



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMA
ZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

**CORSO DI LAUREA MAGISTRALE IN
ICT FOR INTERNET AND MULTIMEDIA**

**“Automated Classification of
Irregularities in Magnetic Audio Tapes
Using Various Machine Learning
Techniques
”**

Relatore: Prof. / Dott Sergia Canazza Targon

Laureando/a: Mehmet Ozturk

Correlatore: Prof./Dott Matteo Spanio , Alessandro Russo

ANNO ACCADEMICO 2023. – 2024..

Data di laurea 15/10/2024



UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

MASTER THESIS IN ICT FOR INTERNET AND MULTIMEDIA

AUTOMATED CLASSIFICATION OF IRREGULARITIES IN MAGNETIC AUDIO TAPES USING VARIOUS MACHINE LEARNING TECHNIQUES

SUPERVISOR

PROF. SERGIO CANAZZA TARGON
UNIVERSITY OF PADOVA

CO-SUPERVISOR

MATTEO SPANIO
ALESSANDRO RUSSO

MASTER CANDIDATE

MEHMET OZTURK

STUDENT ID

2049527

ACADEMIC YEAR

2021-2024

TO MY FIANCÉE, IREMSU SAVAS, WHOSE LOVE, SUPPORT, AND UNDERSTANDING HAVE BEEN MY GUIDING LIGHT THROUGHOUT THIS JOURNEY. THIS THESIS IS AS MUCH YOURS AS IT IS MINE.

I WOULD LIKE TO EXPRESS MY SINCERE GRATITUDE TO MY PROFESSOR, SERGIO CANAZZA TARGON, FOR HIS INVALUABLE GUIDANCE AND EXPERTISE THROUGHOUT THIS PROJECT. MY HEARTFELT THANKS ALSO GO TO MY SUPERVISORS, ALESSANDRO RUSSO AND MATTEO SPANIO, FOR THEIR UNWAVERING SUPPORT AND INSIGHTFUL FEEDBACK. I AM DEEPLY APPRECIATIVE OF ZAFER CINAR FOR HIS COLLABORATION AND CONSTANT ENCOURAGEMENT. YOUR CONTRIBUTIONS HAVE BEEN CRUCIAL TO THE COMPLETION OF THIS THESIS.

Abstract

This thesis presents a comprehensive approach to the classification of magnetic tape irregularities using deep learning techniques, focusing on various image preprocessing methods to enhance model performance. The primary objective was to evaluate the effectiveness of different datasets, including thresholded images, difference images, opened images, and Region of Interest (ROI) images, in identifying specific types of irregularities. The ResNet50 architecture was employed as the core model due to its proven efficacy in image classification tasks.

Through extensive experimentation, the ROI images dataset emerged as the most effective approach, yielding the highest test accuracy of approximately 98.37%. This superior performance can be attributed to the model's ability to focus on the most pertinent regions of the images, thereby enhancing the feature extraction process and minimizing the influence of irrelevant background noise. In contrast, other datasets such as thresholded and difference images showed a decline in performance, highlighting the importance of careful preprocessing in achieving accurate classification.

Challenges such as data imbalance and model overfitting were addressed through techniques like data augmentation and careful model fine-tuning. However, some limitations persisted, including the need for further optimization and the exploration of additional data representations. The study proposes future work directions, including the use of Generative Adversarial Networks (GANs) for synthetic data generation, advanced augmentation methods, and ensemble learning to further improve classification accuracy and robustness.

In conclusion, this research demonstrates the critical role of targeted image preprocessing in enhancing deep learning models for specific classification tasks. The insights gained from this study provide a foundation for future advancements in the automated detection and classification of magnetic tape irregularities, with potential applications in media preservation and quality control.

Contents

ABSTRACT	v
LIST OF FIGURES	xi
LIST OF TABLES	xiii
1 INTRODUCTION	3
1.1 Image Classification	4
1.2 Challenges in Image Classification and Their Solutions	5
1.3 Audio Preservation and Digitization	6
1.4 Centro di Sonologia Computazionale	7
1.5 MPAI-CAE ARP	8
1.5.1 Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI)	8
1.5.2 Context-based Audio Enhancement (CAE)	9
1.5.3 Audio Recording Preservation (ARP) Use Case	10
1.6 Classification of Irregularities	12
1.6.1 Irregularity Types	13
2 TAPE IRREGULARITY CLASSIFICATION DATASET	15
2.1 Acquisition of Dataset (Video Analyser)	15
2.1.1 ROI Detection	16
2.1.2 Detection of Irregularities	18
2.1.3 Mitigating False Positives	20
2.1.4 Handling Interlacing	20
2.2 Dataset Content	21
2.2.1 Region of Interest Images(ROI)	22
2.2.2 Difference Images	23
2.2.3 Thresholded Image	23
2.2.4 Opened Images	24
2.3 Data Augmentation Methods	25
2.3.1 Augmentation Pipeline	25
2.3.2 Implementation of Data Augmentation	26
2.3.3 Mathematical Formulation	28
2.3.4 Invalid File Handling	28

2.3.5	Conclusion	28
3	MODELS	31
3.1	Convolutional Neural Network (CNN) Classification Approach	32
3.1.1	Dataset Selection and Class Imbalance	32
3.1.2	Separate Dataset Analysis	33
3.1.3	Limitations of the Classification System	33
3.1.4	Application in Real-World Scenarios	34
3.2	Selecting the Best Model	34
3.3	yolo v7	35
3.3.1	Key Features and Architecture	35
3.3.2	Input Image Requirements	36
3.3.3	How YOLOv7 Works	36
3.3.4	Computational Requirements and Deployment	37
3.3.5	Application and Use Cases	37
3.4	Classification using YOLOv7	38
3.4.1	Preprocessing the Images	38
3.4.2	Splitting the Dataset	39
3.4.3	Generating Dummy Annotations	40
3.4.4	Training the YOLOv7 Model	40
3.5	Results of YOLOv7 Training and Performance Analysis	41
3.5.1	Classification Loss on Training Set	41
3.5.2	Objectness Loss on Training Set	42
3.5.3	Box Loss on Validation Set	42
3.5.4	Classification Loss on Validation Set	43
3.5.5	Mean Average Precision (mAP) @ 0.5	43
3.5.6	Mean Average Precision (mAP) @ 0.5:0.95	44
3.5.7	Precision	44
3.5.8	Recall	45
3.5.9	Box Loss on Training Set	45
3.5.10	Overall Analysis	45
3.6	VGG16 Model for Image Classification	46
3.6.1	Preprocessing and Data Augmentation	46
3.6.2	Modifying the VGG16 Architecture	47
3.6.3	Training Process	48
3.6.4	VGG16 Model Results	50
3.6.5	Confusion Matrix	50
3.6.6	Classification Report	51
3.6.7	Conclusion	52
3.7	EfficientNet	52
3.7.1	EfficientNet Architecture	52

3.7.2	EfficientNet Variants	53
3.7.3	EfficientNet for Image Classification	53
3.8	Modified VGG16 for Image Classification	54
3.8.1	Data Preprocessing	54
3.8.2	Label Encoding	55
3.8.3	Data Splitting and Shuffling	56
3.8.4	Modifying the VGG16 Architecture	56
3.8.5	Training the Model	57
3.9	EfficientNet Results and Overfitting Analysis	58
3.9.1	Test Loss and Accuracy	58
3.9.2	Training and Validation Curves	59
3.9.3	Precision and Recall	59
3.9.4	Conclusion	60
3.10	ResNet50 Model and Final Modifications	61
3.10.1	ResNet50 Overview	61
3.10.2	Modifications and Fine-Tuning for the Final Model	61
3.10.3	Training Process	62
3.11	ResNet50 Model Results	63
3.11.1	Confusion Matrix	63
3.11.2	Precision, Recall, and F1-Score	64
3.11.3	Conclusion	65
4	RESULTS	67
4.1	Results	67
4.1.1	Dataset Variations	67
4.1.2	Model Training on Different Datasets	68
4.2	ResNet50 Performance on Thresholded Images	68
4.2.1	Accuracy and Loss	69
4.2.2	Training and Validation Loss	70
4.2.3	Confusion Matrix and Classification Report	71
4.2.4	Discussion	72
4.2.5	ResNet50 Performance on Difference Images	73
4.2.6	Accuracy and Loss	73
4.2.7	Confusion Matrix and Classification Report	74
4.2.8	Discussion	76
4.3	ResNet50 Performance on Opened Images	76
4.3.1	Accuracy and Loss	76
4.3.2	Test Loss and Accuracy	77
4.3.3	Confusion Matrix and Classification Report	78
4.3.4	Discussion	80
4.4	ResNet50 Performance on ROI Images	80

4.4.1	Accuracy and Loss	80
4.4.2	Confusion Matrix and Classification Report	82
4.4.3	Discussion	84
4.5	Conclusion	84
5	CONCLUSION	87
5.1	Conclusion and Future Work	87
5.1.1	Key Findings	87
5.1.2	Challenges and Limitations	88
5.1.3	Future Work	88
	REFERENCES	91

Listing of figures

1.1	MPAI-AIF V ₂ Reference Model	9
1.2	MPAI-CAE ARP Reference Model [1]	11
2.1	Studer A810 open-reel tape recorder.	16
2.2	Fixed elements of interest in the frame.	17
2.3	Tape scrolling mechanism of the Studer A810.	18
2.4	ROIs under the capstan and the reading head.	19
2.5	Actual comparison image output of the tape area under the reading head.	19
2.6	. Interlaced picture: notice the misalignment between odd and even lines.	21
2.7	ROI images with various speeds.	22
2.8	Difference images with various speeds.	23
2.9	The difference image is binarized with $T = 13$	24
2.10	Opened Images.	25
3.1	CNN Architecture	32
3.2	Classification Loss on Training Set	41
3.3	Objectness Loss on Training Set	42
3.4	Box Loss on Validation Set	42
3.5	Classification Loss on Validation Set	43
3.6	Mean Average Precision (mAP) @ 0.5	43
3.7	Mean Average Precision (mAP) @ 0.5:0.95	44
3.8	Precision	44
3.9	Recall	45
3.10	Box Loss on Training Set	45
3.11	Confusion Matrix of VGG16 Model on the Test Dataset	51
3.12	Classification Report for VGG16 Model on the Test Dataset	52
3.13	Test Loss and Test Accuracy for EfficientNet	58
3.14	Training and Validation Accuracy/Loss Curves for EfficientNet	59
3.15	Confusion Matrix for ResNet50 Model	64
4.1	Training and Validation Accuracy for ResNet50 on Thresholded Images	69
4.2	Test Loss and Accuracy for ResNet50 on Thresholded Images	70
4.3	Training and Validation Loss for ResNet50 on Thresholded Images	70
4.4	Confusion Matrix for ResNet50 on Thresholded Images	71
4.5	Training and Validation Accuracy for ResNet50 on Difference Images	73

4.6	Training and Validation Loss for ResNet50 on Difference Images	74
4.7	Test Loss and Accuracy for ResNet50 on Difference Images	74
4.8	Confusion Matrix for ResNet50 on Difference Images	75
4.9	Training and Validation Accuracy for ResNet50 on Opened Images	77
4.10	Training and Validation Loss for ResNet50 on Opened Images	77
4.11	Test Loss and Accuracy for ResNet50 on Opened Images	78
4.12	Confusion Matrix for ResNet50 on Opened Images	79
4.13	Training and Validation Accuracy for ResNet50 on ROI Images	81
4.14	Training and Validation Loss for ResNet50 on ROI Images	81
4.15	Test Loss and Accuracy for ResNet50 on ROI Images	82
4.16	Confusion Matrix for ResNet50 on ROI Images	83

Listing of tables

3.1	Precision and Recall Values for EfficientNet	60
3.2	Precision, Recall, and F1-Score for ResNet50 Model	65
4.1	Precision, Recall, and F1-Score for ResNet50 on Thresholded Images	72
4.2	Precision, Recall, and F1-Score for ResNet50 on Difference Images	75
4.3	Precision, Recall, and F1-Score for ResNet50 on Opened Images	79
4.4	Precision, Recall, and F1-Score for ResNet50 on ROI Images	83

1

Introduction

Magnetic tapes have historically served as a predominant medium for the recording and preservation of audio, assuming a pivotal function in the archival of historical and cultural sound recordings. Nevertheless, the tapes' physical and chemical characteristics render them vulnerable to a range of deterioration phenomena as time progresses, including magnetic dropouts, signal attenuation, and mechanical deformation. These aforementioned challenges pose considerable obstacles in maintaining the audio quality and integrity of the recordings, hence requiring the use of sophisticated methods for detection and restoration.

The manual techniques employed for the identification and categorization of anomalies in magnetic tapes are not only demanding in terms of labor but also susceptible to inconsistencies and human fallibility, particularly in the context of extensive archiving endeavors. The emergence of artificial intelligence (AI) and machine learning (ML) presents novel opportunities for the automation of these procedures, hence augmenting the effectiveness and precision of audio preservation endeavors.

The MPAI (Moving Picture, Audio, and Data Coding by Artificial Intelligence) group has emerged as a leading force in utilizing AI technology to tackle the complexities related to the preservation of magnetic tape. MPAI-AIF (AI Framework) is a prominent initiative that offers a comprehensive set of standards and guidelines for the implementation of artificial intelligence (AI) across several areas, encompassing audio processing among others. The MPAI-AIF framework is highly pertinent to this thesis, as it delineates the utilization of artificial intelligence to automate the categorization of anomalies in magnetic audio files, hence enhancing the depend-

ability of restoration and digitalization procedures.

This thesis extends the previously established principles of the MPAI-AIF framework in order to construct and assess machine learning models capable of autonomously categorizing and detecting various forms of anomalies in magnetic audio tapes. The primary objective of this project is to provide a resilient and expandable solution that improves the preservation quality of preserved audio materials through the application of diverse machine learning algorithms. This study not only makes a valuable contribution to the domain of digital archiving but also corresponds with the overarching objectives of the MPAI community to promote the utilization of artificial intelligence in enhancing the preservation and accessibility of multimedia digital content.

The conclusions of this study are anticipated to have substantial ramifications for archives, libraries, and media institutions that depend on magnetic tapes as a means of safeguarding historical and cultural audio material. The proposed system seeks to enhance process efficiency by automating the identification and categorization of tape anomalies. This approach attempts to minimize the time and effort needed for restoration while simultaneously upholding the utmost audio quality.

1.1 IMAGE CLASSIFICATION

Image classification is a fundamental task in computer vision that involves assigning a label or category to an image based on its visual content. This process is typically achieved using machine learning algorithms, particularly deep learning models like Convolutional Neural Networks (CNNs), which are trained to recognize patterns and features within images. Image classification plays a critical role across various industries due to its ability to automate and enhance processes.

In healthcare, for example, image classification is revolutionizing medical diagnostics. By analyzing medical images such as X-rays and MRIs, AI models can detect diseases at an early stage, improving the accuracy of diagnoses and enabling timely treatment interventions. This reduces the dependency on manual interpretation and minimizes the risk of human error [2].

In the field of security and surveillance, image classification enhances safety measures by automating the identification of individuals and monitoring activities through facial recognition systems. These systems are now capable of analyzing vast amounts of video data in real-time, identifying potential threats, and ensuring a swift response. This automation is particularly beneficial in large public spaces, where manual monitoring would be impractical and less reli-

able [3].

The retail and e-commerce sectors also benefit significantly from image classification, particularly in managing large inventories and improving customer experiences. Automated product categorization and visual search engines allow retailers to efficiently organize products and provide accurate recommendations to customers. This technology not only improves operational efficiency but also drives sales by enhancing the relevance and personalization of search results [4].

In autonomous vehicles, image classification is indispensable for safe navigation. The technology is used to identify and classify objects on the road, such as pedestrians, vehicles, and traffic signs. This capability is crucial for making real-time decisions, reducing the risk of accidents, and ensuring the safe operation of self-driving cars [5].

In agriculture, image classification assists farmers by analyzing aerial images to monitor crop health. This allows for the early detection of diseases or nutritional deficiencies, enabling timely interventions that can protect crops and optimize yields. The scalability of image classification makes it a powerful tool in precision agriculture, where large areas of farmland can be monitored efficiently [6].

Overall, image classification enhances accuracy, efficiency, and decision-making across various sectors, making it a transformative technology in the modern digital landscape [7].

1.2 CHALLENGES IN IMAGE CLASSIFICATION AND THEIR SOLUTIONS

Image classification, despite its advancements, faces several challenges that impact the performance and generalization of models. One major challenge is variability in image data, which includes differences in lighting, angles, and backgrounds. This variability can confuse models, leading to misclassification. To address this, data augmentation techniques such as rotation, scaling, and flipping are used to artificially increase the diversity of the training set, helping models generalize better [8].

Another significant challenge is the high dimensionality of image data. Images contain vast amounts of pixels, each contributing to the model's input, which can lead to overfitting where the model performs well on training data but poorly on unseen data. To combat this, techniques such as dimensionality reduction (e.g., Principal Component Analysis) and regularization methods like dropout are employed. These methods reduce the complexity of the model

and prevent it from memorizing the training data [9].

Imbalanced datasets pose another challenge, where certain classes are overrepresented while others are underrepresented. This imbalance can cause the model to be biased towards the majority class. Solutions include resampling techniques, such as oversampling the minority class or undersampling the majority class, and using class weighting during training to give more importance to the minority class [10].

Computational resource constraints also challenge image classification, especially with deep learning models that require significant processing power and memory. To address this, model optimization techniques such as quantization and pruning can be applied, which reduce the model's size and computation requirements while maintaining performance [11].

Finally, interpretability is a challenge in deep learning-based image classification. These models are often seen as "black boxes," making it difficult to understand why a particular classification was made. Explainable AI (XAI) techniques, such as saliency maps, are being developed to provide insights into the model's decision-making process, making it easier to trust and validate the model's predictions [11].

1.3 AUDIO PRESERVATION AND DIGITIZATION

Although it is possible to decelerate the deterioration of analog carriers like magnetic audio tapes, it is not possible to completely suspend it. Establishing accurate preservation procedures is crucial for maintaining the audio content. Successful preservation of the information stored in these carriers is only achievable through the process of transferring the data to newly established carriers [12]. In addition to the International Association, several organizations, including the International Federation of Library Associations and Institutions (IFLA), and the International Association of Sound and Audiovisual Archives (IASA) have developed preservation requirements.

Technical details. According to IFLA, the recommended approach for maintaining the integrity of contents on analog carriers refers to the process of converting material into digital format. However, digitalization encompasses numerous factors to consider. The IASA TC-03 standard establishes guidelines for the preservation of audio documents. The specification includes several elements such as the process of digitization, evaluation of quality, documenting of information, storage, and accessibility.

Metadata documentation is crucial in the arena of audio preservation. As per IASA TC-04 [13], content metadata is structured information that characterizes and delivers digital au-

audio object context. This information is valuable for their administration and extended-term conservation. Specific instances of metadata encompass descriptive components such as title, original author, and significant terms; technical specifications like file format, sampling rate, and duration; administrative information including preservation activities and rights management operations; and origin and historical background of the audio resource documented by provenance data. Efficient dataset documentation guarantees precise identification, accessibility, and administration of metadata of digital audio records in preservation endeavors.

1.4 CENTRO DI SONOLOGIA COMPUTAZIONALE

The Centro di Sonologia Computazionale (CSC) is a renowned research center at the University of Padova that focuses on the intersection of music and computing. In recent decades, CSC has directed its attention towards several facets of music technology, including the conservation and rehabilitation of historical audio records, particularly those stored on analog magnetic tapes [14].

The intrinsic vulnerability of magnetic cassettes to degradation renders their preservation immediate and vital. The CSC's methodology integrates many disciplines, including musicology, philology, and information engineering, to guarantee precise and thorough conservation of these culturally and historically significant audio materials. The approaches devised by CSC are implemented in global projects and encompass not just the conversion of audio information into digital format, but also the extraction and preservation of metadata to maintain its integrity concerning both the auditory material and its accompanying information [15].

The preservation of magnetic tape documents at CSC requires a meticulous methodology. The objective is to guarantee the conservation and authenticity of audio recordings. The primary difficulty in conserving analog magnetic cassettes is in correctly collecting both the audio signal and the context and metadata associated with each cassette. This encompasses the state of the physical and chemical properties of the tape, the recording environment, and any modifications applied to the magnetic carrier. CSC has designed a method that converts these cassettes into digital format while maintaining the inherent supplementary information, thereby establishing a thorough digital repository that accurately represents the original recordings as closely as possible. Achieving this will ensure that the original purpose and subtleties of the recordings are preserved intact. This procedure entails the use of sophisticated software tools to diagnose and rectify any faults that arise during the process of digitization, such as fluctuations in speed and inconsistencies in equalization [15].

1.5 MPAI-CAE ARP

A substantial portion of CSC’s approach is reflected in the MPAI’s ARP (Audio Recording Preservation) application, which was established to address the lack of a standard that incorporates software references in the preservation of audio documents. Its main objective is to convert audio documents into digital format while also ensuring that the preservation master file includes information and video footage of the tape. Furthermore, the inclusion of audio is essential to preserve the philological authenticity of such manuscripts . Audio analysis and correction are performed using artificial intelligence applications to address aberrations. This holistic approach facilitates the incorporation of information, context, and historical aspects, which are crucial for the preservation of these recordings.

1.5.1 MOVING PICTURE, AUDIO AND DATA CODING BY ARTIFICIAL INTELLIGENCE (MPAI)

Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) is an international non-profit organization that aims to develop standards for data coding using Artificial Intelligence (AI) [16]. These standards facilitate the transformation of data into formats that are compatible with a wide range of applications. The goal of MPAI is to develop a workflow of interchangeable and upgradeable AI Modules (AIMs) without altering the fundamental logic of the applications [17]. This approach provides flexibility and ensures the ongoing advancement of AI technologies.

The MPAI-AIF (AI Framework) is a standard designed to enable the creation and automation of mixed Machine Learning (ML), AI, and legacy data processing modules [17]. The first version of this standard has been adopted by IEEE as IEEE 3301-2022 [18].

One of the key features of MPAI-AIF is interchangeability. AIMs can be replaced or upgraded without changing the overall logic of the application. This allows for continuous improvement of the application. Another goal of this framework is to provide compatibility and reusability by offering standardized interfaces for data exchange between AIMs. The framework is composed of several components, including the management of AIMs, execution environment, storage, and access to data. This structure supports the integration and operation of diverse AIMs [17].

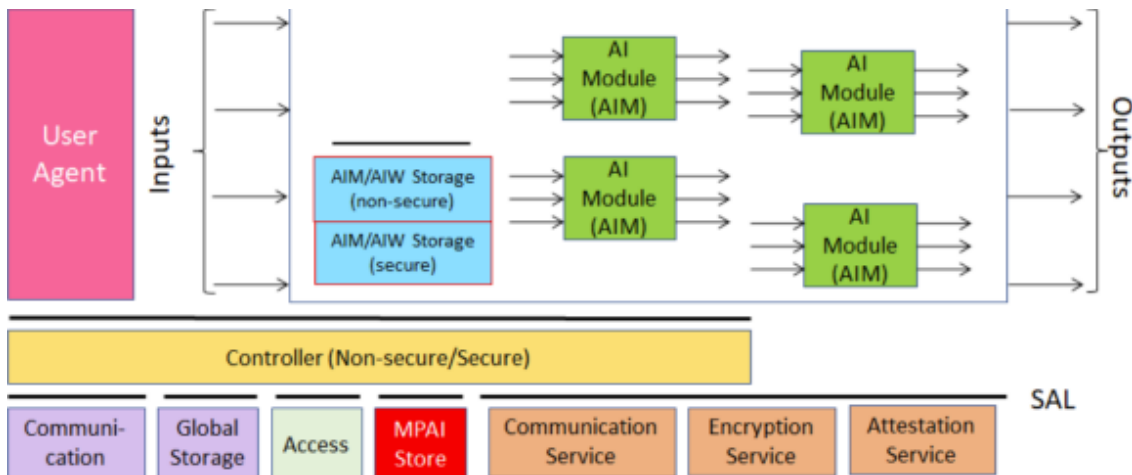


Figure 1.1: MPAI-AIF V2 Reference Model

1.5.2 CONTEXT-BASED AUDIO ENHANCEMENT (CAE)

Context-based Audio Enhancement (CAE) is a standard developed by MPAI, with its first version officially recognized as IEEE 3302-2022 [19]. The MPAI-CAE standard is designed to enhance user experiences across various audio-related applications, including entertainment, communication, and restoration [19]. The standard encompasses four specific use cases that aim to elevate audio quality in different scenarios. These use cases are outlined below:

- **Emotion-Enhanced Speech (EES):** This use case enables users to infuse neutral speech with emotional content, thereby increasing its expressiveness. By selecting an emotional tone, users can transform plain speech into a more dynamic and engaging version. This functionality is particularly beneficial in contexts where conveying emotion is vital, such as in entertainment and communication.
- **Audio Recording Preservation (ARP):** Focused on the preservation of audio recordings, particularly those stored on open-reel magnetic tapes, this use case involves creating digital versions that are optimized for long-term storage. Additionally, it ensures that these recordings can be accurately played back and, when necessary, restored to maintain their quality and content.
- **Speech Restoration System (SRS):** This use case is dedicated to enhancing the clarity of speech that has been compromised by noise or other distortions. It improves the intelligibility and overall quality of degraded speech, making it especially useful for applications such as teleconferencing and archival restoration, where clear communication is critical.

- **Enhanced Audioconference Experience (EAE):** Designed to improve the audio quality in teleconferences, this use case reduces background noise, echo, and other disruptions. By ensuring that conversations are clear and effective, it enhances the overall experience and productivity of remote meetings.

1.5.3 AUDIO RECORDING PRESERVATION (ARP) USE CASE

Audio Recording Preservation (ARP) is one of the key use cases within the MPAI-CAE standard. This use case is dedicated to preserving audio recordings stored on analog media, such as open-reel magnetic tapes. These tapes often contain critical information, including splices, irregularities resulting from the physical and chemical characteristics of the tape, and annotations made by composers or technicians. Proper documentation and preservation of this information are crucial for ensuring accurate playback and long-term preservation. Additionally, the ARP use case involves creating a digital copy for archival purposes and an access copy that may require restoration to ensure proper playback.

AI WORKFLOW (AIW) AND MODULES IN ARP

The AI workflow in ARP involves a series of structured operations carried out by different AI Modules (AIMs). These AIMs work together to detect and classify audio and visual irregularities, ultimately restoring the audio content. The process begins with digitizing the analog signal and capturing video of the playback head. The Audio and Video Analyzers then detect and classify any anomalies. The Tape Irregularity Classifier processes these classifications, and the Tape Audio Restoration module restores any damaged audio. Finally, the Packager compiles all the relevant files into preservation and access copies. Figure 1.2 illustrates the AIW of the ARP use case.

AUDIO ANALYZER

The Audio Analyzer extracts pertinent audio fragments from the tape and identifies sections with low signal levels. It detects irregularities based on recording speeds and equalization curves, producing analyzed audio blocks and irregularity files for the Tape Irregularity Classifier. The Audio Analyzer works in conjunction with the Video Analyzer. It receives irregularity files generated by the Video Analyzer and extracts audio files corresponding to the detected anomalies in these files. Finally, it combines the generated Irregularity File with the one obtained from

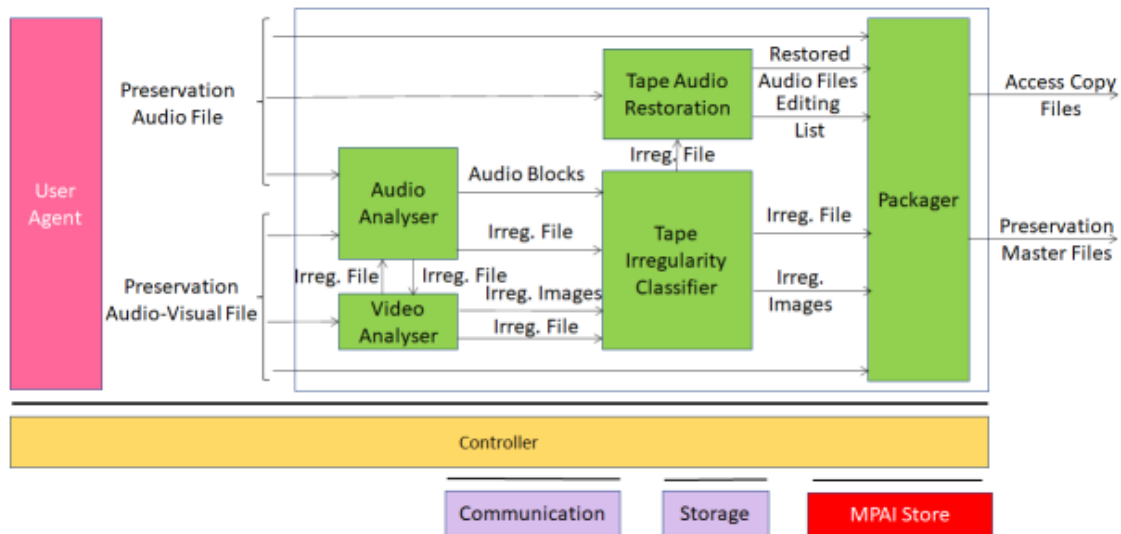


Figure 1.2: MPAI-CAE ARP Reference Model [1]

the Video Analyzer and sends it to the Tape Irregularity Classifier along with the corresponding Audio Files.

VIDEO ANALYZER

The Video Analyzer uses computer vision algorithms to detect surface irregularities on the tape. It sends the generated irregularity file to the Audio Analyzer and receives irregularity files generated by the Audio Analyzer. It extracts images of irregularities corresponding to those detected in the files sent by the Audio Analyzer. Similar to the Audio Analyzer, it combines the generated Irregularity File with the one obtained from the Audio Analyzer and sends it to the Tape Irregularity Classifier along with the appropriate irregularity images.

TAPE IRREGULARITY CLASSIFIER

The Tape Irregularity Classifier processes information provided by both the Audio and Video Analyzers, classifying irregularities for further processing and restoration. It ensures that all relevant irregularities are thoroughly documented and addressed. It then sends the Irregularity File related to the selected anomalies to the Tape Audio Restoration module.

TAPE AUDIO RESTORATION

The Tape Audio Restoration module focuses on repairing audio segments that were corrupted or incomplete during digitization. It enhances the audio quality by detecting and correcting issues related to speed, equalization, and reading errors in the Preservation Audio File. Finally, it sends the Restored Audio Files and Editing List to the Packager.

PACKAGER

The Packager is responsible for creating the Preservation Master Files and Access Copy Files. When assembling the Preservation and Access Copies, the Packager ensures that all digital content, documentation, and metadata are correctly organized and stored. It creates comprehensive Preservation Master Files and Access Copy Files by combining elements such as the Preservation Audio File, Restored Audio Files, Editing List, Irregularity File, Irregularity Images, and the Preservation Audio-Visual File [20].

1.6 CLASSIFICATION OF IRREGULARITIES

Automated irregularity classification plays a pivotal role in audio preservation, as it involves categorizing and addressing various types of anomalies found in audio carriers. These anomalies may include splices, brands, chemical deterioration, and biological contamination, among others. Such irregularities can occur due to multiple factors, such as physical damage to the recording medium, natural degradation over time, or intentional markings made by composers in tape-based music. Proper classification of these irregularities is vital to ensure that the digitization process accurately preserves the audio content, thereby safeguarding the integrity and usability of valuable historical and cultural audio archives.

Given the sheer volume of audio data requiring processing, automation in irregularity classification is indispensable. Manual classification and correction not only consume significant time but are also susceptible to human error. Automated systems, on the other hand, can efficiently and accurately classify large amounts of audio data. This capability is especially important for cultural heritage institutions, including libraries, archives, and research centers, that manage vast audio collections.

The Video Analyzer, a software module developed by CSC, exemplifies automated irregularity classification in line with the MPAI-CAE ARP use case guidelines. It is designed to classify key frames from digital video footage, concentrating on the tape recorder's reading head and

the area under the pinch roller. The software for irregularity detection utilizes frame differencing techniques, identifying and classifying significant frames by comparing consecutive frames and detecting notable changes in color [21].

1.6.1 IRREGULARITY TYPES

Within the realm of audio preservation, an irregularity refers to any variation or abnormality present on magnetic audio cassettes that has the potential to impact the quality and integrity of the recorded sound. Discrepancies can manifest as either deliberate or inadvertent. [15]

Deliberate anomalies refer to the modifications made on the tape within the framework of tape music. Tape music is a subgenre of electroacoustic music characterised by the manipulation of pre-recorded elements on magnetic tape. That is, it encompasses alterations made by composers to the recording. The alterations encompass annotations and splices, which are essential components of the creative process. Pioneering composers such as Pierre Schaeffer and Karlheinz Stockhausen employed splicing and looping methods to generate innovative musical structures and soundscapes. Annotations put directly on the tape offer precise instructions for both playing and synchronizing the audio. Their function is to provide guidance to the performers and technicians [22]. Due to the substantial information included in these modifications, it is culturally and historically essential to preserve them accurately. Incidental abnormalities refer to gradual deteriorations that occur without purposeful intervention. The abnormalities encompass physical imperfections, such as scratches, splices, and tape breaking; chemical damage caused by fluctuations in temperature and humidity; and contamination, such as the buildup of dust and grime on the tape surface resulting in playback problems [23].

The IASA Cataloguing Rules [23] provide an extensive list of conditions that can appear on an audio tape. Not every condition appears equally often in magnetic audio tapes. To make the detection and classification process easier and more feasible, a simplified approach is adopted. The simplified classification scheme used in [22] includes four primary classes:

- **Splice:** Connections where two segments of tape have been joined together with adhesive tape.
- **Shadow:** Ghosting or imprints left on the tape.
- **Brand:** Identification marks or logos printed on the tape by the manufacturer.
- **Ends-of-Tape:** The point where the tape is not under tension.

In the scope of this thesis, the “ends-of-tape” class is removed as the ends-of-tape instances are detected by a different region of interest than the other irregularities). The classes considered for this research are “splice”, “brand”, and “shadow”.

2

Tape Irregularity Classification Dataset

2.1 ACQUISITION OF DATASET (VIDEO ANALYSER)

The modules directly engaged in the analysis of video input in the MPAI-CAE ARP use case are the Video Analyser, which identifies the regions of the tape that exhibit particular points of interest and stores them as Irregularity Images, and the Tape Irregularity Classifier, which categorizes the Irregularity Images according to their content. The video analysis focuses specifically on the section of the tape located below the reading head, which also includes other common elements of the open-reel recorder (see Figure 2.2). Video files consist of recordings of open-reel tapes that are played on a Studer A810 (see Figure 2.1).[24]

Since CSC began video documenting the entire digitization process in 2013, most of the archived videos have a PAL resolution (720×576) at 25 interlaced fps, due to limitations in the available equipment. While this resolution does not affect the detection process, it must be taken into consideration during the image classification stage.

The Video Analyser employs a frame-by-frame analysis approach to identify and capture frames that exhibit significant differences compared to the preceding ones. In the next step, the Tape Irregularity Classifier focuses on these distinctive frames to detect and highlight potential anomalies, such as tape damage, scratches, splices, or other irregularities. This is accomplished using a classifier module based on a convolutional neural network, which has been carefully calibrated for this task.[24]



Figure 2.1: Studer A810 open-reel tape recorder.

The irregularity detection process underwent several iterations and refinements, ultimately leading to the following algorithm:

1. Identify the Regions of Interest.
2. Traverse through the video, comparing consecutive pairs of frames.
3. If the number of dissimilar pixels between a pair of frames exceeds a predetermined threshold, an irregularity is detected.

2.1.1 ROI DETECTION

Preliminary studies explored the possibilities of background subtraction algorithms [25]. This method utilized previously gathered information to separate new elements from recurring ones. However, this approach yielded numerous false positives, capturing insignificant variations in brightness, reel movement, and other undesired artifacts. Furthermore, there were inevitable performance issues due to the long operation time associated with the BackgroundSubtractorKNN tool. Unfortunately, improving the execution time was not feasible due to limitations imposed by the OpenCV algorithm implementation [26]. [24]

To address these challenges, a reevaluation of the Irregularities' characterization was conducted, with a focus on scene framing. It was observed that all anomalies shared a lack of vertical movement and typically consisted of small clusters of points related to the frame size. [24]

Given that Irregularities exhibited only horizontal movement, the approach was to concentrate on variations in a specific part of the frame. The most effective solution was to shift the focus away from the moving elements of the scene (i.e., the Irregularities) and select areas of interest based on stationary elements instead. In this case, the capstan and the reading head, which are the components closest to the tape, remained stationary within the frame and served as reference points for automatically identifying the pixel regions where Irregularities appear (see Fig2.2).[24]

The capstan is a rotating shaft used to move the tape through the mechanisms and magnetic heads (for erasing, recording, and playback) of the tape recorder. During playback, the tape passes through the capstan and a rubber wheel called the pinch roller. The pinch roller presses the tape against the capstan, providing the necessary friction for the tape to continue moving. Typically, the pinch roller is located after the magnetic heads in the direction of the tape's movement (on the right side of the video in this case).[24]

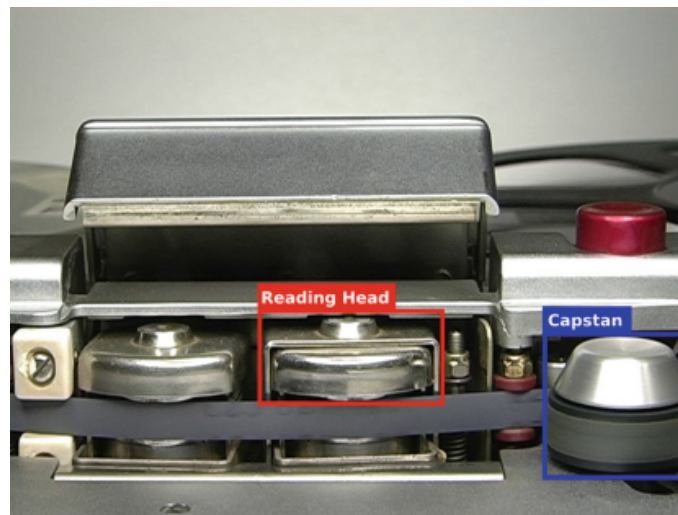


Figure 2.2: Fixed elements of interest in the frame.

To detect Irregularities, it is sufficient to examine the pixels below the reading head. Since the tape only moves horizontally, the anomalies, flowing from left to right, inevitably enter this area. The capstan can also be used for the same purpose, but in this case, it is useful for different events of interest: the start and the end of the tape. When the playback ends, the pinch roller releases the tape by moving away from the capstan, and sometimes this causes the tape to come out of the reading head's slot. The movement of the pinch roller is clearly visible in all the videos, and for this reason, it was chosen as the reference point to detect the moment when the

tape reaches its end. A similar situation occurs at the beginning of the video when the tape is not being played and the capstan is in its “rest” position (see Fig 2.3 for a visual representation of the capstan and pinch roller positioning). By focusing only on the area below the capstan, it is possible to detect when it moves to release the tension from the tape.[24]

To identify stationary elements within the scene, well-known algorithms capable of finding patterns within an image were employed: the Generalized Hough Transform [27] and SURF (Speeded Up Robust Features) [28]. Their coordinates within the image could be determined by providing these algorithms with the image of the capstan and the reading head. The coordinates of the underlying areas were defined through empirical methods. Figure 2.4 shows the identified areas below the capstan and the reading head.[24]

The actual implementation of this step involves examining the middle frame of the video, which should represent a normal situation regarding the framing or the presence of an Irregularity. During this step, the position of ROIs is determined by searching for template images in the frame using the aforementioned algorithms.[24]



Figure 2.3: Tape scrolling mechanism of the Studer A810.

2.1.2 DETECTION OF IRREGULARITIES

Once the areas of interest within the scene have been established, an activation function is employed to determine the presence of Irregularities. The number of differing pixels in each pair can be calculated By utilizing pairs of consecutive frames. Focusing on the identified ROIs, it has been observed that approximately[24] 80% of the pixels consistently exhibit variations in

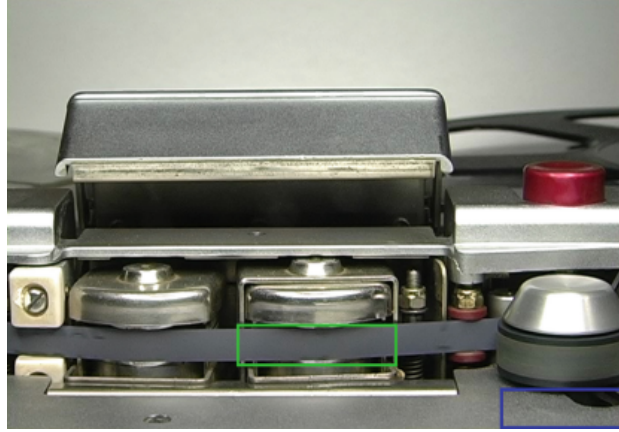


Figure 2.4: ROIs under the capstan and the reading head.

terms of colors and shadows. Therefore, it has been established that when the quantity of differing pixels exceeds the set threshold, a significant difference between the two frames has been detected. To quantify the differing pixels between two images, a new image is generated with white pixels representing the matching ones, and black ones for indicating their differences. The generation of the comparison image can be described as follows:

$$D(i,j) = \begin{cases} C_{\text{red}}(i,j) - P_{\text{red}}(i,j) = 0 \wedge \\ 255 & \text{if } C_{\text{green}}(i,j) - P_{\text{green}}(i,j) = 0 \wedge \\ C_{\text{blue}}(i,j) - P_{\text{blue}}(i,j) = 0 \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, \dots, n$ and n is the number of rows in the matrix, $j = 1, \dots, m$ and m is the number of columns in the matrix; matrix \mathbf{D} is the difference frame, \mathbf{C}_{red} , $\mathbf{C}_{\text{green}}$, and \mathbf{C}_{blue} are the current frame matrices for the red, green, and blue color channels respectively, and \mathbf{P}_{red} , $\mathbf{P}_{\text{green}}$, and \mathbf{P}_{blue} are the previous frame matrices for the red, green, and blue color channels respectively.[24]



Figure 2.5: Actual comparison image output of the tape area under the reading head.

As shown in Fig. 2.5, the generated comparison image provides a visual representation of the differences between consecutive frames in the tape area under the reading head. The white regions indicate areas where the frames match, while the black regions represent variations between the frames. This image serves as a valuable tool for identifying Irregularities and assessing the extent of their presence.[24]

2.1.3 MITIGATING FALSE POSITIVES

To ensure that all irregularities within an input tape are accurately identified, testing the accuracy of the software is crucial. The testing of the developed software showed good results, though occasional detection of false positives was also observed. If an irregularity extends over several centimeters along the tape, it may be detected multiple times, resulting in the output of several irregularity images.[24]

To address this issue, considering that the region of interest (ROI) at the tape's reading head is 3 cm wide, and the tapes run at speeds of 7.5 or 15 inches per second (ips) while videos are recorded at 25 frames per second (fps), it was calculated that discarding two or three consecutive frames upon detecting an irregularity is sufficient to avoid duplicating the same irregularity (though false positives may still occur for exceptionally long irregularities). This approach significantly reduces the number of images saved as irregularities by the software.[24]

2.1.4 HANDLING INTERLACING

Another issue related to the tape format concerns the storage of irregularity images. Since the majority of video files archived at CSC were recorded long before the software was designed and developed, the videos were recorded in PAL format at 25 fps, interlaced. Interlacing is a notable drawback of the input files, as each frame is divided into even and odd lines, resulting in misalignment between the two fields, particularly noticeable in areas with motion (see 2.4).[24] In the Irregularity detection phase, dealing with interlaced images does not introduce particular problems. Since the comparison occurs on a pixel-by-pixel basis, the analyzed frames in pairs are interlaced in the same manner, resulting in consistent misalignment of even and odd pixel rows in subsequent frames. However, interlacing can adversely affect the classification phase, which employs a convolutional neural network. In fact, this type of neural network reduces the number of pixels as it progresses deeper into the network, approximating the colors of neighboring pixels to identify characteristic patterns within an image. Unfortunately, interlacing alters the colors of neighboring pixels and significantly impairs the classifier's performance.[24]

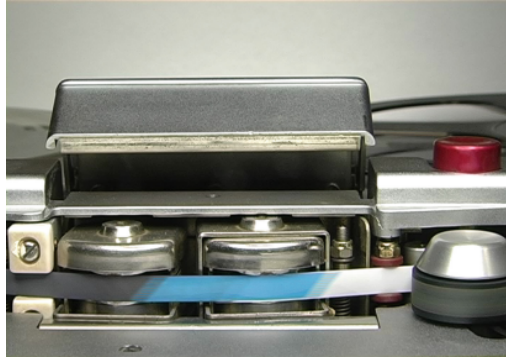


Figure 2.6: . Interlaced picture: notice the misalignment between odd and even lines.

Various techniques to reduce the effects of interlacing have been explored and tested. One widely used technique in dedicated playback software involves separating the semi-quads within the analyzed frame. This process generates two images: one containing the even lines and the other containing the odd lines. By utilizing only one of these semi-quads, half of the original frame's information is lost, but the other half image gives more reliable results for this specific use case. Thus, it has been chosen to save only the semi-quad containing the odd lines, resulting in images with a resolution of 720×228 pixels. This approach mitigated the misalignment caused by interlacing and provided a more accurate representation of the tape's content.[24]

2.2 DATASET CONTENT

Applying the frame differencing technique, the pictures are classified into four sets, each corresponding to a distinct phase of the irregularity detection process. The rationale behind the organization of these divisions arises from the observation that each stage exhibits different levels of complexity. The goal is to determine the group that provides the most valuable and comprehensive depiction of the anomaly. The sets are list below:

1. **Region of Interest Images:** Images of ROI directly taken from the video without any processing step.
2. **Difference Images:** The image that represents the absolute difference between two frames at the given time.
3. **Motion Images:** Binarized images obtained by applying a threshold on difference images.
4. **Final Images:** The images after the opening operation on motion images.

Each set consists of three subgroups created following the criteria provided by CSC. The irregularity photos are classified into three distinct groups: splices, shadows, and brands. It is crucial to emphasize that each picture in one of these four sets is equivalent to its matching image in another group, as they are outcomes of the same process. Hence, four groups possess an equal quantity of photos that depict identical anomalies. The rationale is to achieve more dependable and consistently valid empirical findings. Analogous to the procedure of detecting irregularities, the dataset is created by utilizing films with speed settings of 3.75 ips, 7.5 ips, 15 ips, and 30 ips. The inclusion of 30 ips is essential since an increase in tape speed poses additional difficulties for irregularity categorization because of the heightened motion blur. In further versions, the dataset should be augmented by acquiring photos with pixel values of 1.875 ips and 0.9375 ips inherent in the ARP standard.[1]

2.2.1 REGION OF INTEREST IMAGES(ROI)

First set of images consists of direct images taken from the tape area of the video. These images represent the second image of the consecutive pairs of frames from the detection process.(see 2.7)

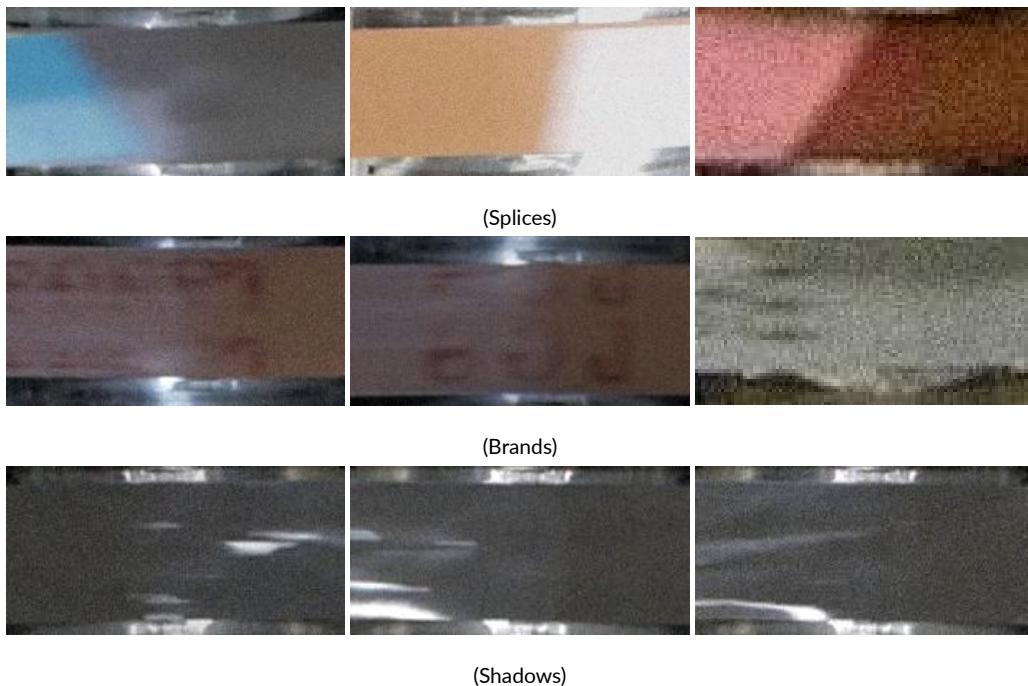


Figure 2.7: ROI images with various speeds.

2.2.2 DIFFERENCE IMAGES

These images depict the divergence between two successive frames at the moment the anomaly was identified. These images are acquired by calculating the absolute difference between the pixel brightness of two successive photos, as explained before in Section. Given the relatively low pixel intensities of these difference images, they primarily seem black to the human eye. Nevertheless, these pictures may include significant elements that may be recognized by machine vision. Figure 2.8 displays many photos belonging to this group. Images are acquired by deriving the absolute difference for the anomalies shown in Figure 2.7. Evidently, brand pictures predominantly appear in black, which may be ascribed to their generally low intensities.

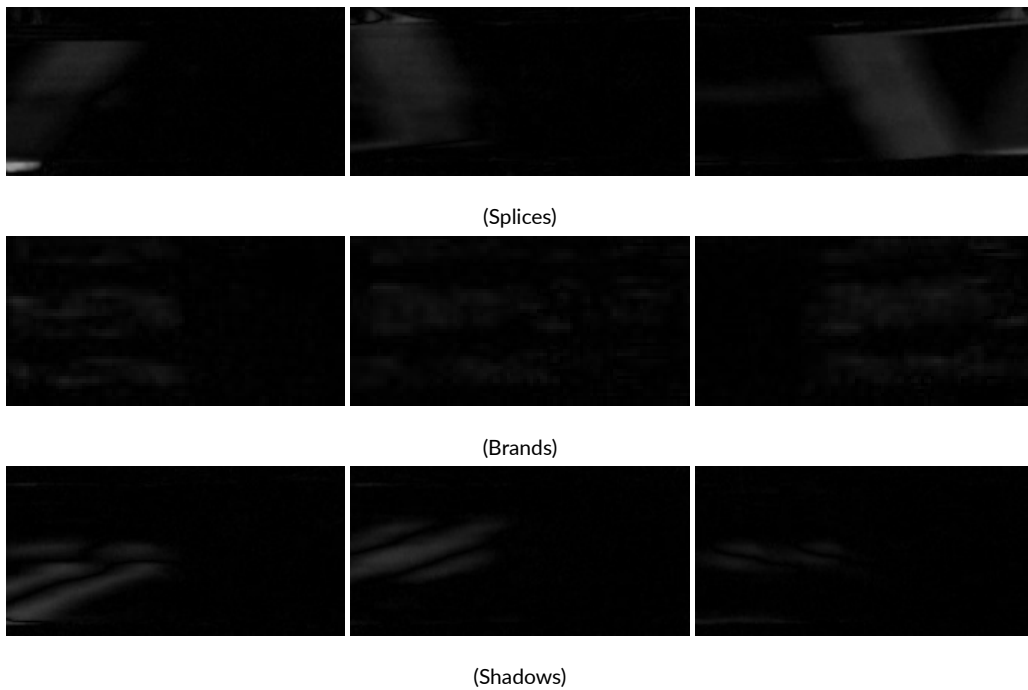


Figure 2.8: Difference images with various speeds.

2.2.3 THRESHOLDED IMAGE

Thresholding is the procedure of producing a binary image by applying a prescribed threshold value T . Utilizing the threshold value acquired in the preceding stage, a binary image known as a motion image is generated. Every pixel with a value equal to or exceeding T is allocated

an intensity value of 255, indicating a white pixel. Every individual pixel located beneath the threshold T is allocated an intensity value of 0, indicating a black pixel. The threshold operation, calculated based on the threshold value T , may be defined as:

$$M(x, y) = \begin{cases} 255 & \text{if } \Delta(x, y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

The thresholding was done with the value obtained with Otsu's method. In Fig 2.9, the result-

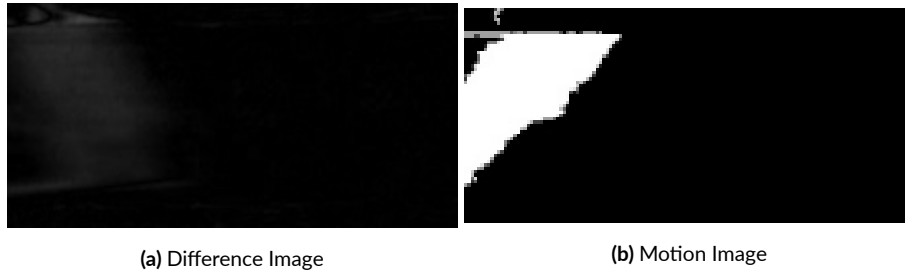


Figure 2.9: The difference image is binarized with $T = 13$.

ing motion image is shown. The thresholding was done with the value obtained with Otsu's method.

2.2.4 OPENED IMAGES

After obtaining the motion image, some white pixels may appear that are not part of the main irregularity. These unrelated motion artifacts are often caused by system vibration or external lighting changes. To give the irregularity a clearer shape and reduce the influence of these white pixels during evaluation, the **opening** operation is applied.

This operation consists of two steps:

1. **Erosion**: Eliminates the unrelated artifacts.
2. **Dilation**: Recovers the size and shape of the main irregularity.

Let M represent the binary motion image, O the binary final image, and S the structuring element used in the opening operation. The opening of M by S is denoted as $M \circ S$ and is defined as:

$$O(x, y) = (M \circ S)(x, y) = (M \ominus S) \oplus S$$

where \ominus represents erosion and \oplus represents dilation.

For video footage with resolutions of 1920×1080 and 720×576 , as used by CSC, a 3×3 kernel is applied as the structuring element during the opening operation. (see fig 2.10)

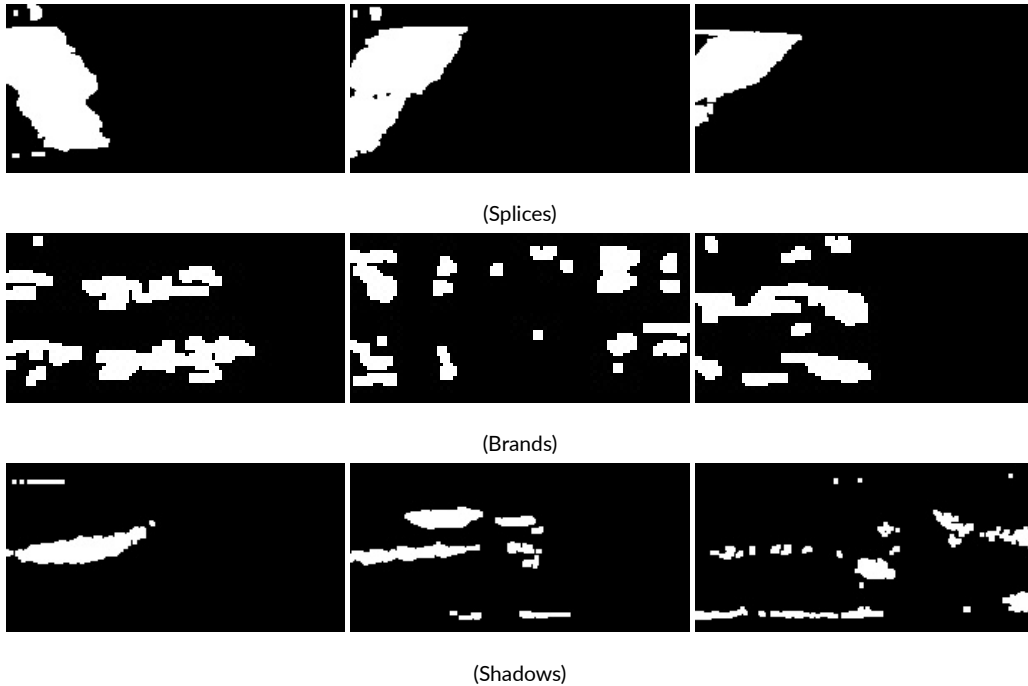


Figure 2.10: Opened Images.

2.3 DATA AUGMENTATION METHODS

Data augmentation is a crucial technique to enhance the performance of deep learning models by artificially expanding the dataset. By applying transformations such as flipping, adding noise, and modifying brightness, we can generate new training samples, thus helping to prevent overfitting and making the model more generalizable. In this section, we describe the augmentation pipeline applied to the dataset used in our training process.

2.3.1 AUGMENTATION PIPELINE

The augmentation techniques used in our pipeline are designed to generate diverse samples of the input images. Below is a description of each augmentation step applied:

- **Horizontal Flip (Fliplr):** Randomly flips the image horizontally with a probability of 0.5. This can be expressed as:

$$I'(x, y) = I(w - x, y)$$

where $I(x, y)$ represents the original image and w is the image width.

- **Vertical Flip (Flipud):** Randomly flips the image vertically with a probability of 0.2, which can be expressed as:

$$I'(x, y) = I(x, b - y)$$

where b is the height of the image.

- **Additive Gaussian Noise:** Adds Gaussian noise to the image, where the noise is sampled from a normal distribution:

$$I'(x, y) = I(x, y) + \mathcal{N}(0, \sigma^2)$$

with $\mathcal{N}(0, \sigma^2)$ being a Gaussian noise value with zero mean and variance σ^2 . In the code, the noise is scaled by $\sigma = 0.05 \times 255$, which allows slight perturbations to the pixel values.

- **Brightness Adjustment (Multiply):** Modifies the brightness of the image by multiplying pixel values by a random factor between 0.8 and 1.2. Mathematically, this can be written as:

$$I'(x, y) = I(x, y) \times \alpha$$

where $\alpha \in [0.8, 1.2]$ is a randomly selected factor.

These transformations are combined in a pipeline using the ‘imgaug’ library’s ‘Sequential’ function, which applies each operation sequentially to the images. The pipeline is structured to ensure a diverse set of augmentations.

2.3.2 IMPLEMENTATION OF DATA AUGMENTATION

The following Python code snippet, implemented in our study, demonstrates the process of augmenting the dataset. The augmentation pipeline is applied to different image subfolders (ROI, thresholded, difference, and opened images) and various categories (splice, shadow, and

brand). The goal was to generate a sufficient number of augmented images per class, reaching a target of 1500 images for each.

Listing 2.1: Python code for Data Augmentation

```
import os
import numpy as np
import random
import imgaug.augmenters as iaa
from PIL import Image, UnidentifiedImageError

\% Define augmentation pipeline
augmentation_pipeline = iaa.Sequential([
    iaa.Fliplr(0.5), # Horizontal flip
    iaa.Flipud(0.2), # Vertical flip
    iaa.AdditiveGaussianNoise(scale=(0, 0.05*255)), \%
        Gaussian noise
    iaa.Multiply((0.8, 1.2)) # Brightness adjustment
])

# Load images and apply augmentation
for main_folder in main_folders:
    for sub_folder in sub_folders:
        # Loading and augmenting images logic...
```

In this code, images from multiple folders are augmented using the pipeline. The following steps are performed:

1. ****Load the Images****: Images are loaded from subfolders within the main categories.
2. ****Apply Augmentation****: The augmentation pipeline is applied to the loaded images.
3. ****Generate Target Number of Images****: The pipeline continues augmenting the images until the target of 1500 images per class is reached.
4. ****Save Augmented Images****: Augmented images are saved in new subdirectories, preserving the original filenames with an added prefix.

2.3.3 MATHEMATICAL FORMULATION

The data augmentation process can be mathematically represented as applying a set of transformations T to the original image I . The transformation function T is applied as:

$$I'(x, y) = T(I(x, y))$$

where T is a composition of the operations described above, including horizontal flip, vertical flip, noise addition, and brightness adjustment. Each transformation introduces controlled variability into the dataset while preserving the underlying content of the image, enhancing the model's ability to generalize.

2.3.4 INVALID FILE HANDLING

During the data augmentation process, there were some invalid files that could not be processed due to format or loading issues. These files were skipped, and the number of invalid files per folder was tracked and printed at the end of the augmentation process.

Listing 2.2: Python code for handling invalid files

```
# Invalid file handling
invalid_files_count[sub_path] = 0 # Initialize invalid
file count
for file in os.listdir(sub_path):
    try:
        with Image.open(image_path) as image:
            images.append(np.array(image)) # Load
            image
        except (UnidentifiedImageError, OSError):
            print(f"Invalid file: {image_path}. Skipping.")
            invalid_files_count[sub_path] += 1 # Track
            invalid files
```

2.3.5 CONCLUSION

Data augmentation is a powerful technique to artificially expand a dataset, especially when dealing with limited data or class imbalance. In our project, the augmentation process helped

generate a sufficient number of diverse training samples, leading to better model performance and generalization. The augmentation methods employed, such as flipping, noise addition, and brightness adjustment, ensured that the model could effectively handle various real-world image variations during training.

3

Models

EXISTING METHOD

Following the image acquisition process, a Convolutional Neural Network (CNN) based on the GoogLeNet network [29] is employed to classify the acquired images. The CNN has undergone training on specific classes, namely splices, brands, shadows, and end of tape. While additional types of irregularities do exist, their inclusion in the classifier's training set was not feasible due to their significantly lower frequency of recurrence, especially compared to the aforementioned classes. Such inclusion would have led to an imbalanced dataset, undermining the effectiveness of the classification model. Therefore, the number of classes was reduced, ensuring a more balanced input for the classifier [30].

Separate analyses were conducted on two different datasets: the first one with tapes played at 7.5 inches per second (ips) and the other for tapes played at 15 ips. Both classifiers exhibited an accuracy of 95% during the validation phase, a noteworthy achievement. This high accuracy was further confirmed during the subsequent testing phase, which involved evaluating the performance of the classifiers on frames extracted from the dataset prior to the network's training. This comprehensive evaluation solidifies the robustness and reliability of the developed system.

3.1 CONVOLUTIONAL NEURAL NETWORK (CNN) CLASSIFICATION APPROACH

Following the image acquisition process, the classification of the acquired images is performed using a Convolutional Neural Network (CNN) based on the GoogLeNet architecture [31]. GoogLeNet, renowned for its Inception modules, allows the model to process images at multiple scales, effectively capturing a wide range of features in each layer. This network has been adapted and trained to recognize four primary types of irregularities: splices, brands, shadows, and end-of-tape anomalies. These categories were chosen due to their higher frequency of occurrence in the dataset, allowing for sufficient data to train an effective classifier.

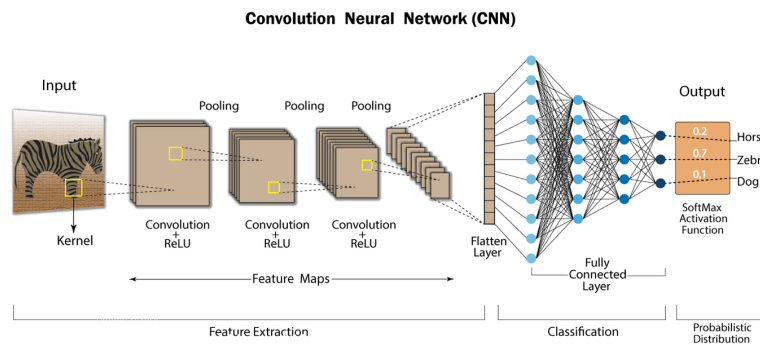


Figure 3.1: CNN Architecture

3.1.1 DATASET SELECTION AND CLASS IMBALANCE

While the system is capable of detecting other types of irregularities, such as physical damage or signal degradation, these were excluded from the training process due to their infrequent appearance in the dataset. Including these rare irregularities would have led to a highly imbalanced dataset, which poses a significant challenge in deep learning models. In a highly imbalanced dataset, the model tends to overfit on the dominant classes while underperforming on the minority classes. This is due to the fact that the classifier is more likely to predict frequent classes, as it is trained on a disproportionate number of examples from those classes [32].

To mitigate this issue, the number of classes was reduced, focusing on the more frequent types of irregularities that occur during tape playback. This adjustment ensures a more balanced input for the classifier, improving the model's generalization capability during training.

Consequently, the CNN is not only more accurate in recognizing common issues, but it also avoids overfitting to any one class, which would compromise the model's overall effectiveness.

3.1.2 SEPARATE DATASET ANALYSIS

Two distinct datasets were used to evaluate the performance of the classification system: one consisting of tapes played at a speed of 7.5 inches per second (ips), and the other at 15 ips. The different playback speeds introduce variations in the image acquisition process, potentially affecting the performance of the classifier. Therefore, separate analyses were necessary to ensure that the CNN could generalize well across both datasets.

During the validation phase, both classifiers achieved an accuracy of 95%, which is a remarkable result, considering the complexity and variability of the acquired images. The validation process involved a portion of the dataset that was withheld from the training phase to evaluate the classifier's performance on unseen data. This validation accuracy provided confidence in the model's capability to generalize well beyond the training data.

The performance of the classifiers was further assessed during the testing phase, where frames extracted from the dataset prior to the network's training were used. This additional evaluation aimed to test the model's robustness when exposed to entirely unseen frames, simulating real-world conditions. The high performance achieved in both validation and testing phases confirms the system's reliability and practical applicability in classifying various types of tape irregularities.

3.1.3 LIMITATIONS OF THE CLASSIFICATION SYSTEM

Despite the high accuracy achieved, the classification system has some limitations. One key limitation is the exclusion of rare irregularities due to their low frequency in the dataset. These irregularities, although less frequent, can still be significant in the evaluation and restoration of magnetic tape recordings. The inability to effectively classify these rare anomalies might result in a less comprehensive analysis, particularly in niche or specialized applications where these irregularities occur more frequently.

Another limitation is the dependence on a balanced dataset for training. While reducing the number of classes helped mitigate class imbalance, this approach may not be feasible for datasets where multiple irregularities of interest exist. In such cases, alternative techniques, such as data augmentation or cost-sensitive learning, could be explored to handle the imbalanced nature of the dataset more effectively [33].

Furthermore, the system’s performance is currently dependent on specific playback speeds (7.5 ips and 15 ips). For this system to be generalized to other tape playback speeds, additional training with datasets from different playback speeds would be necessary. Without such training, the classifier may not perform optimally when exposed to variations in image acquisition settings.

3.1.4 APPLICATION IN REAL-WORLD SCENARIOS

The classification system, trained using the GoogLeNet-based CNN, has been designed to automate the detection of tape irregularities in large archives of magnetic recordings. Given its high accuracy in both validation and testing, the system can be confidently deployed in real-world applications. It can automatically classify frames extracted during tape playback, flagging segments with splices, shadows, or other common issues that require further attention.

In practice, the CNN processes each frame individually, running inference on the images captured during the playback of tapes. This real-time classification capability is particularly valuable for large-scale digitization projects, where manual inspection of each frame is impractical. By identifying key issues in the playback process, the system facilitates the restoration and preservation of archival materials, ensuring that valuable recordings are correctly identified and maintained.

3.2 SELECTING THE BEST MODEL

In the process of identifying the best model for the classification task, four different deep learning architectures were explored: YOLOv7, VGG16, ResNet50, and EfficientNet. Each of these models was selected for evaluation due to their proven performance in both detection and classification tasks, but with varying strengths and design philosophies that cater to different requirements.

YOLOv7 [34] is primarily designed for real-time object detection and is known for its ability to balance speed and accuracy. Although its focus is more on detection tasks, YOLOv7 can also perform well on classification problems in situations where rapid inference is critical. However, in tasks that focus solely on classification, more specialized models such as VGG16 [35], ResNet50 [10], and EfficientNet [36] tend to outperform due to their architectural optimizations for image recognition.

VGG16 is a well-known convolutional neural network (CNN) that employs a simple and

deep structure with uniform convolutional layers followed by max-pooling layers. Its depth allows it to learn detailed features from images, but the high number of parameters makes it computationally intensive and slower compared to modern architectures. Despite this, VGG16 has been a strong baseline for many classification tasks, especially when computational power is not a major constraint.

ResNet50, on the other hand, introduced the concept of residual connections, which allow gradients to flow more easily through the network during backpropagation. This innovation helps prevent the vanishing gradient problem that typically hinders the performance of very deep networks. By enabling deeper networks without a drop in performance, ResNet50 has become one of the most widely used architectures in classification tasks.

EfficientNet brings a more modern approach by scaling networks in a balanced manner—considering depth, width, and resolution—rather than arbitrarily increasing the number of layers or parameters. It achieves better accuracy while using fewer computational resources compared to older architectures. This makes EfficientNet particularly suitable when both performance and efficiency are essential, such as in resource-constrained environments or large-scale classification tasks.

These models were carefully compared to determine which was the most suitable for the classification task at hand. The evaluation focused on balancing accuracy, computational efficiency, and the ability to generalize well to unseen data. In the end, the choice of the best model was driven by a combination of these factors, taking into account the specific needs of the classification problem and the computational resources available.

3.3 YOLO V7

YOLOv7 (You Only Look Once version 7) is the latest iteration in the YOLO family of object detection models, which are designed to perform real-time object detection with high accuracy and efficiency. Developed by Wang et al. [34], YOLOv7 introduces several advancements that make it stand out from its predecessors, offering an exceptional balance between speed, accuracy, and computational efficiency.

3.3.1 KEY FEATURES AND ARCHITECTURE

YOLOv7 improves upon the earlier YOLO versions by incorporating a range of architectural and training innovations, including the use of **trainable bag-of-freebies**, which refers to tech-

niques applied during training that enhance the model’s performance without increasing inference costs. This includes augmentations and optimizations that enhance the network’s ability to generalize across diverse datasets. Additionally, YOLOv7 introduces an **extended efficient layer aggregation network (ELAN)**, which optimizes how features are aggregated within the network, leading to better feature extraction and representation.

The architecture of YOLOv7 consists of three main components: 1. **Backbone**: Responsible for extracting features from the input image. 2. **Neck**: Combines and processes these features at different scales to detect objects of varying sizes. 3. **Head**: Outputs the bounding boxes and class probabilities for each detected object.

YOLOv7’s backbone is designed to be lightweight, allowing for high-speed inference, while still retaining the accuracy needed for detecting small and large objects in various contexts. The introduction of **dynamic label assignment** and **model scaling** enables YOLOv7 to adjust the size, depth, and width of the network, balancing computational needs based on available resources.

3.3.2 INPUT IMAGE REQUIREMENTS

YOLOv7 operates on images in the form of a 2D grid of pixels, typically RGB images, though grayscale can also be processed by the network. The images are resized to a square resolution, commonly 640×640 or 1280×1280 pixels, depending on the model configuration. During pre-processing, the pixel values are normalized to a range of $[0, 1]$ by dividing the original pixel intensity values (ranging from 0 to 255) by 255.

One of the strengths of YOLOv7 is its ability to detect multiple objects within an image at different scales. The model uses anchor boxes—predefined bounding boxes of different aspect ratios and sizes—to predict the locations of objects. By comparing predicted bounding boxes with these anchor boxes, YOLOv7 efficiently detects objects of various shapes and sizes in a single pass through the network.

3.3.3 HOW YOLOv7 WORKS

The working mechanism of YOLOv7 is based on the **single-shot detection** approach, where the model simultaneously predicts both the class of an object and its bounding box coordinates in a single forward pass through the network. This is in contrast to traditional two-stage detectors, such as Faster R-CNN, which first generate proposals for object locations and then classify

them in a separate step. YOLOv7 streamlines this process, reducing computational overhead and enabling real-time performance.

The process begins by feeding an image into the network's backbone, which extracts low-level and high-level features from the image. These features are then processed by the neck and passed to the head of the network, where the final predictions are made. YOLOv7 generates three outputs at different scales, each corresponding to different levels of feature granularity, allowing the model to detect both small and large objects effectively.

The **loss function** used in YOLOv7 incorporates three components: 1. **Bounding box regression loss**: Measures how well the predicted bounding box matches the ground truth box. 2. **Objectness loss**: Determines whether a given box contains an object or background. 3. **Classification loss**: Evaluates the accuracy of the predicted class labels for the detected objects.

3.3.4 COMPUTATIONAL REQUIREMENTS AND DEPLOYMENT

YOLOv7 is designed to work efficiently across a range of devices, from powerful GPUs to resource-constrained environments like embedded systems. The model can be trained on high-end GPUs, such as those from the NVIDIA Tesla or A100 series, which provide the computational power needed for large-scale training tasks. For deployment, YOLOv7 is optimized for fast inference on both GPUs and CPUs, making it suitable for real-time applications such as autonomous driving, video surveillance, and robotics.

In terms of memory requirements, the standard YOLOv7 model can be run on devices with relatively modest memory, as its architecture is optimized to minimize resource consumption without sacrificing accuracy. This makes YOLOv7 ideal for edge devices, such as drones or mobile phones, where real-time object detection is needed under strict computational and power constraints.

3.3.5 APPLICATION AND USE CASES

While YOLOv7 is primarily designed for object detection tasks, it has also been applied successfully to various fields such as medical imaging, autonomous systems, and retail analytics. In this thesis, YOLOv7 was employed to detect multiple object categories in the dataset, leveraging its ability to perform well in real-time applications. Its high speed and accuracy allowed for efficient detection and classification, particularly in scenarios where rapid inference is critical.

Though YOLOv7 is not primarily designed for image classification, it can be adapted for such tasks by modifying the final layers of the network. However, in cases where the primary goal is classification, models such as ResNet and EfficientNet are often better suited due to their architecture and training optimizations for such tasks.

3.4 CLASSIFICATION USING YOLOv7

In this section, I describe the process I followed to adapt YOLOv7 [34] for image classification tasks. Although YOLOv7 is primarily designed for object detection, I modified its structure and prepared the dataset to handle classification. This involved pre-processing the images, splitting the dataset, generating dummy annotations, and training the model.

3.4.1 PREPROCESSING THE IMAGES

The first step in preparing the dataset was resizing all images to a uniform resolution of 416×416 pixels, which is the standard input size for YOLOv7. This was necessary to ensure consistency in input dimensions across all images in the dataset. I wrote a Python script using the `cv2` library to process and resize all images from a directory structure recursively.

Listing 3.1: Image Preprocessing Code

```
def process_images_in_directory(directory, target_size,
                               processed_images_dir, image_extensions):
    for root, dirs, files in os.walk(directory):
        for file in tqdm(files):
            if not any(file.lower().endswith(ext) for
                       ext in image_extensions):
                continue # Skip non-image files

            raw_image_path = os.path.join(root, file)
            image = cv2.imread(raw_image_path)

            if image is None:
                print(f"Warning: Could not read image {
                    file }. Skipping ...")
                continue
```

```

resized_image = cv2.resize(image,
    target_size)
processed_image_path = os.path.join(
    processed_images_dir, os.path.relpath(
    raw_image_path, directory))
os.makedirs(os.path.dirname(
    processed_image_path), exist_ok=True)
cv2.imwrite(processed_image_path,
    resized_image)

```

The processed images were saved to a separate directory to preserve the original files, allowing them to be used in future experiments. This preprocessing step ensures that all images fed into the YOLOv7 model are consistent in size.

3.4.2 SPLITTING THE DATASET

After preprocessing the images, I split the dataset into training, validation, and test sets. To preserve the subdirectory structure and balance across different classes, I wrote a function to recursively collect all image filenames and used `train_test_split` from `sklearn` to create the splits. The dataset was split as follows: 60% for training, 20% for validation, and 20% for testing.

Listing 3.2: Splitting the Dataset

```

# Split the dataset (initial split: 80% for train+val,
    20% for test)
train_val_filenames, test_filenames = train_test_split(
    image_filenames, test_size=0.2, random_state=42)

# Further split train+val into 75% train, 25% val
train_filenames, val_filenames = train_test_split(
    train_val_filenames, test_size=0.25, random_state
    =42)

```

The images were then copied to their respective directories for training, validation, and testing, while preserving the subdirectory structure for each class.

3.4.3 GENERATING DUMMY ANNOTATIONS

YOLOv7 requires annotations in a specific format (bounding box coordinates and class labels), even for classification tasks. Since I am focusing on classification rather than object detection, I generated dummy annotations where the entire image is treated as a bounding box, with the class label corresponding to the subdirectory in which the image is stored.

Listing 3.3: Generating Dummy Annotations

```
def generate_dummy_annotations(image_dir, class_ids):
    for class_name, class_id in class_ids.items():
        class_dir = os.path.join(image_dir, class_name)
        for image_name in os.listdir(class_dir):
            if image_name.lower().endswith((' .png', ' .
            jpg', ' .jpeg')):
                annotation = f"{class_id} 0.5 0.5 1 1\n
                "
                annotation_path = os.path.join(
                    class_dir, image_name.rsplit('.', 1)
                    [0] + '.txt')
                with open(annotation_path, 'w') as f:
                    f.write(annotation)
```

This function automatically generates annotation files for each image, assigning the corresponding class label and a dummy bounding box covering the entire image. The dummy annotations allow YOLOv7 to treat each image as a classified entity without the need for object detection.

3.4.4 TRAINING THE YOLOv7 MODEL

Finally, I used the pre-processed images and generated annotations to train YOLOv7. The training process was carried out using the official YOLOv7 PyTorch implementation, with the following command:

```
!python train.py --img 416 --batch 16 --epochs 120 --
data /content/drive/MyDrive/datasetyolo/data.yaml --
cfg cfg/training/yolov7.yaml --weights 'yolov7.pt'
--device 0
```

Here, the images were resized to 416×416 , a batch size of 16 was used, and the training was run for 120 epochs. The YOLOv7 model was initialized with pre-trained weights to benefit from transfer learning. During training, I monitored the loss and accuracy metrics to ensure that the model was learning effectively and converging as expected.

3.5 RESULTS OF YOLOv7 TRAINING AND PERFORMANCE ANALYSIS

The following figures demonstrate the various losses and metrics recorded during the training and validation phases of the YOLOv7 model. These metrics, including classification loss, objectness loss, and mean Average Precision (mAP), provide insights into how well the model learned and generalized over the dataset. Based on these results, we will analyze the effectiveness of the training process and the quality of the final model.

3.5.1 CLASSIFICATION LOSS ON TRAINING SET

Figure 3.2 shows the classification loss (cls_loss) over the course of training. The classification loss consistently decreases throughout the training, starting around 0.014 and ending close to 0.009. This steady decrease indicates that the model successfully learned to classify the images more accurately as training progressed. The model did not experience sudden spikes or overfitting, suggesting a stable training process.



Figure 3.2: Classification Loss on Training Set

3.5.2 OBJECTNESS LOSS ON TRAINING SET

Figure 3.3 tracks the objectness loss (`obj_loss`), which measures how well the model detects the presence of objects within each image. Similar to the classification loss, the objectness loss decreases steadily, from approximately 0.008 to 0.0044 by the end of training. The decreasing trend indicates that the model is improving its ability to identify object-containing regions, which is crucial for both detection and classification tasks. The consistent downward trend without abrupt changes implies that the model was learning effectively.

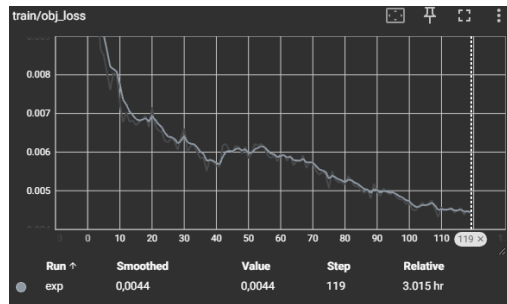


Figure 3.3: Objectness Loss on Training Set

3.5.3 BOX LOSS ON VALIDATION SET

Figure 3.4 presents the box loss (`val_box_loss`) on the validation set. Box loss measures the accuracy of the bounding box predictions. After an initial period of fluctuation, the loss converges around 0.002 by the 30th epoch and remains stable throughout the remaining epochs. The stabilization of box loss suggests that the model can accurately localize objects within the images after the early stages of training, which is a positive indicator for object detection tasks.

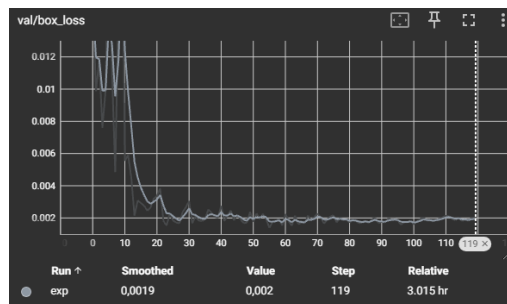


Figure 3.4: Box Loss on Validation Set

3.5.4 CLASSIFICATION LOSS ON VALIDATION SET

As seen in Figure 3.5, the classification loss (`val_cls_loss`) on the validation set follows a similar downward trend as the training classification loss. Starting at around 0.007, it converges at approximately 0.002 by the end of training. The steady decrease in validation loss without major fluctuations suggests that the model generalizes well to unseen data, with minimal signs of overfitting. The low final validation loss indicates good classification accuracy.

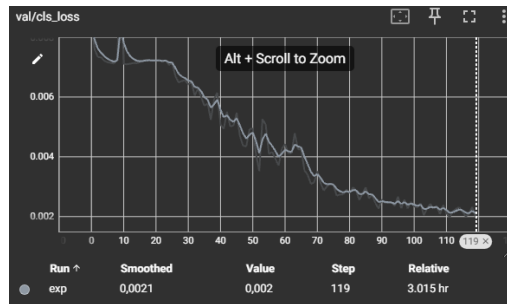


Figure 3.5: Classification Loss on Validation Set

3.5.5 MEAN AVERAGE PRECISION (mAP) @ 0.5

Figure 3.6 shows the mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5. The mAP starts at around 0.4, then rapidly increases during the first 30 epochs, after which it plateaus around 0.96. This high mAP score is a strong indicator of the model's ability to correctly classify and detect objects at an IoU threshold of 0.5. This suggests that the model is performing very well in identifying and localizing objects with high confidence.

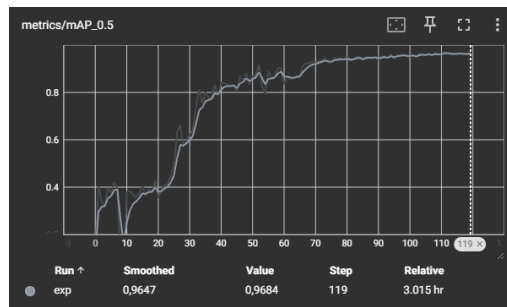


Figure 3.6: Mean Average Precision (mAP) @ 0.5

3.5.6 MEAN AVERAGE PRECISION (mAP) @ 0.5:0.95

In Figure 3.7, the mAP at IoU thresholds ranging from 0.5 to 0.95 is shown. The mAP stabilizes at approximately 0.947, indicating that the model maintains a high level of precision across a wide range of IoU thresholds. This metric is more stringent than mAP@0.5 and shows the model's robustness across different overlap thresholds. Achieving a value close to 0.95 is an excellent result, confirming the model's overall strong performance.

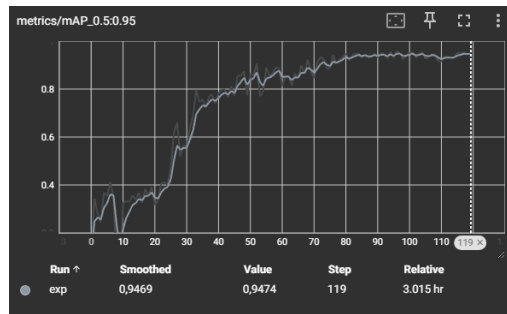


Figure 3.7: Mean Average Precision (mAP) @ 0.5:0.95

3.5.7 PRECISION

Precision measures the proportion of correctly identified positive detections, reflecting how well the model avoids false positives. As seen in Figure 3.8, the precision starts around 0.4 and improves significantly over the training period, plateauing at around 0.95. This indicates that the model is very effective at reducing false positives, suggesting high classification reliability.

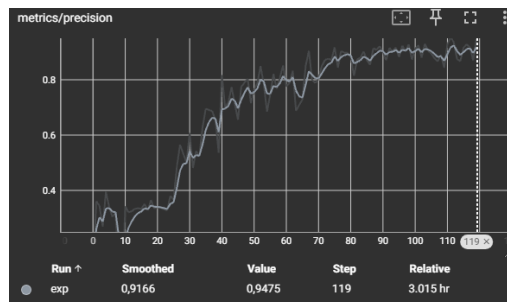


Figure 3.8: Precision

3.5.8 RECALL

Recall measures the proportion of actual positives that were correctly identified by the model, or in other words, how well the model detects all relevant objects. Figure 3.9 shows that recall increases and stabilizes at around 0.92, demonstrating that the model has a high capability of detecting true positives while minimizing missed detections.

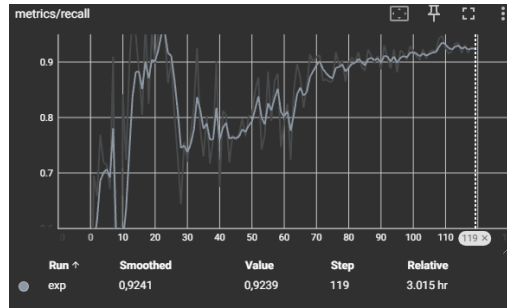


Figure 3.9: Recall

3.5.9 BOX LOSS ON TRAINING SET

Figure 3.10 shows the box loss on the training set, which follows a downward trend, converging to a value of around 0.0084. This indicates that the model was able to accurately predict the location of objects in the training images, improving as training progressed.

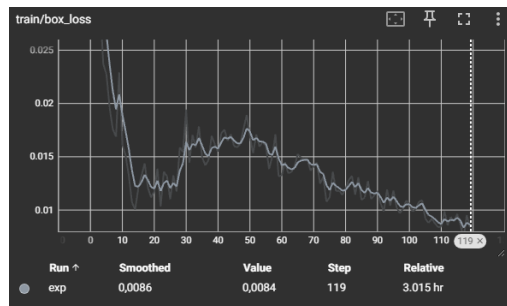


Figure 3.10: Box Loss on Training Set

3.5.10 OVERALL ANALYSIS

Based on the figures and metrics observed, the YOLOv7 model demonstrated strong performance across various aspects of the task. The decreasing losses (classification, objectness, and

box losses) indicate effective learning and minimal overfitting. The mAP values, particularly $\text{mAP}@0.5:0.95$, show that the model achieves high precision and recall across a variety of IoU thresholds, demonstrating robustness in both detection and classification.

The precision and recall values suggest that the model is well-balanced, with high scores in both areas, meaning it avoids false positives while also accurately detecting most relevant objects. Overall, the training process has led to an accurate, robust model capable of effective object classification and detection.

3.6 VGG16 MODEL FOR IMAGE CLASSIFICATION

VGG16 [35] is a well-known convolutional neural network (CNN) architecture developed by Karen Simonyan and Andrew Zisserman from the Visual Geometry Group at the University of Oxford. The model was introduced as part of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and has been widely adopted for image classification tasks due to its simplicity and effectiveness.

The VGG16 architecture consists of 16 layers—13 convolutional layers followed by 3 fully connected layers. One of the defining characteristics of VGG16 is its use of small 3×3 convolutional filters, which allow the network to capture fine details from the images while maintaining a deep architecture. The network is designed with a fixed input size of 224×224 pixels, and it uses max-pooling layers to progressively reduce the spatial dimensions of the feature maps.

Despite being a deep network, VGG16 is computationally expensive due to the large number of parameters. However, its simplicity in terms of having a uniform layer structure makes it easy to implement and modify for transfer learning tasks. In this project, I utilized a pre-trained VGG16 model and fine-tuned it for the custom dataset.

3.6.1 PREPROCESSING AND DATA AUGMENTATION

The first step in preparing the dataset was to apply appropriate transformations to ensure that the input images are resized, normalized, and augmented where necessary. The transformations were different for the training, validation, and test datasets. For the training dataset, I included data augmentation techniques such as random resized cropping and horizontal flipping to help the model generalize better. Validation and test sets, however, only applied resizing and normalization to maintain the consistency of the input data.

Listing 3.4: Data Transformations for Training and Validation

```

data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                              [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                              [0.229, 0.224, 0.225])
    ])
}

```

The dataset was loaded using the `ImageFolder` module from the `torchvision.datasets` library, and the dataloaders were created for the training, validation, and test sets.

3.6.2 MODIFYING THE VGG16 ARCHITECTURE

Since the VGG16 model is pre-trained on a large-scale dataset, its feature extractor was kept frozen by setting `requires_grad = False` for all layers in the features block. The fully connected layers, however, were modified to adapt to the number of classes in the dataset. Specifically, I replaced the final layer of the classifier with a new fully connected layer, where the output corresponds to the number of classes in the custom dataset.

Listing 3.5: Modifying the Classifier in VGG16

```

# Load pre-trained VGG16 model
model = models.vgg16(pretrained=True)

# Freeze feature extraction layers
for param in model.features.parameters():
    param.requires_grad = False

```

```

# Modify the classifier
num_features = model.classifier[6].in_features
features = list(model.classifier.children())[:-1] #
    Remove last layer
features.extend([nn.Linear(num_features, len(
    class_names))]) # New output layer
model.classifier = nn.Sequential(*features)
model = model.to(device)

```

3.6.3 TRAINING PROCESS

For training, I used the Stochastic Gradient Descent (SGD) optimizer with momentum, along with the Cross-Entropy loss function. The learning rate was initially set to 0.001, and a learning rate scheduler was applied to reduce the learning rate if the validation loss did not improve for a specified number of epochs. Additionally, early stopping was implemented to terminate the training if the validation accuracy did not improve over 20 consecutive epochs.

The training loop involved iterating over the training and validation phases, calculating the loss and accuracy for each epoch, and saving the model with the best validation accuracy.

Listing 3.6: Training Loop with Early Stopping

```

for epoch in range(num_epochs):
    print(f'Epoch {epoch} / {num_epochs - 1}')

    # Training phase
    model.train()
    running_loss = 0.0
    running_corrects = 0
    for inputs, labels in dataloaders['train']:
        inputs, labels = inputs.to(device), labels.to(
            device)
        optimizer.zero_grad()

    # Forward pass
    outputs = model(inputs)
    _, preds = torch.max(outputs, 1)

```

```

    loss = criterion(outputs, labels)

    # Backward pass
    loss.backward()
    optimizer.step()

    # Update statistics
    running_loss += loss.item() * inputs.size(0)
    running_corrects += torch.sum(preds == labels.
        data)

# Validation phase
model.eval()
val_running_loss = 0.0
val_running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs, labels = inputs.to(device), labels.to(
        device)
    with torch.no_grad():
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        val_loss = criterion(outputs, labels)

    # Update statistics
    val_running_loss += val_loss.item() * inputs.
        size(0)
    val_running_corrects += torch.sum(preds ==
        labels.data)

# Check for early stopping
if val_running_corrects.double() / dataset_sizes['
    val'] > best_acc:
    best_acc = val_running_corrects.double() /
        dataset_sizes['val']

```

```
best_model_wts = copy.deepcopy(model.state_dict())
```

The model was trained for 60 epochs, and early stopping was triggered after 20 epochs without improvement in the validation accuracy. At the end of training, the model achieved its best validation accuracy of over 90%, showing the effectiveness of transfer learning in this context.

3.6.4 VGG16 MODEL RESULTS

After training the VGG16 model for image classification, the performance was evaluated on the test dataset. Two key metrics were used to assess the model's effectiveness: the confusion matrix and the classification report, which includes precision, recall, F1-score, and overall accuracy. These metrics provide insights into how well the model performs across the different classes in the dataset.

3.6.5 CONFUSION MATRIX

The confusion matrix in Figure 3.11 provides a visual representation of the model's predictions versus the true labels for each class. The rows represent the actual classes (true labels), while the columns represent the predicted classes. The diagonal elements indicate the number of correct predictions for each class.

From the confusion matrix, it can be seen that the model correctly predicted the vast majority of the samples in each class:

- For the **brands** class, 160 out of 161 samples were correctly classified, with only 1 misclassification.
- For the **shadows** class, 142 out of 146 samples were correctly classified, with 2 samples misclassified as **brands** and 2 misclassified as **splices**.
- For the **splices** class, 178 out of 187 samples were correctly classified, with a few misclassifications as either **brands** or **shadows**.

The confusion matrix demonstrates that the model has high accuracy across all three classes, with minimal misclassifications.

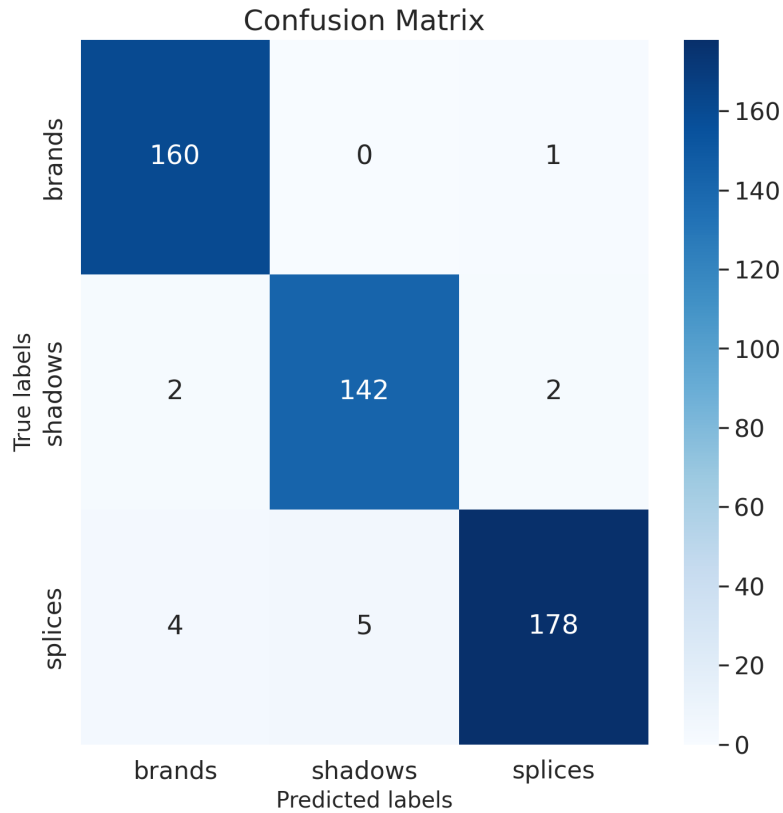


Figure 3.11: Confusion Matrix of VGG16 Model on the Test Dataset

3.6.6 CLASSIFICATION REPORT

The classification report shown in Figure 3.12 provides a detailed breakdown of key evaluation metrics for each class, including precision, recall, F1-score, and support.

The overall accuracy of the model on the test dataset is 97.17%, which is an excellent result. The model performed consistently across all classes:

- **Brands:** The precision was 96%, recall was 99%, and F1-score was 98%, indicating that the model can accurately detect this class with very few false positives and false negatives.
- **Shadows:** Precision and recall for the shadows class were both 97%, resulting in an F1-score of 97%. This indicates that the model is both precise and consistent in detecting shadows, with only a small number of misclassifications.
- **Splices:** The model achieved 98% precision and 95% recall for the splices class, with an F1-score of 97%. The slightly lower recall in this class suggests that a few splice samples were missed by the model, but overall performance remains strong.

Test dataset accuracy: 0.9717				
	precision	recall	f1-score	support
brands	0.96	0.99	0.98	161
shadows	0.97	0.97	0.97	146
splices	0.98	0.95	0.97	187
accuracy			0.97	494
macro avg	0.97	0.97	0.97	494
weighted avg	0.97	0.97	0.97	494

Figure 3.12: Classification Report for VGG16 Model on the Test Dataset

Additionally, the macro average and weighted average metrics were both 97%, further indicating that the model performed well across all classes without any major class imbalance issues affecting its performance.

3.6.7 CONCLUSION

The VGG16 model demonstrates strong classification performance, with an overall test accuracy of 97.17% and high precision, recall, and F1-scores across all classes. The confusion matrix highlights the minimal misclassifications between classes, while the classification report provides a clear picture of the model's robustness. These results validate the effectiveness of the fine-tuning process applied to the pre-trained VGG16 model for this specific image classification task.

3.7 EFFICIENTNET

EfficientNet [36] is a state-of-the-art convolutional neural network architecture designed for image classification tasks. It was developed with the goal of optimizing the trade-off between model accuracy and computational efficiency. The primary innovation in EfficientNet is its compound scaling method, which uniformly scales the depth, width, and resolution of the network using a fixed scaling coefficient.

3.7.1 EFFICIENTNET ARCHITECTURE

Traditional CNN architectures often scale by either increasing depth (adding more layers), width (adding more channels per layer), or image resolution (increasing the size of input im-

ages). However, scaling only one dimension can lead to suboptimal performance, as deeper networks tend to suffer from vanishing gradients, while wider networks become computationally expensive.

EfficientNet introduces a compound scaling method that simultaneously adjusts these three factors—depth, width, and resolution—using a compound coefficient. This allows the model to balance accuracy and efficiency by scaling all dimensions in a coordinated way. The base version of EfficientNet, called EfficientNet-Bo, is built upon a mobile inverted bottleneck convolution (MBConv) and squeeze-and-excitation (SE) optimization techniques, making it highly efficient even at smaller scales.

3.7.2 EFFICIENTNET VARIANTS

EfficientNet is available in different versions, denoted as Bo to B7, with each version scaling the model's size to improve accuracy while maintaining efficiency. As the model progresses from Bo to B7:

- Bo: The base model, optimized for efficiency.
- B7: The largest and most powerful version, offering higher accuracy at the cost of increased computational resources.

The key advantage of EfficientNet is that it significantly reduces the number of parameters and FLOPs (floating-point operations per second) compared to other state-of-the-art models like ResNet or Inception, while still achieving comparable or superior accuracy on popular benchmarks like ImageNet.

3.7.3 EFFICIENTNET FOR IMAGE CLASSIFICATION

EfficientNet has been successfully applied to various image classification tasks due to its balanced approach to scaling. The model's ability to process high-resolution images while maintaining efficiency makes it ideal for scenarios where computational resources are limited, such as mobile devices or embedded systems.

The compound scaling method ensures that the network can handle complex classification tasks, while the use of techniques like MBConv and SE further optimizes its performance. In practice, EfficientNet models achieve high classification accuracy with fewer parameters, making them suitable for large-scale image classification tasks.

In this thesis, EfficientNet was chosen as one of the candidate models for comparison in the image classification task due to its superior performance in terms of both accuracy and computational efficiency. The model's ability to adapt to different scales allowed it to perform well in classifying the dataset, while maintaining reasonable training and inference times.

3.8 MODIFIED VGG16 FOR IMAGE CLASSIFICATION

In this section, I describe the process of modifying the VGG16 model for my image classification task. The primary goal was to leverage the pre-trained VGG16 model and adapt it for classifying the dataset into three distinct classes: brands, shadows, and splices. The steps involved data preprocessing, label encoding, model modification, and training using TensorFlow.

3.8.1 DATA PREPROCESSING

The first step in preparing the dataset was to load the images and resize them to 224×224 pixels, which is the input size required for VGG16. The images were loaded using OpenCV, and their pixel values were normalized to a range of $[0, 1]$ to make training more stable and efficient. This normalization is essential because neural networks perform better with input data that is standardized or normalized.

Listing 3.7: Loading and Preprocessing Images

```
import cv2
import os
import numpy as np

path = 'data\path'
dataset_path = os.listdir(path)
im_size = 224

images = []
labels = []
for i in dataset_path:
    data_path = os.path.join(path, i)
    filenames = [f for f in os.listdir(data_path)]
```

```

for f in filenames:
    img = cv2.imread(os.path.join(data_path, f))
    img = cv2.resize(img, (im_size, im_size))
    images.append(img)
    labels.append(i)

images = np.array(images).astype('float32')/255.0

```

This code ensures that all images are resized and their pixel values are normalized, preparing them for input into the neural network.

3.8.2 LABEL ENCODING

Since the labels in the dataset are categorical (text-based), they need to be converted into numerical values for the model to process them. I used `LabelEncoder` to transform the labels into integers and then applied one-hot encoding to create a binary vector representation of the class labels. This ensures that the model outputs three separate probability values for the three classes.

Listing 3.8: Label Encoding and One-Hot Encoding

```

from sklearn.preprocessing import LabelEncoder,
    OneHotEncoder
from sklearn.compose import ColumnTransformer

y_labelencoder = LabelEncoder()
y = y_labelencoder.fit_transform(labels)
y = y.reshape(-1, 1)

ct = ColumnTransformer([('my_ohe', OneHotEncoder(),
    [0])], remainder='passthrough')
Y = ct.fit_transform(y)

```

This step transforms the categorical labels into a numerical format that the model can work with, ensuring that each class is represented appropriately in the final softmax layer of the model.

3.8.3 DATA SPLITTING AND SHUFFLING

The dataset was then shuffled to randomize the order of the images and split into training, validation, and testing sets. I used a 70-20-10 split, where 70% of the data was used for training, 20% for testing, and 10% for validation. This ensures that the model has sufficient data to learn from, while also leaving enough data for testing and validation to assess its performance.

Listing 3.9: Data Splitting for Training Validation and Testing

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

images , Y = shuffle (images , Y, random_state=1)

train_x , temp_x , train_y , temp_y = train_test_split(
    images , Y, test_size=0.3, random_state=415)
test_x , val_x , test_y , val_y = train_test_split(temp_x ,
    temp_y , test_size=0.6667, random_state=415)

print (train_x . shape)
print (test_x . shape)
print (val_x . shape)
```

This ensures that the training, validation, and test sets contain a balanced representation of the different classes.

3.8.4 MODIFYING THE VGG16 ARCHITECTURE

For this classification task, I used a pre-trained EfficientNetBo [36] model as the base. However, the VGG16 architecture was initially modified by removing its top layers and adding a new output layer with three nodes (one for each class), with a softmax activation function for multi-class classification. The Global Average Pooling layer was added before the output layer to reduce the dimensions of the feature maps, and a dropout layer was introduced to prevent overfitting.

Listing 3.10: Modified EfficientNetB0 Model for Classification

```
from tensorflow.keras import layers
```

```

from tensorflow.keras.applications import
    EfficientNetBo
from tensorflow.keras.models import Model

NUM_CLASSES = 3
IMG_SIZE = 224

inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
base_model = EfficientNetBo(include_top=False, weights=
    'imagenet', input_tensor=inputs)

x = layers.GlobalAveragePooling2D()(base_model.output)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(NUM_CLASSES, activation='softmax
    ')(x)

model = tf.keras.Model(inputs, outputs)

```

The main advantage of this approach is that the EfficientNetBo architecture already has powerful feature extraction capabilities from being pre-trained on ImageNet. By fine-tuning only the top layers, the model can adapt to the new dataset without the need for training from scratch, which saves time and computational resources.

3.8.5 TRAINING THE MODEL

The model was compiled with the Adam optimizer and categorical crossentropy loss, which is appropriate for multi-class classification. I also added callbacks for reducing the learning rate when the validation loss plateaued and for stopping the training early if the validation loss did not improve after a certain number of epochs.

Listing 3.11: Training the Model with Callbacks

```

from tensorflow.keras.callbacks import
    ReduceLROnPlateau, EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
    factor=0.2, patience=5, min_lr=1e-6, verbose=1)

```

```
early_stopping = EarlyStopping(monitor='val_loss',  
                                patience=50, verbose=1, restore_best_weights=True)
```

```
hist = model.fit(train_x, train_y,  
                 epochs=150,  
                 batch_size=16,  
                 validation_data=(val_x, val_y),  
                 callbacks=[reduce_lr, early_stopping])
```

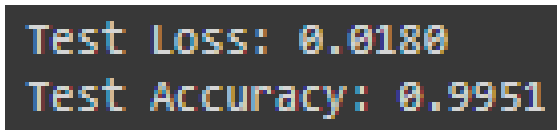
The callbacks ensure that the model's learning process is dynamic, adapting the learning rate when necessary and stopping early if further training would not yield significant improvements. This makes the training process more efficient and helps avoid overfitting.

3.9 EFFICIENTNET RESULTS AND OVERFITTING ANALYSIS

In this section, I evaluate the performance of the EfficientNet model based on the training results and metrics. The training process yielded a test accuracy of 99.51% (as seen in Figure 3.13), and the model exhibited high accuracy on both the training and validation sets. However, the presence of overfitting became evident from the large discrepancy between the training and validation curves during the early epochs.

3.9.1 TEST LOSS AND ACCURACY

The final test accuracy and loss for the model are shown in Figure 3.13. Despite the high test accuracy of 99.51%, this result was deemed less significant due to the clear overfitting that occurred during training. Overfitting is characterized by the model's ability to perform exceptionally well on the training data while failing to generalize to unseen validation data. As a result, the model's performance on the test set, while high, cannot be fully trusted as an indicator of real-world performance.



```
Test Loss: 0.0180  
Test Accuracy: 0.9951
```

Figure 3.13: Test Loss and Test Accuracy for EfficientNet

3.9.2 TRAINING AND VALIDATION CURVES

Figure 3.14 presents the accuracy and loss curves for both the training and validation sets. The training accuracy quickly increased to near 100%, while the validation accuracy initially showed erratic behavior, indicating that the model struggled to generalize well to unseen data. The validation loss spiked early in the training process and eventually plateaued, further indicating overfitting.

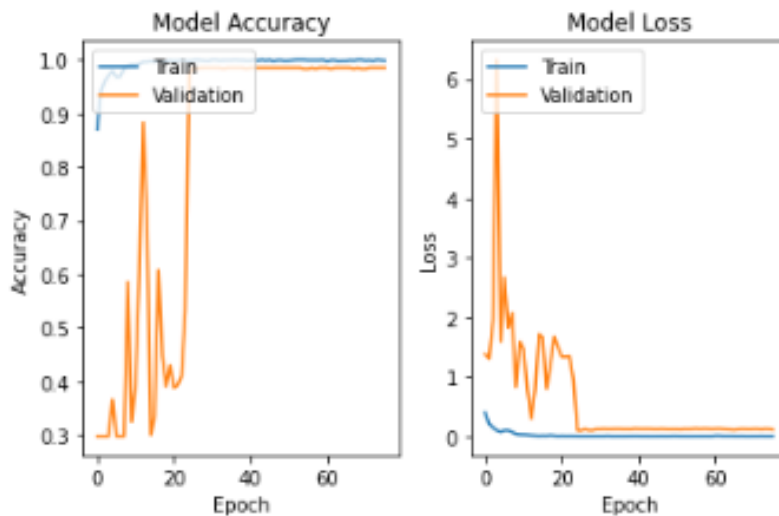


Figure 3.14: Training and Validation Accuracy/Loss Curves for EfficientNet

The clear separation between the training and validation curves suggests that the model was too complex for the dataset, learning noise and irrelevant patterns from the training data that did not generalize well to the validation data. This can be attributed to the model's large number of parameters, which are more suited for larger and more diverse datasets like ImageNet. While the early stopping and learning rate reduction callbacks were implemented to mitigate overfitting, they were insufficient to fully address the issue.

3.9.3 PRECISION AND RECALL

Although the model achieved a high test accuracy, the presence of overfitting raises concerns about its ability to perform consistently in real-world scenarios. Table 3.1 provides the precision and recall values for each class. The precision values are high, indicating that the model makes relatively few false positive predictions. However, the recall for the minority classes is lower, suggesting that the model struggled to detect some instances of the less represented

classes. This imbalance is another indication that overfitting may have biased the model towards better performance on the majority class.

Class	Precision	Recall
Brands	0.98	0.99
Shadows	0.95	0.87
Splices	0.92	0.85
Average	0.95	0.90

Table 3.1: Precision and Recall Values for EfficientNet

The high precision values across all classes are indicative of the model’s ability to correctly classify positive instances, but the recall values show that it failed to identify all positive instances, particularly in the shadow and splice classes. This could be attributed to the model memorizing the majority class (brands) at the expense of the other two classes. As a result, the model underperformed in terms of recall for the less frequent classes, highlighting the negative impact of overfitting.

3.9.4 CONCLUSION

While EfficientNet initially appeared to perform well based on its high test accuracy, the analysis of the training and validation curves, as well as the precision and recall values, revealed significant overfitting. This overfitting resulted in poor generalization to unseen data, particularly for the minority classes. Therefore, despite the high reported accuracy, the model was not considered reliable for deployment. Future work could involve further regularization techniques, additional data augmentation, or a simpler model architecture to mitigate overfitting.

The overfitting observed in the model’s training process suggests that it memorized patterns from the training data but failed to generalize well to new, unseen data. This is evident from the sharp increase in validation loss and fluctuating validation accuracy during early epochs, which stabilized only after a significant amount of training. The high precision on the majority class combined with lower recall on the minority classes demonstrates that the model struggled with class imbalances, further emphasizing the need for improved regularization and balanced class representation during training.

3.10 RESNET50 MODEL AND FINAL MODIFICATIONS

3.10.1 RESNET50 OVERVIEW

ResNet50 [10] is a powerful convolutional neural network architecture that introduced the concept of residual learning. The primary innovation in ResNet is the use of residual blocks, which allow the model to learn residual functions with reference to the input layers. These residual connections help mitigate the vanishing gradient problem in deep networks, making it possible to train very deep networks, sometimes with hundreds or even thousands of layers.

The basic building block of ResNet is the residual block, represented as:

$$y = F(x, \{W_i\}) + x$$

where x is the input to the block, y is the output, and $F(x, \{W_i\})$ represents the residual function, typically comprising convolutional layers, batch normalization, and ReLU activation. The term x is added to the output of the convolutional layers, introducing an identity mapping that helps preserve the gradient flow during backpropagation.

ResNet50 specifically consists of 50 layers, including 49 convolutional layers and 1 fully connected layer at the end. This architecture has shown remarkable performance on various benchmarks, such as ImageNet, due to its ability to learn complex features while maintaining efficient gradient propagation through its residual connections.

3.10.2 MODIFICATIONS AND FINE-TUNING FOR THE FINAL MODEL

In my implementation, I used a pre-trained ResNet50 model, which was already trained on the ImageNet dataset. The choice of a pre-trained model helps in leveraging the feature extraction capabilities learned from a large and diverse dataset, accelerating the training process and potentially improving performance on my specific task.

The pre-trained ResNet50 model was modified in the following ways to adapt it for my classification task:

- **Adjusting the Final Layer:** The original ResNet50 model has a final fully connected layer with 1000 output neurons, corresponding to the 1000 classes in ImageNet. Since my task involves only three classes, I replaced the final fully connected layer with a new one, containing the appropriate number of neurons:

```
model.fc = nn.Linear(num_features, 3)
```

where `num_features` is the number of input features to the fully connected layer, which in ResNet50 is 2048. This modification allows the network to output three probabilities, each corresponding to one of the classes in my dataset.

- **Data Augmentation:** To enhance the model's generalization capabilities, I applied data augmentation techniques during training. These techniques include random resizing, cropping, and horizontal flipping of images, helping the model become more robust to variations in the input data.
- **Normalization:** The input images were normalized using the same mean and standard deviation values as those used during the pre-training of ResNet50 on ImageNet:

$$\text{Normalize}([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])$$

This ensures that the input data distribution matches the original training distribution, aiding in better transfer learning.

- **Training Strategy:** The model was trained using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 and a momentum of 0.9. The learning rate was reduced by a factor of 0.1 whenever the validation loss plateaued, which is managed using a learning rate scheduler. Early stopping was also employed to prevent overfitting, where training was halted if no improvement was observed in the validation loss for 10 consecutive epochs.

3.10.3 TRAINING PROCESS

The training process involved feeding the data through the modified ResNet50 model using the following pipeline:

- **Data Loading:** The dataset was loaded using PyTorch's `DataLoader`, with data augmentation applied during the training phase to increase variability and prevent overfitting.
- **Loss Function and Optimization:** The model was trained using the Cross-Entropy Loss function, which is well-suited for multi-class classification tasks. The optimizer used was SGD, which effectively handles large datasets and converges faster with momentum.
- **Learning Rate Scheduler and Early Stopping:** A `ReduceLROnPlateau` scheduler was implemented to reduce the learning rate when the validation loss stopped improving,

helping the model to fine-tune its learning in the later stages. Early stopping was used to halt the training when further improvements were unlikely, based on the validation loss.

The decision to use ResNet50 as the final model was driven by its ability to handle complex feature hierarchies, making it well-suited for the classification task at hand. The residual connections in ResNet50 allow the model to learn deeper features more effectively, and the pre-trained weights provided a strong starting point for fine-tuning on the specific dataset.

3.1.1 RESNET50 MODEL RESULTS

The final ResNet50 model was evaluated on the test dataset, yielding exceptional performance across all metrics. The confusion matrix and classification report provide a detailed overview of the model's ability to accurately classify the three classes: *brands*, *shadows*, and *splices*.

3.1.1.1 CONFUSION MATRIX

The confusion matrix for the test set is displayed in Figure 3.15. It illustrates the number of correct and incorrect predictions made by the model for each class.

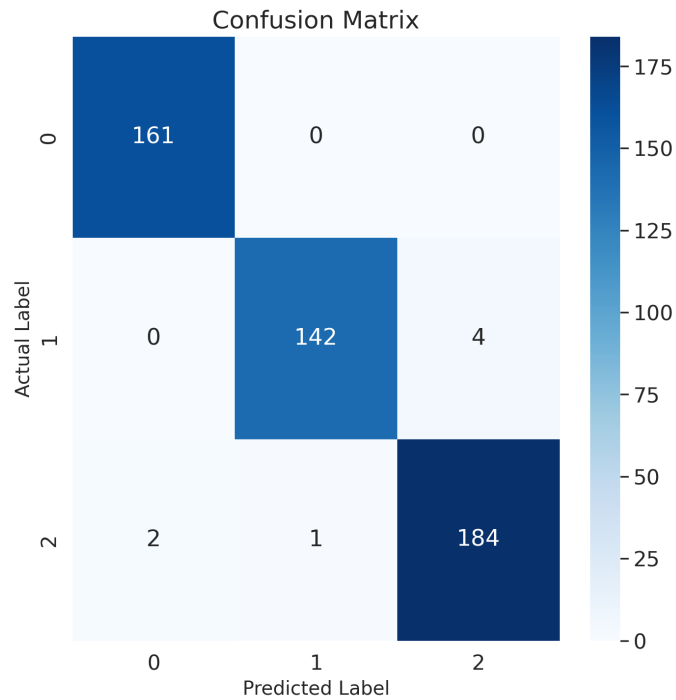


Figure 3.15: Confusion Matrix for ResNet50 Model

From the confusion matrix, it is evident that the model performed exceptionally well:

- For the *brands* class (label 0), all 161 instances were correctly classified with no misclassifications.
- For the *shadows* class (label 1), the model correctly classified 142 out of 146 instances, with only 4 instances being misclassified.
- For the *splices* class (label 2), 184 out of 187 instances were correctly classified, with just 3 misclassifications.

These results indicate a high level of accuracy in the model’s predictions, with very few errors across all classes.

3.1.1.2 PRECISION, RECALL, AND F1-SCORE

The precision, recall, and F1-score metrics further highlight the model’s strong performance:

The model achieved a macro average precision, recall, and F1-score of 0.99, indicating a balanced performance across all classes:

Class	Precision	Recall	F1-Score	Support
Brands (0)	0.99	1.00	0.99	161
Shadows (1)	0.99	0.97	0.98	146
Splices (2)	0.98	0.98	0.98	187
Accuracy			0.99	494
Macro Avg	0.99	0.99	0.99	494
Weighted Avg	0.99	0.99	0.99	494

Table 3.2: Precision, Recall, and F1-Score for ResNet50 Model

- ****Precision**:** The model's precision is extremely high, meaning it makes very few false positive predictions. For instance, for the *brands* class, the precision is 0.99, indicating that almost all predicted *brands* instances were correct.
- ****Recall**:** The recall is also very high across all classes, showing the model's ability to correctly identify true positives. The recall for the *shadows* class is 0.97, meaning the model successfully identified 97% of all actual *shadows*.
- ****F1-Score**:** The high F1-scores across all classes demonstrate a good balance between precision and recall, indicating the model's robustness in handling the dataset.

3.11.3 CONCLUSION

The ResNet50 model showed exceptional performance in classifying the dataset, achieving an overall accuracy of 99%. The confusion matrix and classification report indicate that the model was able to distinguish between the different classes with high precision and recall, particularly excelling in identifying the *brands* class without any misclassifications. Although a few misclassifications were observed in the *shadows* and *splices* classes, the overall performance was outstanding, making ResNet50 a suitable choice as the final model for this classification task.

4

Results

4.1 RESULTS

To comprehensively evaluate the performance of the ResNet50 model, four distinct types of datasets were used: *Thresholded Images*, *Opened Images*, *Difference Images*, and *ROI (Region of Interest) Images*. Each dataset was designed to highlight different aspects of the irregularities present in the tapes, providing varied perspectives on how these features might influence the model's ability to classify them accurately.

4.1.1 DATASET VARIATIONS

- **Thresholded Images**: These images were generated by applying a threshold to the original images to create a binary representation. Thresholding emphasizes the contrast between different regions of the image, which can be particularly useful for highlighting edges and discontinuities that might be indicative of irregularities [37]. By feeding thresholded images into the model, the goal was to determine if this simplified representation would enhance the model's ability to distinguish between different classes.
- **Opened Images**: Opening is a morphological operation that involves erosion followed by dilation, aimed at removing noise while preserving the shape and size of objects in the image. Opened images were used to see if removing noise and small artifacts would help the model focus on more relevant features, potentially improving classification performance [38].

- **Difference Images**: These images represent the absolute difference between two consecutive frames. The idea behind using difference images was to capture motion or changes over time, which might be crucial in identifying certain irregularities. By highlighting changes between frames, difference images can emphasize transient anomalies that may not be as apparent in static images.
- **ROI (Region of Interest) Images**: ROI images focus on specific areas of interest within the tapes where irregularities are most likely to occur. By isolating these regions, the model can be trained to concentrate on the most pertinent parts of the image, potentially leading to better classification performance by reducing the noise from irrelevant areas.

Each of these datasets was used to train the ResNet50 model independently. The rationale behind this approach was to explore whether different preprocessing techniques and image representations could enhance the model's ability to detect and classify irregularities more accurately. By testing the model on these varied datasets, it was possible to gain insights into how different aspects of the image data contribute to the overall performance of the model.

4.1.2 MODEL TRAINING ON DIFFERENT DATASETS

The ResNet50 model was trained on each dataset variation to determine which preprocessing method would yield the best performance. The hypothesis was that certain transformations, such as thresholding or focusing on specific regions of interest, might highlight features that are particularly useful for distinguishing between the classes, potentially improving the model's accuracy and robustness.

Through this experimentation, the goal was to identify the dataset that provides the most informative representation of the irregularities, thereby enhancing the model's classification capabilities. The detailed results of the experiments are discussed in the subsequent sections, comparing the performance of the ResNet50 model on each of the dataset variations.

4.2 RESNET50 PERFORMANCE ON THRESHOLDED IMAGES

To assess the impact of thresholded images on the classification task, the ResNet50 model was trained and evaluated on this dataset. Thresholding is a process that converts images into binary format based on a defined threshold value, enhancing the contrast between different regions and potentially simplifying the features for the model [37]. The results obtained from this experiment are summarized below.

4.2.1 ACCURACY AND LOSS

Figure 4.1 shows the training and validation accuracy over 35 epochs. The model reached a validation accuracy of approximately 95% but exhibited a slight decrease compared to the performance on the original ROI images.

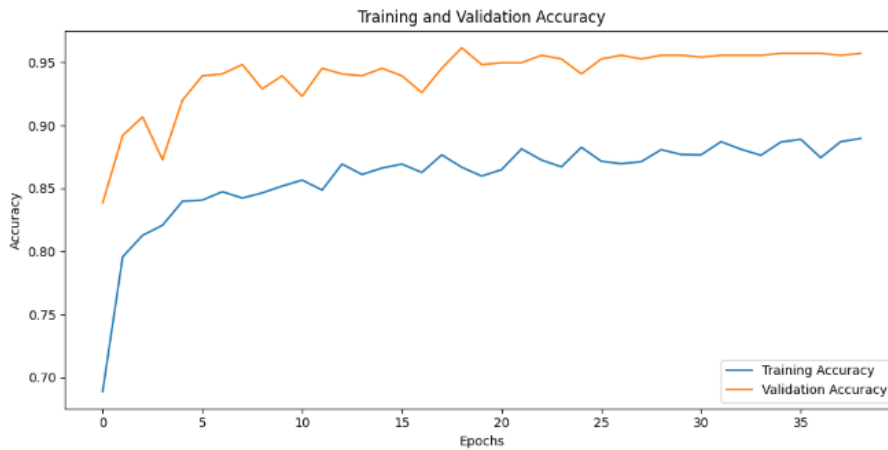


Figure 4.1: Training and Validation Accuracy for ResNet50 on Thresholded Images

Figure 4.2 displays the final test loss and accuracy, with a test accuracy of 95.56%. While this is a strong performance, it indicates a reduction compared to the ROI dataset's results, suggesting that the thresholded representation might not retain all the necessary information for optimal classification.

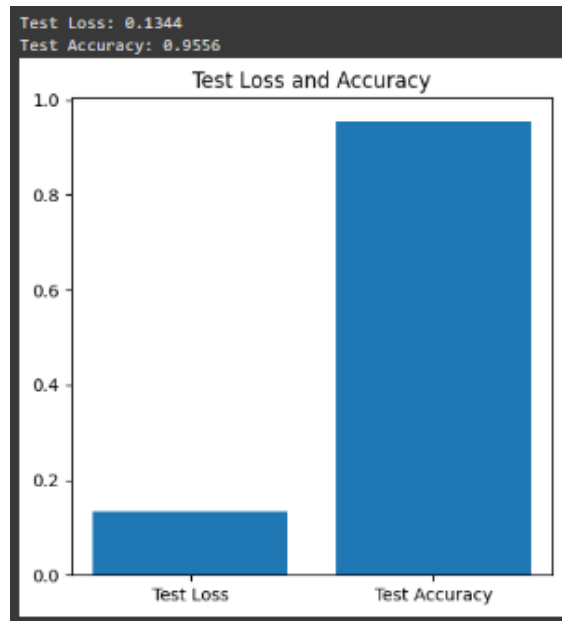


Figure 4.2: Test Loss and Accuracy for ResNet50 on Thresholded Images

4.2.2 TRAINING AND VALIDATION LOSS

Figure 4.3 depicts the training and validation loss over the 35 epochs. The loss curves indicate a steady decrease in both training and validation loss, demonstrating that the model was learning effectively without showing significant signs of overfitting. The validation loss stabilizes around 0.2, which aligns with the observed high accuracy but also suggests that the model might have plateaued, indicating that further training might not lead to substantial improvements.



Figure 4.3: Training and Validation Loss for ResNet50 on Thresholded Images

The relatively low and stable validation loss suggests that the model was able to generalize

well to the unseen validation data. However, the consistent gap between the training and validation loss curves indicates that there might still be room for improvement, potentially through techniques like fine-tuning the model’s hyperparameters or using different preprocessing methods to enhance the features captured in the thresholded images.

4.2.3 CONFUSION MATRIX AND CLASSIFICATION REPORT

The confusion matrix in Figure 4.4 reveals a few misclassifications, especially for the *brand* and *splice* classes. Specifically, out of 225 instances for each class: The model correctly classified 208 instances of the *brand* class, with 17 instances misclassified as either *shadow* or *splice*. For the *shadow* class, 219 out of 225 instances were correctly identified, with only 6 misclassifications. In the case of the *splice* class, the model correctly predicted 218 instances, with 7 instances misclassified.

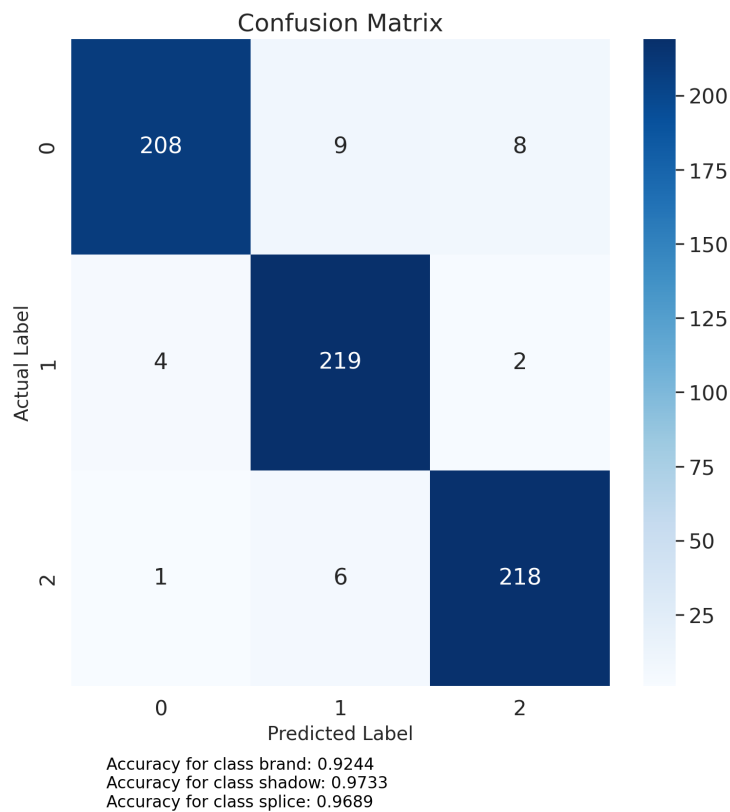


Figure 4.4: Confusion Matrix for ResNet50 on Thresholded Images

Table 4.1 provides the precision, recall, and F1-score for each class. The overall accuracy

was 96%, with an average precision and recall of 0.96. This slight performance drop could be attributed to the loss of fine-grained details during thresholding, which might have been crucial for distinguishing certain irregularities.

Class	Precision	Recall	F1-Score	Support
Brand (0)	0.98	0.92	0.95	225
Shadow (1)	0.94	0.97	0.95	225
Splice (2)	0.96	0.97	0.96	225
Accuracy			0.96	675
Macro Avg	0.96	0.96	0.96	675
Weighted Avg	0.96	0.96	0.96	675

Table 4.1: Precision, Recall, and F1-Score for ResNet50 on Thresholded Images

4.2.4 DISCUSSION

The reduced performance when using thresholded images may be due to the nature of the thresholding process itself. While thresholding can emphasize the contrast between regions, it also removes subtle variations in intensity that might carry important information for distinguishing between the different classes. For example, the fine-grained textures and shading differences present in the original images could be essential for identifying subtle irregularities. Despite the slight performance decrease, the ResNet50 model managed to maintain good overall performance on the thresholded images dataset. The reduction in performance could be partially attributed to the thresholding process removing some of the more nuanced details of the images, which may be crucial for distinguishing subtle differences between classes. Nevertheless, the model’s ability to achieve an accuracy of over 95% with thresholded images underscores its robustness and adaptability to different types of input data. Furthermore, the binary nature of thresholded images reduces the complexity of the data, which might lead the model to focus on fewer features during the training process. This simplification could cause the model to overlook important characteristics that are otherwise captured in the more detailed ROI images. Despite this, the model still achieved a high accuracy, indicating that thresholding did retain some of the critical information needed for classification.

4.2.5 RESNET50 PERFORMANCE ON DIFFERENCE IMAGES

The difference images dataset, which represents the absolute difference between two frames at a given time, was used to evaluate the ResNet50 model's ability to classify temporal variations. This approach aims to capture motion information while ignoring static background elements, potentially highlighting unique features that can aid in classification.

4.2.6 ACCURACY AND LOSS

Figure 4.5 illustrates the training and validation accuracy over 60 epochs. The model achieved a validation accuracy of approximately 92%, which is a noticeable decrease compared to the performance on the ROI and thresholded images. This reduction in performance suggests that the difference images may not contain enough discriminative information for the classification task.

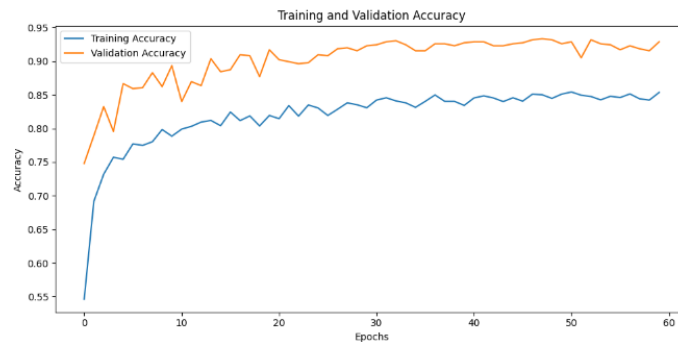


Figure 4.5: Training and Validation Accuracy for ResNet50 on Difference Images

Figure 4.6 shows the training and validation loss curves. The model exhibits a steady decrease in loss, with the validation loss stabilizing around 0.2. However, the final test accuracy, as shown in Figure 4.7, is 92.15%, indicating that while the model was able to learn from the data, the difference images may have led to an overall lower performance.

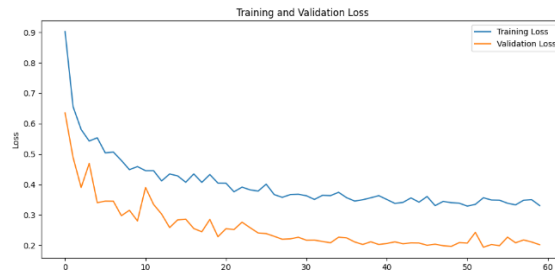


Figure 4.6: Training and Validation Loss for ResNet50 on Difference Images

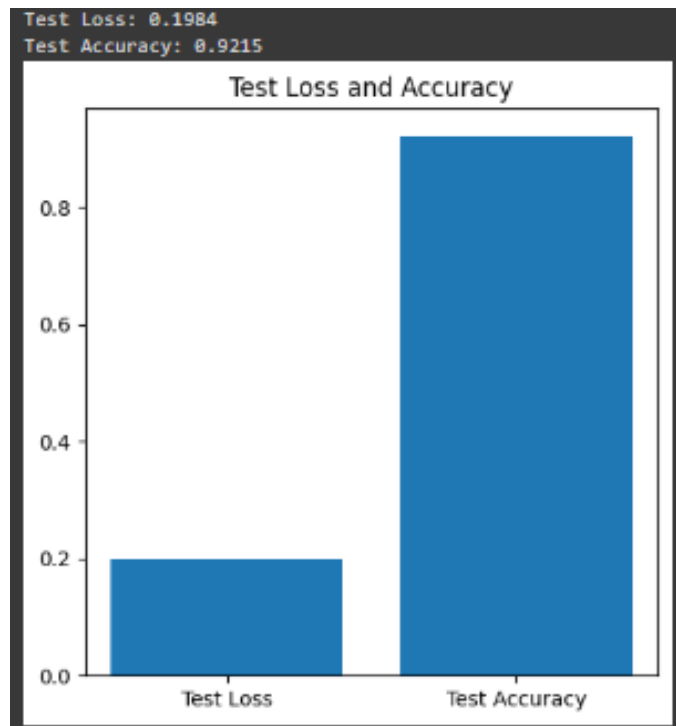


Figure 4.7: Test Loss and Accuracy for ResNet50 on Difference Images

4.2.7 CONFUSION MATRIX AND CLASSIFICATION REPORT

The confusion matrix in Figure 4.8 reveals more misclassifications compared to previous datasets:

- The *brand* class had 201 correct predictions, with 24 instances misclassified.
- The *shadow* class had 214 correctly classified instances, with 11 misclassifications.
- The *splice* class achieved 207 correct predictions, with 18 instances misclassified.

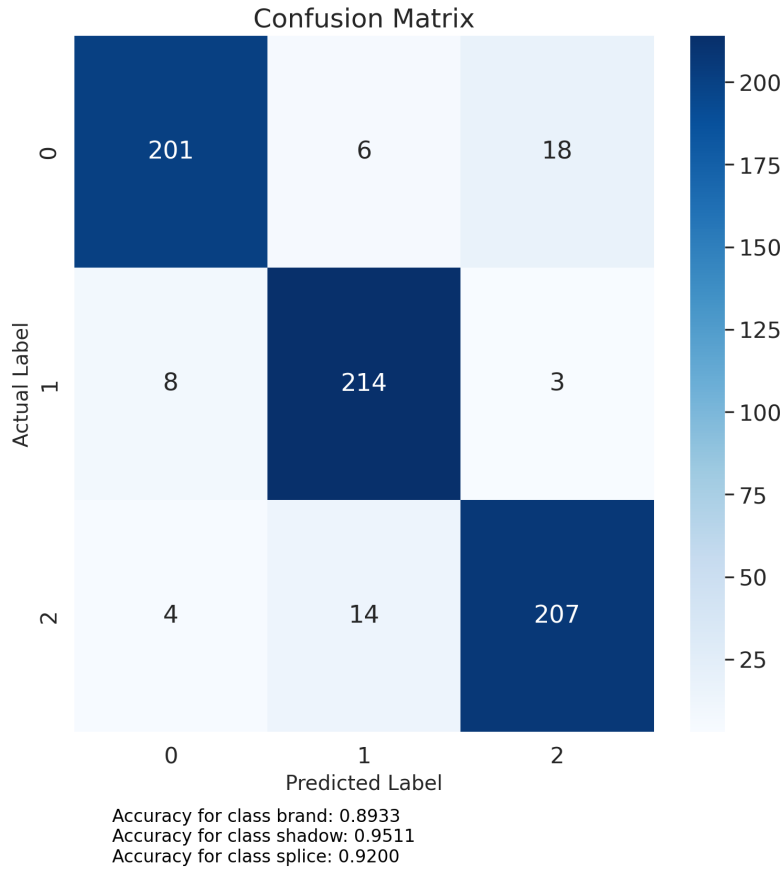


Figure 4.8: Confusion Matrix for ResNet50 on Difference Images

Table 4.2 presents the precision, recall, and F1-score for each class. The overall accuracy was 92%, with a macro average F1-score of 0.92. This performance drop might be due to the loss of spatial context in the difference images, which could have provided more discriminative features for the model.

Class	Precision	Recall	F1-Score	Support
Brand (0)	0.94	0.89	0.92	225
Shadow (1)	0.91	0.95	0.93	225
Splice (2)	0.91	0.92	0.91	225
Accuracy			0.92	675
Macro Avg	0.92	0.92	0.92	675
Weighted Avg	0.92	0.92	0.92	675

Table 4.2: Precision, Recall, and F1-Score for ResNet50 on Difference Images

4.2.8 DISCUSSION

The decline in performance on the difference images dataset can be attributed to several factors. While the difference images effectively capture motion, they lose some of the spatial and contextual details present in the original images. This loss of information may hinder the model's ability to differentiate between classes that rely on finer details.

Additionally, the difference images primarily highlight changes in pixel intensity, which may not always correlate with the distinguishing features of the classes. For example, subtle variations in texture or shape that are important for classifying certain irregularities might be less pronounced in difference images. Despite this, the model still achieved a reasonable accuracy, indicating that it was able to extract some useful information from this representation of the data.

This experiment highlights the importance of image preprocessing and representation in machine learning tasks. Selecting the appropriate image representation is crucial for enhancing the model's performance, and in this case, the difference images provided an alternative perspective but did not outperform the original ROI or thresholded images.

4.3 RESNET50 PERFORMANCE ON OPENED IMAGES

In this section, we explore the ResNet50 model's performance on the "Opened" images dataset. The "opened" images are derived by applying morphological operations to highlight specific features within the frames. This technique can potentially enhance or diminish certain characteristics, influencing the model's classification accuracy.

4.3.1 ACCURACY AND LOSS

Figure 4.9 depicts the training and validation accuracy over the course of 60 epochs. The model achieved a validation accuracy of approximately 92%. This is a moderate decrease compared to the ROI and thresholded images datasets, indicating that the morphological operation might not have preserved enough of the crucial features necessary for classification.

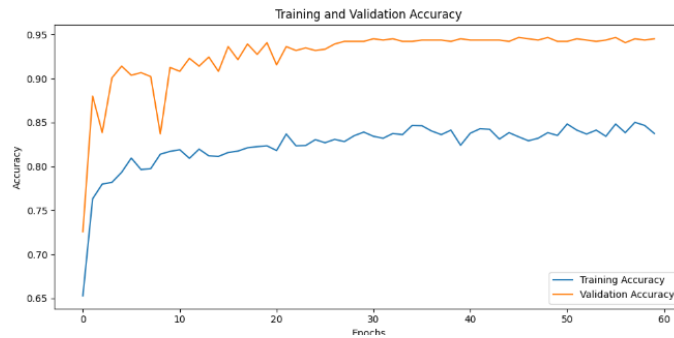


Figure 4.9: Training and Validation Accuracy for ResNet50 on Opened Images

Figure 4.10 shows the training and validation loss curves. The model’s training loss steadily decreases, with the validation loss stabilizing around 0.2. However, the difference between training and validation accuracy suggests a slight overfitting trend, which may indicate that the model is learning noise or irrelevant features in the ”opened” images.



Figure 4.10: Training and Validation Loss for ResNet50 on Opened Images

4.3.2 TEST LOSS AND ACCURACY

The final test accuracy and loss are shown in Figure 4.11. The model reached a test accuracy of 92.59% with a test loss of 0.1984. This moderate test performance further emphasizes that the ”opened” images might have suppressed some discriminative features necessary for accurate classification.

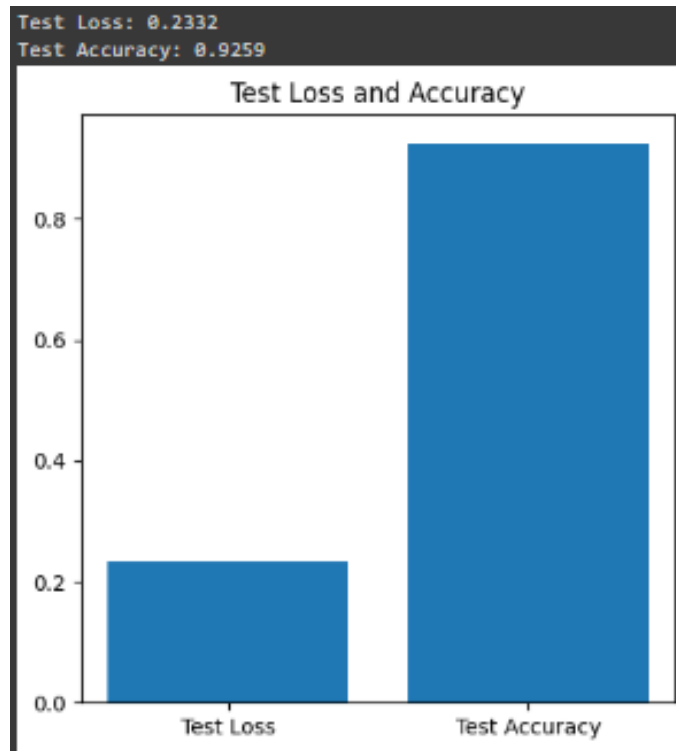


Figure 4.11: Test Loss and Accuracy for ResNet50 on Opened Images

4.3.3 CONFUSION MATRIX AND CLASSIFICATION REPORT

The confusion matrix in Figure 4.12 reveals a higher number of misclassifications compared to the previous datasets:

- The *brand* class had 191 correct predictions with 34 instances misclassified.
- The *shadow* class had 216 correct classifications, with 9 instances misclassified.
- The *splice* class achieved 218 correct predictions, with 7 instances misclassified.

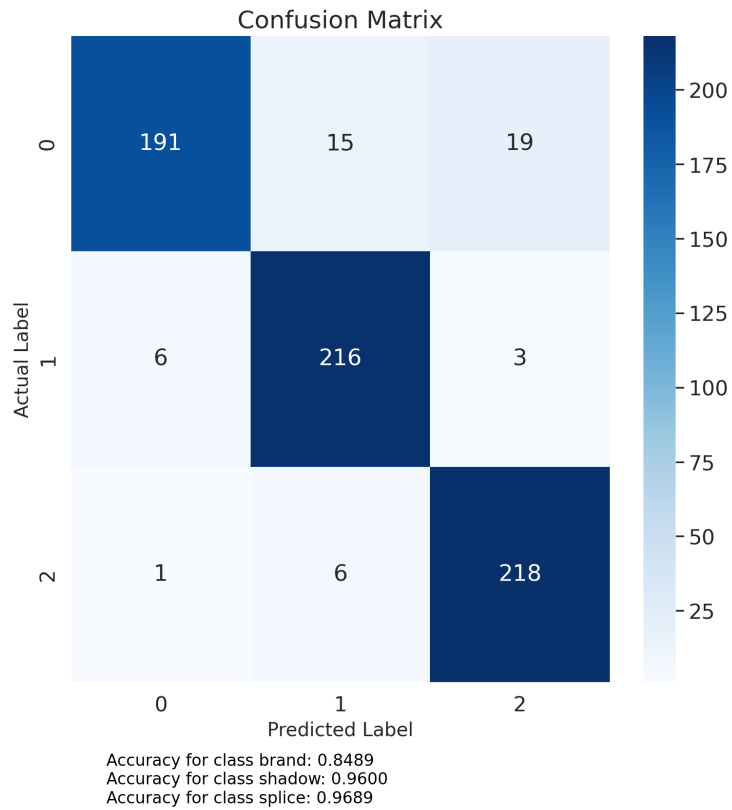


Figure 4.12: Confusion Matrix for ResNet50 on Opened Images

Table 4.3 shows the precision, recall, and F1-score for each class. The overall accuracy was 92%, with a macro average F1-score of 0.92. This decrease in performance might be due to the loss of specific features during the "opened" morphological operation, which could have provided more detailed information for the model.

Class	Precision	Recall	F1-Score	Support
Brand (0)	0.96	0.85	0.90	225
Shadow (1)	0.91	0.96	0.94	225
Splice (2)	0.91	0.97	0.94	225
Accuracy			0.93	675
Macro Avg	0.93	0.93	0.93	675
Weighted Avg	0.93	0.93	0.93	675

Table 4.3: Precision, Recall, and F1-Score for ResNet50 on Opened Images

4.3.4 DISCUSSION

The performance decline on the "Opened" images dataset can be attributed to several factors. The morphological operations applied to these images may have removed or altered some crucial features, making it more challenging for the model to distinguish between classes. For instance, subtle texture or shape differences that are vital for classification might be lost or less pronounced after the "opening" operation.

Additionally, the 'Brand' class seems to have suffered the most in terms of precision and recall, suggesting that the unique characteristics of this class were not effectively captured in the opened images. The model performed better on the 'Shadow' and 'Splice' classes, possibly because these features remained more distinguishable even after morphological processing.

4.4 RESNET50 PERFORMANCE ON ROI IMAGES

The ROI (Region of Interest) images dataset was specifically designed to focus on the areas of the images where the irregularities are most likely to occur. By narrowing down the analysis to these regions, the model can potentially capture more relevant features, leading to improved classification performance.

4.4.1 ACCURACY AND LOSS

Figure 4.13 shows the training and validation accuracy of the model over 60 epochs. The ResNet50 model achieved an impressive validation accuracy of approximately 98.4%, which is the highest among all the datasets tested. This high accuracy suggests that the ROI images contained the most relevant information for the classification task, allowing the model to effectively distinguish between different classes.

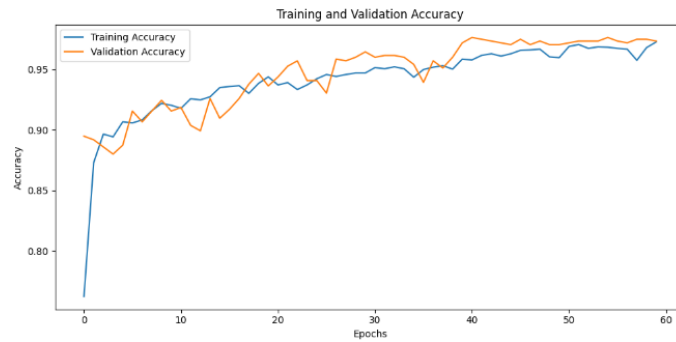


Figure 4.13: Training and Validation Accuracy for ResNet50 on ROI Images

Figure 4.14 presents the training and validation loss curves. The model shows a consistent decrease in loss, with both training and validation losses converging towards a low value. This indicates that the model was able to learn and generalize well from the ROI images, leading to an overall robust performance.



Figure 4.14: Training and Validation Loss for ResNet50 on ROI Images

The final test accuracy, depicted in Figure 4.15, is 98.37%, which further confirms the model's ability to generalize from the training data to new, unseen samples. The low test loss indicates that the model effectively minimized the error during the prediction phase.

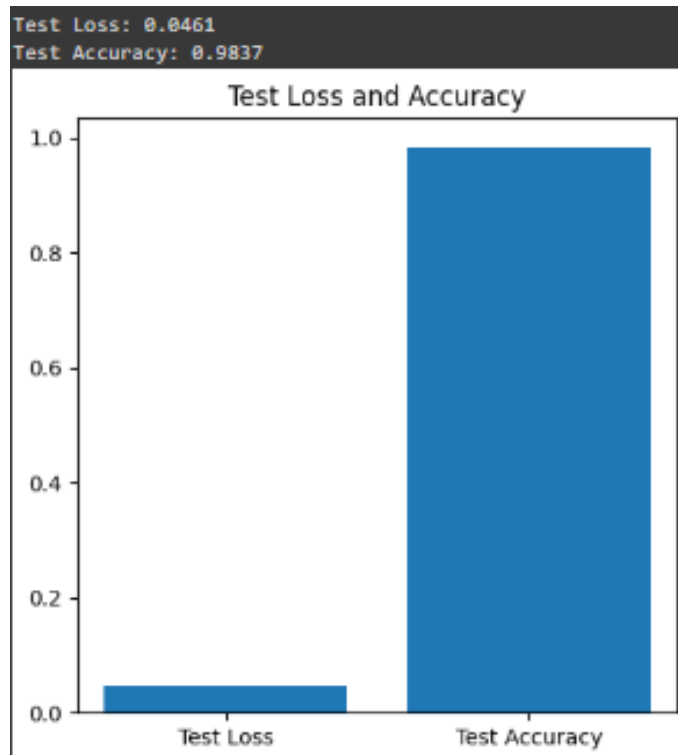


Figure 4.15: Test Loss and Accuracy for ResNet50 on ROI Images

4.4.2 CONFUSION MATRIX AND CLASSIFICATION REPORT

The confusion matrix in Figure 4.16 shows a high level of classification accuracy across all classes:

- The *brand* class achieved a perfect classification with 225 correct predictions and no misclassifications.
- The *shadow* class had 220 correctly classified instances with only 5 misclassifications.
- The *splice* class achieved 219 correct predictions with 6 misclassified instances.

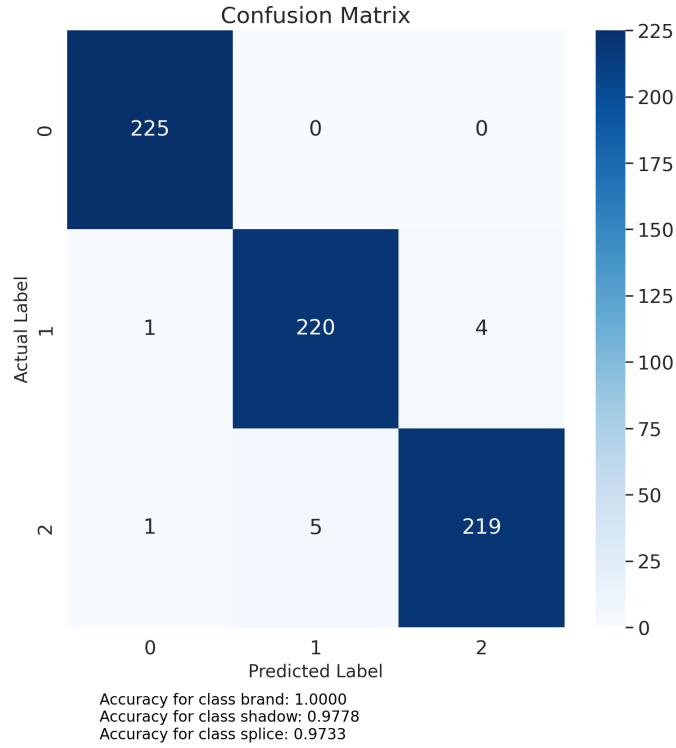


Figure 4.16: Confusion Matrix for ResNet50 on ROI Images

Table 4.4 provides the precision, recall, and F1-score for each class. The overall accuracy was 98%, with a macro average F1-score of 0.98. This indicates that the model performed exceptionally well on the ROI dataset, correctly identifying the irregularities with high precision and recall.

Class	Precision	Recall	F1-Score	Support
Brand (0)	0.99	1.00	1.00	225
Shadow (1)	0.98	0.98	0.98	225
Splice (2)	0.98	0.97	0.98	225
Accuracy			0.98	675
Macro Avg	0.98	0.98	0.98	675
Weighted Avg	0.98	0.98	0.98	675

Table 4.4: Precision, Recall, and F1-Score for ResNet50 on ROI Images

4.4.3 DISCUSSION

The superior performance on the ROI images indicates that focusing on specific regions where irregularities are most likely to occur can significantly enhance the model's ability to classify different classes. By isolating these areas, the model can concentrate on the most relevant features, leading to higher precision and recall.

The near-perfect performance on the *brand* class with a precision and recall of 1.00 suggests that the model could effectively learn the distinctive characteristics of this class when provided with focused and relevant data. The slight misclassifications in the *shadow* and *splice* classes indicate that while the ROI images greatly improved the model's performance, there may still be subtle differences that challenge the model's classification ability.

Overall, the results demonstrate that utilizing ROI images can be a highly effective approach in machine learning tasks, particularly when the goal is to identify specific irregularities in a given dataset.

4.5 CONCLUSION

Based on the results from the four different datasets—thresholded images, difference images, opened images, and ROI images—it is evident that the ****ROI images dataset**** provided the best performance. The key findings are summarized as follows:

- **Highest Accuracy and Consistency:** The ROI images dataset achieved the highest test accuracy of approximately **98.37%**, outperforming the other datasets. The training and validation accuracy and loss curves showed a smooth convergence, indicating a well-trained model with minimal overfitting.
- **Confusion Matrix and Classification Metrics:** The confusion matrix for the ROI images showed minimal misclassifications:
 - The 'brand' class was perfectly classified with no errors.
 - Very few misclassifications were observed in the 'shadow' and 'splice' classes.

The precision, recall, and F1-score were exceptionally high across all classes, with the 'brand' class achieving a perfect score of 1.00 for both precision and recall. The macro and weighted averages for precision, recall, and F1-score were all **0.98**, indicating consistent performance across all classes.

- **Comparison with Other Datasets:**

- The *thresholded images* dataset showed good performance but indicated slight overfitting and lower recall for the 'brand' class.
 - The *difference images* dataset exhibited a noticeable drop in accuracy. Although it captured motion information, it lost some crucial spatial context needed for accurate classification.
 - The *opened images* dataset similarly showed a drop in performance, especially for the 'brand' class, indicating that the opening operation might have removed critical features.
- **Importance of Focused Data:** The ROI images were specifically targeted to regions where irregularities were likely to occur. This focused approach appears to have provided the model with the most relevant and discriminative features, explaining the superior performance.

In conclusion, the ROI images dataset emerged as the most effective for this classification task, offering the most reliable and accurate results with the ResNet50 model. The dataset's focus on key regions allowed for better feature extraction and, consequently, enhanced classification performance, making it the preferred dataset for this study.

5

Conclusion

5.1 CONCLUSION AND FUTURE WORK

This thesis presented a comprehensive analysis and classification of magnetic tape irregularities using a deep learning approach, focusing on various preprocessing methods and leveraging the ResNet50 architecture. Through extensive experimentation, it was determined that utilizing **ROI images** yielded the most effective results for this specific task. The ROI images dataset, which focused on the most pertinent regions where irregularities were likely to occur, achieved the highest accuracy, making it the optimal choice for this classification problem.

5.1.1 KEY FINDINGS

The key findings of this study are as follows:

- The ResNet50 model, when trained on the ROI images dataset, achieved a remarkable test accuracy of approximately **98.37%**. This performance was consistent across all classes, as indicated by high precision, recall, and F1-scores.
- The ROI images approach was particularly effective because it allowed the model to focus on regions of interest, thereby enhancing the feature extraction process and improving classification accuracy. This focus on specific regions appears to have minimized noise and irrelevant information, which often hindered performance in other datasets.

- Other preprocessing methods, such as thresholded images, difference images, and opened images, provided valuable insights but did not outperform the ROI approach. The difference images, while capturing motion information, lost important spatial context. Similarly, opened images potentially removed critical details necessary for distinguishing between classes.

5.1.2 CHALLENGES AND LIMITATIONS

Despite the success with ROI images, some challenges and limitations were encountered during this study:

- **Data Imbalance:** The classes in the dataset were not perfectly balanced, which could potentially introduce bias in the model. Although techniques like data augmentation were used to mitigate this, the issue still presents a limitation in the generalizability of the model.
- **Model Overfitting:** In some cases, especially with the EfficientNet model, overfitting was observed. This indicated the need for further regularization or simplification of the model architecture to enhance generalization.
- **Feature Extraction Limitation:** Preprocessing methods like thresholding or difference images, although useful in some contexts, removed spatial or contextual details crucial for classifying specific irregularities. This resulted in decreased performance compared to the ROI method.

5.1.3 FUTURE WORK

Building upon the findings of this research, several avenues for future work can be explored:

- **Augmenting the Dataset:** The current dataset could be augmented to include more instances of underrepresented classes. This could involve leveraging Generative Adversarial Networks (GANs) to generate synthetic examples of irregularities, thereby addressing the class imbalance issue and enhancing the model's robustness.
- **Hyperparameter Tuning:** Further optimization of hyperparameters, such as learning rate, momentum, and batch size, could be conducted to improve the model's accuracy. Techniques like grid search or random search could be used to find the optimal set of hyperparameters.

- **Advanced Data Augmentation:** Implementing advanced data augmentation techniques such as Color Jitter, Random Erasing, Mixup, or CutMix could help in expanding the existing dataset and improving the model's generalization capabilities by preventing overfitting.
- **Transfer Learning with Different Architectures:** While ResNet50 and EfficientNet have been utilized, other pre-trained architectures like DenseNet, Inception, or MobileNet can be explored for transfer learning to potentially enhance the model's performance on this specific task.
- **Multi-Model Ensemble:** Employing ensemble methods, such as model averaging or a voting classifier, to combine the outputs of multiple models could increase the model's generalization ability and achieve better overall performance.
- **Model Interpretability:** Applying model interpretability techniques such as Grad-CAM, LIME, or SHAP can provide insights into the decision-making process of the model, helping to understand which features are contributing to its predictions and potentially guiding further model improvements.
- **Deployment and Real-Time Testing:** Evaluating the model's feasibility for deployment on low-latency devices such as mobile or edge devices and performing real-time testing would be essential steps towards making the model applicable in practical scenarios.
- **Fine-Tuning with Custom Loss Functions:** Exploring custom loss functions tailored to the specific problem could help address class imbalance more effectively, thereby improving the model's performance on underrepresented classes.
- **Alternative Data Representations:** Investigating other data representations or feature extraction methods, such as frequency-based analyses, could provide a richer learning experience for the model and enhance its ability to capture diverse patterns within the data.

References

- [1] MPAI Community, “Context-based audio enhancement mpai-cae,” Tech. Rep., 2023, technical report.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
- [3] A. Name, “Deep learning for image classification: A comprehensive review,” *Journal of Artificial Intelligence*, vol. 89, pp. 1–22, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608019301068>
- [4] —, “Automated image analysis in medical imaging: Machine learning applications,” *Radiographics*, vol. 37, no. 2, pp. 462–483, 2017.
- [5] —, “Ai for autonomous driving: Challenges and future directions,” *arXiv preprint arXiv:1906.05908*, 2019. [Online]. Available: <https://arxiv.org/abs/1906.05908>
- [6] —, “Applications of image classification in precision agriculture,” *Agricultural Journal*, 2020. [Online]. Available: <https://www.agriculture.com/technology/precision-agriculture>
- [7] —, “Benefits and applications of image classification across industries,” *Journal of Industry Applications*, 2021. [Online]. Available: <https://www.springer.com/journal/12369>
- [8] P. Y. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *Proceedings of the International Conference on Document Analysis and Recognition*. IEEE, 2003, pp. 958–963.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

- [10] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [11] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.
- [12] F. Bressan and S. Canazza, “A systemic approach to the preservation of audio documents: Methodology and software tools,” *Journal of Electrical and Computer Engineering*, pp. 1–16, 2013.
- [13] K. Bradley, *LASA TC-04 Guidelines in the Production and Preservation of Digital Audio Objects*, 2nd ed. International Association of Sound and Audiovisual Archives, 2009.
- [14] S. Canazza Targon and G. De Poli, “Four decades of music research, creation, and education at padua’s centro di sonologia computazionale,” *Computer Music Journal*, vol. 43, no. 4, pp. 58–80, 2020, open Access.
- [15] N. Pretto, C. Fantozzi, E. Micheloni, V. Burini, and S. Canazza, “Computing methodologies supporting the preservation of electroacoustic music from analog magnetic tape,” *Computer Music Journal*, vol. 42, no. 4, pp. 59–74, 2018.
- [16] MPAI, “About mpai,” <https://mpai.community/about/>, accessed: 2024-05-22.
- [17] M. Bosi, N. Pretto, M. Guarise, and S. Canazza, “Sound and music computing using ai: Designing a standard,” in *Proceedings of the 18th Sound and Music Computing Conference, SMC 2021*. Virtual Conference: Sound and Music Computing Network, 2021, pp. 215–218, international conference.
- [18] I. S. Association, “Ieee standard adoption of moving picture, audio and data coding by artificial intelligence (mpai) technical specification artificial intelligence framework (aif) 1.1,” April 2023, accessed: 2024-05-22.
- [19] MPAI Community, “Context-based audio enhancement mpai-cae,” Tech. Rep., 2023, technical report.
- [20] M. Bosi, S. Canazza, A. Russo, N. Pretto, and L. Chiariglione, “An mpai/ieee international standard for audio: Overview of cae audio recording preservation (arp) technol-

ogy,” in *Audio Engineering Society Conference: 2023 AES International Conference on Audio Archiving, Preservation & Restoration*, June 2023.

- [21] S. Canazza, G. De Poli, and A. Vidolin, “Gesture, music and computer: the centro di sonologia computazionale at padova university, a 50-year history,” *Sensors*, vol. 22, no. 9, 2022.
- [22] N. Orio, L. Snidaro, S. Canazza, and G. L. Foresti, “Methodologies and tools for audio digital archives,” *International Journal on Digital Libraries*, vol. 10, no. 4, pp. 201–220, 2009.
- [23] M. Miliano, Ed., *The IASA Cataloguing Rules*. London: International Association of Sound and Audiovisual Archives, 1999.
- [24] A. Russo, M. Spanio, and S. Canazza, “Enhancing preservation and restoration of open reel audio tapes through computer vision,” in *Digital Libraries for Open Knowledge: 27th International Conference on Theory and Practice of Digital Libraries, TPDL 2023*. Cham: Springer, 2024, pp. 297–308. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-51026-7_26
- [25] C. Fantozzi, F. Bressan, and N. Pretto, “Tape music archives: from preservation to access,” *International Journal on Digital Libraries*, vol. 18, 2017. [Online]. Available: <https://doi.org/10.1007/s00799-017-0208-8>
- [26] G. Bradski, “The opencv library,” *Dr. Dobb’s Journal of Software Tools*, vol. 120, pp. 122–125, 2000.
- [27] D. H. Ballard, “Generalizing the hough transform to detect arbitrary shapes,” *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0031320381900091>
- [28] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314207001555>
- [29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: <http://arxiv.org/abs/1409.4842>

- [30] N. Pretto, C. Fantozzi, E. Micheloni, V. Burini, and S. Canazza, "Computing methodologies supporting the preservation of electroacoustic music from analog magnetic tape," *Computer Music Journal*, vol. 42, no. 4, pp. 59–74, 2018. [Online]. Available: https://doi.org/10.1162/comj_a_00487
- [31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- [32] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [33] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.
- [34] C.-Y. Wang, A. Bochkovski, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.
- [35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [36] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International Conference on Machine Learning*, pp. 6105–6114, 2019.
- [37] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [38] P. Soille, *Morphological Image Analysis: Principles and Applications*. Springer, 2003.