cluster is selected based on the auto-encoder, which gives the least reconstruction error - as formulated in Equation (35).

$$\hat{c}_t = \arg \min_{i=1}^k d(x_t, \hat{x}_t(i)), t = 1, \dots, n. \quad (35)$$

## Chapter 13. Conclusion and Future Work

This research provided a generic framework for clustering the acoustic calls for wildlife monitoring. An in-depth analysis of bat calls clustering while interpreting the clusters formed had also been presented. Steps for formulating the problem of ecological surveillance into an image clustering problem were described in detail. This work was built on state-of-the-art research in wildlife monitoring and utilized batdetect [22] to get potential bat calls from raw audio files. Then the recent developments in the field of deep learning (from the last 5 years) were used to formulate a practical use-case of segmenting the behavior of bats into different clusters (based on their acoustic calls).

This was the first study to compare the relative performance of well-known unsupervised deep learning algorithms for bat calls clustering. Essentially, an end-to-end system was developed for an uncontrolled environment (considering the practicality constraints) which could potentially lead to a major development in wildlife monitoring. The results were established by utilizing only two audio augmentation techniques. This approach is not limited to singular use-case but can be expanded based on the application of interest. For example, in the case of bats, species identification is also performed with this method because each species has a different acoustic signature. The signals of interest are highlighted through pre-processing before applying the clustering algorithms.

In the future, more experimental data could be collected, and by varying the target number of clusters, its correlation to the behavioral characteristics could be observed. Right now, it requires an expert (or as in our case, a coding scheme suggested by the domain expert) to interpret the results. In the future, labeling techniques could be used to help in interpretation.

To improve the results, it has been observed that IIC gave better results, for species clustering than SCAN, even though SCAN's performance on image clustering algorithms is higher. We hypothesize that this difference could be overcome by utilizing more robust and different augmentation techniques. The author's implementation of SCAN utilized the permutation of 14 different forms of augmentations. However, in IIC and IMSAT, the different types of augmentations used were only two and one respectively. Our technique also utilized two different augmentations – described in

section 3.4. The results could be improved if more augmentation techniques are devised, in the future, for bats acoustic data. In contrast to other algorithms, SCAN utilized a multi-step approach where the feature learning stage was separate from the clustering stage. This also exacerbates the problem of feature learning when less augmentations were available. So, to make the learning more robust for SCAN, it is crucial that more augmentation techniques are developed.

To improve SCAN, a promising avenue is using to use transfer learning. More specifically, in the case of SCAN, a pre-trained Resnet-50 model trained on audio's spectrogram features should be used as a base/initial model. Recently, [85] used Resnet-50 on embedding features of the AudioSet dataset for audio pattern recognition. Another work also used a similar approach [86]. The problem is that these models are not built on the spectrogram's features, rather on the embeddings of the dataset – released by [87]. According to [88], transfer learning could only work if the learned features are meaningful to both base and the target task, and not only the base domain. Although it does not matter if the spectrograms consisted of bats' data or not, it is still essential that the representation of the feature is consistent in both the base and target domain. Unfortunately, any such models do not exist in literature right now because deep learning has only emerged to be used in the domain of audio, very recently [89]. So, in the future, when this field of research matures, it is recommended to run SCAN on the pre-trained Resnet-50 model, that's trained for spectrograms features, to improve results.

Another direction to improve the results could be to experiment with different algorithms. Different training paradigms could be experimented with, to see if any other algorithm will provide better results than the already explored algorithms. Few examples of these clustering algorithms are given from sections 12.1 to 12.3.

## References

- [1] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, “Learning Discrete Representations via Information Maximizing Self-Augmented Training,” *arXiv:1702.08720 [cs, stat]*, Jun. 2017.
- [2] X. Ji, A. Vedaldi, and J. Henriques, “Invariant Information Clustering for Unsupervised Image Classification and Segmentation,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 9864–9873. doi: 10.1109/ICCV.2019.00996.
- [3] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, “SCAN: Learning to Classify Images without Labels,” *arXiv:2005.12320 [cs]*, Jul. 2020.
- [4] J. Yang, D. Parikh, and D. Batra, “Joint Unsupervised Learning of Deep Representations and Image Clusters,” *arXiv:1604.03628 [cs]*, Jun. 2016.
- [5] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep Clustering for Unsupervised Learning of Visual Features,” *arXiv:1807.05520 [cs]*, Mar. 2019.
- [6] E. B. Morrison and C. A. Lindell, “Birds and bats reduce insect biomass and leaf damage in tropical forest restoration sites,” *Ecological Applications*, vol. 22, no. 5, pp. 1526–1534, 2012, doi: 10.1890/11-1118.1.
- [7] R. Muscarella and T. H. Fleming, “The Role of Frugivorous Bats in Tropical Forest Succession,” *Biological Reviews*, vol. 82, no. 4, pp. 573–590, 2007, doi: 10.1111/j.1469-185X.2007.00026.x.
- [8] D. H. Kelm, K. R. Wiesner, and O. von Helversen, “Effects of Artificial Roosts for Frugivorous Bats on Seed Dispersal in a Neotropical Forest Pasture Mosaic,” *Conservation Biology*, vol. 22, no. 3, pp. 733–741, 2008, doi: 10.1111/j.1523-1739.2008.00925.x.
- [9] R. M. Nowak and E. P. Walker, *Walker’s Bats of the World*. JHU Press, 1994.
- [10] M. de la Peña-Domene, C. Martínez-Garza, S. Palmas-Pérez, E. Rivas-Alonso, and H. F. Howe, “Roles of Birds and Bats in Early Tropical-Forest Restoration,” *PLOS ONE*, vol. 9, no. 8, p. e104656, Aug. 2014, doi: 10.1371/journal.pone.0104656.
- [11] E. C. Braun de Torrez, H. K. Ober, and R. A. McCleery, “Restoring historical fire regimes increases activity of endangered bats,” *Fire Ecology*, vol. 14, no. 2, p. 9, Dec. 2018, doi: 10.1186/s42408-018-0006-8.
- [12] A. M. Bailey, H. K. Ober, A. R. Sovie, and R. A. McCleery, “Impact of land use and climate on the distribution of the endangered Florida bonneted bat,” *Journal of Mammalogy*, vol. 98, no. 6, pp. 1586–1593, Dec. 2017, doi: 10.1093/jmammal/gyx117.
- [13] G. Lippi, C. Mattiuzzi, C. Bovo, and M. Plebani, “Current laboratory diagnostics of coronavirus disease 2019 (COVID-19),” *Acta Bio Medica Atenei Parmensis*, vol. 91, no. 2, pp. 137–145, May 2020, doi: 10.23750/abm.v91i2.9548.
- [14] M. Z. Magnino, K. A. Holder, and S. A. Norton, “White-nose syndrome: a novel dermatomycosis of biologic interest and epidemiologic consequence,” *Clinics in Dermatology*, Aug. 2020, doi: 10.1016/j.clindermatol.2020.07.005.
- [15] J. Hu, W. Huang, Y. Su, Y. Liu, and P. Xiao, “BatNet++: A Robust Deep Learning-Based Predicting Models for Calls Recognition,” in *2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA)*, Kunming, China, Jun. 2020, pp. 260–263. doi: 10.1109/ICSGEA51094.2020.00062.

- [16] I. Zualkernan, J. Judas, T. Mahbub, A. Bhagwagar, and P. Chand, “A Tiny CNN Architecture for Identifying Bat Species from Echolocation Calls,” in *2020 IEEE/ITU International Conference on Artificial Intelligence for Good (AI4G)*, 2020, pp. 81–86.
- [17] L. Schmarje, M. Santarossa, S.-M. Schröder, and R. Koch, “A survey on Semi-, Self- and Unsupervised Learning for Image Classification,” *arXiv:2002.08721 [cs]*, Jul. 2020.
- [18] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, “A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture,” *IEEE Access*, vol. 6, pp. 39501–39514, 2018, doi: 10.1109/ACCESS.2018.2855437.
- [19] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, “Deep Adaptive Image Clustering,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Oct. 2017, pp. 5880–5888. doi: 10.1109/ICCV.2017.626.
- [20] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised Deep Embedding for Clustering Analysis,” *arXiv:1511.06335 [cs]*, May 2016.
- [21] Y. Opoichinsky, S. E. Chazan, S. Gannot, and J. Goldberger, “K-Autoencoders Deep Clustering,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020, pp. 4037–4041. doi: 10.1109/ICASSP40776.2020.9053109.
- [22] O. Mac Aodha, R. Gibb, K. E. Barlow, E. Browning, M. Firman, R. Freeman, B. Harder, L. Kinsey, G. R. Mead, S. E. Newson, I. Pandourski, S. Parsons, J. Russ, A. Szodoray-Paradi, F. Szodoray-Paradi, E. Tilova, M. Girolami, G. Brostow, and K. E. Jones, “Bat detective—Deep learning tools for bat acoustic signal detection,” *PLoS computational biology*, vol. 14, no. 3, p. e1005995, Mar. 2018, doi: 10.1371/journal.pcbi.1005995.
- [23] “Sound reception - Echolocation in bats,” *Encyclopedia Britannica*. <https://www.britannica.com/science/sound-reception> (accessed Apr. 18, 2021).
- [24] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, doi: 10.1186/s40537-019-0197-0.
- [25] J. Schlüter and T. Grill, “Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks,” in *ISMIR*, Malaga, Spain, 2015, pp. 121–126.
- [26] Z. Tariq, S. K. Shah, and Y. Lee, “Lung Disease Classification using Deep Convolutional Neural Network,” in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, CA, USA, Nov. 2019, pp. 732–735. doi: 10.1109/BIBM47256.2019.8983071.
- [27] C.-P. Tsai, Y.-L. Tuan, and L.-S. Lee, “Transcribing Lyrics from Commercial Song Audio: the First Step Towards Singing Content Processing,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, Apr. 2018, pp. 5749–5753. doi: 10.1109/ICASSP.2018.8462247.
- [28] R. Hyder, S. Ghaffarzagdegan, Z. Feng, J. H. L. Hansen, and T. Hasan, “Acoustic Scene Classification Using a CNN-SuperVector System Trained with Auditory and Spectrogram Image Features,” in *Interspeech 2017*, Stockholm, Sweden, Aug. 2017, pp. 3073–3077. doi: 10.21437/Interspeech.2017-431.
- [29] J. Zhao, D. Lu, K. Ma, Y. Zhang, and Y. Zheng, “Deep Image Clustering with Category-Style Representation,” *arXiv:2007.10004 [cs]*, Jul. 2020.

- [30] Y. Ren, N. Wang, M. Li, and Z. Xu, “Deep density-based image clustering,” *Knowledge-Based Systems*, vol. 197, p. 105841, Jun. 2020, doi: 10.1016/j.knsys.2020.105841.
- [31] H.-U. Schnitzler and E. K. V. Kalko, “Echolocation by Insect-Eating Bats,” *BioScience*, vol. 51, no. 7, p. 557, 2001, doi: 10.1641/0006-3568(2001)051[0557:EBIEB]2.0.CO;2.
- [32] M. L. Perkins, H. K. Frank, J. M. Pauly, and E. A. Hadly, “Frequency shifting reduces but does not eliminate acoustic interference between echolocating bats: A theoretical analysis,” *The Journal of the Acoustical Society of America*, vol. 142, no. 4, pp. 2133–2142, Oct. 2017, doi: 10.1121/1.5006928.
- [33] T. Hügél, V. van Meir, A. Muñoz-Meneses, B.-M. Clarin, B. M. Siemers, and H. R. Goerlitz, “Does similarity in call structure or foraging ecology explain interspecific information transfer in wild *Myotis* bats?,” *Behav Ecol Sociobiol*, vol. 71, no. 11, p. 168, Oct. 2017, doi: 10.1007/s00265-017-2398-x.
- [34] D. A. Schmieder, T. Kingston, R. Hashim, and B. M. Siemers, “Breaking the trade-off: rainforest bats maximize bandwidth and repetition rate of echolocation calls as they approach prey,” *Biology Letters*, vol. 6, no. 5, pp. 604–609, Oct. 2010, doi: 10.1098/rsbl.2010.0114.
- [35] M. K. Obrist, “Flexible Bat Echolocation: The Influence of Individual, Habitat and Conspecifics on Sonar Signal Design,” *Behavioral Ecology and Sociobiology*, vol. 36, no. 3, pp. 207–219, 1995.
- [36] H. Romer and J. Lewald, “High-frequency sound transmission in natural habitats: implications for the evolution of insect acoustic communication,” *Behav Ecol Sociobiol*, vol. 29, no. 6, pp. 437–444, Feb. 1992, doi: 10.1007/BF00170174.
- [37] J. P. Bunkley, C. J. McClure, N. J. Kleist, C. D. Francis, and J. R. Barber, “Anthropogenic noise alters bat activity levels and echolocation calls,” *Global Ecology and Conservation*, vol. 3, pp. 62–71, 2015.
- [38] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.
- [39] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.
- [40] X. Jin and J. Han, “K-Medoids Clustering,” in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 564–565. doi: 10.1007/978-0-387-30164-8\_426.
- [41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.
- [42] “Structural Similarity Index,” *GitHub*, Jul. 13, 2021. [https://github.com/scikit-image/scikit-image/blob/c480ca41bc9887e42d32f5434ecd90dedac13217/skimimage/metrics/\\_structural\\_similarity.py](https://github.com/scikit-image/scikit-image/blob/c480ca41bc9887e42d32f5434ecd90dedac13217/skimimage/metrics/_structural_similarity.py) (accessed Jul. 13, 2021).
- [43] E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection,” in *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006, pp. 430–443. doi: 10.1007/11744023\_34.
- [44] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in *Computer Vision – ECCV 2010*, Berlin, Heidelberg, 2010, pp. 778–792. doi: 10.1007/978-3-642-15561-1\_56.

- [45] “OpenCV: ORB (Oriented FAST and Rotated BRIEF),” *OpenCV*. [https://docs.opencv.org/master/d1/d89/tutorial\\_py\\_orb.html](https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html) (accessed Jul. 13, 2021).
- [46] B. Waggener, W. N. Waggener, and W. M. Waggener, *Pulse code modulation techniques*. Springer Science & Business Media, 1995.
- [47] “Image Similarity Measure,” *GitHub*, Jul. 07, 2021. <https://github.com/up42/image-similarity-measures> (accessed Jul. 13, 2021).
- [48] “KMedoids,” *Scikit-Learn*. [https://scikit-learn-extra.readthedocs.io/en/latest/generated/sklearn\\_extra.cluster.KMedoids.html](https://scikit-learn-extra.readthedocs.io/en/latest/generated/sklearn_extra.cluster.KMedoids.html) (accessed Jul. 13, 2021).
- [49] M. D. L. Srinivasulu and M. A. V. Reddy, “Improving The Scalability And Efficiency Of K-Medoids By Map Reduce,” vol. 2, no. 4, p. 3, 2015.
- [50] F. Zhdanov, “Diverse mini-batch Active Learning,” *arXiv:1901.05954 [cs, stat]*, Jan. 2019.
- [51] P. Arora, Deepali, and S. Varshney, “Analysis of K-Means and K-Medoids Algorithm For Big Data,” *Procedia Computer Science*, vol. 78, pp. 507–512, Jan. 2016, doi: 10.1016/j.procs.2016.02.095.
- [52] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for K-medoids clustering,” *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, Mar. 2009, doi: 10.1016/j.eswa.2008.01.039.
- [53] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii, “Distributional Smoothing with Virtual Adversarial Training,” *arXiv:1507.00677 [cs, stat]*, Jun. 2016.
- [54] “IMSAT’s Pytorch Implementation for Unsupervised Clustering,” *GitHub*. <https://github.com/arbab97/ImSAT-1> (accessed Apr. 21, 2021).
- [55] J. A. Hartigan, “Direct Clustering of a Data Matrix,” *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972, doi: 10.2307/2284710.
- [56] I. S. Dhillon, S. Mallela, and D. S. Modha, “Information-theoretic co-clustering,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, D.C., Aug. 2003, pp. 89–98. doi: 10.1145/956750.956764.
- [57] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” *arXiv:1808.06670 [cs, stat]*, Feb. 2019.
- [58] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” *arXiv:1312.6114 [cs, stat]*, May 2014.
- [59] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *arXiv:1511.06434 [cs]*, Jan. 2016.
- [60] A. Alguwaizani, “Degeneracy on K-means clustering,” *Electronic Notes in Discrete Mathematics*, vol. 39, pp. 13–20, Dec. 2012, doi: 10.1016/j.endm.2012.10.003.
- [61] A. Coates, A. Ng, and H. Lee, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Jun. 2011, pp. 215–223.
- [62] “TensorFlow Implementation of IIC,” *GitHub*. <https://github.com/arbab97/IIC-1> (accessed Apr. 22, 2021).

- [63] “Deep Clustering for Unsupervised Learning of Visual Features,” *GitHub*. <https://github.com/arbab97/deepcluster> (accessed Apr. 22, 2021).
- [64] “Implementation of SCAN for bats audio data,” *GitHub*. <https://github.com/arbab97/SCAN-algorithm> (accessed Apr. 22, 2021).
- [65] D. Zhao and X. Tang, “Cyclizing clusters via Zeta function of a graph,” in *Proceedings of the 21st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2008, pp. 1953–1960.
- [66] “Implementation of JULE for Bats Calls Classification In Tensorflow,” *GitHub*. <https://github.com/arbab97/joint-cluster-cnn> (accessed Apr. 22, 2021).
- [67] S. Romano, N. X. Vinh, J. Bailey, and K. Verspoor, “Adjusting for chance clustering comparison measures,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 4635–4666, 2016.
- [68] O. M. Aodha, “Bat Echolocation Call Detection in Audio Recordings,” May 18, 2021. <https://github.com/macaodha/batdetect> (accessed Aug. 28, 2021).
- [69] M. Schultz and T. Joachims, “Learning a distance metric from relative comparisons,” *Advances in neural information processing systems*, vol. 16, pp. 41–48, 2004.
- [70] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, Dec. 2006, pp. 153–160.
- [71] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen, and S. Li, “Optimized Cartesian K-Means,” *arXiv:1405.4054 [cs]*, May 2014.
- [72] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *arXiv:1511.06434 [cs]*, Jan. 2016.
- [73] D. Petrov, A. Ivanov, J. Faskowitz, B. Gutman, D. Moyer, J. Villalon, N. Jahanshad, and P. Thompson, “Evaluating 35 Methods to Generate Structural Connectomes Using Pairwise Classification,” in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2017*, Cham, 2017, pp. 515–522. doi: 10.1007/978-3-319-66182-7\_59.
- [74] S.-H. Park and J. Fürnkranz, “Efficient Pairwise Classification,” in *Machine Learning: ECML 2007*, Berlin, Heidelberg, 2007, pp. 658–665. doi: 10.1007/978-3-540-74958-5\_65.
- [75] H. Bao, T. Shimada, L. Xu, I. Sato, and M. Sugiyama, “Similarity-based Classification: Connecting Similarity Learning to Binary Classification,” *arXiv:2006.06207 [cs, stat]*, Jun. 2020.
- [76] K. Kim, S. Rohatgi, and C. L. Giles, “Hybrid deep pairwise classification for author name disambiguation,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, New York, United States, 2019, pp. 2369–2372.
- [77] J. Tanha, M. van Someren, and H. Afsarmanesh, “Semi-supervised self-training for decision tree classifiers,” *Int. J. Mach. Learn. & Cyber.*, vol. 8, no. 1, pp. 355–370, Feb. 2017, doi: 10.1007/s13042-015-0328-7.
- [78] J. Yadav and M. Sharma, “A Review of K-mean Algorithm,” *Int. J. Eng. Trends Technol.*, vol. 4, no. 7, pp. 2972–2976, 2013.
- [79] X. Guo, E. Zhu, X. Liu, and J. Yin, “Deep Embedded Clustering with Data Augmentation,” in *Asian Conference on Machine Learning*, Beijing, China, Nov. 2018, pp. 550–565.

- [80] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive Mixtures of Local Experts,” *Neural Computation*, vol. 3, no. 1, pp. 79–87, Feb. 1991, doi: 10.1162/neco.1991.3.1.79.
- [81] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” *arXiv:2003.05991 [cs, stat]*, Apr. 2021.
- [82] D. Zhang, Y. Sun, B. Eriksson, and L. Balzano, “Deep Unsupervised Clustering Using Mixture of Autoencoders,” *arXiv:1712.07788 [cs, stat]*, Dec. 2017.
- [83] S. E. Chazan, S. Gannot, and J. Goldberger, “Deep Clustering Based On A Mixture Of Autoencoders,” in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, PA, USA, Oct. 2019, pp. 1–6. doi: 10.1109/MLSP.2019.8918720.
- [84] B. Schölkopf, J. Platt, and T. Hofmann, “Greedy Layer-Wise Training of Deep Networks,” in *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, British Columbia, Canada, 2007, pp. 153–160.
- [85] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition,” *arXiv:1912.10211 [cs, eess]*, Aug. 2020.
- [86] Q. Kong, C. Yu, T. Iqbal, Y. Xu, W. Wang, and M. D. Plumbley, “Weakly Labelled AudioSet Tagging with Attention Neural Networks,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 11, pp. 1791–1802, Nov. 2019, doi: 10.1109/TASLP.2019.2930913.
- [87] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Louisiana, United States, Mar. 2017, pp. 131–135. doi: 10.1109/ICASSP.2017.7952132.
- [88] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *arXiv:1411.1792 [cs]*, Nov. 2014.
- [89] S. Latif, H. Cuayáhuítl, F. Pervez, F. Shamshad, H. S. Ali, and E. Cambria, “A Survey on Deep Reinforcement Learning for Audio-Based Applications,” *arXiv:2101.00240 [cs, eess]*, Jan. 2021.

## **Vita**

Muhammad Arbab Arshad was born in 1997, in Faisalabad, Pakistan. He received his primary and secondary education in Rawalpindi, Pakistan. He received his B.Sc. degree in Computer Sciences from the Lahore University of Management Sciences in 2019.

In September 2019, he joined the Computer Engineering master's program in the American University of Sharjah as a graduate teaching assistant. During his master's study, he co-authored 2 papers which were presented in international conferences. His research interests are in unsupervised deep learning and multicore computing.