Open Set Classification

Open set classification refers to the scenario where a model must identify whether a given instance belongs to a class it has been trained on (**known class**) or to a previously unseen class (**unknown class**). This capability is important in real-world applications where it's unrealistic to assume that all possible classes are known during the training phase.

Using CNN for Feature Extraction and Generative Models for Classification

In the context of open set classification, one effective approach involves the following steps:

1. **Feature Extraction with CNN:** A Convolutional Neural Network (CNN) is employed to **extract meaningful features** from the input data. These features are then used as **input** for the classification stage.
2. **Classification with Generative Models:** For each class in the training set, a generative model is trained. Generative models, such as Gaussian Mixture Models (GMMs) or Variational Autoencoders (VAEs), learn the **distribution of the features** for each known class. By modeling the probability distribution of the features for each class, these models can generate new instances or **calculate the likelihood** of a given feature vector belonging to a specific class.
3. **Unknown Detection with a Threshold:** To handle unknown classes (classes not seen during training), a **threshold is defined on the likelihood** scores or distances produced by the generative models. When a new instance is classified, its likelihood or distance to the nearest class is calculated. If this value is under the predefined threshold, the instance is flagged as belonging to an unknown class. This threshold-based approach allows the model to distinguish between known and unknown classes effectively.

ResNet50 for Feature Extraction and GMM for Generative Modeling

The combination of **ResNet50** for feature extraction and **GMM** for generative modeling offers a robust framework for open set classification. ResNet50 provides deep, hierarchical features that capture the complexity of the input data, while GMM models the distribution of these features for each known class, allowing for effective classification and unknown detection.

The Importance of Outlier Removal in Threshold Setting

In open set classification, setting an appropriate threshold is critical for effectively **distinguishing** between known and unknown classes. The **removal of outliers** before setting the threshold is crucial for the model's performance. Outliers can **distort the distribution** of the data, leading to inaccurate thresholds that either allow too many unknown instances into the

known classes or incorrectly reject known instances. By eliminating these outliers, the model can set a more accurate and balanced threshold, improving its ability to classify known instances correctly and detect unknown instances effectively.

Openness

Openness is a key concept in open set classification, **quantifying the degree** to which a classification problem involves unknown classes that were not present during the training phase. Openness is important because it directly affects the design and evaluation of open set classification models. A model designed for a **high-openness** environment must prioritize the detection of unknown classes, while a model for **low-openness** may focus more on accurately classifying known classes. Understanding the openness of a problem helps in selecting appropriate modeling strategies, such as **adjusting the threshold** for unknown detection or **choosing models** that can effectively generalize to unseen classes.

Saying "I Don't Know" Is Better Than Misclassification

In open set classification, it is often better for a model to respond with "I don't know" or to flag an instance as unknown rather than risk misclassifying it. This approach has several key advantages:

1. Avoiding Misleading Results:

- **False Confidence:** Misclassifying an unknown instance as a known class can lead to a false sense of confidence in the model's predictions. This is particularly dangerous in applications where incorrect classifications can have serious consequences, such as in medical diagnosis or security systems.

2. Enabling Further Analysis:

- **Trigger for Human Intervention:** When a model indicates that it does not know how to classify an instance, it can serve as a trigger for further analysis, often involving human experts. This allows for a more thorough examination of the data and the possibility of discovering new classes or patterns that were not initially considered during training.
- **Learning Opportunity:** Marking instances as unknown provides an opportunity to expand the model's knowledge base. These instances can be used to update the model, either by incorporating them into the training set or by adjusting the model's parameters to improve its ability to handle similar cases in the future.

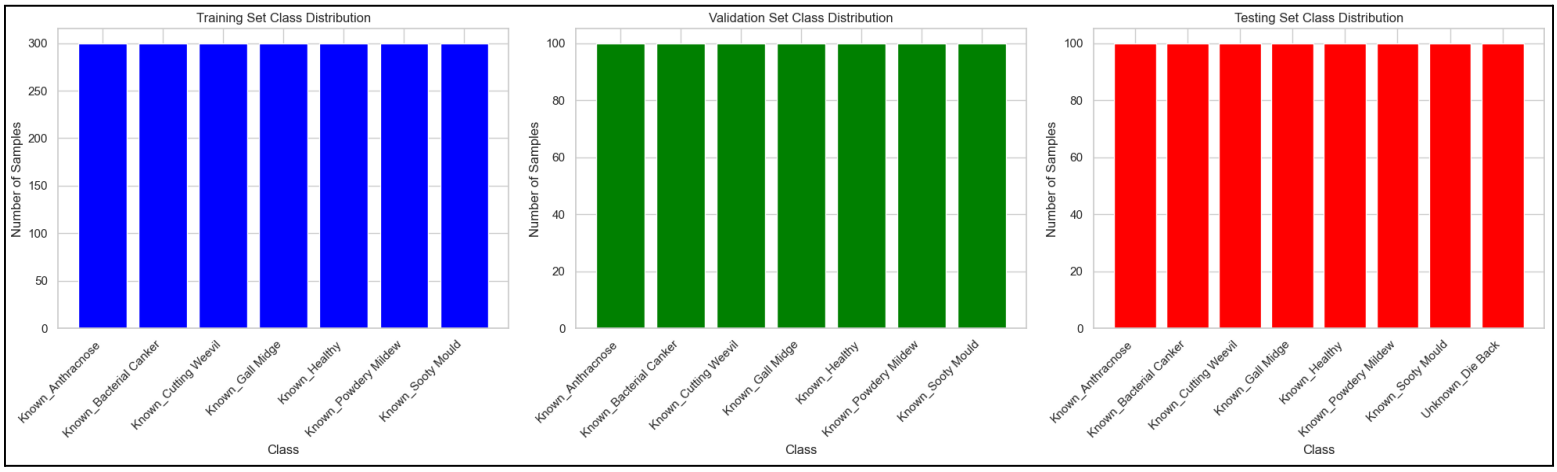
Open Set Classification Report

This report provides a detailed analysis of an open set classification task, focusing on key aspects such as class distribution, feature extraction, model performance, and evaluation. The following sections present graphical representations and interpretations of the data and model results.

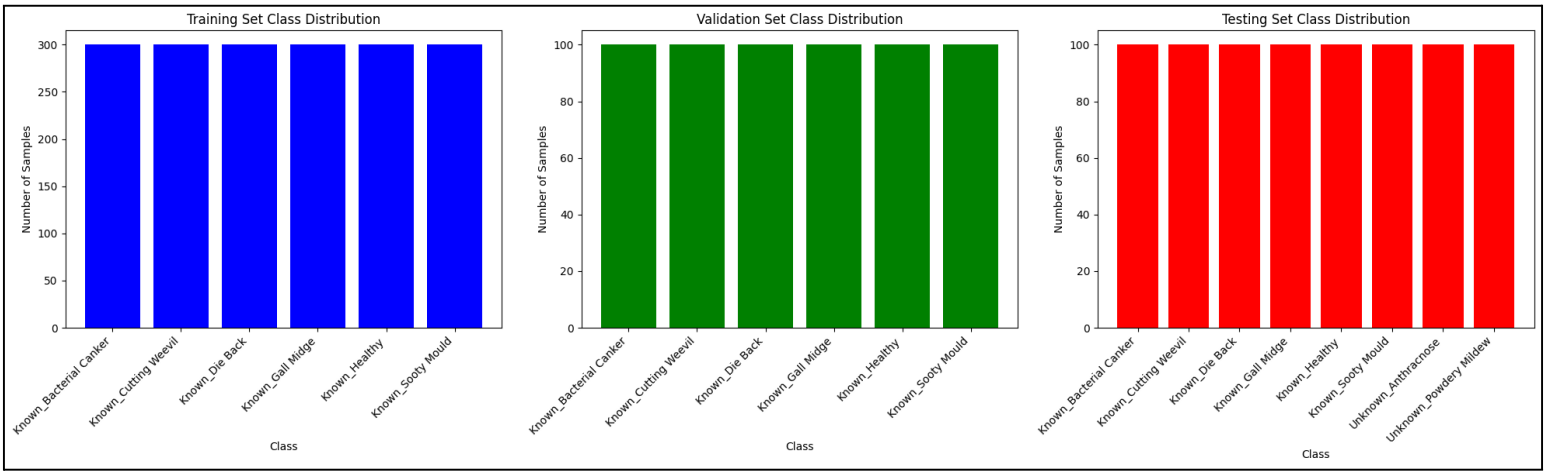
1. Dataset: Class Distribution Graph

The class distribution graph visualizes the number of samples per class in the dataset. This distribution is crucial for understanding the balance of the dataset and identifying any potential class imbalances that might affect the model's performance.

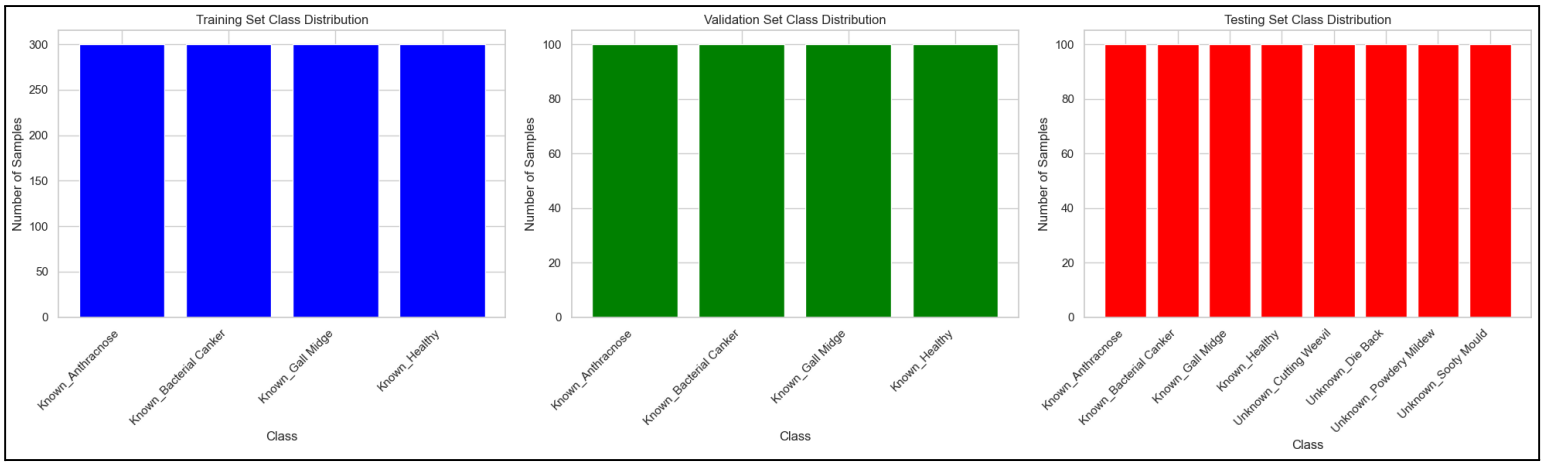
- Openness 10



- Openness 22



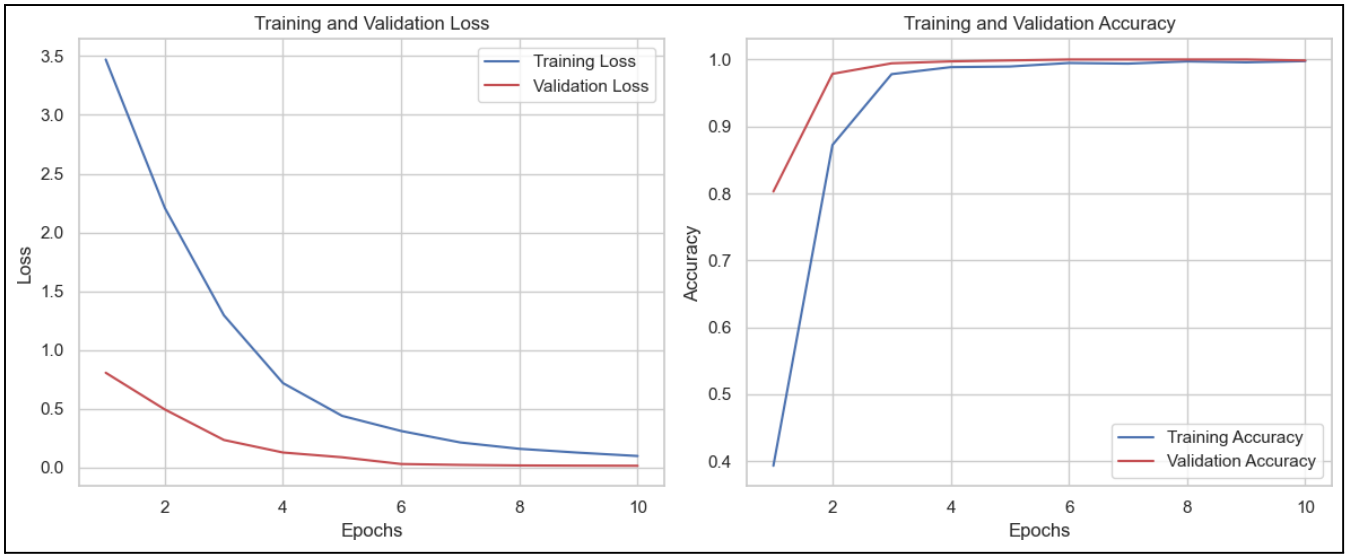
- Openness 45



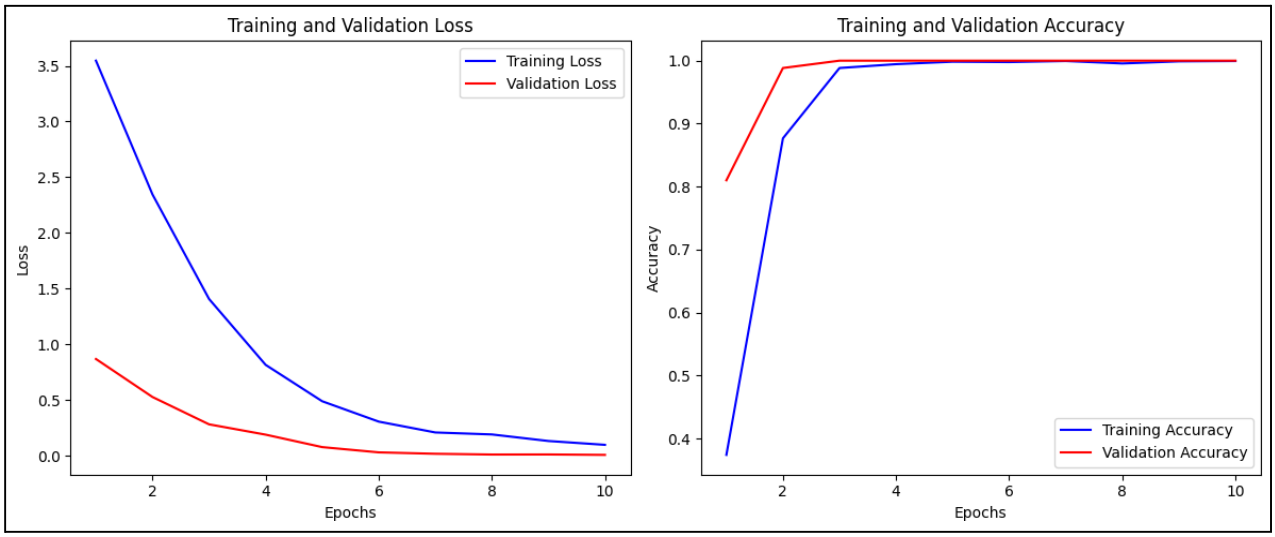
2. ResNet Model: Training and Validation Loss-Accuracy Graph

The training and validation loss and accuracy graphs depict the performance of the ResNet model during training.

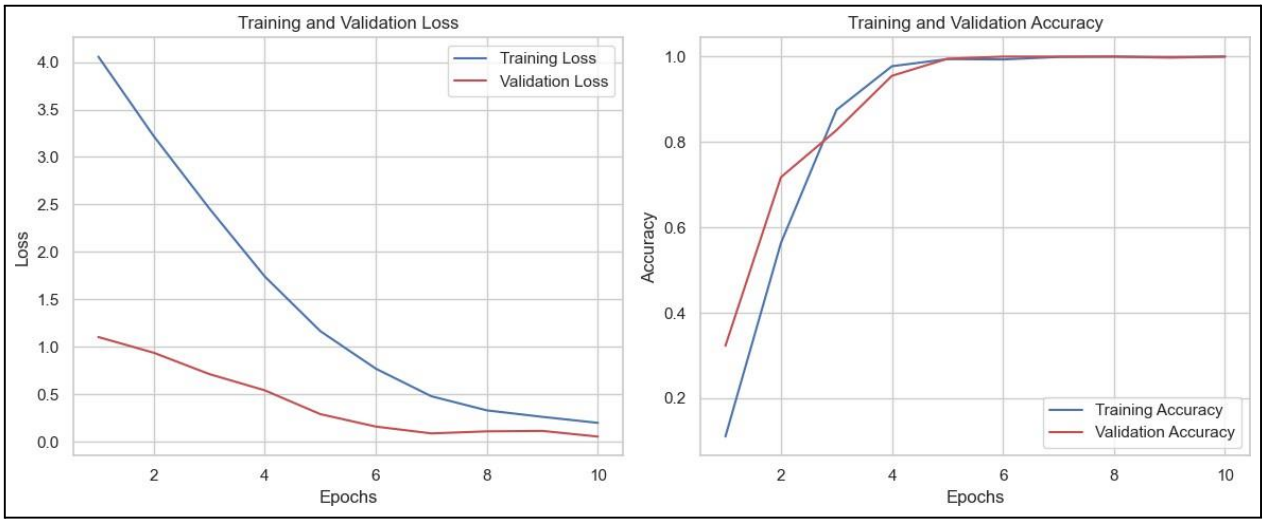
- Openness 10



- Openness 22



- Openness 45

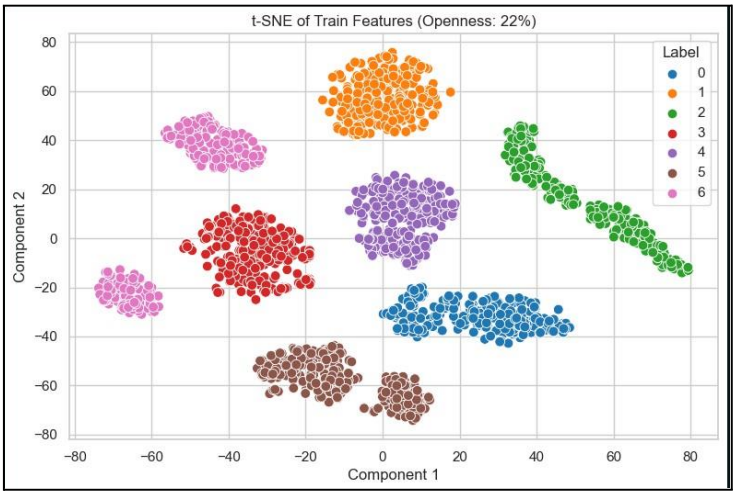


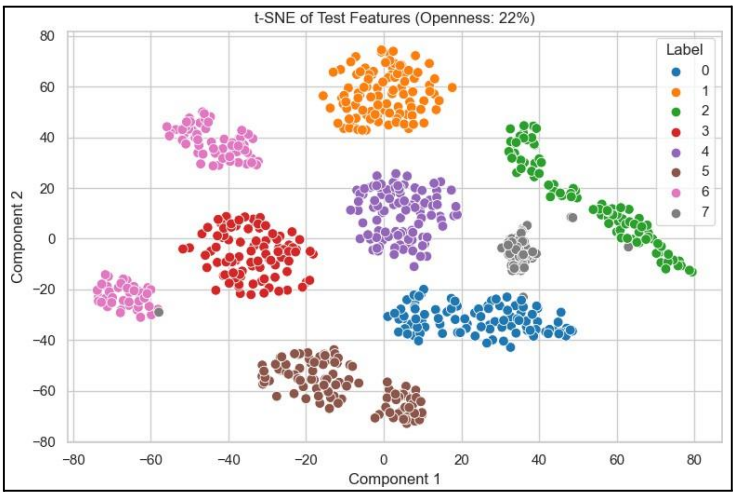
3. Extracted Features: t-SNE of Train Features Graph

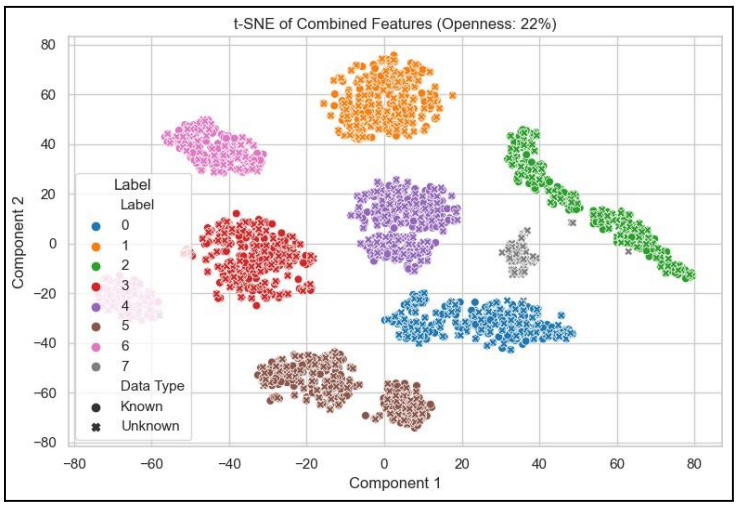
A t-SNE plot of the features extracted by the ResNet model from the training data is presented. t-SNE reduces the high-dimensional feature space into two or three dimensions for visualization. This graph provides insights into **how well the model's feature extractor** clusters data points from the same class while separating those from different classes. Effective clustering

indicates that the extracted features are discriminative and suitable for the subsequent classification task.

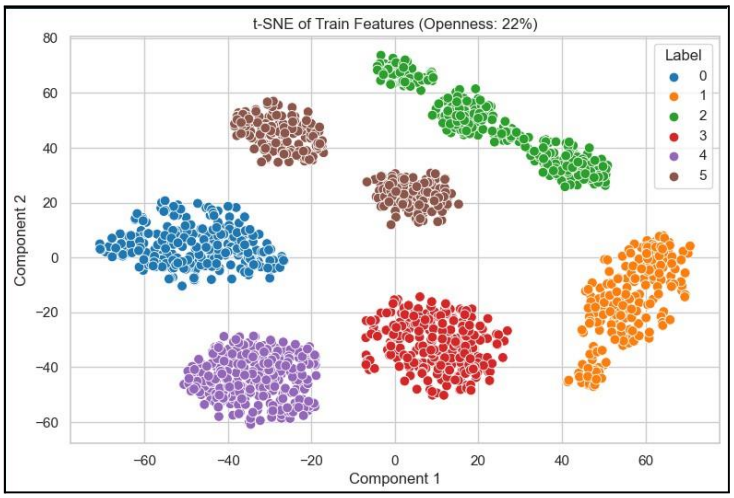
Openness 10

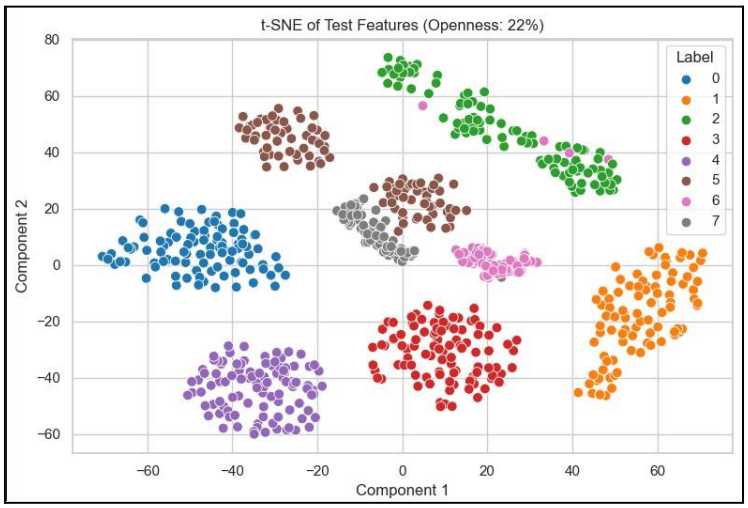


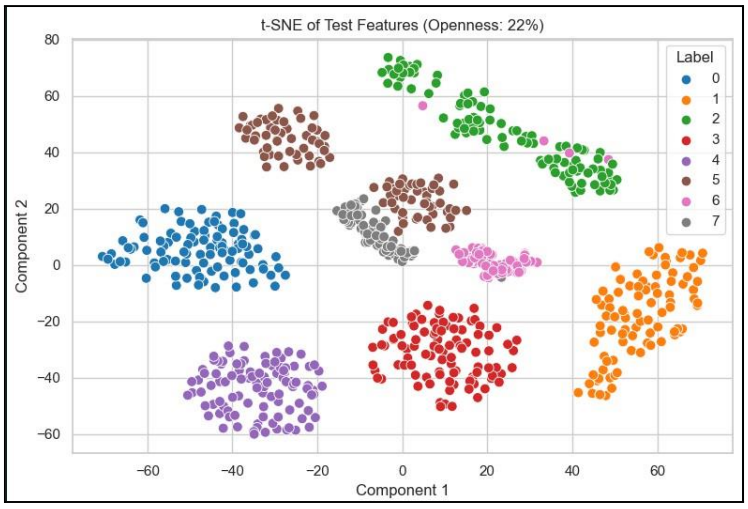




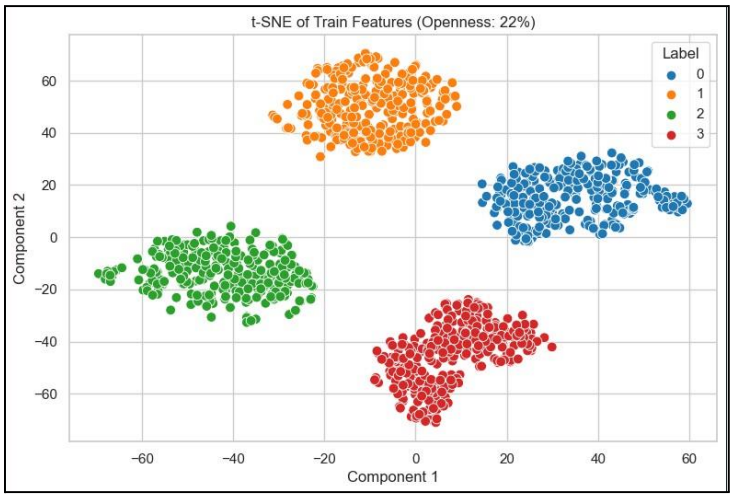
Openness 22

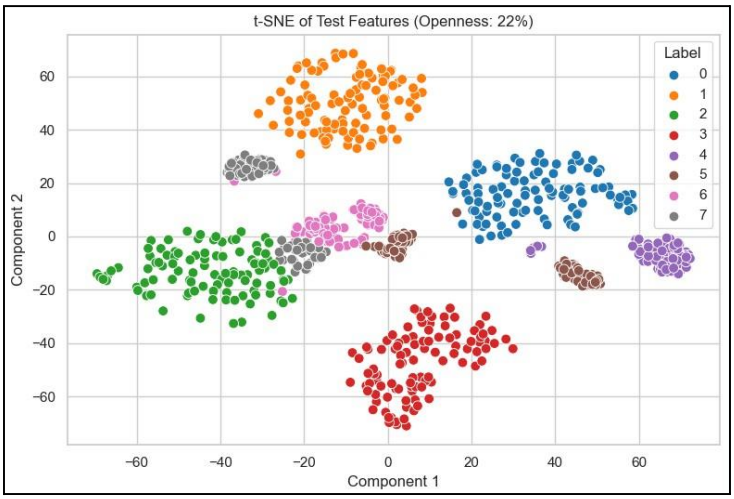


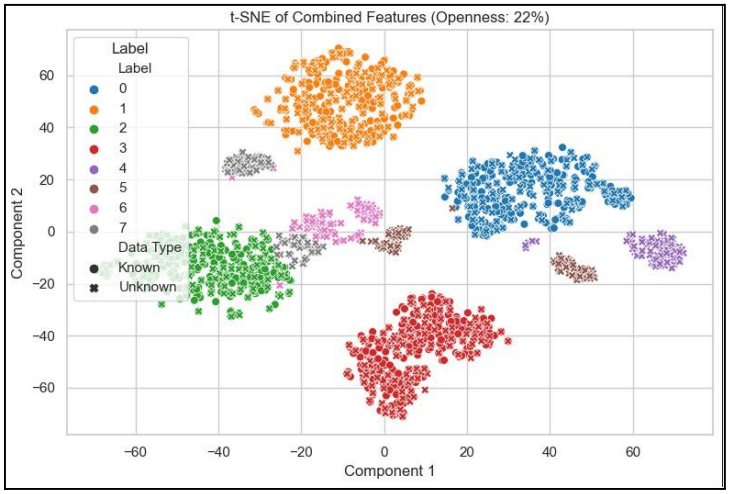




Openness 45



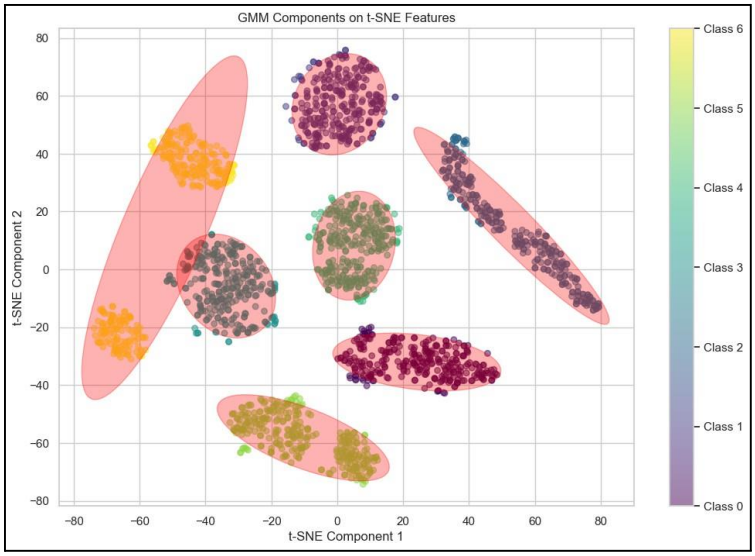




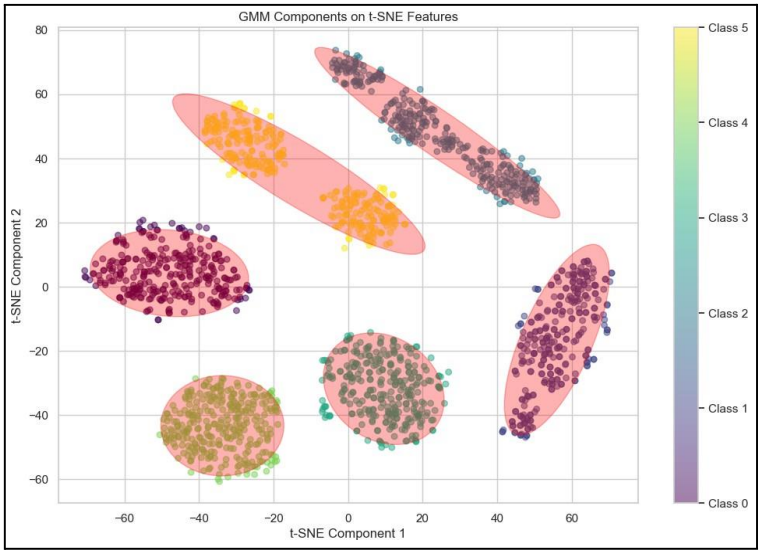
4. GMM: GMM Components on t-SNE Features Graph

The Gaussian Mixture Model (GMM) components are overlaid on the t-SNE plot of the extracted features. These graphs show how the GMM fits the distribution of the data in the feature space. Each component represents a Gaussian distribution learned from the features of a particular class. The visualization helps in understanding how well the GMM captures the underlying structure of the data and whether the model can distinguish between classes effectively.

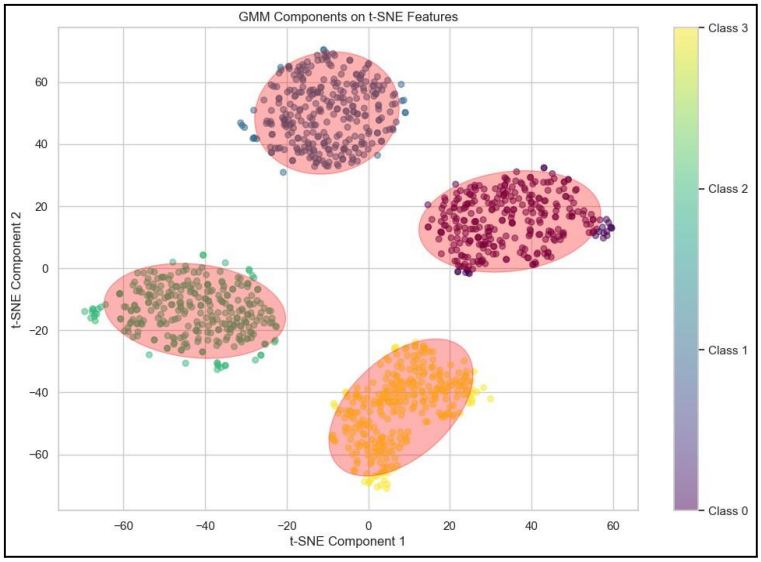
Openness 10



Openness 22



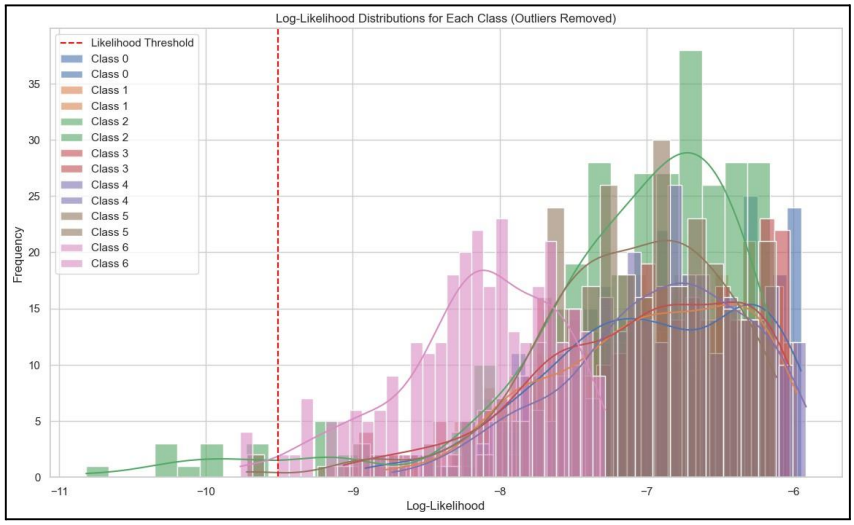
Openness 45



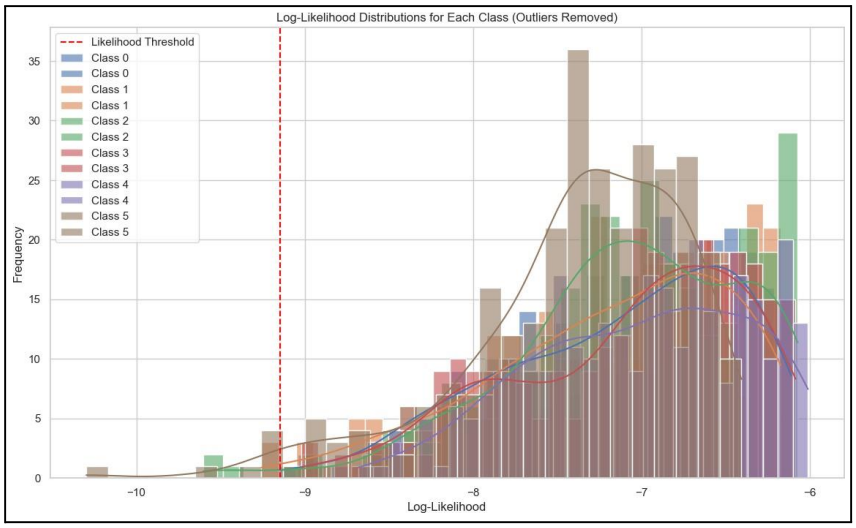
5. Threshold: Log-Likelihood Distribution for Each Class with Threshold Line Graph

These graphs show the log-likelihood distribution of each class, with a threshold line indicating the **decision boundary** for unknown detection. The log-likelihood is a measure of how likely a data point belongs to a particular class based on the GMM. The threshold line is critical for **distinguishing** known classes from unknown instances. Data points with log-likelihoods below the threshold are flagged as unknown, helping to prevent misclassification of novel inputs.

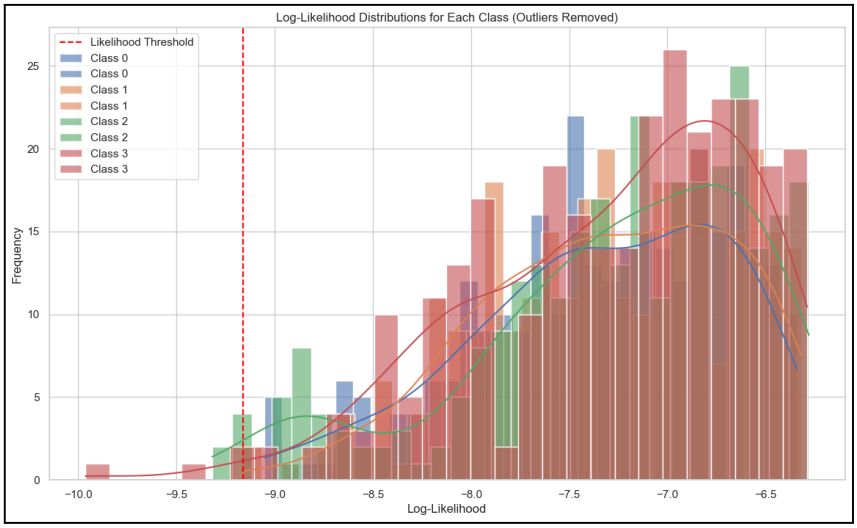
- Openness 10



- Openness 22

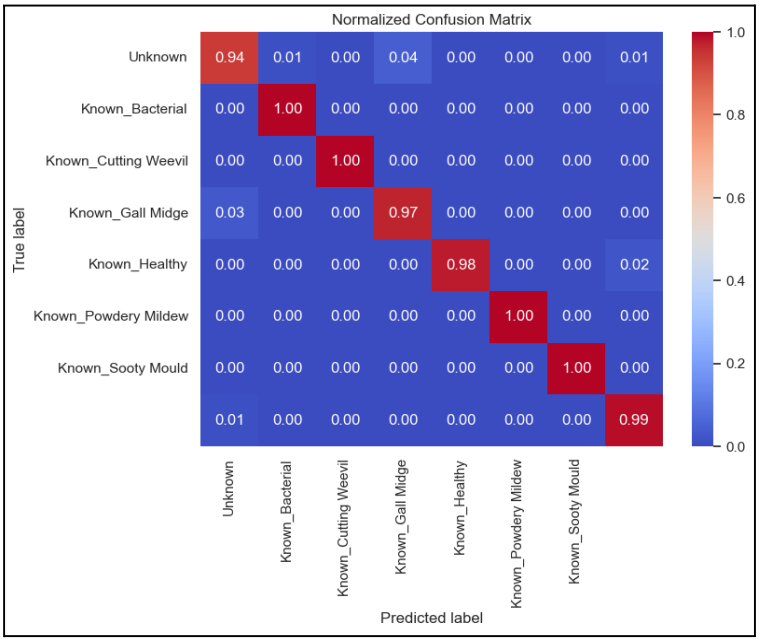


- Openness 45

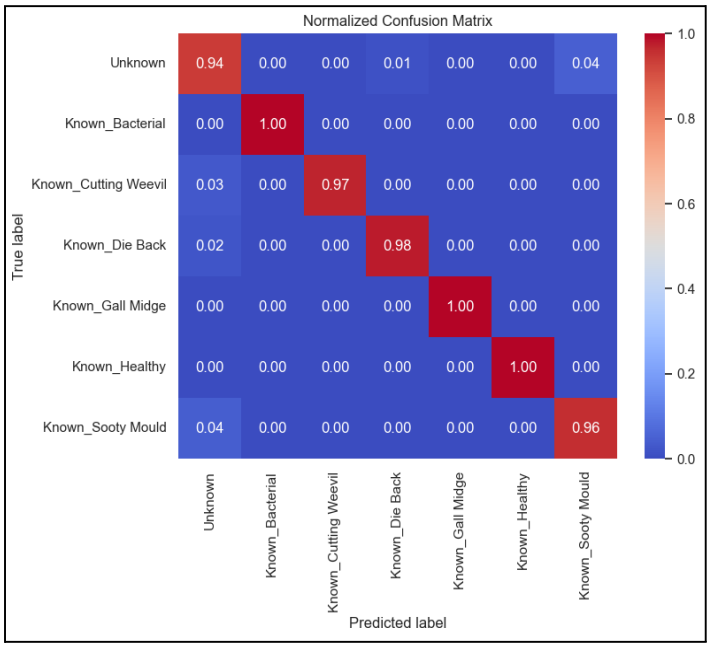


6. Evaluation: Confusion Matrix

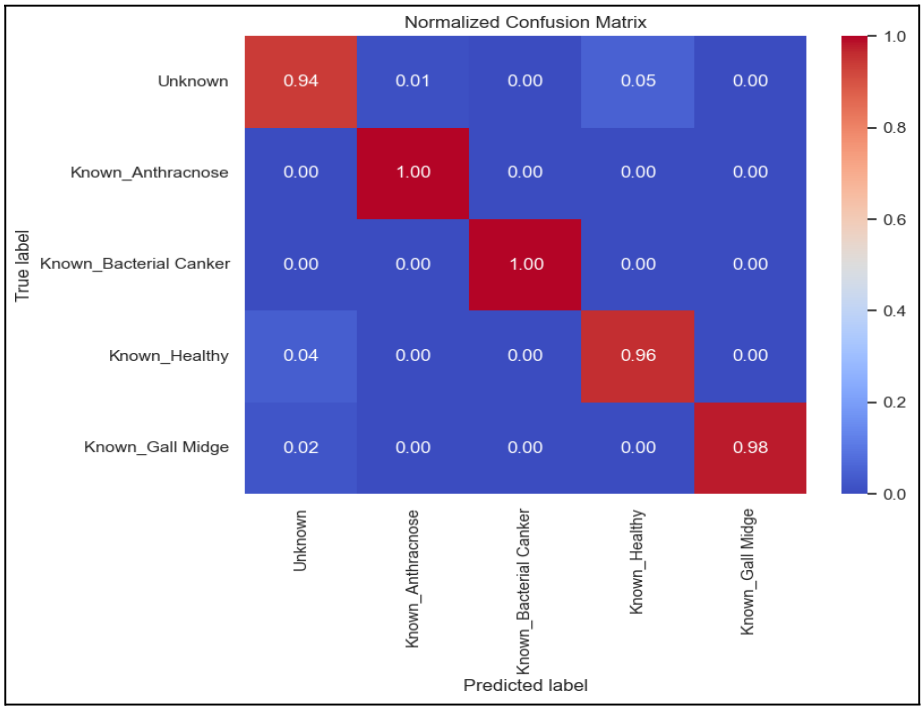
Openness 10



Openness 22



Openness 45



Open Set Classification with Signal Data

Data Processing and Interpolation

The IQ signal data is stored as **binary files**, where the data represents the **in-phase (I) and quadrature (Q) components** of a signal. The key challenge in working with IQ signals is ensuring that the input data is **consistent in terms of length** across all samples. To address this, we perform the following steps:

- **Loading IQ Data**: We read the signal data from binary files where each pair of elements corresponds to the I and Q components.
- **Interpolation**: We identify the longest signal and interpolate the shorter signals to match this length using linear interpolation. This ensures that all signals have the same length.
- **Stacking I and Q Components**: Once the data is interpolated, we stack the I and Q components into a **2xN matrix** for each sample, where N is the length of the signal.

Key Differences Between Signal Data and Image Data

1. Data Representation and Structure

- **Image Data**: Images are typically **two-dimensional arrays** where pixel intensity values are **spatially correlated**. For example, adjacent pixels in an image are often part of the same object or feature, and **convolutional neural networks** (CNNs) are excellent at capturing these spatial dependencies through filters and convolutions.
- **Signal Data**: IQ signals, in contrast, are **time-series data** where the relationship between consecutive data points is **temporal, not spatial**. The I and Q components represent the signal's magnitude and phase over time, and their patterns are better suited for models that can capture temporal correlations, such as **recurrent neural networks** (RNNs) or **transformers**, rather than spatial patterns.

2. Dimensionality and Information Encoding

- **Image Data:** In images, the structure is typically fixed (e.g., 224x224 for ResNet50), and each pixel represents a distinct point in space, contributing to an overall visual pattern. CNNs are designed to down-sample and detect increasingly abstract patterns from these fixed dimensions.
- **Signal Data:** IQ signals are often variable in length, and while resizing these signals into a fixed 2D structure (like 224x224) allows them to be processed by CNNs, this resizing can distort the underlying temporal information. Important features related to the signal's frequency, amplitude, and phase can be lost or misrepresented during interpolation and resizing, leading to poor performance.

3. Spectral and Temporal Information

- **Image Data:** Convolutional layers are effective at detecting edges, textures, and shapes, which are relevant for image classification.
- **Signal Data:** In signal processing, crucial information is often contained in the frequency and phase domains rather than spatial patterns. For instance, Fourier transforms are commonly used to analyze signal data, as they capture frequency characteristics that CNNs are not inherently designed to detect. The ResNet50 model, trained on image data, lacks mechanisms to directly interpret these frequency-domain features, limiting its effectiveness on signal data.

4. Feature Distribution and Representation

- **Image Data**: Visual features in images, like edges and textures, are robust and invariant across different scales, and CNNs are adept at extracting these features through hierarchical layers.

- **Signal Data:** IQ signals often exhibit very different feature characteristics, such as signal power, frequency shifts, or modulation patterns. These features require specialized models designed to interpret time-varying signals and frequency distributions, which CNNs struggle to handle without appropriate pre-processing (e.g., converting signals to spectrograms).

Alternatives and Solutions

1. Using Signal-Specific Models

- **Recurrent Neural Networks (RNNs):** RNNs and LSTMs are better suited for time-series data like IQ signals because they can capture temporal dependencies and patterns over time.
- **Spectrograms and CNNs:** One approach is to transform the IQ signal data into spectrograms (time-frequency representations) and then use CNNs. This method preserves the frequency domain information and allows the CNN to learn relevant patterns.
- **Transformers for Sequence Modeling:** Transformer models, designed to capture long-range dependencies, have shown strong performance on time-series data and can be a better fit for signal data.

2. Feature Engineering

- **Fourier Transform:** Applying a Fourier transform to the signal data before feeding it into a deep learning model could help capture important frequency-domain features.
- **Wavelet Transform:** Wavelet transforms provide a time-frequency representation of the signal and can be useful for identifying features that change over time.