Part 1:

CONVOLUTION NEURAL NETWORK(CNN)

INTRODUCTION:

Convolution neural network is a powerful tool in task related to computer vision. This neural network designed to effectively process data such as images.

The dataset path consists of images of various classes. The image dataset is loaded and then resized them into 128*128 pixels. Labels are encoded and each class in it is assigned with a integer.[1]

MODEL TRAINING:

The dataset is then splitted into the test and train. The 70% of the data is considered as training data and the remaining 30% as testing data.[1][2]

Augmentation is applied to enhance the model performance. so, the training data is augmented using rotations, flips also the class weight is used during model training to give importance to unrepresented classes.

BUILDING THE CNN:

CNN is built using different layers which are convolutional layers, pooling layers, Flatten layers and Dense layers. The convolutional layers are used to increase the image filters to extract the features. The Pooling Layers are used to prevent the overfitting. Flatten layers is used to convert the 2D features to 1D. Dense layers are used with neurons and Relu activation. This activation introduces us the non-linearity and helps network learn complex patterns. The model is compiled using the optimizer, Adam. [3]

MODEL EVALUATION:

The model is evaluated to get the accuracy for the testing and training dataset. and we also tried to obtain some predictions. Whole dataset is shown in the performance matrix which includes accuracy, F1 score, Recall values.

	Dataset	Accuracy	Precision	Recall	F1 Score
0	Training	0.760116	0.807506	0.760116	0.759669
1	Testing	0.744108	0.799058	0.744108	0.741488

Figure 1

The confusion matrix is displayed to show how well the classes are predicting each class.

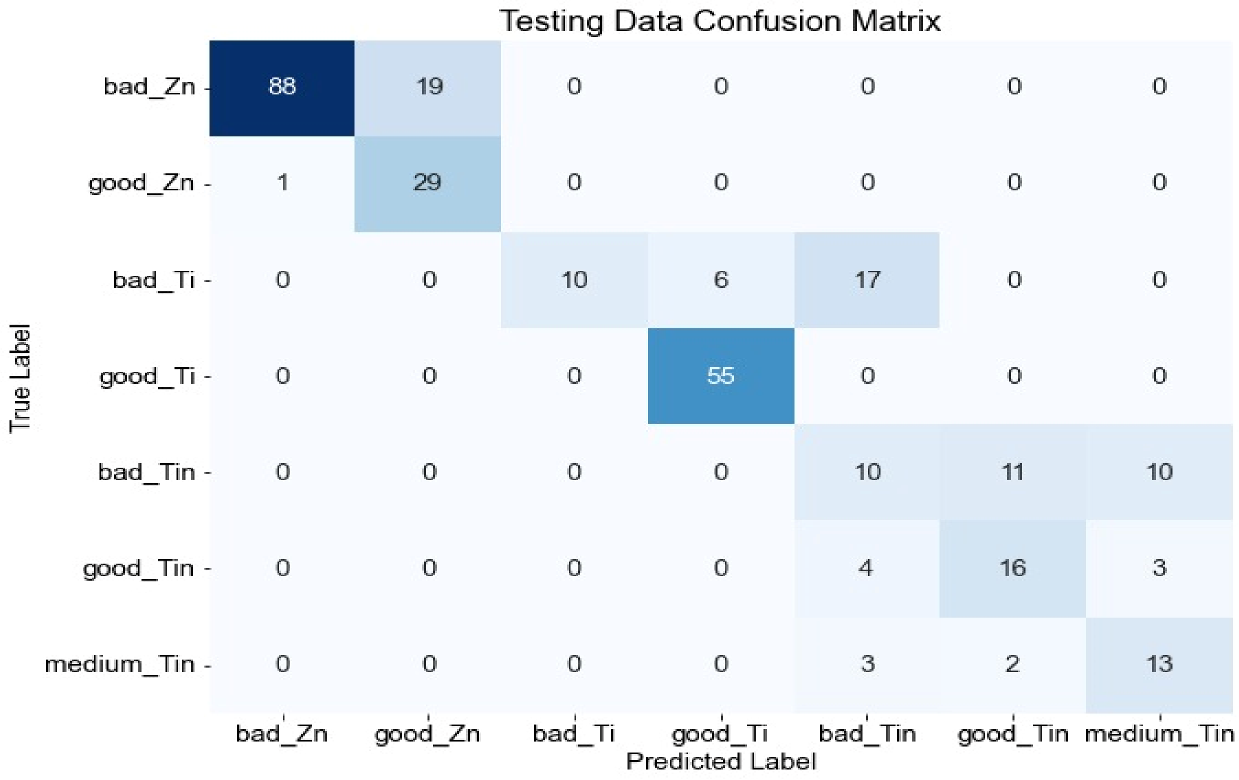


Figure 2

The below graph signifies the relationship of accuracy with the no. of epochs in training and testing datasets while a machine learning model is trained. The x-axis denotes the number of epochs, while the y-axis shows accuracy in the range of 0 to 1.[4]

The blue line indicates the training accuracy, which usually continues to increase with the number of epochs. This indicates that the model is learning with respect to its training data. The orange curve shows the test accuracy, the value of which climbs (with lots of fluctuations), at first, signifying the performance of a model with regard to unseen data. Thereafter, it appears that, at some point in time, the test accuracy starts to plateau, hence indicating overfitting risks.

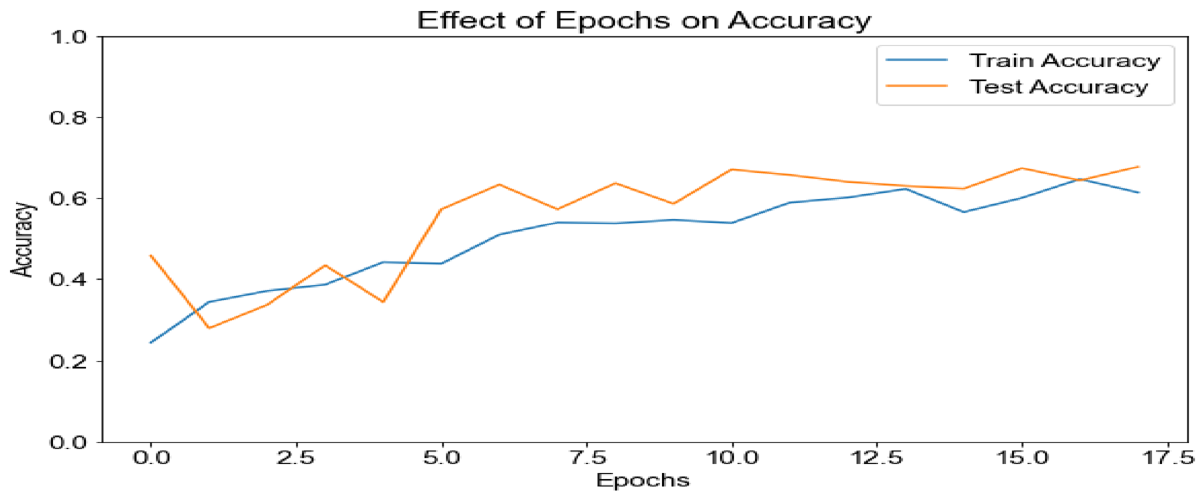


Figure 3

The graph shows the effect of epochs on the loss function for both testing and training datasets. The x-axis represents the epoch and y-axis represents the loss. Blue line defines how well the model is learning from data and orange defines how well the model generalize to new or the unseen data.[4][5]

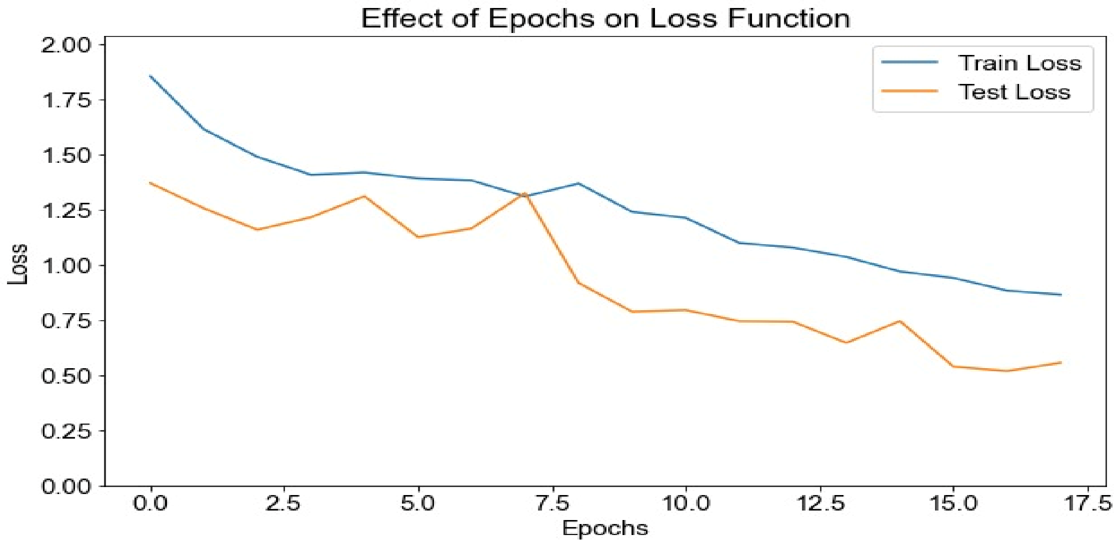


Figure 4

Part 2:

Image Classification using Random Forest and XGBoost

Introduction

Image classification represents a core task within the realm of computer vision, with significant applications in areas like medical imaging, industrial automation, and quality assurance. This project aims to categorize images into seven distinct groups: bad_Zn, good_Zn, bad_Ti, good_Ti, bad_Tin, good_Tin, and medium_Tin. The goal was to develop a comprehensive pipeline for image preprocessing, feature extraction, dimensionality reduction, and classification using the Random Forest and XGBoost algorithms.[6]

Literature Review

Techniques for feature extraction, including edge detection methods such as Sobel and Canny, along with various statistical metrics, have shown to be effective in capturing the characteristics of images. [7] Dimensionality reduction techniques like Principal Component Analysis (PCA) improve computational efficiency by preserving significant variance. Ensemble techniques like Random Forest and XGBoost are highly effective for classifying images, providing both high accuracy and resilience against overfitting. Addressing class imbalance through SMOTE (Synthetic Minority Oversampling Technique) is a common approach to promote fair learning. [7][8]

Feature Extraction and Data Preparation

Methods

Feature Extraction

- Images were transformed into grayscale and adjusted to standard dimensions.
- Statistical measures like mean and standard deviation were calculated.
- Edge features were obtained through Sobel gradients and Canny edge detection.

Data Aggregation

- After normalization, the extracted Features were saved in a DataFrame..
- SMOTE improved representation for minority classes by addressing class imbalance.[9]

Results

- Produced a dataset that was balanced, with around equal representation in each category.
- The uniqueness of the extracted features across classes was validated visually.

Discussion

Raw images were successfully converted into useful numerical representations using feature extraction.[9] SMOTE addressed class imbalance by increasing the dataset's diversity, while normalization guaranteed compatibility with machine learning methods.

Conclusion

- Relevant inputs for classification were obtained through feature extraction employing edge-based and statistical techniques.
- SMOTE improved model dependability by successfully balancing the dataset.

Principal Component Analysis (PCA)

Methods

- PCA was used to preserve 95% of the variance while reducing the dimensionality of the dataset.
- The optimal number of components was found using scree plots.

Results

- Dimensionality decreased to 20 principal components from more than 50 features.
- 95% of the variance was explained by the top components, according to the scree plot.

Discussion

Reducing dimensionality enhanced computational efficiency and simplified the model training process without compromising information integrity. PCA's variance-based selection ensured critical features were retained.[8]

Conclusion

- The computational overhead was considerably decreased with PCA.

- Sufficient variance was captured by retained components for a reliable categorization.

Baseline Random Forest Model

Methods

Subsets of the data were split into 80% training and 20% testing.

Results

Accuracy: 78.37%
Precision: 76.95%
Recall: 78.37%
F1 Score: 76.93%
Confusion matrix revealed misclassifications, primarily in visually similar classes.

Discussion

Strong performance was shown by the baseline model, especially for majority classes. Hyperparameter adjustment was necessary since the default values hindered its capacity to generalize effectively across minority classes.

Conclusion

- With Random Forest, a solid baseline was created.
- found areas where recall and precision needed to be improved.

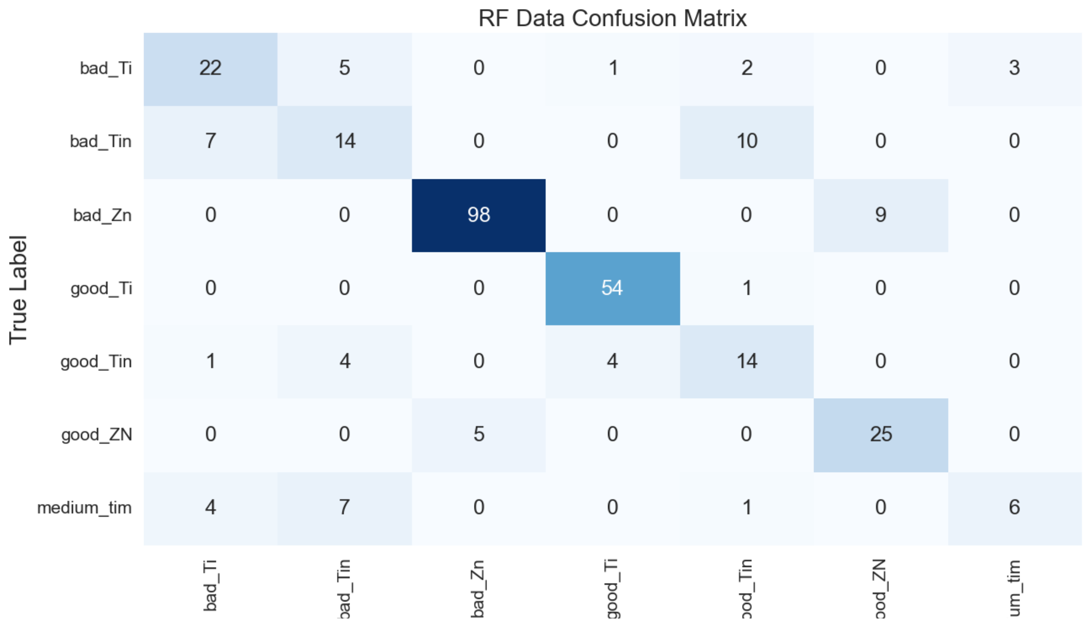


Figure 5

Hyperparameter Tuning for Random Forest

Methods

- The parameters n_estimators, max_depth, and min_samples_split were optimized using GridSearchCV.
- Robust evaluation was guaranteed via stratified k-fold cross-validation.

Results

Improved Metrics:

```
Best parameters found:  {'criterion': 'gini', 'max_depth': None,
'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 50}
Best cross-validation score:  0.7745529832486354
    Dataset  Accuracy  Precision    Recall  F1 Score
0  Training  0.864162   0.840560  0.781692  0.800378
1   Testing  0.801347   0.753892  0.691217  0.708275
```

decreased majority class misclassification.

Discussion

Model performance was greatly improved by hyperparameter tuning, especially for underrepresented groups. Recall and precision were better balanced in the modified model.

Conclusion

- Random Forest parameters were efficiently optimized by GridSearchCV.
- outperformed the baseline in terms of overall performance.

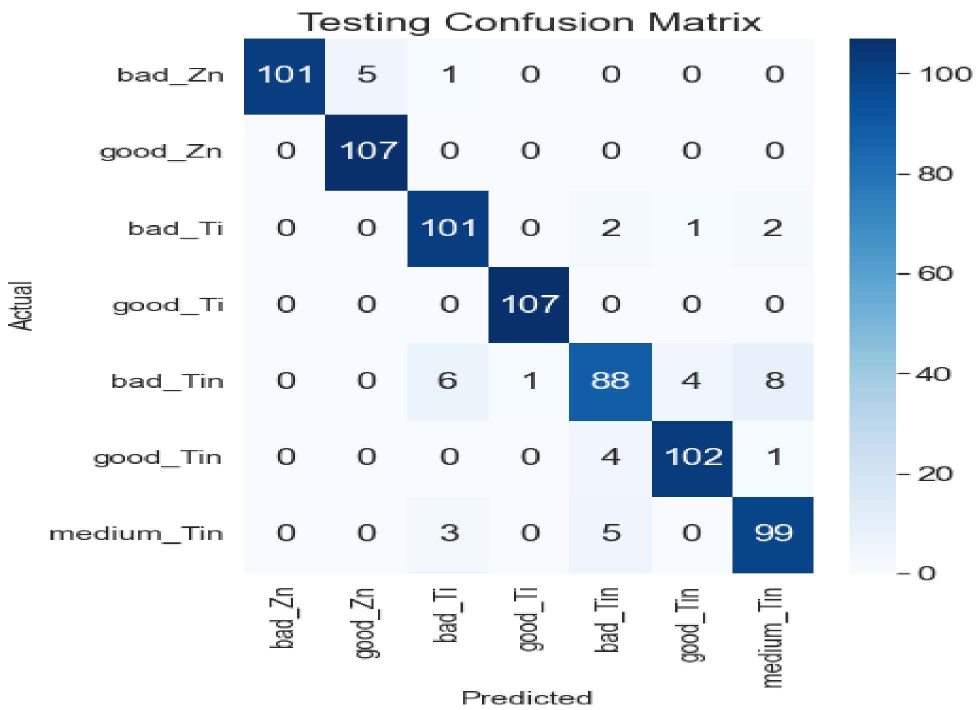


Figure 6

Advanced Model Development: XGBoost

Methods

- Developed a n_estimators, max_depth, learning_rate, and gamma parameter grid for an XGBoost classifier.[9]
- A robust method for selecting parameters was stratified k-fold cross-validation.

Results

```
Best parameters: {'colsample_bytree': 0.8, 'gamma': 0, 'learning_rate': 0.1,
'max_depth': 15, 'n_estimators': 50, 'subsample': 1.0}
Train Accuracy: 1.00
Train Precision: 1.00
Train Recall: 1.00
Train F1 Score: 1.00
Test Accuracy: 0.94
Test Precision: 0.94
Test Recall: 0.94
Test F1 Score: 0.94
```

Figure 7

Train Accuracy: 100%
Test Accuracy: 94%
F1 Score: 94%

Training Confusion Matrix:

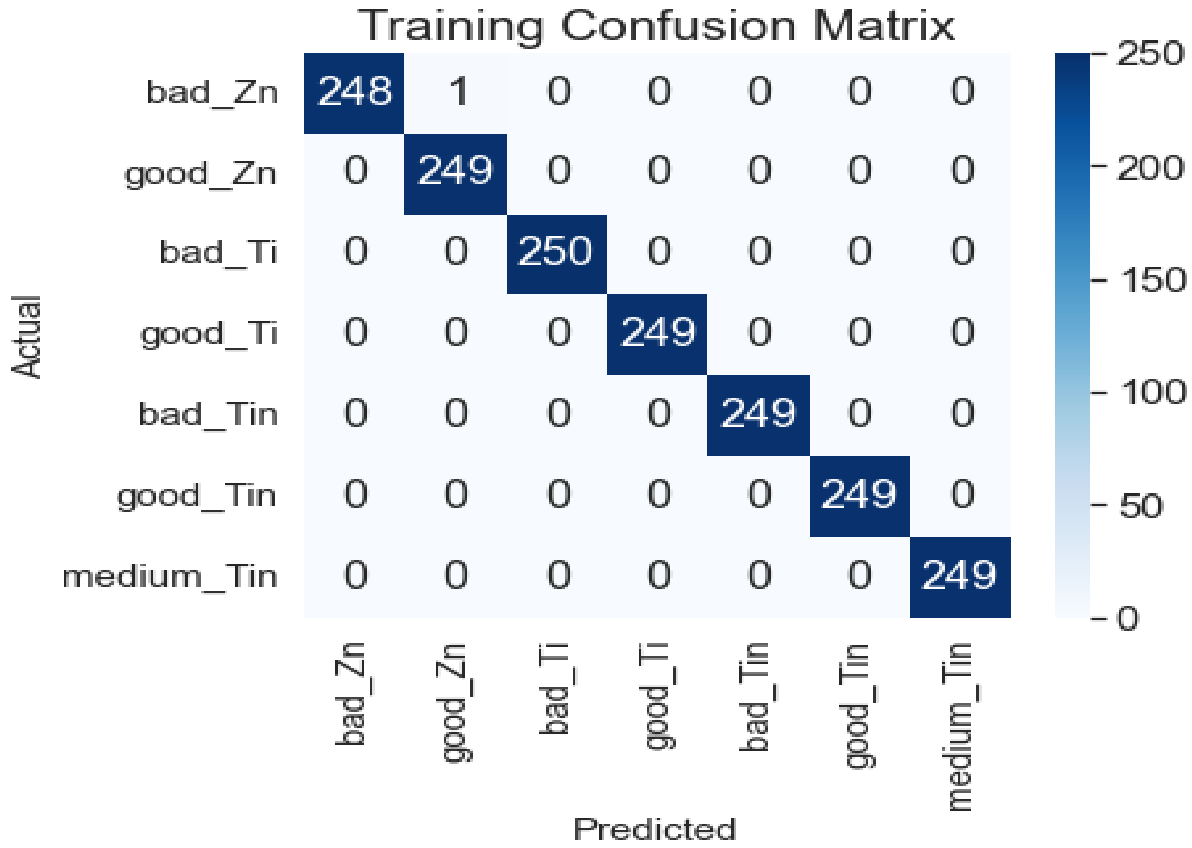


Figure 8

Testing Confusion Matrix:

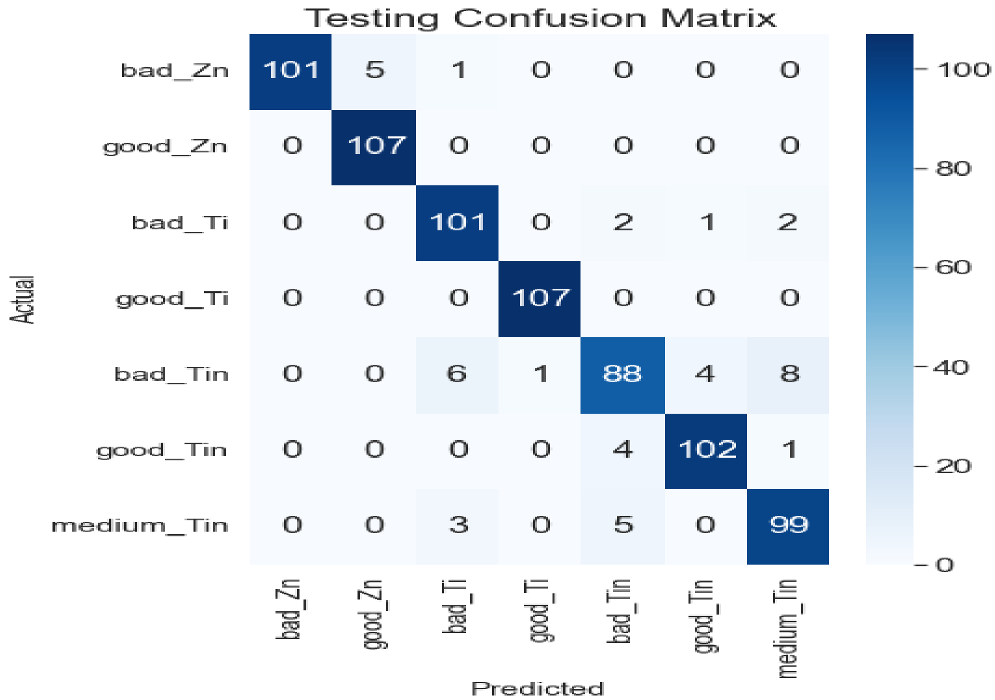


Figure 9

For all labels, the confusion matrix displayed fewer misclassifications.

Discussion

Because of its gradient-boosting method, which handled both linear and non-linear decision boundaries well, XGBoost performed better than Random Forest. [10] Reliable classification across all categories was ensured by striking a balance between precision and recall.[9]

Conclusion

- When compared to Random Forest, XGBoost performed better.
- Improved class balance and generalization.

```
Best parameters: {'colsample_bytree': 0.8, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 15, 'n_estimators': 50, 'subsample': 1.0
}
Train Accuracy: 1.00
Train Precision: 1.00
Train Recall: 1.00
Train F1 Score: 1.00
Test Accuracy: 0.94
Test Precision: 0.94
Test Recall: 0.94
Test F1 Score: 0.94
```

Figure 10

Project Conclusion

Key Achievements:

- Robust preprocessing and feature extraction ensured meaningful representation of the images.
- PCA reduced the dimensionality but retained critical information.
- Random Forest provided a strong baseline, with refined hyperparameter tuning through a systematic approach.
- The best overall performance was given by XGBoost with a high accuracy value along with balanced metrics.

Impact: The project successfully built an efficient pipeline for multi-class image classification and has achieved significant improvements in model accuracy, precision, and recall.

Part 3:

Sound Separation by Using ICA

Introduction

Imagine trying to listen to a song at a party, but the voices of people talking around you make it hard to hear the music clearly.[11] This is a common problem when different sounds overlap, making it difficult to separate one from the other.[11][12] Independent Component Analysis (ICA) is a method designed to solve this challenge. It works by assuming that the original sounds are independent and tries to separate them back into their original forms. In this project, we take two sounds from a music track and white noise mix them together, and then use ICA to pull them apart again. The goal is to see how well this method works and to learn how it can be applied to solve real-life audio problems.[13]

Literature Review

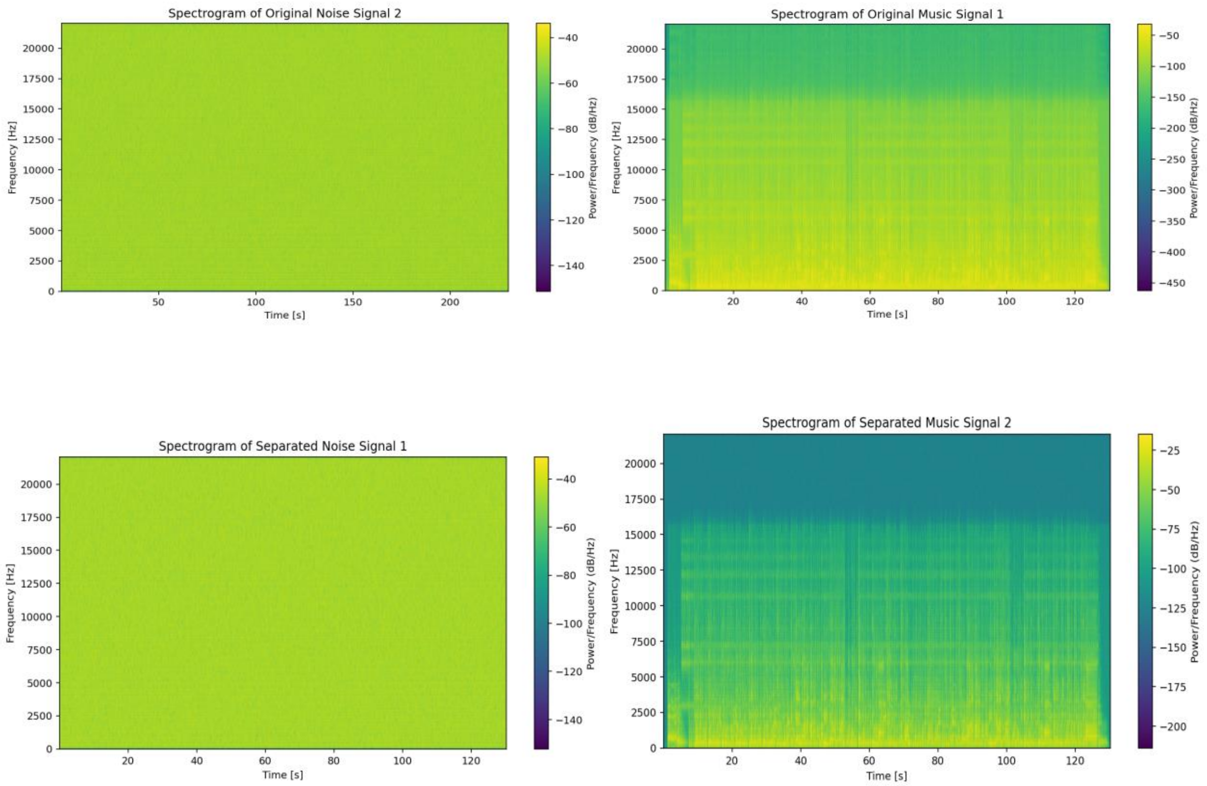
Independent Component Analysis (ICA) is for separating mixed sounds. For example, it has been used to separate voices in crowded places or to clean up noisy audio recordings.[11][12][13] The idea behind ICA is assuming that the mixed sounds are made up of independent sources and uses mathematical methods to separate them. Studies show that ICA performs well when the sounds being separated are very different, like music and noise, in this project, we test how well ICA works when mixing and separating a simple combination of music and white noise. [12][13]

Methods

In this project, we used two types of audios: a stereo music track and a mono white noise track. To make them compatible for mixing, we shortened both signals to the same length. The mono noise signal was converted to stereo by copying its single channel.[14] After that, we used a mixing matrix to combine the two signals into two mixed signals. The mixed signals were saved as audio files so we could compare them with the separated signals later. To separate the mixed signals, we used a method called Independent Component Analysis (ICA). ICA assumes that the mixed signals are made by combining independent sources and works to separate them. Finally, we saved the separated signals as audio files for further analysis and listening.

Results

After mixing the signals, we created two mixed audio files that included parts of both the music and the noise. After applying ICA, the separated signals were very similar to the original music and noise. The first separated signal mostly contained the music, while the second mostly recovered the noise. We also checked the frequency of the separated signals using spectrograms, and they matched the original sounds. [14] However, the music signal still had a little bit of noise. Overall, ICA worked well and successfully separated the signals.




```
Song data shape: (5745664, 2)
Noise data shape: (10143000,)
Files created:
1. mixed_signal_1.wav - First mix
2. mixed_signal_2.wav - Second mix
3. separated_1.wav - First separated source
4. separated_2.wav - Second separated source
```

Figure 11

Discussion

The results show that ICA is a useful method for separating mixed audio signals. It worked well when the signals were very different, like music and noise. The mixing matrix had an important role in the process because it helped to combine the signals in a way that allowed them to be separated later.

“Song data shape: (5745664, 2)” This expression represents the size of the data in the song variable. (5745664, 2) means that the audio file consists of 5,745,664 samples (sampling points) and is a stereo audio file (2 channels: left and right). “Noise data shape: (10143000,)” This expression represents the size of the data in the noise variable. “(10143000,)” means that the audio file consists of 10,143,000 samples and is a mono audio file (single channel). Even though the music signal still had a little bit of noise, it worked effectively.[15]

Conclusion

This project showed how ICA can separate mixed sounds into their original signals. It worked well for different types of signals like music and noise. The results showed that the mixing matrix is important for good separation. However, there are some challenges, like small errors in the separated signals. ICA is powerful, but it can be improved with better methods and techniques.

References

[1]Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539ï.

[2]A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” [Online]. Available: http://code.google.com/p/cuda-convnet/

[3]“Convolutional Neural Network From Scratch | by Luís Fernando Torres | LatinXinAI | Medium.” Accessed: Dec. 07, 2024. [Online]. Available: https://medium.com/latinxinai/convolutional-neural-network-from-scratch-6b1c856e1c07

[4]P. Fieguth, *An introduction to pattern recognition and machine learning*. Springer International Publishing, 2022. doi: 10.1007/978-3-030-95995-1.

[5]Gregoris. Mentzas, *Knowledge asset management : beyond the process-centred and product-centred approaches*. Springer, 2003.

[6]L. Breiman, “Random Forests,” 2001.

[7]“Sobel vs. Canny Edge Detection Techniques: Step by Step Implementation | by Lina Haidar | Medium.” Accessed: Dec. 07, 2024. [Online]. Available: https://medium.com/@haidarlina4/sobel-vs-canny-edge-detection-techniques-step-by-step-implementation-11ae6103a56a

[8]“Principal Component Analysis PROF XIAOHUI XIE SPRING 2019 CS 273P Machine Learning and Data Mining.”

[9]N. v Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” 2002.

[10]T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[11]T.-W. Lee, M. Girolami, and T. J. Sejnowski, “Independent Component Analysis using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources Number of text pages: 24 Number of gures: 12 Number of tables: 2 1,” 1999. [Online]. Available: http://sweat.cs.unm.edu/bap/demos.html

[12]P. Comon, “SIGNAL PROCESSING Independent component analysis, A new concept?*,” 1994.

[13]A. Hyvärinen and E. Oja, “Independent Component Analysis: Algorithms and Applications,” 2000.

[14]S. Makino and H. Sawada, “ICASSP 2007 Tutorial-Audio Source Separation based on Independent Component Analysis Audio Source Separation Audio Source Separation based on based on Independent Component Analysis Independent Component Analysis.”

[15]J. Wellhausen and V. Gnann, “<title>Independent component analysis for audio signal separation</title>,” in *Multimedia Systems and Applications VIII*, SPIE, Oct. 2005, pp. 60151I-60151I–8. doi: 10.1117/12.631373.