Comprehensive Analysis and Development of Deep Learning Models for Bengali Character's Spectrogram Image Classification in Child Speech: Introduction of Spectro SETNet

by

Syed Istiaque Ahmed
20101273
Md. Jubayer Hossain
20101470
Kayes Mohammad Bin Hoque
20101471
Mahmadur Rahman Tusher
20101005
Sajedur Islam
23141093

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Syed Istiaque Ahmed
20101273

Kayes Mohammad Bin Hoque
20101471

MD. Jubayer Hossain
20101470

Mahmadur Rahman Tusher
20101005

Sajedur Islam
23141093

Approval

The thesis/project titled "Comprehensive Analysis and Development of Deep Learning Models for Bengali Character's Spectrogram Image Classification in Child Speech: Introduction of Spectro SETNet" submitted by

1. Syed Istiaque ahmed (20101439)
2. Kayes Mohammad Bin Hoque (20101471)
3. MD. Jubayer Hossain (20101470)
4. Mahmadur Rahman Tusher (20101005)
5. Sajedur Islam (23141093)

Of spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in May, 2024.

Examining Committee:

Supervisor:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Nishat Nayla
Contractual Faculty
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, Ph.D.
Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

In a rapidly developing linguistic technology, the key role of phoneme recognition consists of understanding language and language learning. The research will be framed where a recognition system is developed for the language of Bangla—vowels, consonants, and numbers for children of age three to six years. By adopting advanced approaches like technological methods and classical phonetic education, the spectrogram images of the Bengali children we investigate are classified. Among the techniques associated with modern machine learning (ML) the pervasive techniques are image recognition and large language models (LLM) which have extended to the less explored domain of Bangla phoneme spectrogram image recognition. From our group of 21 participants, we have generated balanced 31,147 spectrogram images - a new dataset that we have created from scratch. This is because the dataset was done meticulously to serve as a complete resource for researchers of Bangla-speaking children's phoneme recognition. Therefore, we then trained ten pre-existing deep learning models that were capable of interpreting and optimizing their performance in Bangla phoneme recognition by using our dataset. Based on these, the SENet model stood out among other existing models with a high performance of 96. 89% accuracy on our testing data set. The ResNet50 and VGG19 models produced the best outcomes among the deep learning models tested which ranked second and third respectively with an accuracy of 88. 8% and 87%. Based on these findings, we propose a novel architecture, Spectrogram SE-Transformer Block Network (Spectro-SETNet), which is a hybrid of the ResNet50 model to which the SE and Transformer blocks have been added, in order to cope with more complicated data and to limit the computational power. The original hypothesis is that the model not only improves the accuracy of Bengali speech recognition for children but also offers a new standard for more complex data processing with less computational power.

Keywords: Automatic Speech Recognition, Character's Recognition, Deep Learning, Mel-Frequency Spectrogram, Spectro-SETNet.

Acknowledgement

First and foremost, we express our deepest gratitude to the Almighty Allah, whose blessings enabled us to complete this thesis without significant obstacles.

We would also like to extend our heartfelt thanks to our supervisor, Md. Golam Rabiul Alam, and co-supervisor, Nishat Nayla, for their invaluable guidance and support throughout this project. Their assistance was crucial and readily available whenever we needed it.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bengali is the fifth most used native language on the globe, and it has around 300 million native speakers, while 37 million of the population has added it as a second language. In the number of speakers counted, it secures the 7th position among all languages of the world. Furthermore, it comes among the top five Indo-European languages.[21] Besides this, speech recognition technology has proven to be a new breakthrough for machine learning and artificial intelligence fields. The applications of this technology are so wide-ranging that they cut through diverse niches, such as virtual assistants, customer service chatbots, accessibility aids, and language learning platforms.[28] The major significance of improving Bangla Character Recognition in children lies in the fact that the child's voice is quite different. The proposed methodology targets their characteristic speech patterns. Consequently, it supports the successful beginning of Bangla language learning. When there is a deficit or absence of tools performing the function of language learning for children this project bridges that gap, which contributes to a better pronunciation and which is key to the academic readiness of the children in the phase of the critical acquisition of language.

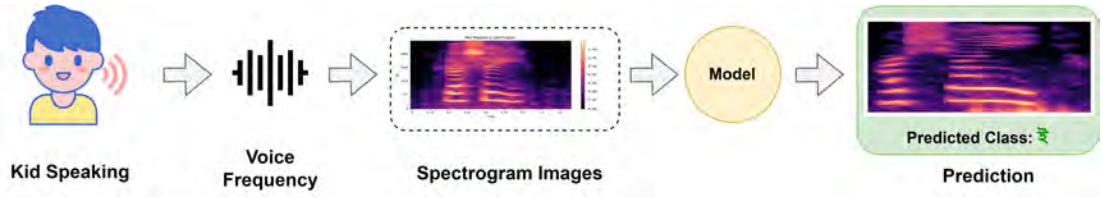

Figure 1.1: Spectogram Image Classification using ML

However, enhancing Bangla character recognition for children faces several challenges and difficulties. Understanding children's speech in Bangla isn't easy. Kids don't sound like adults, they may not speak clearly, and not much has been done in Bangla before. Too few resources, linguistic complexity, and the demand for an easy and noise-resistant system make it harder. Utilizing what's special about how kids talk, understanding their pronunciation, and tackling all these difficulties are key. We need to face all of these barriers to develop a perfect voice recognition system for the children who started learning Bangla. Furthermore, there is a significant lack of proper datasets capturing the pronunciation of children, let alone Bangla-speaking children.

1.1 Role of Machine Learning in Enhancing a Bangla character Recognition

Machine learning plays a big part in shaping character recognition systems. Considering numerous Bangla voice samples, for instance, machine learning algorithms can pick out unique audio patterns from various characters. It's simpler for the system to spot and break down the phonetic sounds from young kids. Boosting machine learning has tons of benefits. In addition, quick feedback helps kids improve their speaking skills. Moreover, the approach provides personal learning experiences by knowing a child's learning style and pace. However, it can be efficient for collecting reliable data regarding children's language development and might prove beneficial in the future for evaluating and exploring how language develops. By harnessing the capabilities of machine learning, this system can adapt to the child's progress, providing customized feedback and adapting the difficulty level to ensure a tailored learning experience.[25]

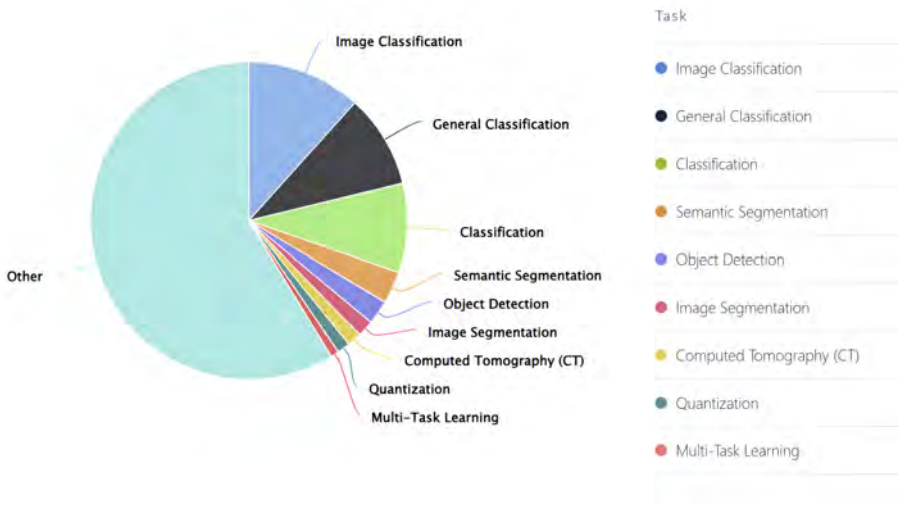

Figure 1.2: Applications of Machine Learning Algorithoms

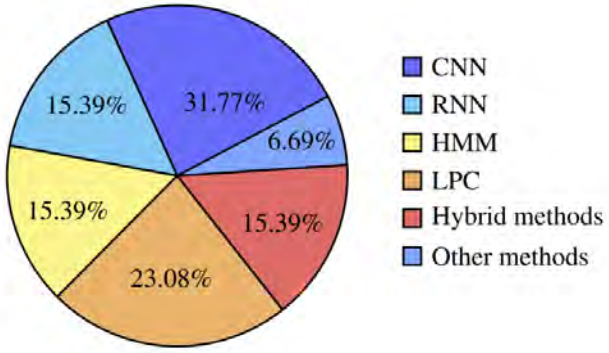

Figure 1.3: Methods used in Bangla speech processing and recognition.

1.2 Advancements in ML for Bangla character Recognition System for Children

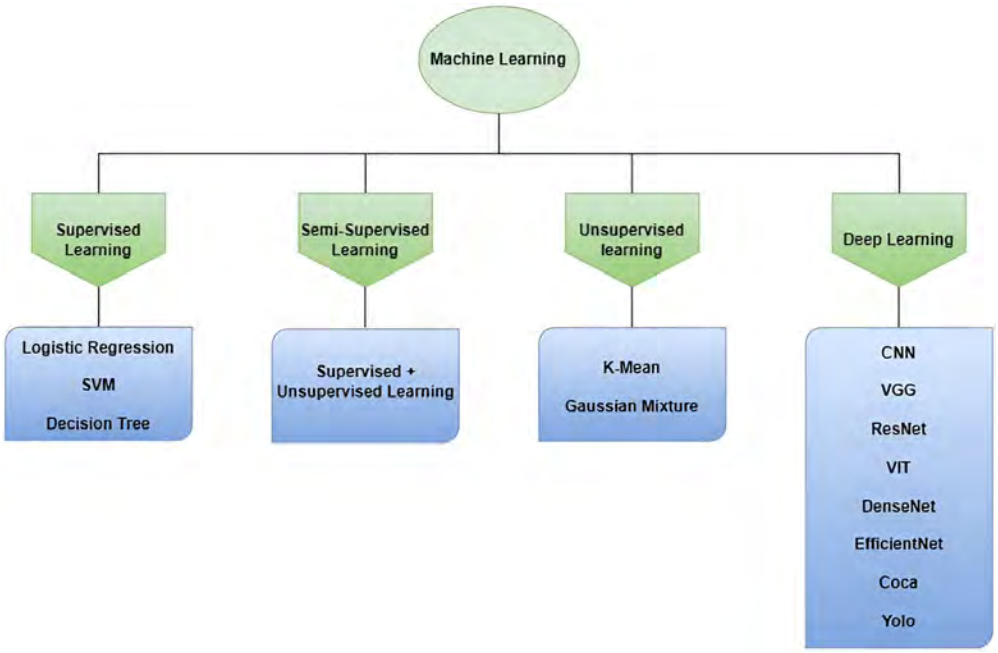

Figure 1.4: Machine Learning Algorithms for Image Classifications

From birth, infants exhibit remarkable proficiency in absorbing linguistic input from their surroundings. Despite their limited ability to pronounce words correctly, which makes it difficult for computer vision systems to interpret.[1] This is where machine learning (ML) steps in—by training with appropriate datasets, ML algorithms can be tailored to detect and understand what these children are attempting to communicate.
Convolutional neural networks, also known as CNNs, have become very popular for computer vision problems. However, CNNs can also be successfully applied to speech-related tasks. For example, CNNs have shown great promise in character recognition from audio signals. CNNs contain multiple layers of convolutional and pooling layers that learn representations of the input data in a hierarchical manner. When used for phoneme recognition, CNNs are able to analyze the spectral patterns and temporal relationships within speech signals. Through this analysis, CNNs extract crucial features that are important for accurate character identification. The features CNNs learn from speech aid in distinguishing between different phonemes.

VGG19 is a CNN framework encompassing 19 layers, incorporating convolutional and pooling layers. It is distinguished for its consistent structure and simplicity, rendering it more accessible to comprehend and execute. VGG19 acquires discriminative attributes from audio spectrograms or alternative speech representations to categorize characters precisely. By employing VGG19, the Bangla character recognition system can derive advantages from its proficiency in capturing intricate patterns and enhancing recognition precision.

SENET, which stands for Squeeze and Excitation Networks, uses a special technique that it bumps up the important features and pushes down the ones that are irrelevant. This makes the network better at picking things apart. SENET is effective for recognizing characters, or sounds, especially in consideration of Bangla spoken by kids. It's good at zeroing in on the sounds when it matters. It could also lead to better accuracy and a model that's easily fine-tuned to the specific speech patterns of children.

ResNet50 is a deeper CNN architecture that addresses the vanishing gradient problem through residual connections. With 50 layers allowing information to pass unimpeded across its depth, ResNet50 has the capability to learn both simple low-level features as well as complex high-level features by capturing both local speech characteristics within segments as well as global characteristics across the full speech signal. This innovative network configuration has demonstrated spectacular outcomes on various computer vision tasks. It holds promise to also prove helpful for a Bangla character recognition system seeking to hone its feature extraction abilities and thereby elevate recognition precision if utilized to identify distinguishing aspects within the input audio and distinguish between different phoneme classes.

On the other side, ResNet18 presents a lighter alternative to the standard ResNet architecture that is better optimized. Specifically created for identifying Bangla phonemes in educational content meant for children, ResNet18 achieves a nice balance between how complex it is and how many resources it needs. This makes it viable for use in settings with constrained processing power, such as schools with older technology. While Inception V3 likely delivers slightly better performance, ResNet18 strikes a fair compromise between effectiveness and feasibility of deployment in environments with limited computational horsepower.

The field of character recognition has seen significant advancements through the integration of sophisticated neural network architectures. Among these, DenseNet has proven especially beneficial due to its unique structure that connects each layer to every other layer in a feed-forward fashion. This connectivity pattern ensures maximum information flow between layers, which is crucial for recognizing the subtle nuances in character sounds. This architecture's efficiency in feature extraction makes it particularly effective in handling the variances in children's speech, which often present challenges in speech recognition systems. Figure 1.5 demonstrates the continuous improvements of machine learning algorithms in image classification.

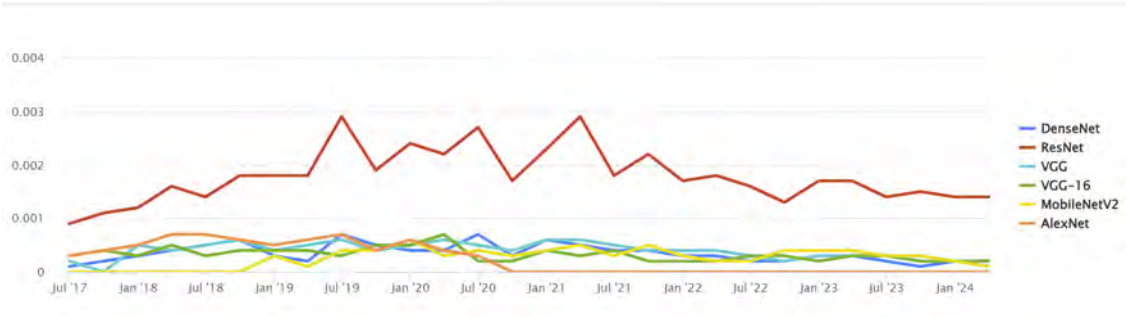

Figure 1.5: Continuous Improvement of Machine Algorithms

1.3 Problem Statement

Recognizing 'voice' in the Bengali language is not an easy task, but it is very enthralling, especially for child-targeted applications. It is hard to improve the Bangla Automatic Speech Recognition (ASR) quality for children as it demands an in-depth comprehension of the patterns of their language. The speech styles of kids are constantly changing due to factors such as their age, developmental stages, and places of residence. These alterations can lead to difficulty for current ASR systems to be effective. While they are good at understanding adult speech in Bengali, they stumble a little with child speech in Bengali.

Prior to a couple of years, Bengali automatic speech recognition methods were quite narrow-minded and they mostly focused on recognizing isolated words that are not in the context or specific accent which leads to a great gap in continuous streams of speech like understanding a speech. To establish an interactive educational environment, first and foremost we should develop technologies that are good at accurately processing the interrupted flows of sound by kids. The lack of a full-fledged Bangla speech sample database, especially one that can be used to tailor for children from different regions of the country, becomes a big hindrance to developing the necessary speech recognition models accurately for child users.

The up-to-date deep learning modes were employed along with a new proposed architecture to complete our work. These tools give them assurance in trying to understand the nature of Bangla-speaking children's complex language. However, to apply them is extremely difficult. This may cause errors, such as overfitting. In addition, an idiosyncratic accent and speech habits are what we have to cope with.

Rephrase with these issues and chances for progress, one important question stands out:

How can deep learning models, utilizing spectrogram image classification, be effectively integrated into the development of a Bangla character Recognition System for children to enhance recognition accuracy and adaptability to diverse speech patterns and regional accents?

The speech recognition system in Bangla that we are developing is going to be solid and easy to use for children. Our mission is to address the different types of child pronunciation and the various Bengali accents. Our goal is to develop the first speech recognition technology, especially for children. This study has been carried out to understand the Bangla language sounds, more specifically for kids. We are employing cutting-edge models such as VGG16, VGG19, ResNet18, ResNet50, DenseNet, EfficientNet, Inception V3, MobileNet, SENet, and our custom-developed architecture model as well. Our goal is to sensitively enhance speech recognition for Bengali-speaking kids between three and five ages. The goals of the project are to improve the accuracy and efficiency of these systems; minor improvements have been made already. The research, thus, can be perceived as one of the milestones of the language technology design of children's language learning in the context of Bangla.

1.4 Our Contributions

The study seeks to leverage the power of cutting-edge deep learning models (VGG16, VGG19, ResNet18, ResNet50, DenseNet, EfficientNet, Inception V3, MobileNet, SENet) and Specto-SETNet for optical character recognition on Bangla characters using children's data by emerging an innovative approach. The objectives are multifaceted and designed to address various aspects of language recognition in young learners:

- We created a prime dataset comprising Bangla characters audio recordings, tailored specifically to the speech patterns of Bangali children by addressing the lack of suitable resources in Bangla, particularly for younger demographics.
- We proposed a new architecture for a precise multi class Bangla spectrogram character image recognition system named, Spectro SETNet. which can effectively handle the complexity and variability of Bangla characters in children's speech.
- We comparatively analyzed the selected ten deep learning models on our prime dataset to evaluate their ability to effectively identify and understand complex Bangla characters that are produced by children.

By achieving these goals, the research aims to facilitate a paradigm change in early childhood language literature with deep learning augmented by advanced deep-learning technologies to create inventive tools for inclusive and efficient learning among young Bengali speakers.

Chapter 2

Literature Review

The development of the models employed in Bangla character recognition presents an interesting view toward the development of computational linguistics and machine learning. In contrast to these, the evolution from conventional machine learning models to progressive deep learning algorithms has led to a marked enhancement in the precision and performance of character recognition, especially for vernaculars such as Bangla. With regard to the number of speakers in general, it occupies the seventh spot among all languages. What is more, it is the fifth most popular Indo-European language.[21] In the early stages of life they undertake the journey to learn the complexities of language and communication. This phase is characterized by swift cognitive development, wherein children naturally acquire the sounds, words, and morphological patterns of their mother tongue [1]. Through the benefit of machine learning, this system can respond to the child's development, offering personalized advice and adjusting the level of difficulty to fit the individual learning experience.[1] By harnessing the capabilities of machine learning, this system can adapt to the child's progress, providing customized feedback and adapting the difficulty level to ensure a tailored learning experience.[25] In addition, voice recognition technology is identified as one of the breakthrough factors in the area of machine learning and artificial intelligence.[28]

2.1 Existing Works

In this literature review, Bengali character recognition is investigated using different models to improve the accuracy of speech recognition. Bangla ASR is concerned with phonetic characteristics, improving sentence accuracy, word precision, and overall accuracy. A Bengali feature phonetic table with 22 unique features is created in the review and the possible use of multilayer neural networks (MLN) to transform acoustic properties such as Mel Frequency Cepstral Coefficients (MFCCs) to character likelihood is discussed. These probabilities, along with the delta (Δ) and delta-delta $\Delta\Delta$) parameters are used as inputs for hidden Markov models (HMMs). Although the improvement in the method shows better performance, no specific accuracy metrics are given. Nevertheless, the research has its limitations, such as the lack of ample Bengali speech corpora and the focus on basic vowel and consonant recognition in certain studies, which may have missed the details of the Bengali language.[8]

2.2 Related Works

2.2.1 Foundational Innovations in Bengali Speech Recognition

In relation to research conducted in [9], the authors discuss the main subject of their paper related to the production and evaluation of a specific CNN aimed at the recognition of short vocal commands in the Bengali language. In order to develop a resilient model, the authors carefully gathered a dataset consisting of real-world utterances with noise that contained 10 different Bangla words. Their approach was three-layered. The study referenced as [2], is a study that uses the Bengali speech database and acoustic features to perform Bengali word recognition, focusing on triphone HMM-based classifiers and MFCC38 and MFCC39 models. It concludes that the MFCC39-based approach is superior in recognizing Bengali words and is likely due to the regional accents in the speech data used. The paper [4], analyzes the phonetic features in the Bangla language such as vowels, fricatives, and nasals and their implementation in an ASR system. It describes the construction of a PF table and the establishment of a PF-driven ASR system, as opposed to conventional MFCC-based approaches. The study highlights the system's three stages: However, the PFs are obtained by deriving MFCC features, integrating PFs with a complicated neural network, and combining with a triphone Hidden Markov Model to reveal that the PF based approach is more accurate than the MFCC method in Bangla speech identification.

A significant portion of the research in BSR has been dedicated to traditional models like HMM and Gaussian Mixture Models (GMM). For instance, the study on "Bangla Word Recognition using Acoustic Features" employed a Triphone HMM-based classifier, which compared different acoustic features for Bangla word recognition.[2] The study found that the MFCC39-based system outperformed the MFCC38-based system. Likewise, another investigation concentrated on the identification of standalone Bengali terms by employing MFCC for extracting features, DTW for feature comparison, and SVM with RBF for categorization, resulting in an 86.08% precision level.[4] This paper provides a unique method to better recognize Bengali speech using an E2E system, having the Bengali audio converted to the text using a transfer learning framework. It uses a trained model derived from a 5-gram language model, while post-processing utilizes a Bengali Unicode normalizer. It involves audio to .wav format conversion, feature extraction, and prediction. However, the system's reliance on vast volumes of training data and speed of processing may limit its implementation in low-resource languages and real-time conditions, which requires further optimization and research.[29]

2.2.2 Gender Neutral Bengali Speech Recognition

In reference [3], a Bangla Automatic Speech Recognition (ASR) system is designed which improve accuracy in word and sentence recognition across genders using gender-specific HMM-based classifiers. This approach applied to a corpus of 3000 male and 3000 female speakers, gets its best results at the third mixture components. Future work involves combining gender-independent classifiers with neural network-

based systems. The Gender-Independent Bangla ASR technique notably achieves sentence accuracy rates of 68.95%, 78.65%, and 87.30% for different mixture components. However, it faces challenges such as the need for a substantial training corpus and limitations in handling non-binary gender identities or unconventional speech patterns. Despite these issues, the technique shows promise in improving speech recognition by addressing gender variations.

Nevertheless, BSR has a lot of challenges despite the progress. Recognition systems' accuracy can be influenced greatly by gender differences, speaker accents, and regional dialects. A paper on Bangla ASR with Gender Neutrality focused on transgender speech patterns and implemented HMM classifiers that correctly identified male and female speakers yielding outstanding sentence accuracy results.[3] In their paper, Das et al. propose a machine-learning approach for generating Bangla text summaries, using a conditional random field model for theme detection and a clustering algorithm for theme grouping. The approach generates summaries with an IR score and obtains an F1 score of 69.65%. Although only single-themed documents at this stage, the study proposes future developments in Bangladesh NLP through statistical models such as CRFs, machine learning methods like SVM and LSTM networks, and hierarchical thematic word clusters. While the summarization system has 72.15% precision, 67.32% recall, and an F1 score of 69.65%, the theme detection method gives an 83.60% precision, 76.44% recall, and 79.85% F-measure. Though limited by available data and annotator experience, the paper identifies avenues for improvement through increased data and linguistic tool development. The study by Das et al. thus adds to the growing branch of machine learning-based Bangla text summarization and developed avenues for future improvement and development.[23]

2.2.3 Entering a New Era of Multilayer Neural Networks (MLN) for Speech Recognition

This study [5], presents a customized ASR system for Bengali with local parameters and K-means clustering that operates better than the traditional MFCC system in terms of sentence accuracy and word accuracy and improves recognition of female speakers due to better acoustic features. In the study, the LF-25 technique and Triphone Hidden Markov Model are presented which are applied for more accurate recognition of Bengali words. Another research, "Bengali Speech Recognition using RNN" was also based on RNN. It had an accuracy rate of 86.058%.[26] These works demonstrate the ability of deep learning algorithms to improve the accuracy of BSR systems and overcome the challenges created by the Bangla voice-to-text transcription, which is spoken by more than three hundred million people yet understudied in the voice recognition literature. Having used the Bengali Common Voice 9.0 dataset, the authors improve ASR models, especially the n-gram language model enhanced version of the wav2vec2 model. This approach, overcoming dialect diversity and audio quality complexities, sets a new standard in Bengali ASR model performance.

The study [11] an SVM and DTW-based automatic speech recognition system for the isolated Bangla words is proposed. It has MFCC for feature extraction, DTW

for feature matching, and SVM with RBF for data classification. Procedures such as MFCC for audio feature extraction, DTW for time-dependent data comparison, and SVM for exact data classification. The methodology involves feature extraction using MFCC, feature matching using an improved DTW technique, and classification based on SVM and RBF. In a controlled test with 40 speakers pronouncing five Bangla words and a 12-person test, the system accuracy was 86.08%. However, the study is only limited to single-word recognition without any comparison with other models in Bangla speech recognition. Deep learning has shifted the face of BSR. The advent of deep learning has revolutionized the field of BSR. Word error rate obtained in the paper [12], titled "Utilizing Deep Learning for Bengali Speech Recognition" was 4.5% using DNNs paired with LSTM networks.

The study [7] study presents the first Bengali Pronunciation Error Detection System using Hidden Markov Models (HMMs) and Bigram language models. It obtained an 85.5% accuracy on sentences and a 91.5% at the word level but also detected limitations involving sentence-level confusions where conjugate words or certain letters at the beginning could be misplaced. This paper describes a novel Bengali speech recognition approach based on the RNN with offline characteristics, minimal preprocessing, and open-source capability. It uses different models including CNN and RNN, character-based and word-based Hidden Markov Models (HMM). The process is composed of speech recording, noise reduction, and MFCC feature extraction reaching an incredible 86.058% accuracy. Still, it is based on a speaker's dictionary for isolated word recognition and may have diminished precision under aged speakers. However, it has the potential to make the Bengali language speech recognition better not only in Bangladesh but also worldwide.[13]

2.2.4 Integration of AI in Speech Recognition

Artificial intelligence (AI) has achieved tremendous progress with one of the most notable milestones being the advent of machine learning. Machine process algorithms and programming methods for the purpose of analysis, cognition, and training of data. This advancement has resulted in the birth of deep learning, an element of machine learning that involves the design of artificial neural networks that can learn and make autonomous decisions. A real-life example of deep learning, especially in image classification, is to increase efficiency in business, eliminating the necessity of manual classification which would have cost time and money. An example of a real-life application is product classification which aims at reducing price comparison optimization. In this regard, different CNN structures, including VGG16 and ResNet, are compared in terms of accuracy on solving the image classification problems. The intention is to find the best architecture for the task, promoting accurate and fast product classification.[22]

2.2.5 Deep Learning: A Significant Leap Forward in Speech Recognition

In his article 'Transfer Learning for Image Classification Using VGG-16 and Deep Convolutional Neural Networks', Srikanth Tammina starts the transfer learning journey to address the limitations of traditional machine learning techniques. She

writes: Transfer learning reuses the knowledge from the pre-trained model to achieve better accuracy than training from scratch. For example, using my VGG-16 network as a pre-trained model to learn the classification of Tamil characters. Due to the scarcity of labeled data in the target domain, Tammina investigated the effectiveness of transfer learning in improving model accuracy despite limited training samples per category. The study outlines a systematic approach that begins with designing a convolutional neural network (CNN) model using TensorFlow and Keras. Tamina then compared the performance of the base CNN model with variants fine-tuned using image augmentation and a pre-trained VGG-16 model. The results show the significant advantages of transfer learning, with the VGG-16-based method providing superior accuracy in classifying images of dogs and cats. But, as the paper also notes, 'the road to such a perfect transfer learning model is far from easy' due to a lack of computational resources and deep-learning domain experts. Despite its limitations, Tammina states that thanks to transfer learning, if 'we can solve some of the dents in outputs of deep learning models, it will transform the capabilities of machine learning and help us to solve real-world problems in image classification problems along with other fields.' Overall, this paper underscores the power of transfer learning for leveraging what is known from training data to learn about novelties and proves useful today for researchers and practitioners alike who work in the field of deep learning and computer vision.[14]

This paper gives a neural network model for image recognition by means of deep learning, which can automatically extract multiple features for recognition, and its advantage over traditional methods is that. Reasonably, our experiments demonstrate the validity of the model, accuracy, and robustness to noise, greatly improving the image recognition. Deep network and vanishing gradients can be addressed for the case of neural networks, with ResNets solving such an issue on the one hand and allowing training deep networks on the other hand.[18] In recent literature [20], several authors of the latest literature have considered the use of deep learning models for image classification analysis related to plant diseases and their diagnosis. Wu et al (2024) aimed to develop a proficient algorithm for the classification of corn leaf diseases by using the ResNet18-based model. The intention is to raise agricultural yield by giving farmers an opportunity to quickly detect and counteract corn leaf diseases. The study authors provide a detailed description of their approach which involves pre-processing parameters, including image adjustments and enhancements to enhance the homogeneity and diversity of the data. Finally, they applied the dataset to train the model with healthy and diseased leaves of corn and used diagnostic parameters such as accuracy, precision, recall, and F1 score to measure the performance. The benefits of their method include high accuracy, and generalization capability as they show this through the continuous increases in model performance during training. Nevertheless, there could also be the requirement for massive and varied datasets to prevent the model from performance incompetence caused by neglect of different types of diseases or meteorological conditions.

Progress made on deep learning architectures in the place recently has guided the significant workout in many image processing duties, such as segmentation and classification. On top of this, customized U-Net series of models have been very active recently for their success in image segmentation and classification (Ronneberger et

al. , 2015). On the downside, these models tend to use transposed convolution (TC) for their encoders resulting in perceptual loss of discriminative features that are critical for adequate representation. One way to handle this problem is the novel DenseUNet model that relies on dilated/dense TC (Ronneberger et al. , 2015) to offer more precise pooling and sharing of information between different levels of the model. It could be done using four blocks that were built up by dense TC operations and they made it possible for the model to restore the size of features at different scales. So, it led the network to better capture low-level features, in turn, that muttons the ability to identify objects more accurately. The cited paper has indeed successively confirmed DenseUNet's capability to perform image classification. The paper has reported good stability of training in several datasets including CIFAR-10, CIFAR-100, SVHN, and FMNIST (Ronneberger et al. , 2015). This justifies the necessity to find ways, like DenseUNet, to have deep learning models that deal with images more efficiently and robustly used in image classification tasks. [20] [30]

In the study by Joshi et al. (Year) presents a robust framework for classifying sports images based on environmental cues and surroundings, utilizing the InceptionV3 architecture for feature extraction and Neural Networks for classification. The framework demonstrates high accuracy in classifying six sport categories—rugby, basketball, tennis, badminton, cricket, and volleyball—achieving an impressive average accuracy of 96.64% (Joshi et al., Year). This study contributes to the literature by showcasing the effectiveness of deep learning techniques, particularly Neural Networks, in handling large image datasets efficiently and accurately. The comparison with other classifiers such as Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) highlights the superiority of Neural Networks in sports image classification tasks (Joshi et al., Year). Furthermore, the study identifies opportunities for future research, including image preprocessing techniques to enhance accuracy, incorporation of additional sports categories for analysis, and exploration of alternative feature descriptors for improved results (Joshi et al., Year).

Chapter 3

Dataset

3.1 Data Collection

In this research, we have created a unique dataset that contains voice samples taken from Bangla language speaking kids. The collection of our dataset is perfect and specialized, also this kind of dataset is first in the Bangla language as we collected from the field. In this dataset, there are audio files that were taken from 21 kids, these represent 47 Bangla characters and 10 Bangla numeric characters, starting from (অ to ঁ) and (০ to ৯) in a total 57 classes. An average of 25 samples from each character for each child exist here and more than 31,147 spectrogram images are here. All the recordings from a child were captured in a single session. And after that, we converted the audio (.mp3) file into (.Wav) files. Figure 3.2, is a polar or radial histogram, of our class distribution of audio samples.

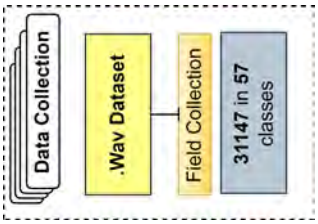

Figure 3.1: Audio Data Collection

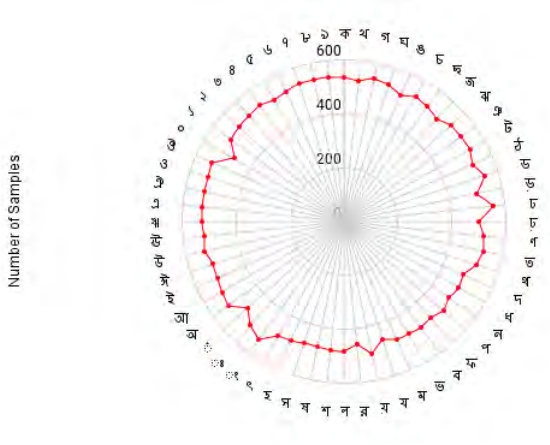

Figure 3.2: Dataset Class Distribution

3.2 Data Cleaning

As shown in Figure 3.3 noise reduction and data cleaning are done with the help of Audacity software. Furthermore, every (.wav) file was segmented into parts to match the individual samples.

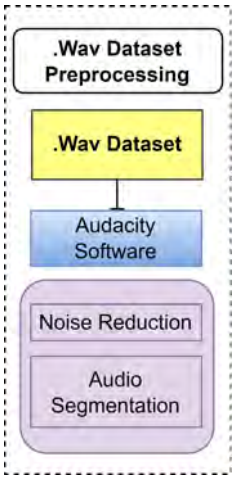

Figure 3.3: Audio Data Cleaning

3.3 Data Conversion

Figure 3.4 illustrates, that we changed the (.wav) files into mel-frequency spectrogram images and kept them in an organized folder management system. For easier access and data handling, a thorough CSV file was created that included the recordings' classifications and also their matching spectrogram image paths.

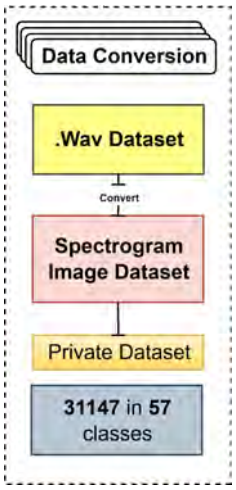

Figure 3.4: Converting Audio Data into Spectrogram Image Data

3.3.1 Dataset Class Distribution

To ensure the dataset's purity and reliability, we have crafted it meticulously from the ground up. Emphasis was placed on recording in a quiet environment to minimize any background noise, thus preserving the clarity of the samples. Recordings compromised by excessive background noise were excluded to maintain the dataset's quality. Subsequently, selected recordings underwent a rigorous cleaning process using Audacity audio software to eliminate any residual background sounds, which might interfere with the analytical processes. This careful curation is vital to boost the precision of any models that will be trained or tested with this dataset.

Character	Number of Samples
অ	528
আ	521
ই	512
ঈ	510
উ	525
ঊ	519
ঋ	526
এ	528
ঐ	525
ও	529
ঔ	534
ক	535
খ	523
গ	541
ঘ	533
ঙ	512
চ	533
ছ	525
জ	511
ঝ	531
ঞ	534
ট	537
ঠ	519
ড	547
ঢ	554
ণ	499
ত	528
থ	512
দ	482

Character	Number of Samples
ধ	489
ন	479
প	494
ফ	478
ব	483
ভ	480
ম	478
য	558
র	454
ল	478
শ	480
ষ	471
স	474
হ	483
ৎ	486
ং	536
ঃ	517
ঁ	477
০	467
১	515
২	523
৩	526
৪	531
৫	518
৬	526
৭	538
৮	538
৯	537
Total	**31147**

Table 3.1: Dataset Class Distribution Table

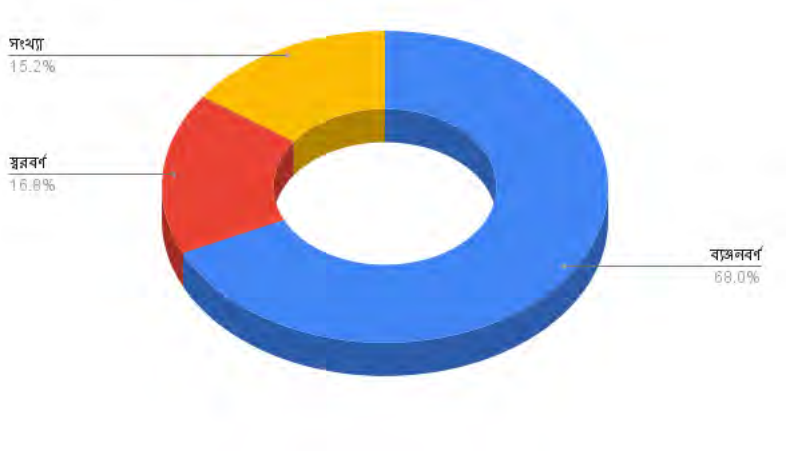

Figure 3.5: Section Distribution

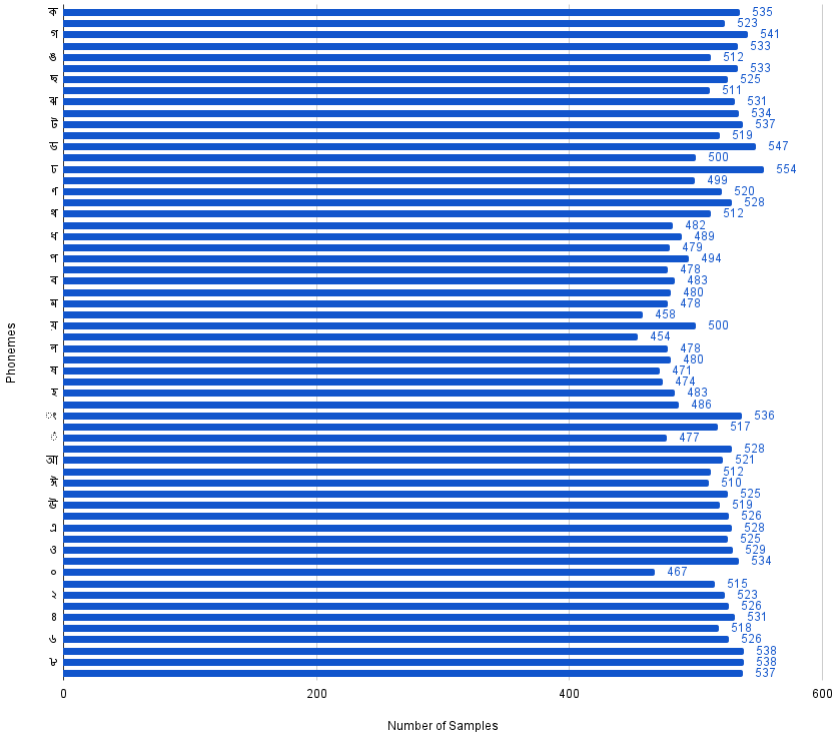

Figure 3.6: Sample Distribution Among Classes

3.3.2 Dataset Gender and Regional Distribution

While gathering data sets, the issue of the impartiality of the data should undoubtedly be in focus. Therefore, the case is always that we give the same measure of data to male, as that of female children. Moreover, to ensure the transparency of the information we provide, we work to include data from as many regions as possible. This method essentially enables us to build an entire and diverse dataset, which is one of the key factors in arriving at representative and unbiased results during our analysis. Through the multicultural and representative aspects that make our findings more accurate and generalizable, we intend to deliver a data set that is specific to the cultural environment we are studying.

Kids (No.)	**Gender**	**Region**
KID (1)	Female	Dhaka
KID (6)	Female	Chittagong
KID (7-8)	Male	Feni
KID (9)	Female	Feni
KID (10)	Female	Shylet
KID (11)	Female	Dhaka
KID (12-15)	Male	Mymenshingh
KID (16-19)	Female	Cumilla
KID (20)	Male	Cumilla
KID (21-22)	Female	Rajshahi
KID (23-25)	Male	Barishal

Table 3.2: Kids Gender and Regional Based Data Distribution

Gender	**Count**
Male	10
Female	11
Total	21

Table 3.3: Kids Gender Based Count

Region	**Count**
Dhaka	2
Chittagong	1
Feni	3
Sylhet	1
Mymenshingh	4
Comilla	5
Rajshahi	2
Barishal	3
Total	21

Table 3.4: Kids Region Based Count

3.3.3 Dataset Folder Organization

Folder Distribution: The dataset is meticulously organized into individual folders for each child participant, labeled from Kid1 through Kid25. This structured distribution facilitates the isolated analysis of spoken number data for each child, enabling precise evaluations and comparisons.

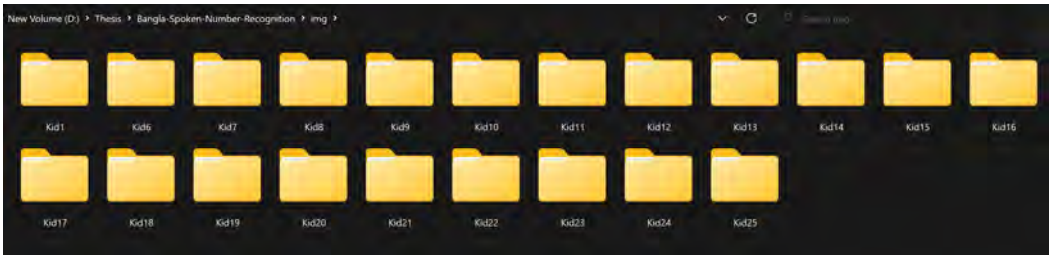

Figure 3.7: Folder Distribution

Sub-Folder Distribution: Within each participant's directory, subfolders are systematically labeled according to a combination of the first digit (representing the child's identifier) and a character.

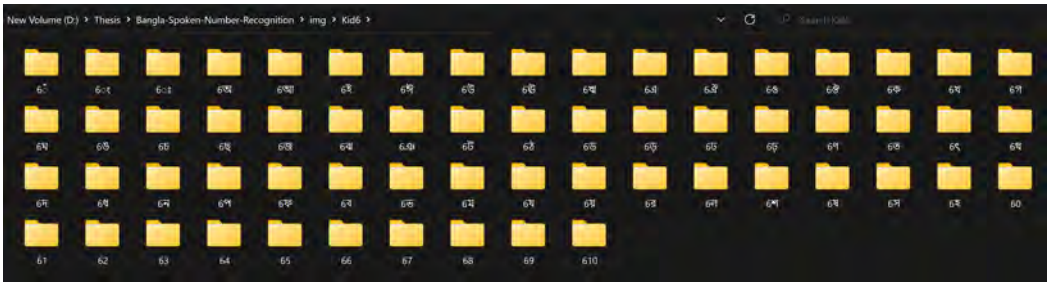

Figure 3.8: Sub-Folder Distribution

Image Distribution: In the subfolder for the Kid folder, each spectrogram image is named solely based on the sequence of its recording, emphasizing the character. This streamlined naming convention facilitates easy identification and access to specific phonetic instances for detailed analysis.

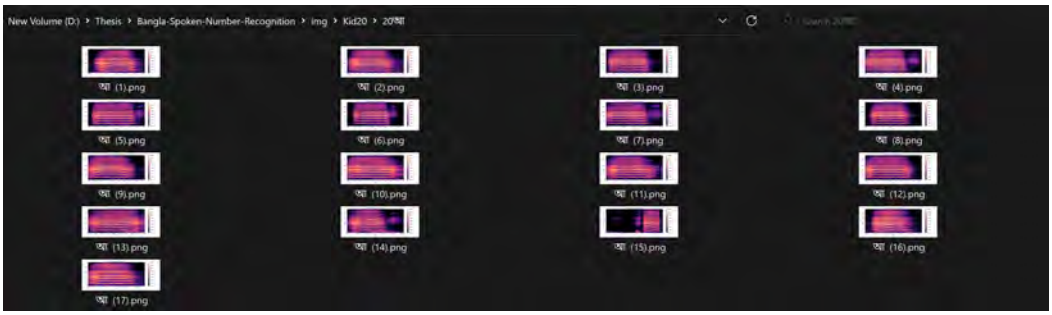

Figure 3.9: Image Distribution

3.4 Data Partitioning (Training and Testing)

Our preprocessing workflow commenced with the retrieval of image paths and their corresponding labels from a carefully assembled CSV file. We maintained the quality of our dataset by eliminating any entries lacking an 'Image Path'. Next, we transformed the categorical labels into a numerical representation, making them suitable for our machine learning algorithms. Employing stratified random sampling, we allocated 80% of the dataset to training and reserved the remaining 20% for testing. Within the training portion, we designated a portion specifically for validation.

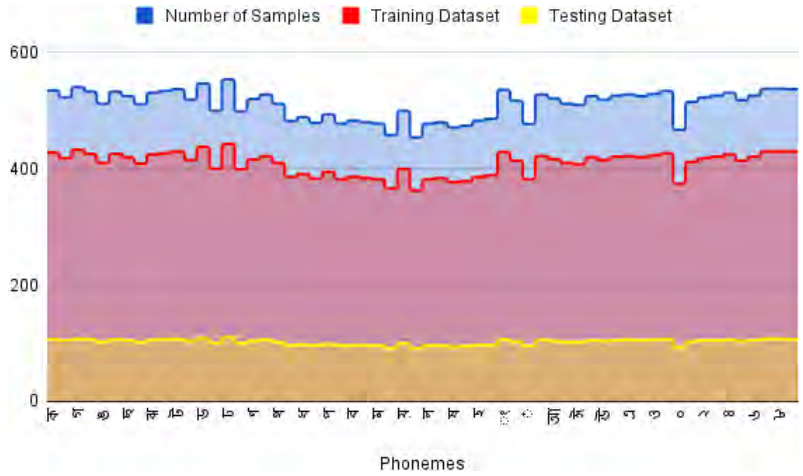

Figure 3.10: Data Partitioning

Characters	Number of Samples	Training Dataset	Testing Dataset
অ	528	422	106
আ	521	417	104
ই	512	410	102
ঈ	510	408	102
উ	525	420	105
ঊ	519	415	103
ঋ	526	421	105
এ	528	422	106
ঐ	525	420	105
ও	529	423	106
ঔ	534	427	107
ক	535	428	107
খ	523	418	105
গ	541	433	108
ঘ	533	426	107
ঙ	512	410	102
চ	533	426	107

Table 3.5: Data Partitioning Table-1

Characters	Number of Samples	Training Dataset	Testing Dataset
ছ	525	420	105
জ	511	409	102
ঝ	531	425	106
ঞ	534	427	107
ট	537	430	107
ঠ	519	415	104
ড	547	438	110
ঢ	554	443	111
ণ	499	416	104
ত	528	422	106
থ	512	410	102
দ	482	386	96
ধ	489	391	98
ন	479	383	96
প	494	395	99
ফ	478	382	96
ব	483	386	97
ভ	480	384	96
ম	478	382	96
য	558	366	92
র	454	363	91
ল	478	382	96
শ	480	384	96
ষ	471	377	94
স	474	379	95
হ	483	386	97
ৎ	486	389	97
ং	536	429	107
ঃ	517	414	103
ঁ	477	382	95
০	467	374	93
১	515	412	103
২	523	418	105
৩	526	421	105
৪	531	425	106
৫	518	414	104
৬	526	421	105
৭	538	430	108
৮	538	430	108
৯	537	430	107

Table 3.6: Data Partitioning Table-2

Chapter 4

Algorithms

4.1 CNN (Convolutional Neural Network)

4.1.1 CNN Preprocessing

In order to improve speech recognition accuracy, the Convolution Neural Network template designed for raw audio should be modified in a different way. The process involves the normalizing of the spectrogram image to make it more uniform across all sections regardless of the time frame. The dataset is comprehensively balanced for all classes of training data by including powerful data augmentation methods that are able to handle variety. We used two augmentation strategies, one of which is a random crop and the other is to make the image laterally flipped so that the CNN can be trained adequately for what diverse data inputs might exist.

4.1.2 CNN Overview

Convolutional Neural Networks are a type of deep learning model. They work like the human visual cortex to recognize images and videos. These networks have convolutional layers with filters that detect things like edges and textures. The filters create feature maps that show the local connections. As the feature maps go through more layers, they become more detailed and specific.[6]

Pooling layers come after the convolutional layers. Their purpose is to make the feature maps smaller and reduce parameters. This helps avoid overfitting and makes computations simpler. Max pooling is commonly used. It selects the highest value from certain regions of the feature maps. CNNs use the Rectified Linear Unit (ReLU) activation function. ReLU brings non-linearity which solves an issue called vanishing gradient. Vanishing gradient is a problem in deep neural networks. ReLU is easy to compute but can still capture complex patterns.

After all the convolutional layers, fully connected layers are used. The job of these layers is to combine and label the features from earlier layers. Neurons in these layers are connected to every neuron in the last convolutional layer. Fully connected layers add composed features together. To stop overfitting, CNNs often use dropout. Dropout randomly ignores some layer outputs during training. This helps the network learn general features instead of memorizing the training data.

CNNs work well with big image data sets. They can see things even when the things move around. Shared weights in layers make CNNs faster. CNNs help spot things in photos and videos. They also help with medical tests and language tasks.

The design of CNNs is inspired by individual cortical neurons of the animal visual cortex, which have receptive fields for stimuli. This biological analogy is reflected in the fact that convolutional layers are local and layered making them efficient and reliable for visual data processing.

4.1.3 CNN Working Procedure

Building a Convolutional Neural Network (CNN) starts with image collection and preprocessing, including normalization (adjusting pixel values within 0 to 1) and resizing for standardization. This helps in gradient descent and corrects input to the network. Second, the CNN architecture is structured with convolutional layers to extract features with filters and activation using the ReLU function. Firstly,Pooling layers reduce dimensions to save from overfitting, fully connected layers infer high level features, while dropout layers prevent overfitting.

Compilation of the model requires optimizers and loss functions, with metrics such as accuracy or precision in evaluating the model. During training, the model will be trained by means of minimizing the loss with the training data, assuming weights based on the predictions and the real labels. The model will be tested on a different dataset, which might cause changes in architecture or hyperparameters. The refined and upgraded model is used in a variety of fields, from the recognition of images to specific uses, which need constant development with new data to maintain its accuracy and relevance. In every step, close monitoring is a must to avoid either overfitting or underfitting and achieve good generalization for new data, hence a productive and satisfying CNN.

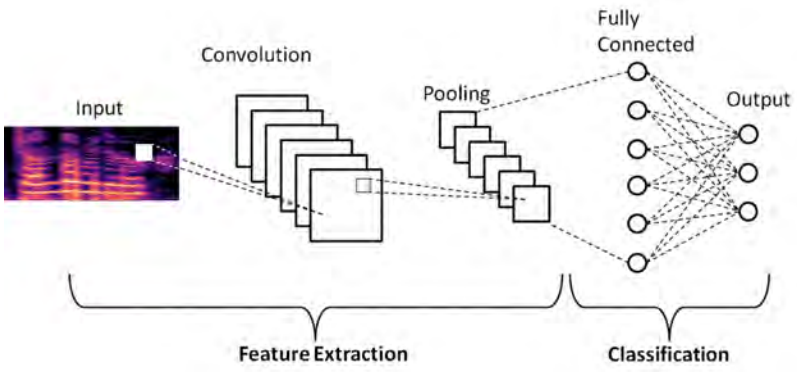

Figure 4.1: CNN Working Procedure

4.2 VGG16

4.2.1 VGG16 Preprocessing

In the VGG16 model, images are to be resized to 224x224 pixels, which match its design. Each pixel value is normalized according to the mean and standard deviation from the ImageNet dataset to emulate the VGG16's training environment with its data. Techniques for data augmentation such as the image scaling, rotations, and horizontal flips are applied to make the model be able to generalize across diverse visual inputs. For that reason, this model became capable of generalizing the data, showing its resilience to various types of dependency.

4.2.2 VGG16 Overview

VGG16 was one of the first models proposed by the Visual Geometry Group (VGG), and is a milestone in the history of Convolutional Neural Networks (CNNs) for image recognition and processing. Its depth and accuracy in the image analysis task provide strong evidence for the power of deep architectures in recognizing highly complex patterns in images. This model is called VGG16 because its original version used 16 convolutional layers (so we could say VGG6 had 6 convolutional layers). VGG16 relies on shall layers of convolutions, each of which detects features (such as lines, edges, textures, and shapes) and smaller 3×3 filters. These are then sensitive to a much more complex range of features, increasing in complexity. The Figure below illustrates these layers with three visual examples that are classified into the same category (airplane type). The colored boxes (blues, greens, and oranges) represent features detected by different layers in the architecture.[15]

If we look closely, we can perceive the sequential nature of these layers: as it goes from left to right, more complex features are detected. VGG16 activations. VGG16 also uses the Rectified Linear Unit (ReLU) activation functions. Remember that activation functions yield a measure of how much each neuron in the neural network is activated. Here, the nonlinearity of each unit in the convolutional layer allows it to learn complex patterns and interactions in the signal. This activation function adds a substantial amount of flexibility to the model and makes it highly appropriate to classify a wide variety of image features. VGG16 was introduced to the ImageNet benchmark dataset for object detection and classification in 2014, which instantly achieved the best results on that day. Nowadays, this model is considered a milestone in the history of CNNs for object detection and classification thanks to its accuracy and low computational capacity. Bottom-up to top-down layers.

4.2.3 VGG16 Working Procedure

VGG16 is an engineered complex convolutional neural network for image processing tasks. It contains a large number of convolutional layers equipped with small filters for extracting features from input images in an increasingly complex manner. Max-pooling layers interspersed between convolutional layers decrease spatial dimensions, hence improving computational efficiency and invariance of features. As input images go through VGG16's convolutional and pooling layers, it progressively identifies

and extracts complex visual features related to classification tasks. The hierarchical process of feature extraction enables VGG16 to develop discrimination based on patterns and attributes from the input images. Later, VGG16 transforms these features into a one-dimensional vector after feature extraction and finally passes them through fully connected layers for feature aggregation and classification. In the last step of classification, SoftMax activation generates class probabilities to conduct accurate image classification. The use of ReLU activation functions in the architecture of VGG16 improves the ability to learn complex patterns and relationships, hence boosting the accuracy of classification. Such a nonlinear activation function is of prime importance in enabling VGG16 to detect and classify various attributes of an image.

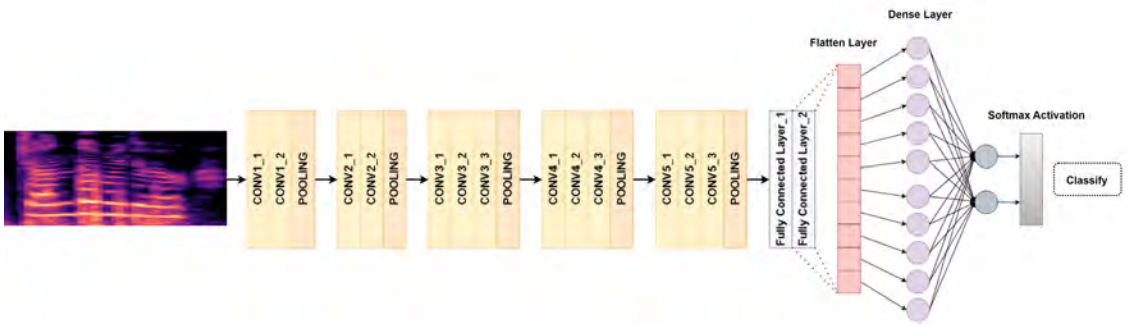

Figure 4.2: VGG16 Working Procedure

4.3 VGG19

4.3.1 VGG19 Preprocessing

In order to ensure that a variety of deep learning models are utilized efficiently, customized preprocessing protocols play a key role. For the VGG19 model, a 224x224 pixel size is first provided for the image. Subsequently, the pixels are scaled after normalization, which follows the ImageNet dataset distribution, and is a stage seamlessly meshing with the VGG19 pre-training. Additionally, a set of data augmentation strategies such as rescaling, rotations, shifts, and flips are put into action. Furthermore, for the images a couple of generator objects are made to do batch processing in parallel mode.

4.3.2 VGG19 Overview

The VGG, developed by the Visual Geometry Group, is an important landmark in the world of CNNs and especially in the domain of image recognition and image processing. It is the deep VGG architecture that makes VGG famous and brings in major progress in the domain when it was introduced. This is manifested by the depth of VGG as demonstrated by the two well-known architectures: VGG-16 and VGG-19, which refer to the number of convolutional layers that they possess. It is these layers that are very important in CNN because they detect features, such as edges, textures, and complex patterns, using different filters. VGG-19, with more layers than VGG-16, is better at understanding and classification of visual

information.[10] Some of these layers contain a convolution with ReLU activation function that introduces nonlinearity to allow the network to learn sophisticated patterns. VGG thus dominated the sphere of object detection, setting new bars in respect to the accuracy-to-efficiency ratio. First and foremost, it flexed its muscles on ImageNet, a huge image database for recognition tasks. The VGG architecture advances from simple to complicated stacks running across a variety of image features and improving object recognition accuracy. It has since pivoted away from its original function and has played a pivotal role in the development of the technology for image classification. Its design principles continue to shape further research and development in this field. One of the main reasons VGGNet holds relevance is that it empowers the ability of CNNs to develop more complex and specialized image recognition and processing systems

4.3.3 VGG19 Working Procedure

VGG19 is a highly complicated deep convolutional neural network that is used for image processing. It uses several convolutional layers of small 3x3 filters that move about the image, progressively extracting features from basic edges and textures in the first layers to more intricate principles in deeper layers. Max-pooling is applied to reduce spatial dimensions, decreasing parameters and enhancing invariance to input shifts. As the generated image passes through these convolutional and pooling layers, VGG19 progressively identifies increasingly complex features, one of the essential ways in which visual information is interpreted. Lastly, classification is based on top-level features after going through all layers. This is achieved by flattening the elements of the ultimate pooling layer into a vector that runs through fully connected layers, where the extracted features are gathered for comprehensive image interpretation. The last one is image classification, where VGG19 transforms the outputs of its last layer into class probabilities using a softmax activation function Neatly, for all layers, VGG19 relies on the ReLU activation function, which introduces nonlinearity into the model if positive, zero otherwise. Deep learning requires this kind of nonlinearity because the data patterns that need to be learned are complex and nonlinear. Overall, VGG19's interpretation consists of successive convolutional and pooling layers for hierarchical feature extraction, flattening, and fully connected layers for information consolidation and classification, based on ReLU activation for nonlinear association learning.

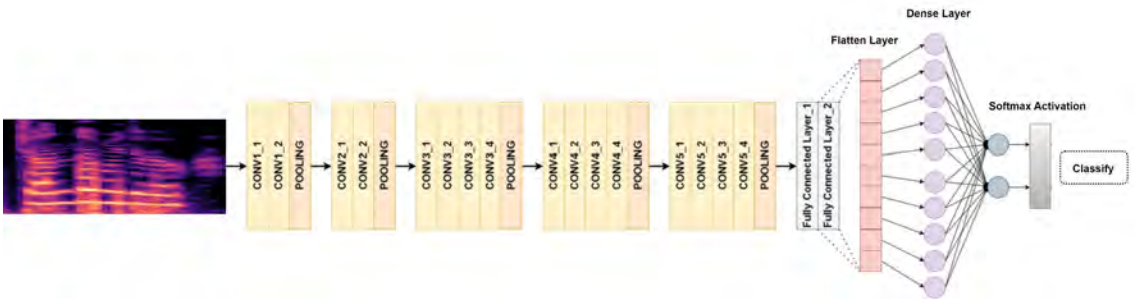

Figure 4.3: VGG19 Working Procedure

4.4 Resnet18

4.4.1 ResNet18 Preprocessing

ResNet18 uses the same preparation steps as other ResNet models like ResNet50. Pictures are made smaller to 224x224 pixels. Normalization adjusts pixel values to match the average and standard deviation of ImageNet. To stop overfitting and speed up training, batch normalization and dropout are used during preparation. These steps help ResNet18 learn better.

4.4.2 Resnet18 Overview

In the domain of deep learning, the ResNet18 architecture acts as a huge contribution, especially in the tasks of image classification. It is a variant of Residual Network, or very commonly known as ResNet, that gives an even more lightweight solution than its deeper counterparts, such as ResNet50, and still delivers astonishing performance. But at its very core, ResNet18 solves a vanishing gradient problem that is very common in deep neural networks: this is done with residual links, which also go by the name of skip connections. These connections create direct paths for the flow of gradients back, thereby enabling the training of deeper networks without having their performance degrade. The architecture of ResNet18 consists of many residual blocks, each comprising a number of convolutional layers. Due to its ability to capture both intricate details and overarching meanings, ResNet18 performs remarkably in extracting hierarchical features from the input images. What sets ResNet18 apart is its accuracy in image classification tasks, whereas it has a relatively shallow depth in comparison to other ResNet models. This, therefore, makes ResNet18 an ideal selection where computational resources are restricted or where faster inference speed is required without hitting performance.[27]

4.4.3 Resnet18 Working Procedure

Working on ResNet18 consists of a logical sequence in how to manage input images: starting with basic preprocessing procedures like resizing and normalization to get rid of variability and create the best conditions for the network's input. Preprocessed input images are fed to the initial convolutional layer of ResNet18, where basic features, such as edges and textures, are identified.

With the process going on, the subsequent convolutional and pooling layers refine and enhance such features, progressively obtaining hierarchical representations of the input images. The main feature of ResNet18 is residual blocks consisting of convolutional layers followed by identity mapping due to skip connections. In fact, these allow gradients to directly flow through the network, thus avoiding the vanishing gradient problem and allowing training with deeper networks. The feature maps, after passing through residual blocks, are fed to global average pooling to collapse the spatial information into a single value per channel. This reduces the computational complexity of the network while retaining critical features.

The output of the global average pooling layer is then flattened and sent to fully connected layers for high-level reasoning and classification. The SoftMax activation

function maps it to yield class probabilities, making the ResNet18 model predict the most likely class for a given input image.

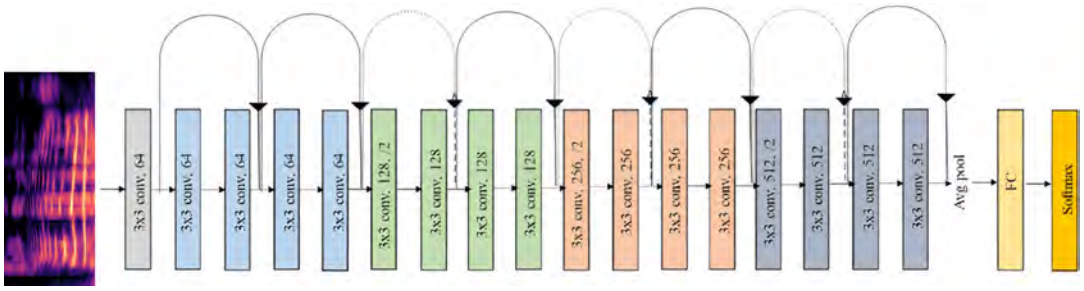

Figure 4.4: ResNet18 Working Procedure

4.5 Resnet50

4.5.1 ResNet50 Preprocessing

ResNet50 needs images to be 224x224 pixels, just like VGG19. First, the pixel values are adjusted based on ImageNet's averages. Then, batch normalization and dropout layers help the training go faster. These layers also prevent overfitting, which is when a model performs poorly on new data.

4.5.2 Resnet50 Overview

ResNet50 is part of the family of ResNets, standing for residual networks, and is represented by 50 layers; these have contributed greatly to the development of deep learning algorithms for image classification tasks. Conventional deep neural networks, as they went deeper, would often have problems with their performance, usually plateauing or degrading. ResNet50 solves this problem, resulting in a deep yet powerful network.

The most important innovation with ResNet is residual connections, also called skip connections. These connections offer solutions to the vanishing gradient problem, a serious problem that arises in deep networks: gradients become ever smaller during backpropagation, making it difficult to learn in the earlier layers. In a typical layer of a neural network, the input is transformed through weights and nonlinear activations to obtain an output. In ResNet, each layer is designed to learn a residual function relative to the layer inputs. In other words, such layers learn the difference between the desired mapping with the identity function. If the identity function is the optimum, the layer can adjust weights toward zero to approximate it. Residual connections create shortcuts to some or even multiple layers, thus easing the direct flow of gradients during backpropagation. It overcomes the vanishing gradient problem because gradients need not pass through a long chain of diminishing transformations. Notably, these residual connections allow the network to learn identity functions when it needs to, thus enabling it to bypass redundant layers while retaining important features.[19] ResNet, including ResNet50, is one of the most excellent state-of-the-art architectures on image recognition tasks and has created records in

many competitions and benchmarks. Its depth and revolutionary residual connections resolve vanishing gradient issues, enabling the training of deep neural networks and setting new records in image recognition tasks.

4.5.3 Resnet50 Working Procedure

ResNet50 is very good at sophisticated image recognition tasks by structurally processing images through defined stages. First, it resizes images uniformly to 224x224 pixels, making input handling easy and standardizing the scale of pixel values to a common range for efficient convergence during training. After preprocessing, it is the beginning of deep learning through the first convolutional layer by filtering the input image to produce different feature maps of the image aspects, such as edges and textures. Convolution and max-pooling steps follow, reducing spatial dimensions, reducing the computational load, and preventing overfitting. ResNet50 is all about the four stages of residual blocks; each stage has numerous residual blocks that are themselves made up of three convolutional layers, which fiddle with the image features by sometimes down-sampling and elsewhere up-scaling the space dimensions, hence controlling network depth with no added complexity. Critical innovations include shortcut connections within residual blocks, allowing the network to skip layers and propagate gradients during training. That helps with the vanishing gradient problem and learning identity functions, where deeper layers are not worse than shallower ones. Toward the end of the network, global average pooling was introduced, mapping each feature map into a single value, reducing parameters and computations, hence helping avoid overfitting. The output is further flattened and fed to a fully connected layer for classification. The SoftMax activation function creates a probability distribution over the recognized classes, where the class that has the highest probability is the final prediction, and the process of image processing and recognition in ResNet50 is complete.

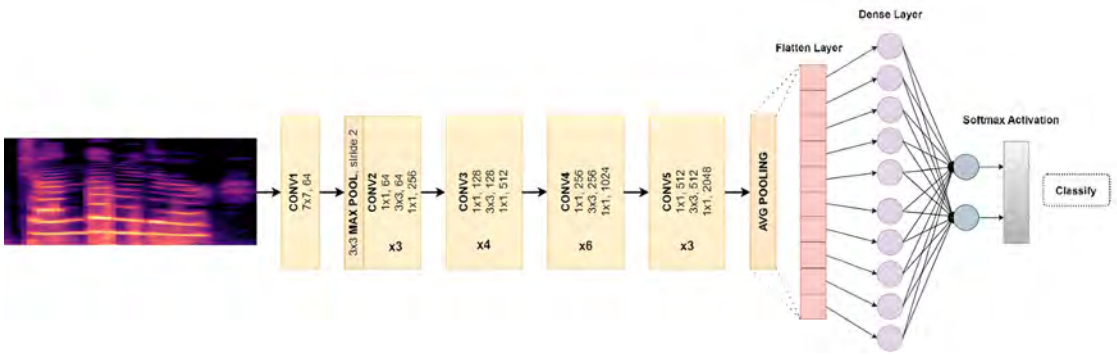

Figure 4.5: ResNet50 Working Procedure

4.6 DenseNet

4.6.1 DenseNet Preprocessing

Getting data ready for DenseNet starts with making images 224x224 pixels big. This matches the input size the network expects. Next, we normalize pixel values by subtracting the mean and dividing by the standard deviation from ImageNet data. To avoid overfitting, we use data augmentation. We randomly crop and flip training images. This helps the model work well with new images it hasn't seen before.

4.6.2 DenseNet Overview

DenseNet, short for Densely Connected Convolutional Networks, represents a breakthrough in convolutional neural network architectures, particularly in addressing the challenges of training very deep networks effectively. Introduced by Huang et al. in 2017, DenseNet proposes a novel connectivity pattern wherein each layer is connected to every other layer in a feed-forward fashion. This densely connected architecture promotes feature reuse and facilitates gradient flow, addressing the vanishing gradient problem while encouraging feature propagation throughout the network. The origin of DenseNet architecture lies in the dense connectivity structure where every layer directly connects with all the others regardless of their depth. In contrast with classical architectures where the layers receive the inputs from the previous block, the DenseNet layers receive inputs from all preceding layers inside the block. The connection scheme produces enhanced feature propagation thus, deeper layers are able to have direct access to features from the early stage of the model. Thus, DenseNet exhibits high feature reuse, parameter efficiency, and gradient flow, which lead to better learning and generalization. DenseNet includes building blocks called dense blocks as input. These blocks each consist of several layers of layers that are created closely together. Through the use of each dense block, previous layer feature maps are concatenated along the depth dimension, giving rise to the subsequent layer's dense connection with preceding layers. The dense blocks are physically separated from each other with interspersing transition layers. Physical separation helps to overcome the problem of spatial dimension and channel number that is controlling model complexity, and at the same time, it facilitates feature compression. The DenseNet design demonstrates a record performance of the previous architectures in the image classification aspects including accuracy and trainable parameters. Its architecture has many neurons connected tightly, so information will spread easily and gradients will flow, thus training deeper networks with fewer parameters is possible. DenseNet's emergence has been a dirt spark as this success led to more and more research on densely connected architectures and mentioned some key elements of deep learning methodologies.[24]

4.6.3 DenseNet Working Procedure

Importantly, DenseNet hinges upon a network of tightly knit blocks of neurons tightly knit to share and reuse the abundance of features and spread them in a wide network. The working procedure can be outlined as follows: The working procedure can be outlined as follows:

- **Input Processing:** The DenseNet starts by preprocessing input images, normalization of pixel values, and resampling to a uniform concentration. This purpose is to get rid of the discrepancies in the training process and to facilitate the formation of the descending gradient.
- **Dense Connectivity:** For all indicated layers, there are dense connections among the previously generated feature maps of a higher layer. In turn, this interconnected pattern supports second-generation repetition, and in being so, sees that features from the earlier sections are directly identifiable. This leads to DenseNet having a condensation effect whereby feature maps are concatenated along the dimension of depth. This creates a rich feature representation and promotes effective gradient flow.
- **Transition Layers:** Following the blocks with a high density, the layers between the sea are used to decrease spatial dimension and to reduce channel numbers. Other layers like calibrating or even shrinking the features as well control the complexities of the model and this way is quite helpful for resource utilization and parameter efficiency.
- **Global Average Pooling:** Terminal networks of the networks are being applied after that average pooling is performed, to aggregate feature maps into a single vector representation. Hence, the pooling process eliminates spatial aspects to a fixed size of spatial dimension which offers feature extraction and classification.
- **Classification:** The aggregated attributes make their way through the fully connected layers for the finessing of the reasoning and classification. The last one, most often, utilizes the SoftMax activation function along with outputting the probabilistic values of classes in order to return the class that has the highest probability.

By using a highly interconnected architecture, DenseNet ensures the wide spread of features to be reused and transmitted, therefore enhancing the learning quality and generalization. The Densenet model relies on the exploitation of the connection patterns and is able to achieve the best results in various image recognition tasks as it is not only very accurate but also parameter efficient.

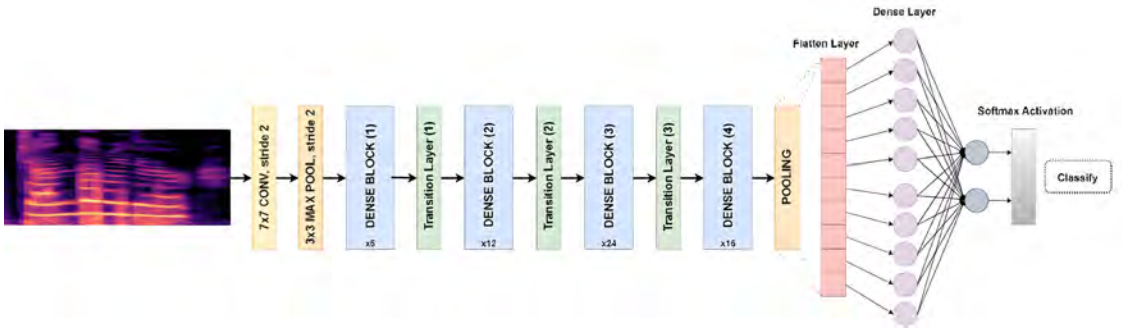

Figure 4.6: DenseNet Working Procedure

4.7 Inception V3

4.7.1 Inception V3 Preprocessing

The Inception V3 model needs images to be resized to 299x299 pixels. This is because the model was designed for high-resolution inputs. The pixel values are normalized using statistical data from ImageNet. The Inception V3 model uses more advanced augmentation techniques. These include changing aspect ratios, rotations, and shifts. This helps the model handle different input conditions. It also improves the model's ability to make accurate predictions in various scenarios.

4.7.2 Inception V3 Overview

Inception v3, an improved version of Inception architecture, has been regarded as a game-changing model in the domain of CNN (Convolutional Neural Networks). Brought about by the minds of Google scientists, this architecture denotes a major uptick in the skill of image recognition and processing. While its previous counterparts have adopted inception modules that combine various kernel sizes in the same layer to produce features at different scales, Inceptionv3 makes use of an ingenious and intricate inception module that primarily focuses on the generation of features at multiple scales within the same layer. With this approach, the network is able to handle the least information and still high-level cognition, which are processes that characterize human mental operations.

The architecture of Inception v3 is defined by its intricacy and depth that respectively implies the vast number of layers and complexity of the links. The combination is involved in the module that allows parallel convolutional layers each with its own filter size to be arranged in such a way to ensure the extraction of features at different levels of abstraction. This multi-layered structure transforms the network enabling Inceptionv3 to achieve excellent performance when working with the various kinds of tasks.

Indeed, one of the most distinguishing features of Inceptionv3 is the utilization of auxiliary classifiers as the intermediate layers of the network are being added. These auxiliary classifiers, associated with additional supervision, give a good guarantee in training and thus help to overcome a vanishing gradient problem and make the model stable and convergent.

What is getting most people excited about Inceptionv3 is that it has shown the highest performance not only in different benchmark datasets, but it has exceeded previous models in terms of accuracy and efficiency. Its robustness and generalization capability grant it a wide space in the tools used in applications such as image classification, object detection, and precise localization.

4.7.3 Inception V3 Working Procedure

The functioning mechanism of Inceptionv3 is designed around its hybrid architecture and the fact that it extracts features. This architecture consists of the inception

modules which are the core of the model and they represent building blocks for capturing multi-scale features within the input images. They are based on convolutional operations implemented in parallel with kernels of different sizes, so the network produces information at different levels of detail or granularity.

In every layer of the network, the feature maps are processed via convolution and/or pooling algorithms, and the complexity increases gradually while the level of abstraction gets higher. The employment of batch normalization together with advanced activation functions such as the ReLU function makes the model both stable and quickly converging during training. Also along with the primary classification outcome, it adds auxiliary classifiers in the middle level which includes extra control and supervision. Such less frequent auxiliary classifiers enhance the propagation of gradients and quick training compared to dense classifiers thereby, improving the learning dynamics and convergence efficiency of the model.

In the early training phase, Inceptionv3 employs stochastic gradient descent (SGD) to optimize the weights along with techniques like momentum and learning rate scheduling. Techniques like regularization that employ dropout and weight decay are also incorporated to reduce overfitting and improve generalization capability.

Figure 4.7: InceptionV3 Working Procedure

4.8 EfficientNet

4.8.1 EfficientNet Preprocessing

During image processing using EfficientNet, the images are resized to the network model-specific dimensions that typically consist of 224x224 pixels for EfficientNet-B0, and images are normalized by employing ImageNet's mean and standard deviation to guarantee consistency with its training conditions. Forming of the pictures is also employed by both flips and random rotations to optimize the model for various conditions, as it is required in real-life scenarios. Besides, screen generator objects are applied exactly for efficient batch processing when the work is done with dynamic, large datasets. Surely, the designed environment will be wholly optimal, as well as comply with reference implementation demands, be it in TensorFlow or PyTorch.

4.8.2 EfficientNet Overview

EfficientNet is a groundbreaking convolutional neural network architecture proposed by Tan and Le in 2019 which is optimized to enhance accuracy while keeping the number of parameters constant or even relatively small in comparison to the existing models. Its development ensures that the models are efficient for important computational resources, precision in conditions variation, and complexities across a vast range of scales. EfficientNet accomplishes this through a multiscale compound scaling method, which consistently modifies height, depth, and resolution using a defined set of fixed terms. Such arrangement promotes efficient use of system resources (computer power), which, in turn, ensures excellence in results.

EfficientNet's architecture is based on the baseline version of this network, EfficientNet-B0, where subsequent variants are created by scaling the network and adding extra layers to facilitate achieving goals like obtaining even higher performance with little computation cost.[16] The scaling coefficients give the model the ability to adjust itself at the level of capacity that matches a specific set of computing requirements, hence making EfficientNet suitable to different hardware platforms and functional operations.

EfficientNet has shown itself to be a very powerful method, which has pushed accuracy and efficiency limits even further among deep learning methods scorers on a variety of computer vision tasks such as image classification, object recognition, and semantic segmentation. Its variety and scaling abilities make it one of the prominent decisions for the app manufacturing in real-time high-performance image segmentation on resource-constrained devices.

4.8.3 EfficientNet Working Procedure

Through effective implementation of the convolutional layers, EfficientNet works by segmenting the input images that are being processed in a sequence consisting of each convolutional layer designed to elicit progressively higher-level features from the data. The topology is enhanced by three factors: the number of layers, the number, and the resolution, which are scaled via the coefficient additionally, to achieve optimal performance under variable computational constraints.

The compound scaling technique is the core mechanism of EfficientNet, allowing for simultaneous growth of the width and depth of the network while maintaining its structural integrity. This scaling strategy allows the network to stay equally focused on efficiency and accuracy, ultimately making the model settle for state-of-the-art performance with minimal computational resources. EfficientNet's training involves pattern recognition and image feature learning by altering the weights of its convolutional layers through back migration. This model is trained using large datasets with labeled images, so it is able to generalize its learning over and to new and unknown data.

Being quite efficient, EfficientNet can very easily be deployed on different hardware platforms, such as mobile phones and embedded systems where computational resources can be scarce. Its capability of obtaining high performance with a low

default level of computer resources makes it a good option in various applications including computer vision and beyond.

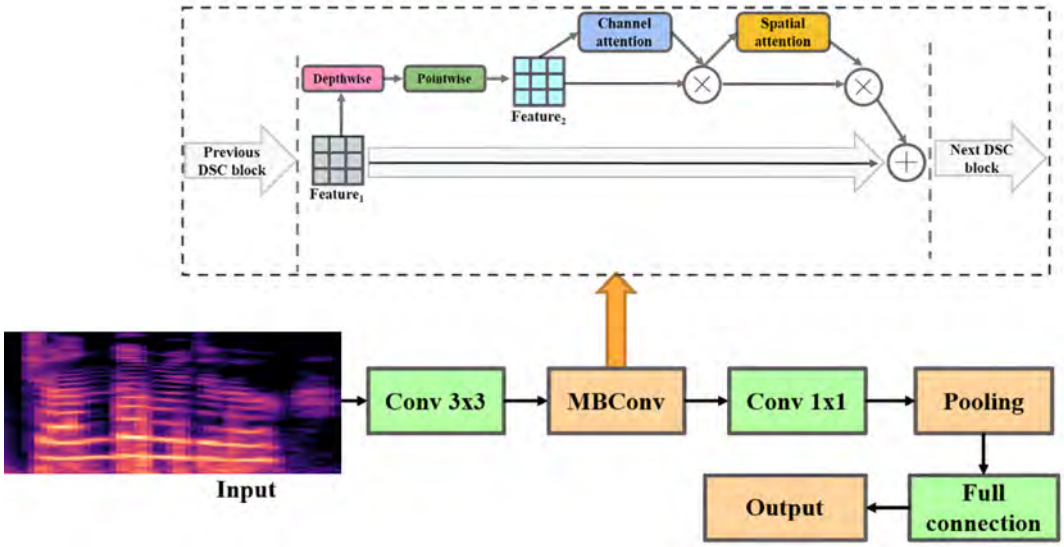

Figure 4.8: EfficientNet Working Procedure

4.9 MobileNetV2

4.9.1 MobileNet V2 Preprocessing

MobileNet V2 is a computer program. It is used on phones and tablets. It needs pictures to be 224 pixels by 224 pixels. MobileNet V2 uses special numbers from ImageNet. The numbers help the program work better. Too much training can make MobileNet V2 not work well. Moving and turning the pictures a little helps. The program also uses batch normalization. Batch normalization makes training faster and better.

4.9.2 MobileNetV2 Overview

MobileNetV2 has introduced an evident improvement in the course of CNNs that is evident in image classification and recognition specifically. It was developed especially to be lightweight and accurate so that it could handle mobile and embedded projects where the computational facilities are restricted.

MobileNetV2 architecture is based on the functions of depth-wise separable convolutions that quickly reduce the number of computations in the traditional convolution layers. Depthwise separable convolutions consist of two distinct layers: depthwise convolutions and 1x1 convolutions.[17] The depthwise convolution step possesses the unique feature of using a single filter with all input channels, whereas the pointwise convolution step acts as a one-dimensional convolution over all the channels, combining and transforming features.

With the help of these depth-wise separable convolutions, MobileNetV2 has a good balance between the accuracy of a model and model efficiency. This allows fine-grained feature extraction, which in turn gives an advantage in small model size and low inference time, making it good for resource-constrained environments.

4.9.3 MobileNetV2 Working Procedure

MobileNetV2 works by running the input images through a sequence of depthwise separable convolutional layers and nonlinear activation functions like for, instance, ReLU (Rectified Linear Unit). This kind of layer extracts more and more abstract symptoms (characteristics) from the input images such as the filtration of the important patterns and characteristics. Moreover, MobileNetV2 has inverted residual blocks with linear bottleneck layers that contribute to achieving feature extraction with more accurate results. Enhanced capacity without a significant increase in computational cost is achieved by using lightweight depthwise convolutions (referring to inverted residuals), and linear bottlenecks conserve feature representation efficiency. As well, the architecture introduces shortcut connections, which function in a similar way to residual networks, to help gradient flow during training and to optimize the network. These improvised connections lessen the danger of the vanishing gradient problem and enable the network to learn more efficiently through deeper layers. In fact, MobileNetV2 is based on the principle of global average pooling at the end of the network and thus the spatial dimensions can be substantially decreased with the generation of a representation of the features that are of low dimensionality.

These pooling functionalities gather spatial information across feature maps that are made to a fixed-length feature vector which enables the classification. The final part of the process is done by passing the result of MobileNetV2 through a softmax activation function thus, getting distributions of probabilities for predefined classes. This gives the model the ability to predict and categorize the input images as they have been trained to recognize features with distinct categories.

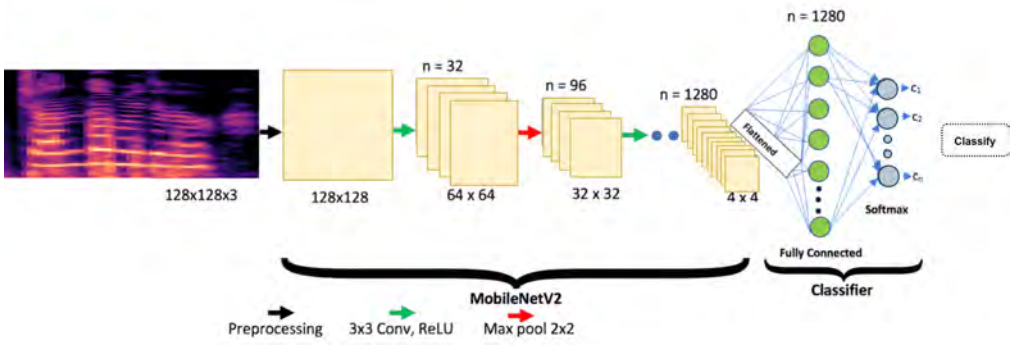

Figure 4.9: MobileNetv2 Working Procedure

4.10 SENet (Squeeze-and-Excitation Network)

4.10.1 SENet Preprocessing

The SENet model leverages the PyTorch framework's functionality, which necessitates image resizing, center-cropping, and tensor transformation. SENet-specific image normalization is conducted using ImageNet-compatible values, and in cases where TensorFlow is also in use, functions are provided to convert PyTorch DataLoaders into TensorFlow datasets.

4.10.2 SENet Overview

In 2018, Hu et al. proposed the Squeeze-and-Excitation Network, which proved to be another milestone of Convolutional Neural Networks. The SENET is designed to enhance the ability of a network to describe the data by adapting interactions between its feature map channels through a block referred to as Squeeze-and-Excitation, which makes three principal actions: squeeze, activation, and scale.

Pooling allows for global averaging of the input feature map, summing up the global spatial information into one value per channel—producing one single descriptor per channel. This compresses the spatial dimensions, keeping the depth.

The squeezed channel descriptors then pass through a gating mechanism, a typically two-layer fully connected network—functioning as channel-wise attention. It learns a weight for each channel, defining its relevance, with sigmoid activation, thus returning values between 0 and 1.

In the last step, the SE block shrinks the feature maps using the weights acquired in the previous step. Independent feature scaling in each channel of the input map is done with the channel-specific weight values. This process of re-calibration gives more importance to the feature channels and lowers the less important ones, strengthening the major feature representation and weakening the secondary ones.

Integrating an SE block into a state-of-the-art CNN architecture will offer better performance by focusing on important features in an even stronger manner. Simplicity and efficiency of the SE block enable embedding different CNN models into it without large additional computational complexity. The method used to upgrade the model's accuracy and performance, mainly in the area of image classification and recognition tasks.

4.10.3 SENet Working Procedure

The Squeeze and Excitation Network (SENet) follows a different method of handling input images by focusing on the idea of channel recalibration. It is aimed at making the network more skillful in isolating the most prominent features of an image and drawing their attention. Key stages involved in this process include:

At the start, the convolutional layers are the standard ones, which process the input images. These layers are important components of Convolutional Neural Networks (CNNs) which extract features from input images. They use various filters that perceive different characteristics ranging from basic edges and textures to more complex patterns. The depth and complexity of these extracted features usually increases as an image passes through subsequent convolutional layers.

In certain areas of the network, Squeeze-and-Excitation (SE) blocks are strategically integrated. These blocks are positioned to augment the process of extracting features. The function of an SE block serves a dual purpose:

- **Squeezing Spatial Information:** Each SE block starts with the squeezing of spatial information. It takes the output from the previous convolutional layers (feature maps) and applies global average pooling. This pooling operation compresses the spatial dimensions of each channel in the feature map, resulting in a channel descriptor. This descriptor is a compact, global representation of the spatial features for each channel.
- **Channel Recalibration (Excitation):** Upon being squeezed, the channel descriptors are employed in order to recalibrate the channels. This 'excitation' marks a recalibration. The learning mechanism (usually a simple neural network) is applied to the channel descriptors which assigns importance weights to each of them. By applying these weights on the original feature maps, it is possible to emphasize or downplay certain channels. Consequently, this allows the network to adapt its feature responses on-the-fly, concentrating more on the most important channels for that particular input.

After convolutional layers and SE blocks have received an image, data flattens out. In order to flatten out multidimensional feature maps into one-dimensional vectors, they must be transformed into a single-dimensional vector.

The flattened information is then passed through one or more fully connected (dense) layers. These layers are typical in neural networks because they are responsible for combining learned features and performing complex decision-making processes. The last layer of the network uses SoftMax activation function which is commonly used in multi-class classification tasks. A probability distribution across various classes is formed by the SoftMax function indicating how likely an input image will fall under each class.

The thing that makes SENet different is its effectiveness in changing the feature responses at channel level dynamically. This feature enables the network to concentrate more on the most important channels for a given input. It thus increases its efficiency in better identifying images accurately and classifying them correctly.

This approach represents a significant advance within deep learning, particularly with regard to tasks like image classification and recognition.

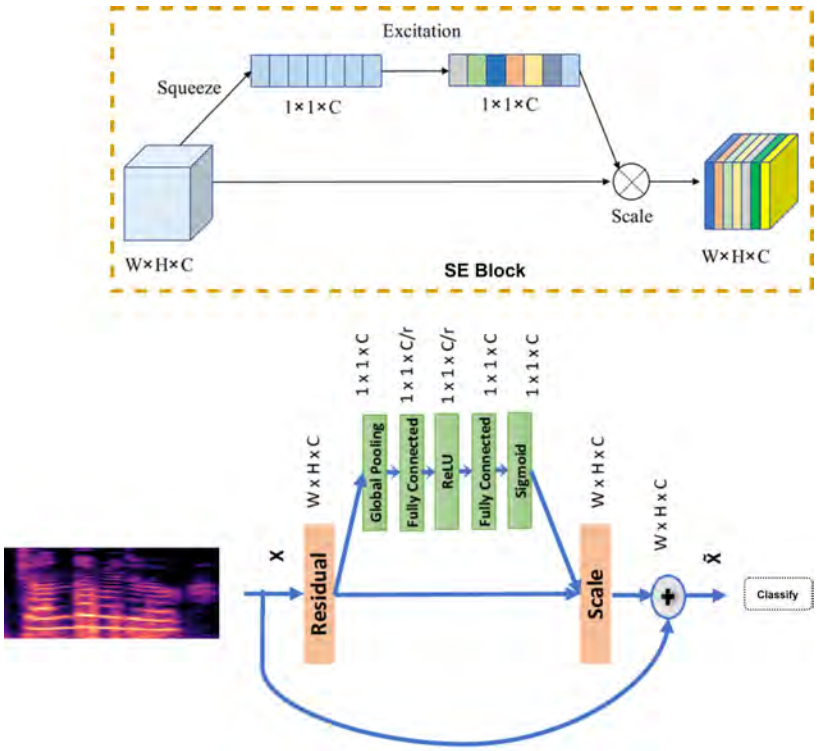

Figure 4.10: SENet Working Procedure

Chapter 5

Proposed Model: Spectro SETNet

From fig 5.1 we can see the top overview of our proposed model. Firstly, we did data acquisition where we gathered audio from 21 children across 57 classes in a controlled setting to ensure data quality. Obtain all necessary consents for recording children, adhering to privacy and ethical guidelines. Secondly, we used Audacity to reduce noise and segment audio into characters, ensuring proper labeling and storage. Thirdly, we converted (.WAV) files to mel frequency spectrogram images using a Python script that utilizes librosa and matplotlib for processing and visualization. Before feeding the images into the model, we preprocessed the images by decoding the image files, resizing them to uniform dimensions, normalizing input values, shuffling the data to minimize order bias, caching for efficiency, and prefetching to streamline the training process. Then, we designed our proposed model Spectro SETNet. Furthermore, we check that spectrogram image dimensions match ResNet50+SE input requirements. Adjust preprocessing as needed and adapt the neural network to handle the spectrogram images. Then after further preprocessing we feed our data transformer block. Lastly, organized spectrogram images into balanced training and testing sets. Implement data augmentation to boost the model's generalization capability.

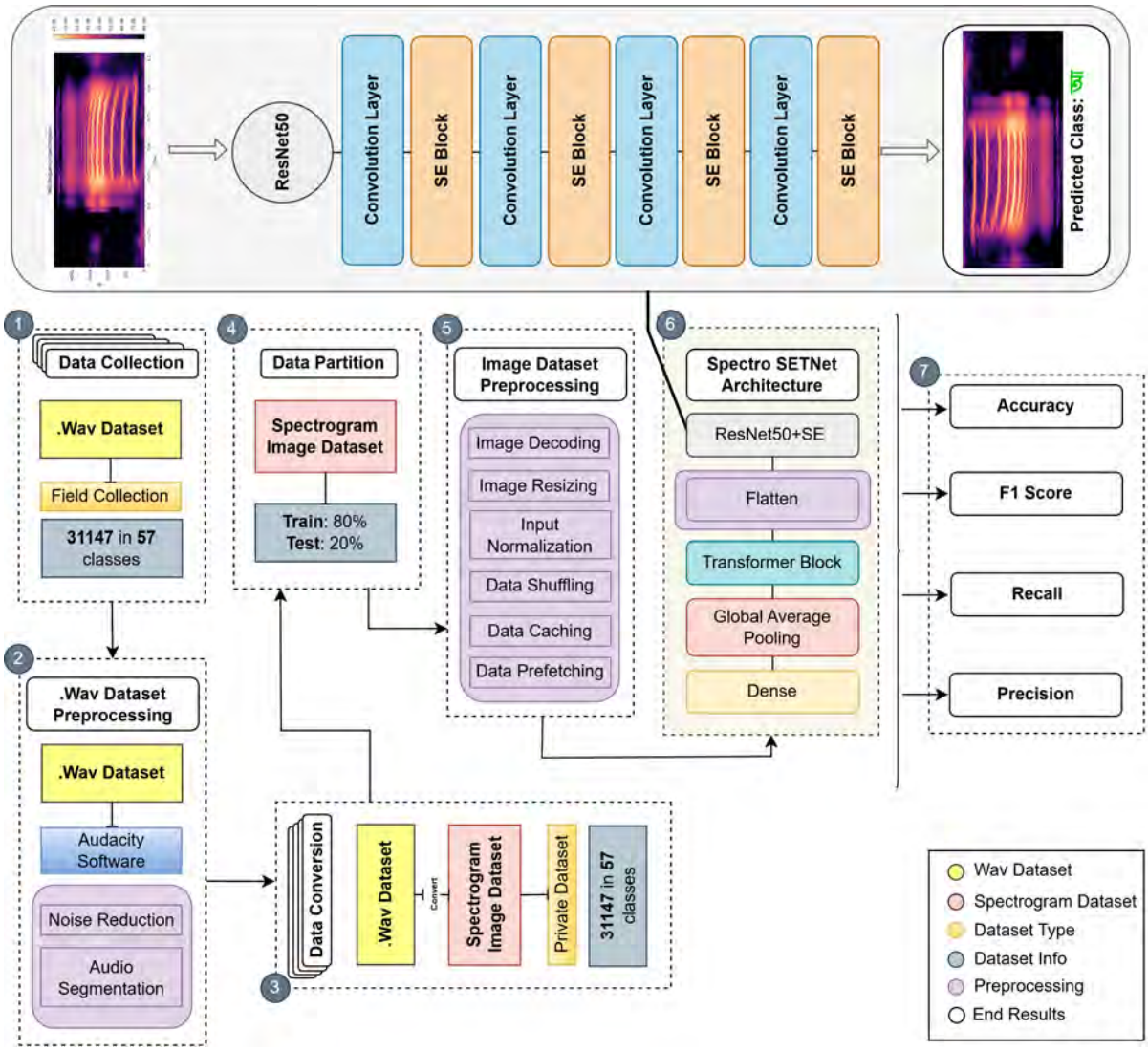

Figure 5.1: Top Level View of the Proposed Model Spectro SETNet

5.1 Data Collection

Data collection has been discussed in chapter 3.1

5.2 .wav Dataset Preprocessing

(.wav) file preprocssing has been discussed in chapter 3.2

5.3 Data Conversion

Converting data from (.wav) file to mel frequency spectrogram image has been discussed in chapter 3.3

5.4 Data Preprocessing for Spectro SETNet

5.4.1 Dataset Splitting

- **Train and Test:**The dataset is split into training and testing sets which segregate the data into separate sets (20% for validation and 80% for training in this case) to ensure the model is tested on unseen data, improving its generalizability.

5.4.2 Image Preprocessing

- **File Reading:** The image file is read into a byte string to be decoded into a tensor, facilitating further image manipulations.
- **Decoding:** Converting the byte string into an RGB image tensor standardizes data format, preparing it for consistent processing and analysis. When a JPEG image is decoded, the byte stream is converted into a 3D tensor of RGB values. Mathematically, this transformation can be represented as a mapping from byte stream B to a tensor T:

$$\boldsymbol{T} = \text{decode}(\boldsymbol{B}) \tag{5.1}$$

  In this context, T is a tensor of shape $(\text{height}, \text{width}, 3)$, where 3 represents the RGB channels.

- **Resizing:** Standardizing images to 224x224 pixels aligns with the input size requirements of the ResNet50 architecture, ensuring that the network receives uniformly sized inputs.
  Resizing involves changing the spatial dimensions of the image. If the original dimensions of the image tensor T are $(H, W, 3)$ and it is resized to $(H', W', 3)$, the resizing operation can be expressed as:

$$\boldsymbol{T'} = \text{resize}(\boldsymbol{T}, \boldsymbol{H'}, \boldsymbol{W'}) \tag{5.2}$$

Where T' is the resized tensor. The most common in deep learning frameworks (like TensorFlow) is bilinear interpolation, where the new pixel value is computed as a weighted average of pixels in the nearest 2×2 neighborhood.

- **Normalization:** Using ResNet50's 'preprocess_input' function, which applies a specific normalization that adjusts pixel values to the model's expected range. This normalization is based on the statistics of the ImageNet dataset, on which ResNet50 was pre-trained, and it helps in reducing internal covariate shifts during training. Normalization adjusts the pixel values of the image to a range that a pretrained network expects. For the case of ResNet50, which is trained with pixel values scaled from -1 to 1, the normalization can be mathematically represented for images initially in the range [0, 255] (standard for RGB images) as follows:

$$\boldsymbol{P}_{\text{normalized}} = \frac{\boldsymbol{P} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{5.3}$$

For ResNet50, the mean μ and standard deviation σ values are calculated channel-wise based on the ImageNet dataset, typically resulting in:

$$\boldsymbol{\mu} = [\mathbf{123.68, 116.779, 103.939}], \quad \boldsymbol{\sigma} = [\mathbf{58.393, 57.12, 57.375}]$$

Alternatively, if using the common practice for some implementations that map [0, 255] directly to [-1, 1], the equation would simplify to:

$$\boldsymbol{P}_{\text{normalized}} = \frac{\boldsymbol{P}}{\mathbf{127.5}} - \mathbf{1} \tag{5.4}$$

5.4.3 Creating TensorFlow Dataset Objects

- By transforming the lists of paths and labels into tf.data. Dataset objects, leverage TensorFlow's built-in functionalities for handling large datasets efficiently. These objects are designed to facilitate complex transformations and batching operations on the data, which are crucial for training deep learning models effectively.

5.4.4 Performance Configuration

- **Caching:** Caching the images after the first epoch prevents repeated disk read operations, speeding up training.
  Caching is a technique used to store previously computed data in a rapidly accessible storage layer, reducing the need for redundant computations and speeding up data retrieval in subsequent operations. While there is no specific equation for caching, its impact can be described in terms of reduction in time and computational resources required for data access across epochs. Conceptually, caching can be thought of as reducing the number of read operations required from the primary storage:

$$\boldsymbol{O}_{\text{new}} = \frac{\boldsymbol{O}_{\text{initial}}}{\text{epochs}} \tag{5.5}$$

where O_{initial} represents the number of operations to read and preprocess the data during the first epoch, and O_{new} is the average number of operations per epoch post-caching. The reduction in operations directly correlates with improvements in training speed, particularly in scenarios where data loading is a significant bottleneck.

- **Shuffling:** Shuffling the data prevents the model from learning unintended patterns from the order of the data, thus helping in better generalization.
  Shuffling the dataset randomly permutes the order of the data points, which is crucial for preventing the model from learning potentially misleading patterns that may arise from the sequence in which data is presented. Mathematically, shuffling can be represented as applying a permutation π to the dataset D:

$$\boldsymbol{D' = \pi(D)} \tag{5.6}$$

  where D' is the shuffled dataset.

- **Batching:** Batching defines how many examples the model sees before updating its weights, balancing learning dynamics and computational efficiency.
  Batching divides the entire dataset into smaller, manageable subsets known as batches. This allows the model to update its parameters iteratively over batches rather than the entire dataset, which can be computationally efficient and manageable for memory resources. If D is divided into k batches of size n, it can be expressed as:

$$\boldsymbol{D = \bigcup_{i=1}^{k} B_i}, \quad \text{where } \boldsymbol{|B_i| = n} \text{ for all } \boldsymbol{i} \tag{5.7}$$

  Each B_i represents a batch from the dataset D.

- **Prefetching:** Prefetching overlaps the preprocessing of data for the next batch with the model's training on the current batch, reducing idle times and improving overall efficiency.
  Prefetching is a technique used to prepare data for upcoming operations while the current operation is still executing. This method helps in keeping the data pipeline busy and reduces the time spent waiting for data to be loaded and processed. The impact of prefetching can be conceptualized as overlapping data loading time T_{load} with computation time T_{compute}, potentially reducing the effective batch processing time:

$$\boldsymbol{T_{\text{new}} = T_{\text{compute}} + \max(0, T_{\text{load}} - T_{\text{compute}})} \tag{5.8}$$

  where T_{new} is the reduced total time per batch due to overlapping operations.

5.5 Overview of the Proposed Spectro SETNet Model

Spectro SETNet, the novel architecture integrates ResNet50 with Squeeze-and-Excitation (SE) blocks and a Transformer block, tailored specifically for the classification of Mel Frequency Spectrogram images. The model harnesses the deep feature extraction capabilities of ResNet50, the adaptive recalibration feature of SE blocks, and the sequence processing strengths of Transformer blocks to create a robust framework for audio classification tasks.

5.5.1 Base Model

The model begins with a standard ResNet50 as the base layer, a deep convolutional neural network known for its efficacy in image recognition tasks. ResNet50 is employed here without the top layer (i.e., it excludes the fully connected layer at the end), allowing for custom layers to be appended for task-specific fine-tuning. The input to this network is a Mel Frequency Spectrogram image, resized to 224x224 pixels, to match the input dimension that ResNet50 expects.

5.5.2 Integration of Squeeze-and-Excitation (SE) Blocks

Following ResNet50, Squeeze-and-Excitation (SE) blocks are applied. These blocks are strategically placed after specific convolutional layers within ResNet50, specifically after the output of each major residual block. The SE blocks serve to recalibrate the feature maps generated by the preceding convolutions, enhancing the model's ability to focus on relevant features while suppressing less useful ones. This recalibration is crucial for audio data, which often contains complex patterns that are more subtle than those found in typical visual data sets.

5.5.3 Transforming Features for Sequence Processing

The outputs from the last SE block are then reshaped to prepare them for sequence processing. This reshaping converts the two-dimensional feature maps into a sequence of vectors, each vector representing a portion of the original image, thereby preserving spatial relationships in a form amenable to sequential processing.

5.5.4 Integration of Transformer Block

A Transformer block, which follows the reshaped output, includes a Multi-Head Attention mechanism and a feed-forward network. This block allows the model to process sequences of feature vectors from the reshaped SE-enhanced ResNet50 outputs. The Multi-Head Attention mechanism in the Transformer allows the model to focus on different parts of the input sequence, capturing internal dependencies without regard to their distance in the sequence. This feature is particularly beneficial for spectrogram images, where temporal and frequency dependencies can span across the entire sequence.

5.5.5 Functional Capabilities

The joint use of ResNet50, SE blocks, and a Transformer block gives a strong solution for the audio signal classification tasks. ResNet50 acquires the strong initial features from the spectrogram images, which are afterward refined by the SE blocks to promote the most salient features for the audio analysis. The Transformer block then continues with the processing of these features, using its attention mechanisms to create complex dependencies and interactions in the data. The model is designed to be computationally efficient while still delivering high accuracy. The use of pretrained weights from ImageNet for initializing ResNet50 speeds up convergence and improves the generalizability of the model. The SE blocks add minimal computational overhead but significantly boost the model's performance by focusing it on relevant features. Finally, the Transformer block, while computationally more intensive than traditional sequence processing methods, offers unparalleled capabilities in handling long-range dependencies in data.

5.5.6 Summary of Proposed Model

Layer (type)	**Output Shape**	**Param**	**Connected to**
input_2 (InputLayer)	(None, 224, 224, 3)	0	[ ]
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	{'input_2[0][0]'}
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	{'conv1_pad[0][0]'}
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	{'conv1_conv[0][0]'}
conv1_relu (Activation)	(None, 112, 112, 64)	0	{'conv1_bn[0][0]'}
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	{'conv1_relu[0][0]'}
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	{'pool1_pad[0][0]'}
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	{'pool1_pool[0][0]'}
conv2_block1_1_bn (BatchNorm)	(None, 56, 56, 64)	256	{'conv2_block1_1_conv[0][0]'}
...	...	...	...

Table 5.1: Summary of Proposed Spectro SETNet Model

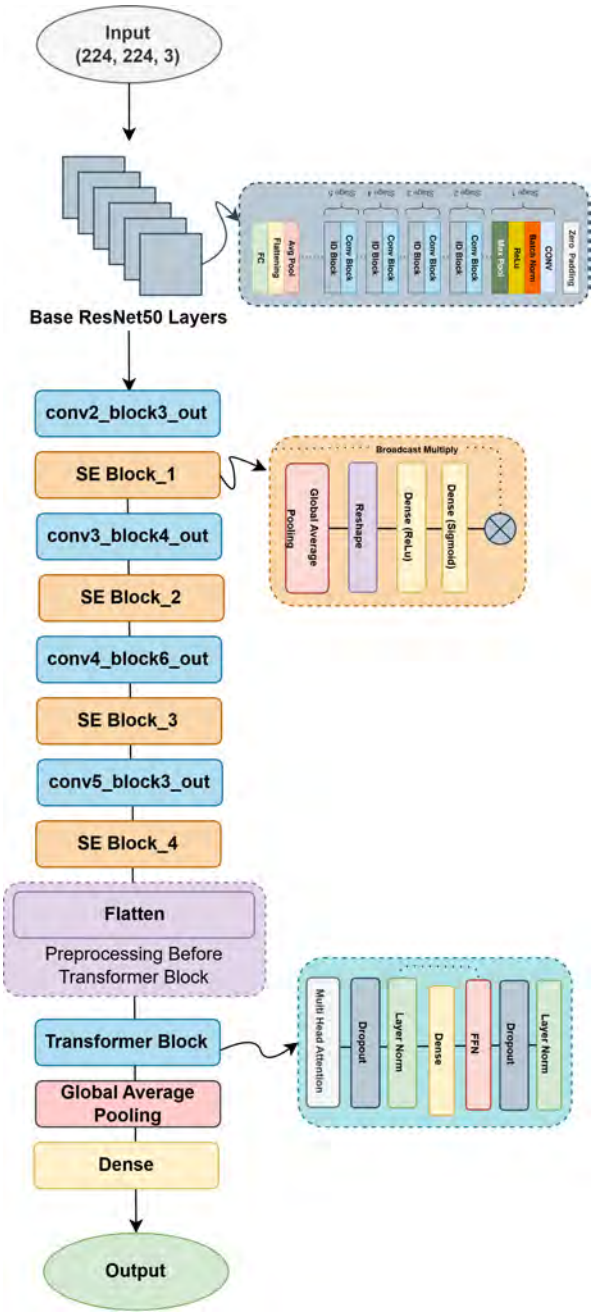

Figure 5.2: Spectro SETNet Architecture

5.6 Spcetro SETNet Detailed Architecture

5.6.1 Architecture of the Base Model

In our new model, using ResNet50 as a base model in a customized architecture tailored for audio analysis via spectrogram input allows we to leverage the advanced visual feature extraction capabilities of ResNet50 for audio data. This is a creative and effective way to apply image classification architecture to audio analysis tasks, exploiting the fact that spectrograms can be treated as images. The initial input to the system is a Mel Frequency Spectrogram, which represents the spectrum of frequencies of a sound signal as it varies with time. This spectrogram undergoes resizing and preprocessing to match the input specifications required by the ResNet50 model. Typically, this involves resizing the image to 224x224 pixels, as ResNet50 is designed to handle this dimension, and normalizing the pixel values.

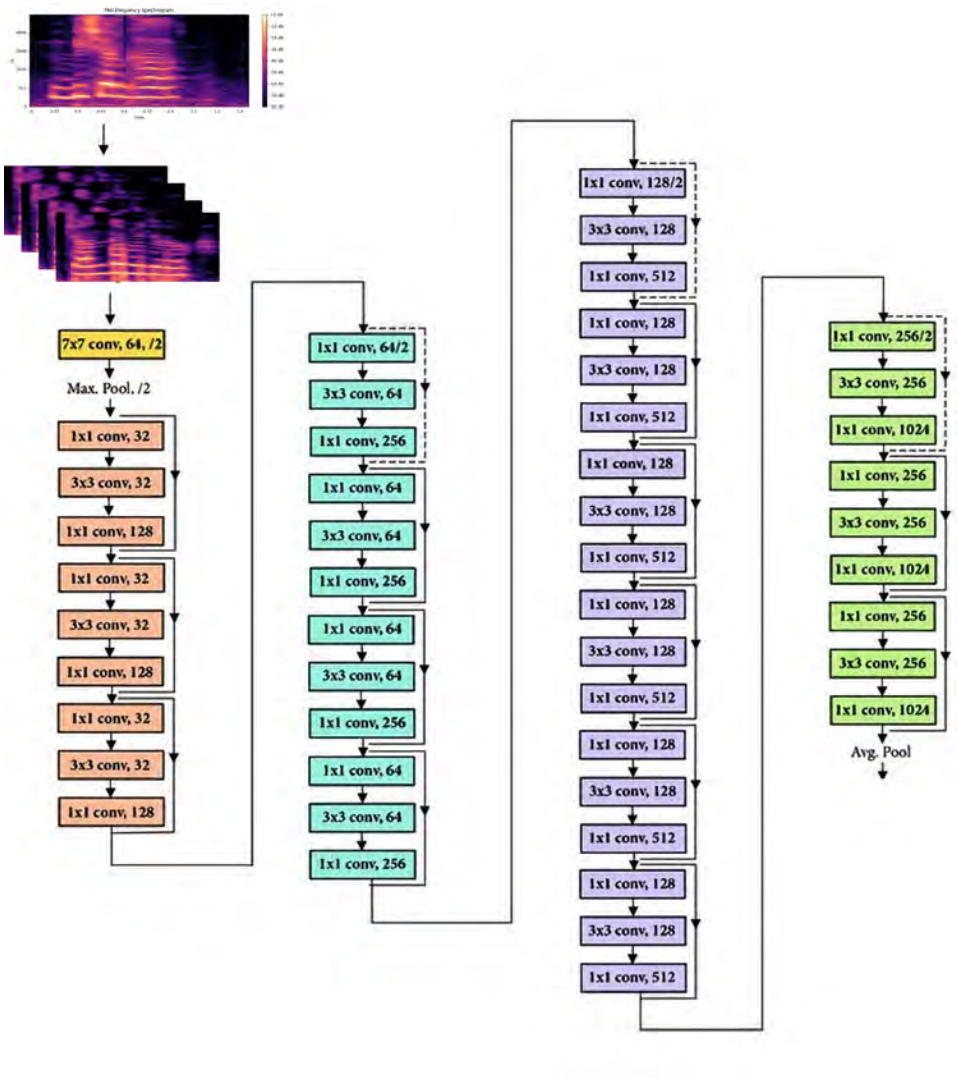

Figure 5.3: Residual Architecture used in Spectro SETNet

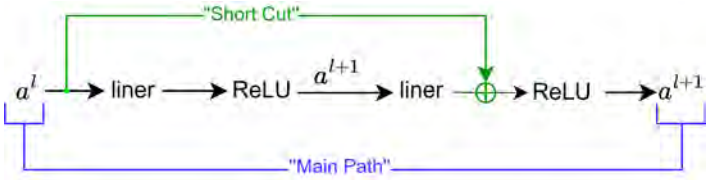

Figure 5.4: Residual Block

This approach is particularly advantageous because it:

- Utilizes pre-trained weights (from ImageNet), which can help in faster convergence and potentially better generalization, especially when audio-related data might be limited.
- Adapts a proven architecture for a novel application, potentially increasing the robustness and accuracy of the audio analysis.

This type of architecture is typically used in advanced sound analysis applications like environmental sound classification.

Layer (type)	**Output Shape**	**Param**	**Connected to**
input_1 (InputLayer)	(None, 224, 224, 3)	0	[ ]
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	{'input_1[0][0]'}
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	{'conv1_pad[0][0]'}
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	{'conv1_conv[0][0]'}
conv1_relu (Activation)	(None, 112, 112, 64)	0	{'conv1_bn[0][0]'}
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	{'conv1_relu[0][0]'}
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	{'pool1_pad[0][0]'}
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	{'pool1_pool[0][0]'}
conv2_block1_1_bn (BatchNorm)	(None, 56, 56, 64)	256	{'conv2_block1_1_conv[0][0]'}
...	...	...	...

Table 5.2: Summary of ResNet50 Model

5.6.2 Achitecture of SE Block

In the processing of Mel Frequency Spectrograms, Squeeze-and-Excitation (SE) blocks play a very important role in the improvement of the performance of convolutional networks as they redefine the feature maps with the unique characteristics of audio signals. The SE blocks, through the process of emphasizing on the main information and suppressing the less important ones, enhance the network's capability to detect the fine details in the audio which is a vital part of the audio classification and analysis tasks. These blocks are located just after the convolutional layers and thus, the features are immediately adjusted. This adjustment is then further refined through repeated convolution and recalibration cycles. This process not only enhances the features in the hierarchy but also preps them for further advanced processing, e. g. by means of transformer blocks for the complex sound classification or the event detection in audio streams.
This approach is particularly advantageous because it:

- Adapts channel-wise feature responses dynamically, focusing more on informative features, crucial for complex analysis tasks.
- Introduces minimal computational overhead, making SE blocks suitable for real-time applications where both speed and accuracy are essential.

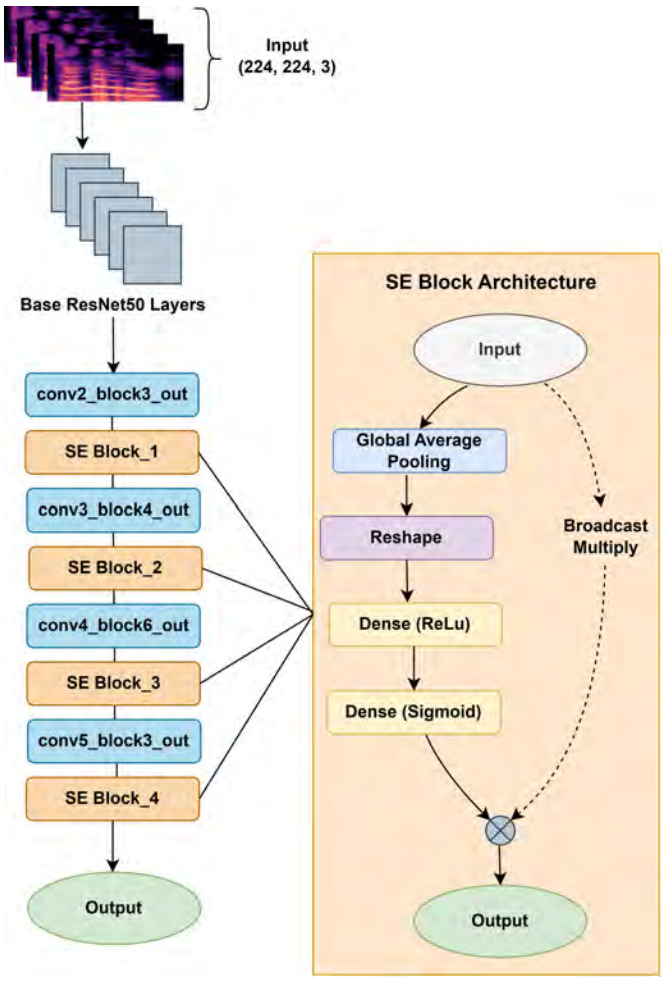

Figure 5.5: SE Block Architecture

5.6.3 Architecture of Transformer Block

The Transformer block is the most important element in the improvement of the Mel Frequency Spectrogram processing in neural network architectures, especially for the tasks that deal with detailed audio analysis. The design of the Transformer block's architecture, that is, the Multi-Head Attention mechanism, is aimed at solving the problems related to audio signal processing.

The core of the Transformer block is the Multi-Head Attention mechanism, which allows the model to focus on different parts of the input sequence simultaneously, a vital feature for capturing the nuances and dependencies in complex data patterns. Applied after the attention mechanism to prevent overfitting during training by randomly setting a fraction of the input units to 0 at each update during training time. Then Layer Normalization normalizes the inputs across the features, stabilizing the learning process and speeding up the training. Such an arrangement not only facilitates detailed attention to relevant sequential features but also allows for efficient processing through parallel computations. The inclusion of Feed-Forward Networks (FFNs) within each block further processes these features to refine the model's output. Each sub-layer (attention and feed-forward) includes a residual connection followed by layer normalization, enhancing the flow of gradients throughout the network, which allows for the training of deeper models.

Prior to the data entering the Transformer block, the data undergoes a preprocessing stage where it is usually flattened and reshaped to match the expected input format of the block. This preprocessing stage is very important as it modifies the data into a form that is suitable for processing by the Transformer Block.
This approach is particularly advantageous because:

- The Multi-Head Attention mechanisms help the model to process and combine the context from different parts of the input sequence which is vital for the tasks that need a deep understanding of the structure of the sequence such as language translation or speech recognition.
- Layer normalization of the features of each sub-layer stabilizes the training dynamics, which in turn results in faster convergence and less sensitivity to the network initializations.
- The capability to process the parts of the input data at the same time, not only accelerates the training but also makes it possible to expand the network to deal with larger datasets or more complex models without a big increase in the computational time.

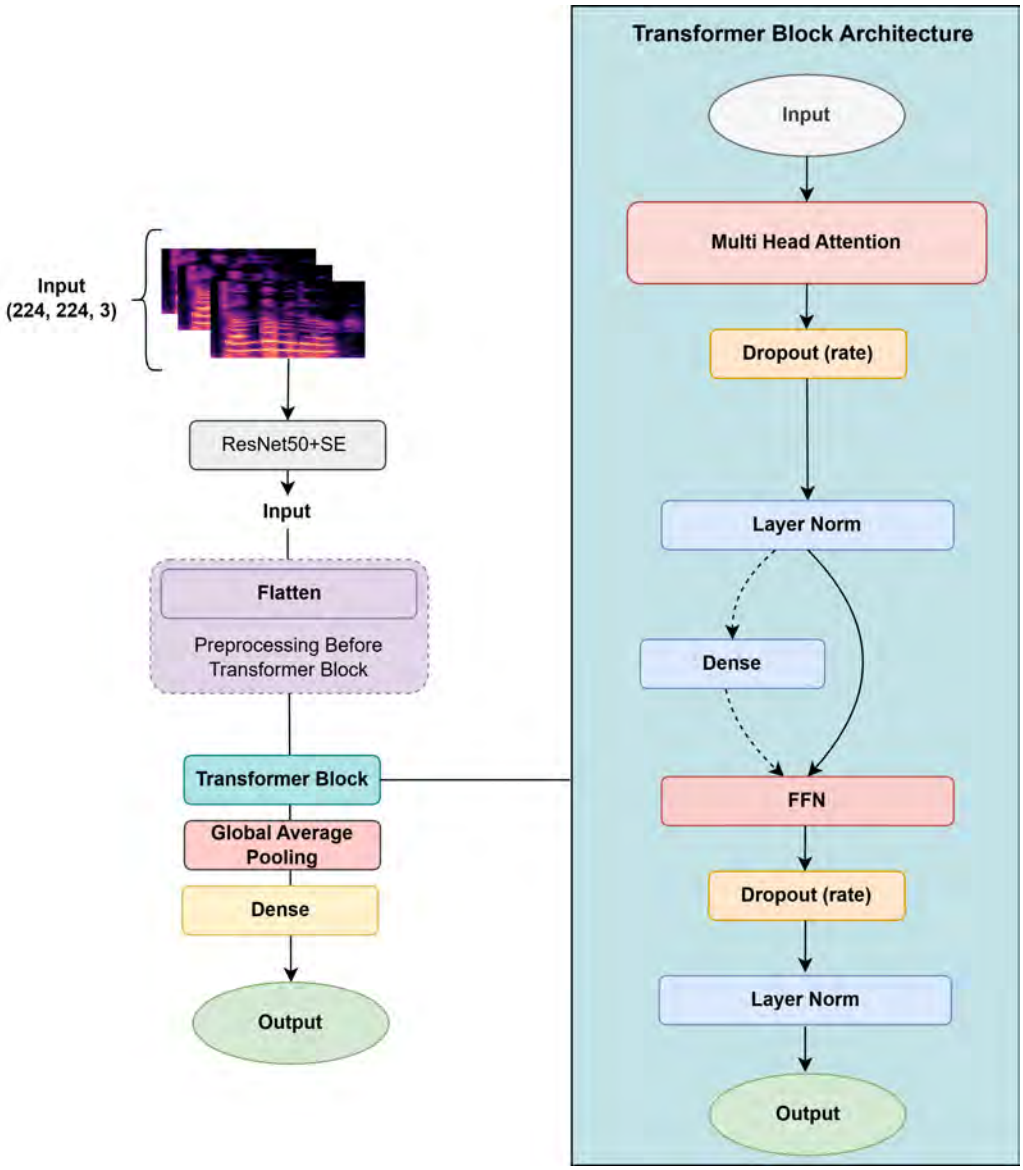

Figure 5.6: Transformar Block Architecture

5.7 Layers Used in Spectro SETNet

5.7.1 Convolution Layer

Convolution layers are a significant part of the neural network architecture and this layer is best suited for processing data with a grid structure, for example – images. It involves capturing the detected features through feedback and highlighting the feature maps summarizing these presences of the features in the input.
The mathematical operation performed by a convolution layer can be described as follows:

$$\boldsymbol{O_{ij} = (K * I + b)_{ij}} \tag{5.9}$$

where O represents the output feature map, K denotes the kernel or filter, I is the input matrix, b is a bias term, and $*$ denotes the convolution operation. The convolution operation involves sliding the kernel K over the input matrix I and computing the dot product at each position, which captures local dependencies within the input data.

5.7.2 Flatten Layer

Flatten layer in neural networks serves the purpose of converting a multi-dimensional input tensor into a one-dimensional vector. This is very effective when going from convolutional layers to fully connected (dense) layers as the pooling operation decreases the dimensions of the feature maps so that they can be processed by the dense layer which expects lower dimensional inputs. If the input tensor has dimensions $a \times b \times c$, the output of the Flatten layer will be a vector of dimension $a \cdot b \cdot c$.
This layer does not modify the data values but rearranges them into a single vector, facilitating different types of processing downstream in the network.

5.7.3 Reshape Layer

The reshape layer molds the shape of an input tensor but does not affect how the data and elements are stored, therefore, the amount of elements stays the same as well. This operation is therefore crucial because it allows the transformation of the tensor dimensions to meet the requirements of the subsequent layers which is a key feature of a neural network architecture. Although there is no specific arithmetic formula for transform operation, we can approach it by rearranging the elements of the input tensor I into a new tensor O that fits into given dimensions, yet ensuring the total number of elements remains intact.
For example, if the input tensor I has dimensions $a \times b \times c$ and needs to be reshaped into dimensions $x \times y \times z$, then:

$$\boldsymbol{a \times b \times c = x \times y \times z} \tag{5.10}$$

where a, b, c are the original dimensions, and x, y, z are the new dimensions, under the constraint that the total number of elements (the product of the dimensions) remains unchanged.
This layer does not have any learning parameters or any computations other than changing the data structure which is computationally efficient but structurally important.

5.7.4 Global Average Pooling

The Global Average Pooling (GAP) layer computes the average of all elements from the feature maps of input tensor spatially, thereby splitting its spatial dimensions (height and width) to one dimension. This operation reduces the amount of model parameters, which offsets an overfitting danger, making the model more tolerant of the input spatial translations.
Mathematically, the operation of a Global Average Pooling layer can be described as follows:

$$\boldsymbol{O_c} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{x_{ic}} \tag{5.11}$$

where O_c is the output for the c-th channel, N is the number of elements in each channel of the input tensor, and x_{ic} are the elements of the c-th channel of the input tensor. This process is repeated for each feature map (channel) in the input, resulting in a tensor where each channel is reduced to a single scalar value that represents the average of all the input values of that channel. Global Average Pooling is especially suitable for applications involving convolutional neural networks in image classification where complexities of the networks can be affected by reducing the model and where the ability of the network to interpret is enhanced by making the feature maps correspond to specific categories.

5.7.5 Dense (ReLU)

A Dense layer, or a fully connected layer, is the one which tends to make a connection between every neuron in the preceding layer to every neuron in the following layer. This is one of the most important components of many neural network architectures that are often used as the main learning mechanism when the complexity of the patterns and the data are extremely difficult to be learned solely from the dataset. The coupling of Dense layer with ReLU (Rectified Linear Unit) activation function brings the additional non-linearity in the output of the network along with the linear combination of the initial inputs which simplifies the network to detect complex subtlety in the data and important features in a given dataset.
The mathematical representation of a Dense layer with ReLU activation can be given as:

$$\boldsymbol{y} = \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{5.12}$$

where y is the output, W represents the weight matrix associated with the layer, x is the input vector, b is the bias vector, and ReLU is defined by the function:

$$\mathrm{ReLU}(\boldsymbol{z}) = \mathbf{max}(\mathbf{0}, \boldsymbol{z}) \tag{5.13}$$

The ReLU function is particularly effective because it introduces non-linearity without affecting the scale of the input, which helps prevent the vanishing gradient problem during backpropagation.

5.7.6 Dense (Sigmoid)

A Dense layer with an activation function of Sigmoid is a typical one in neural networks to deal with binary classification issues. The sigmoid activation function

is a mapping function that maps the input values to an output range of 0 to 1 which is very useful for probability estimation. The mathematical representation of a Dense layer with Sigmoid activation is given by:

$$\boldsymbol{y = \sigma(Wx + b)} \tag{5.14}$$

where y is the output, W represents the weight matrix of the layer, x is the input vector, b is the bias vector, and σ is the sigmoid function defined as:

$$\boldsymbol{\sigma(z) = \frac{1}{1 + e^{-z}}} \tag{5.15}$$

5.7.7 Multi-Head Attention

Multi-Head Attention is a mechanism in neural networks that allows the model to jointly attend to information from different representation subspaces at different positions. It is a key component of transformer models, used primarily in natural language processing tasks.
The operation can be conceptualized mathematically as:

$$\text{MultiHead}(\boldsymbol{Q, K, V}) = \text{Concat}(\text{head}_\mathbf{1}, \ldots, \text{head}_\boldsymbol{h})\boldsymbol{W^O} \tag{5.16}$$

where each head head_i is computed as:

$$\text{head}_\boldsymbol{i} = \text{Attention}(\boldsymbol{QW_i^Q, KW_i^K, VW_i^V}) \tag{5.17}$$

Attention function is typically scaled dot-product attention:

$$\text{Attention}(\boldsymbol{Q, K, V}) = \text{softmax}\left(\frac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}\right)\boldsymbol{V} \tag{5.18}$$

Q, K, V are queries, keys, and values matrices respectively; W^O, W_i^Q, W_i^K, W_i^V are parameter matrices.

5.7.8 Layer Normalization

Layer Normalization is a technique used to stabilize the training of deep neural networks by normalizing the inputs across the features instead of the batch dimension. Mathematically, Layer Normalization is described by:

$$\boldsymbol{y = \frac{x - \mu}{\sigma}\gamma + \beta} \tag{5.19}$$

where x is the input to the layer, μ and σ are the mean and standard deviation computed across the features, γ and β are learnable parameters that rescale and shift the normalized value respectively.

5.7.9 Dropout

Dropout is a regularization technique used in neural networks to prevent overfitting. It works by randomly setting a fraction of input units to 0 at each update during training time, which helps to make the model robust.

5.7.10 Sequential Feedforward Network (FFN)

A Sequential Feedforward Network (FFN), often used within larger models like transformers, consists of several fully connected (dense) layers stacked sequentially. This structure allows the network to learn complex patterns in the data through successive transformations.
Typically, an FFN might be described by:

$$\boldsymbol{y = f(W_2 \cdot \mathrm{ReLU}(W_1 x + b_1) + b_2)} \tag{5.20}$$

where W_1, W_2 are weight matrices, b_1, b_2 are biases, and f is an activation function applied to the output of the last layer, often a non-linearity like ReLU.

5.8 Hyper Parameters and Auxiliary Functions

5.8.1 Optimizer

The optimizer updates the weights of the network based on the gradients of the loss function. In this model, the Adam optimizer is used, which is an adaptive learning rate optimizer:

$$\textbf{Optimizer: Adam} \tag{5.21}$$

Adam combines the best properties of the AdaGrad and RMSProp algorithms to handle sparse gradients on noisy problems.

5.8.2 Loss Function

The loss function measures how well the model performs during training by comparing the model's predictions with the true data. The chosen loss function is sparse categorical crossentropy:

$$\textbf{Loss Function: Sparse Categorical Crossentropy} \tag{5.22}$$

This loss function is suitable for classification problems with many classes where each class is mutually exclusive.

5.8.3 Metrics

Metrics are used to evaluate the performance of the model. Here, accuracy is used as a metric:

$$\textbf{Metric: Accuracy} \tag{5.23}$$

Accuracy measures the proportion of correct predictions over total predictions, providing an intuitive interpretation of the model's performance.

Chapter 6

Result Analysis

6.1 Performance Metrics

6.1.1 Recall

Recall, also known as sensitivity, measures the comprehensiveness of a classifier's predictions. It quantifies the ability of a model to identify all relevant instances in the dataset. The formula for recall is given by:

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{6.1}$$

where TP represents the true positives, or correctly identified positive cases, and FN denotes the false negatives, or positive cases that the model incorrectly classified as negative.

6.1.2 Precision

Precision measures the accuracy of the positive predictions made by a classifier. It is calculated as the ratio of correctly identified positive samples to the total number of samples that were classified as positive, whether correctly or incorrectly. The formula for precision is given by:

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{6.2}$$

where TP represents true positives, or cases correctly identified as positive, and FP denotes false positives, or negative cases incorrectly classified as positive.

6.1.3 F1 Score

The harmonic mean known as the F1 Score is considered to be the balance between precision and recall in a classification model. Overall, a single metric is provided to assess its accuracy and its ability to correctly recognize the given significant cases. The formula for the F1 Score is given by:

$$\textbf{F1 Score} = 2 \times \frac{\textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}} \tag{6.3}$$

where Precision is the ratio of correctly predicted positive instances to all instances predicted as positive, and Recall is the ratio of correctly predicted positive instances to all actual positive instances.

6.1.4 Accuracy

Accuracy is a metric that quantifies the overall effectiveness of a classification model in correctly identifying both positive and negative cases. It is computed as the percentage of true results (both true positives and true negatives) among the total number of cases examined. The formula for accuracy is given by:

$$\textbf{Accuracy} = \left(\frac{TP + TN}{TP + TN + FP + FN} \right) \times 100\% \tag{6.4}$$

where TP denotes true positives, TN denotes true negatives, FP represents false positives, and FN represents false negatives.

6.1.5 Confusion Matrix

The Confusion Matrix is a very important technique that helps us to measure the effectiveness of classifying models. It allows one to conduct a visual and quantitative analysis, so one can know how well the trained model performed and at which places the model failed. The matrix is structured as follows:

- **True Positive (TP)**: Observations correctly predicted as positive.
- **True Negative (TN)**: Observations correctly predicted as negative.
- **False Positive (FP)**: Negative observations incorrectly classified as positive.
- **False Negative (FN)**: Positive observations incorrectly classified as negative.

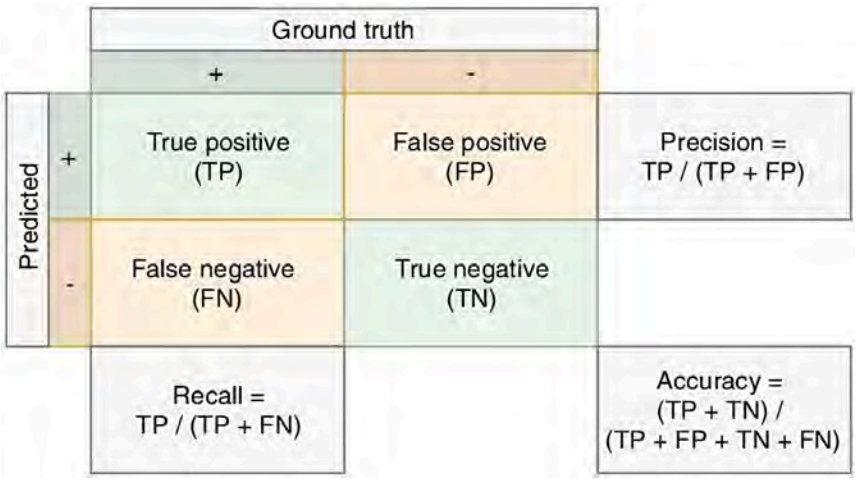

Figure 6.1: Performence Metrics

6.2 Performance Analysis

6.2.1 CNN

The CNN model has an 84.87% training success rate and an accompanying 84.86% testing success rate. It is periodically validated against a test set to ensure a wide applicability. The model remains spot on when tested rigorously for the task of predicting new images.

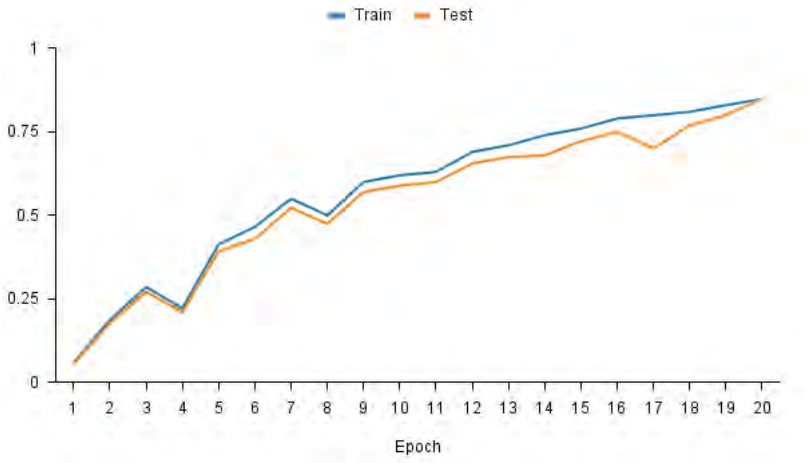

Figure 6.2: CNN Train and Test Accuracy

6.2.2 VGG16

The VGG16 model gets a training precision of 80.81% and a testing accuracy of 79.61%. The model's ability to generalize from the data it has learned to unseen data is what makes its performance robust. VGG16 performance during testing is slightly less than that of the training however, the model still retains a strong predictive ability which is of utmost importance for successful image classification.

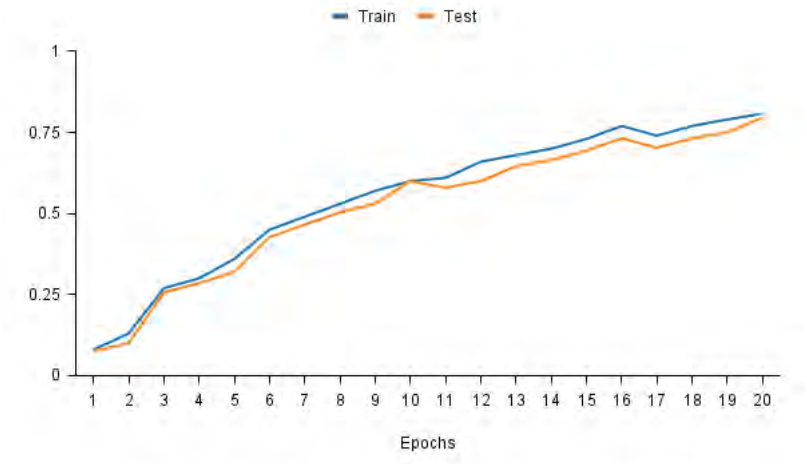

Figure 6.3: VGG16 Train and Test Accuracy

6.2.3 VGG19

For our implementation, we used various techniques including rotation and image shift on the training dataset. Model showed a training accuracy of 87.87% and a testing precision of 80.96%.

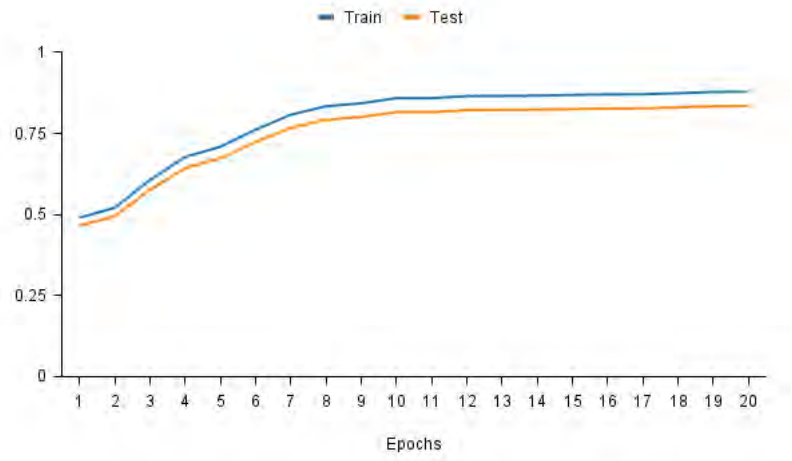

Figure 6.4: VGG19 Train and Test Accuracy

6.2.4 ResNet18

ResNet18 demonstrates the training accuracy of 83.33% and the testing accuracy of 83.27%. This model exhibits strong generalization, which is supported by the small gap between the training and the test results. Processing both training and new, unseen data with similar accuracy indicates that the ResNet18 model is consistent and robust, ready to process even complicated image recognition tasks.

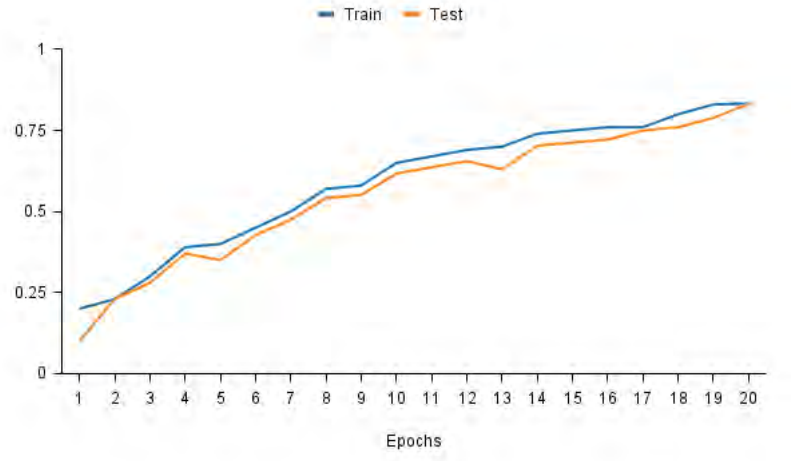

Figure 6.5: ResNet18 Train and Test Accuracy

6.2.5 ResNet50

In this implementation, we used a number of techniques to enhance the training data set, including rotating and shifting the pictures. This was done in order to make the model robust to image variations. It was a good approach as ResNet50 attained 88.70% accuracy in training and testing proving its valid generalization ability. Data augmentation was very essential in being able to work with new unseen images.

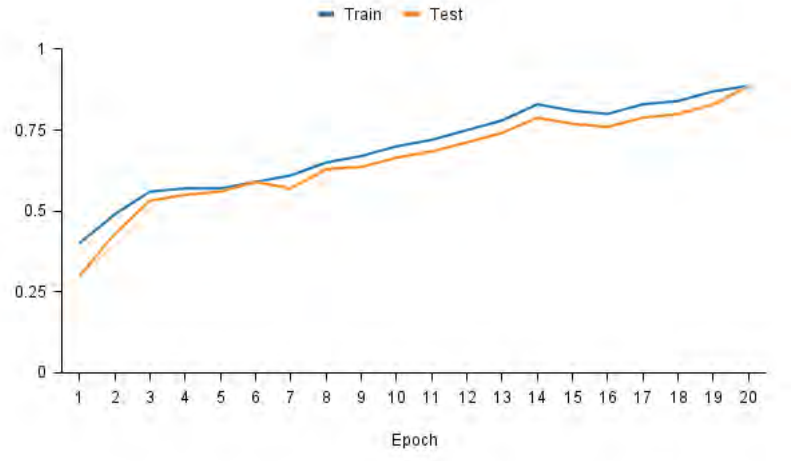

Figure 6.6: Resnet50 Train and Test Accuracy

6.2.6 DenseNet

DenseNet achieves a training accuracy of 86.37% and a testing accuracy of 85%. This model demonstrates strong capability in learning and generalizing from the training dataset to the testing dataset. The relatively close performance metrics indicate that DenseNet effectively handles overfitting, making it a reliable choice for tasks requiring robust image recognition capabilities.

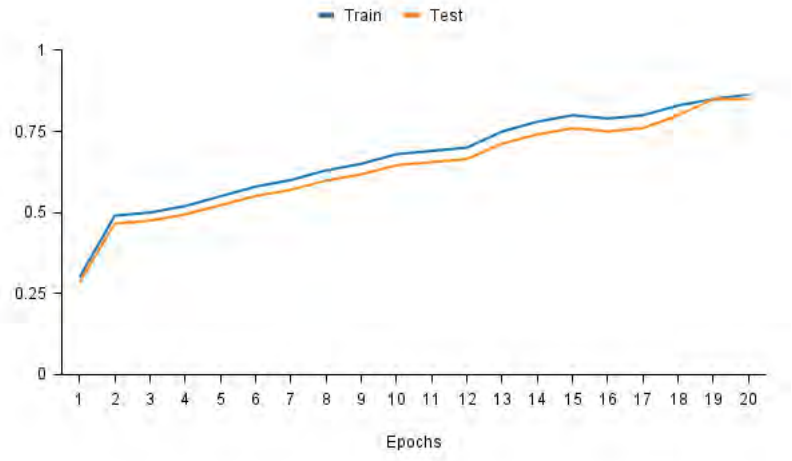

Figure 6.7: DenseNet Train and Test Accuracy

6.2.7 Inception V3

Inception V3 records a training accuracy of 69.75% and a testing accuracy of 57%. This significant drop between training and testing performance suggests challenges in the model's ability to generalize to new data, which might indicate overfitting or an inefficiency in capturing the necessary features from the training data effectively.

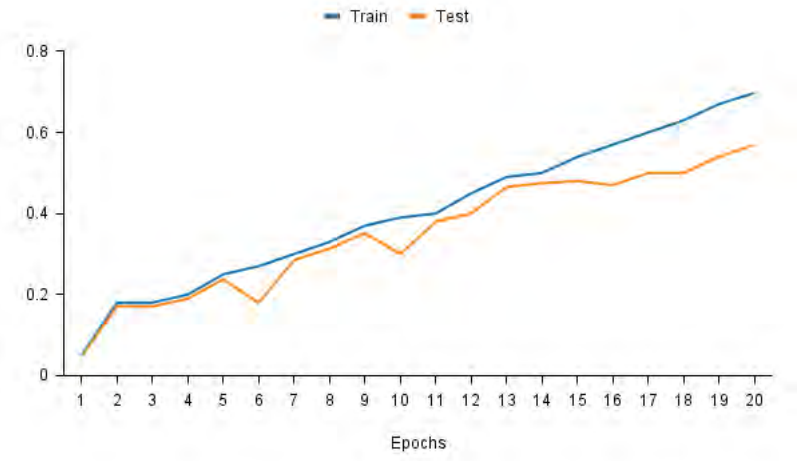

Figure 6.8: Inception V3 Train and Test Accuracy

6.2.8 EfficientNet

EfficientNet shows a training accuracy of 83.47% and a testing accuracy of 80%. This model exhibits a good balance between learning from the training data and generalizing to unseen data, with a moderate decrease in performance from training to testing. EfficientNet's architecture allows it to maintain a strong predictive performance, making it a suitable option for various image processing tasks.

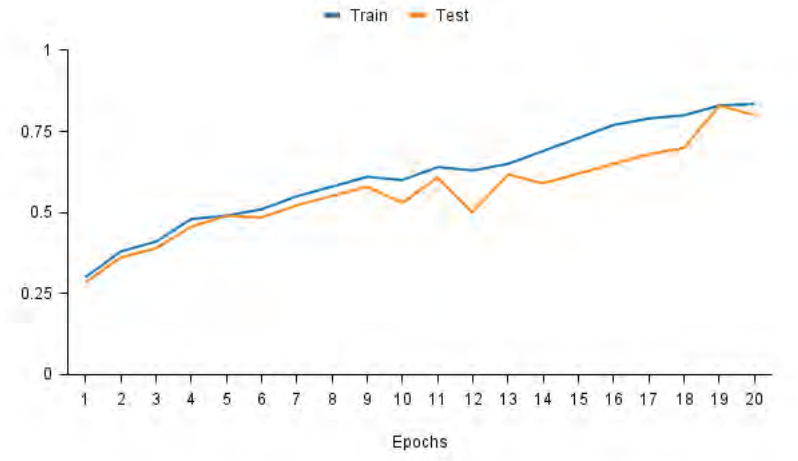

Figure 6.9: EfficientNet Train and Test Accuracy

6.2.9 MobileNet V2

MobileNet V2 achieves a training accuracy of 78.34% and a testing accuracy of 75.67%. This model is designed for environments where computational resources are limited, and it performs admirably given these constraints. The close alignment of training and testing accuracies suggests that MobileNet V2 effectively avoids overfitting, providing reliable and efficient performance in mobile or embedded applications.

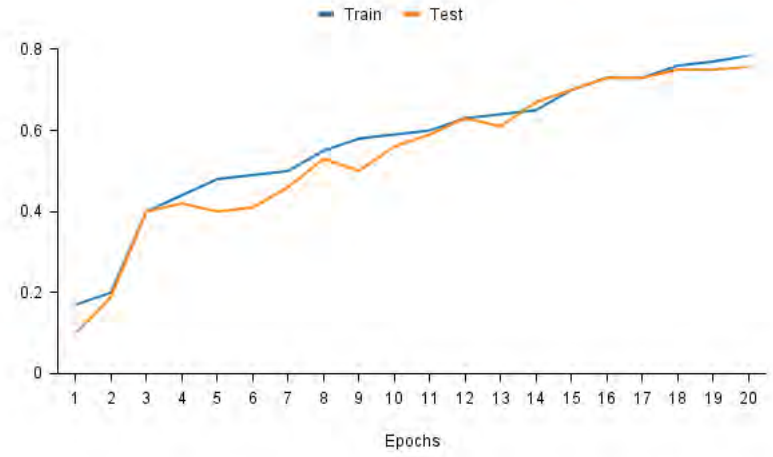

Figure 6.10: MobileNet V2 Train and Test Accuracy

6.2.10 SENet

Our specially-designed SENet model exhibited remarkable performance reaching 96.89% training and testing accuracy, which proves its usefulness and validity when applied to our dataset.

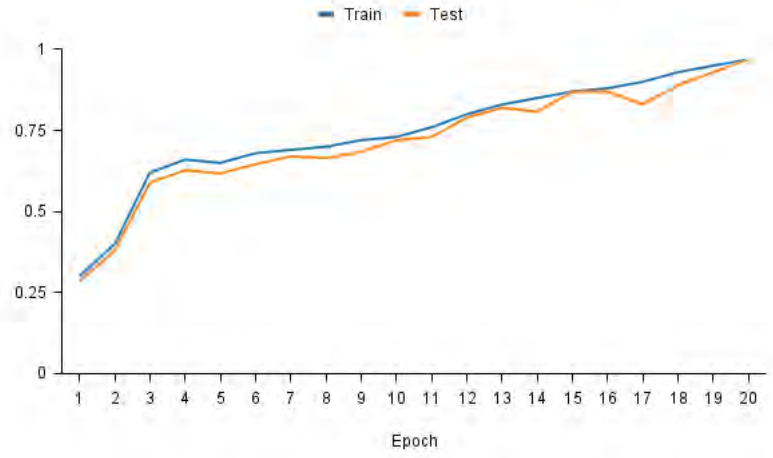

Figure 6.11: SENet Train and Test Accuracy

6.2.11 Proposed Spectro SETNet

Spectro SETNet, the proposed model, demonstrates exceptional performance with a training accuracy of 97.50% and a testing accuracy of 97%, achieved in just 10 epochs. This model not only outperforms the other existing models in terms of accuracy but also in efficiency, requiring fewer epochs to reach high levels of accuracy. Comparatively, SENet, which is the next best performing model, also achieves a high testing accuracy of 96.89% but requires double the number of epochs (20 epochs) to train. Spectro SETNet's capability to deliver superior performance with significantly fewer epochs, especially with complex data, highlights its advanced efficiency and effectiveness. This makes it an ideal choice for applications demanding quick and accurate image processing with limited computational time.

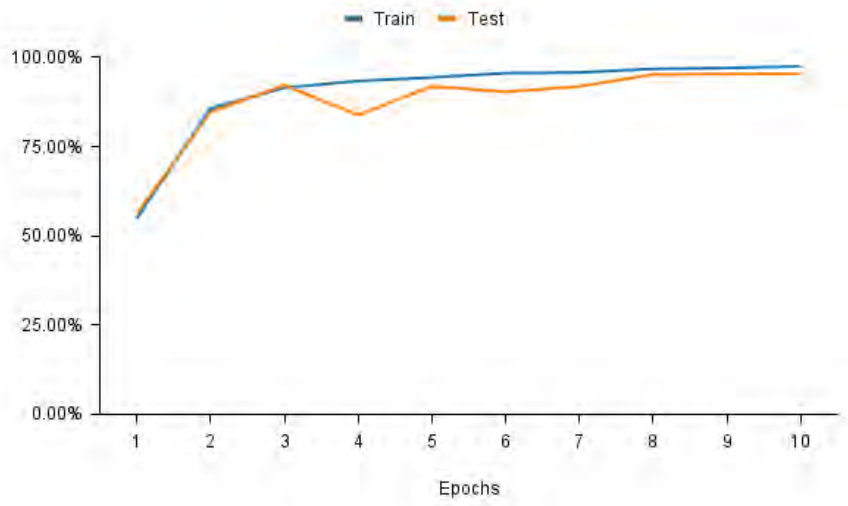

Figure 6.12: Spectro SETNet Train and Test Accuracy

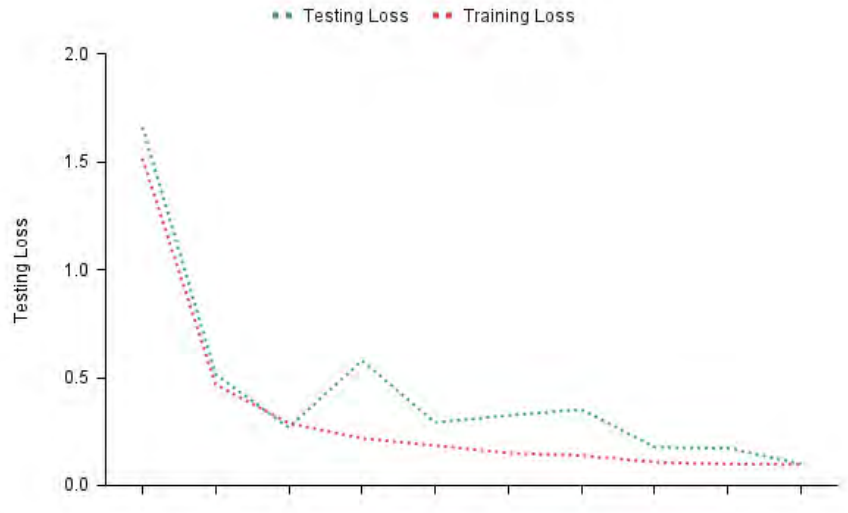

Figure 6.13: Spectro SETNet Train and Test Loss

Figure 6.14 that depicts a comparison between the training and testing accuracies of the Spectro SETNet model through 10 epochs. Initially, the value of training accuracy was at 54.81% and for test accuracy it was slightly higher at 56.24%. In every epoch, both metrics show positive growth where in many cases test accuracy outperforms training accuracy. When the model reaches its 10th epoch, the figures have changed significantly— with training accuracy now standing at a remarkable 97.50% while test accuracy is noted at 95.39%. This shows a model that has performed well throughout the ten epochs without being greatly affected by overfitting problems; hence demonstrating success in creating a generalizable model from this data.

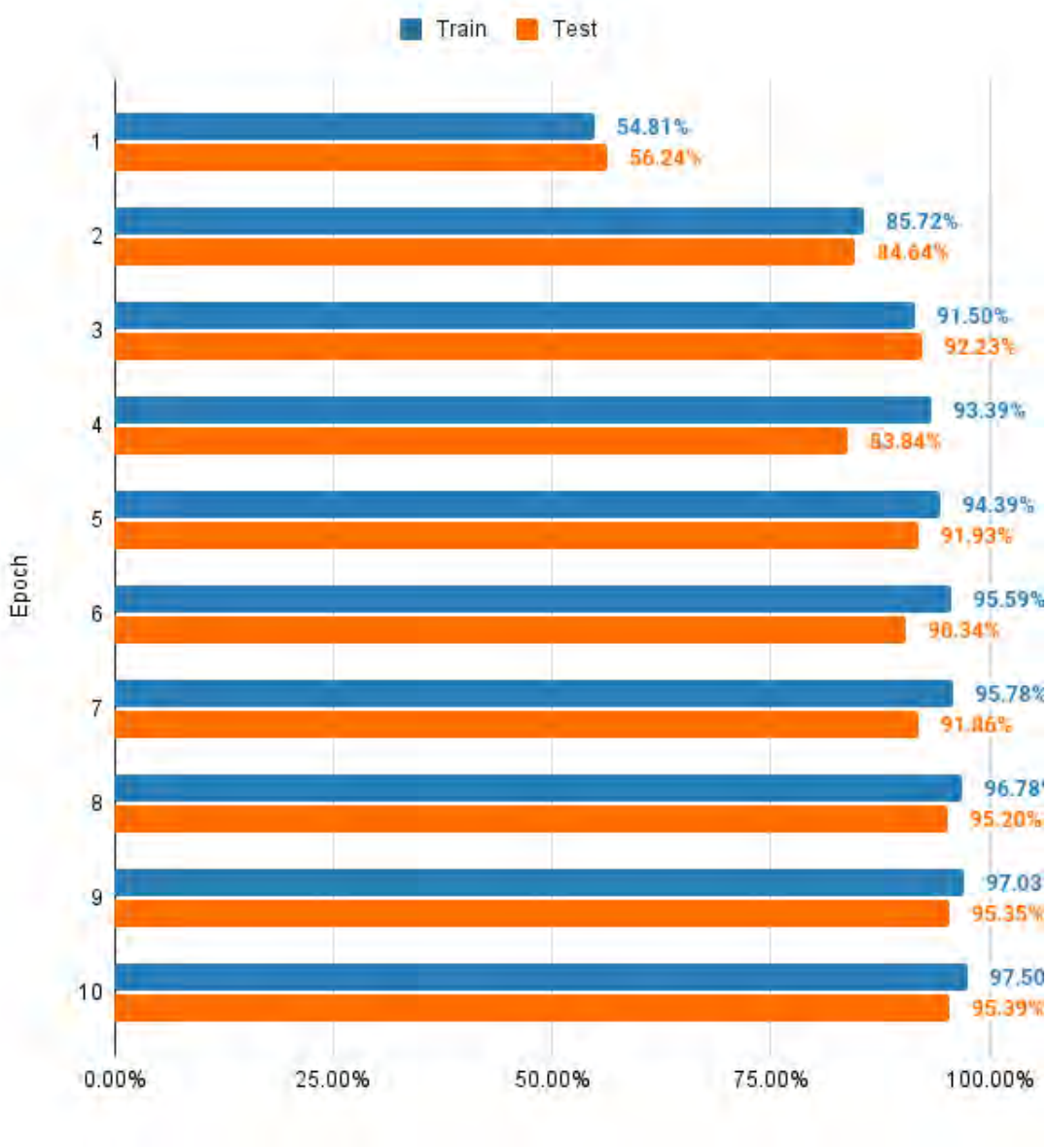

Figure 6.14: Spectro SETNet Train and Test Accuracy Bar Chart

The bar chart shown in Figure 6.15, indicates the progression of training and testing losses for the Spectro SETNet model across 10 epochs. At the start, the training loss is at a value of 1.517 while the testing loss is slightly higher at 1.6629; both values see a considerable decrease throughout the epochs— suggesting advancement in model performance. Upon reaching the final epoch, it is observed that the training loss dwindles to a mere 0.1 (with the testing loss marginally exceeding this value at around 0.1); this parity unveils successful learning dynamics coupled with generalization phenomena while striving to keep overfitting aberrations at bay.

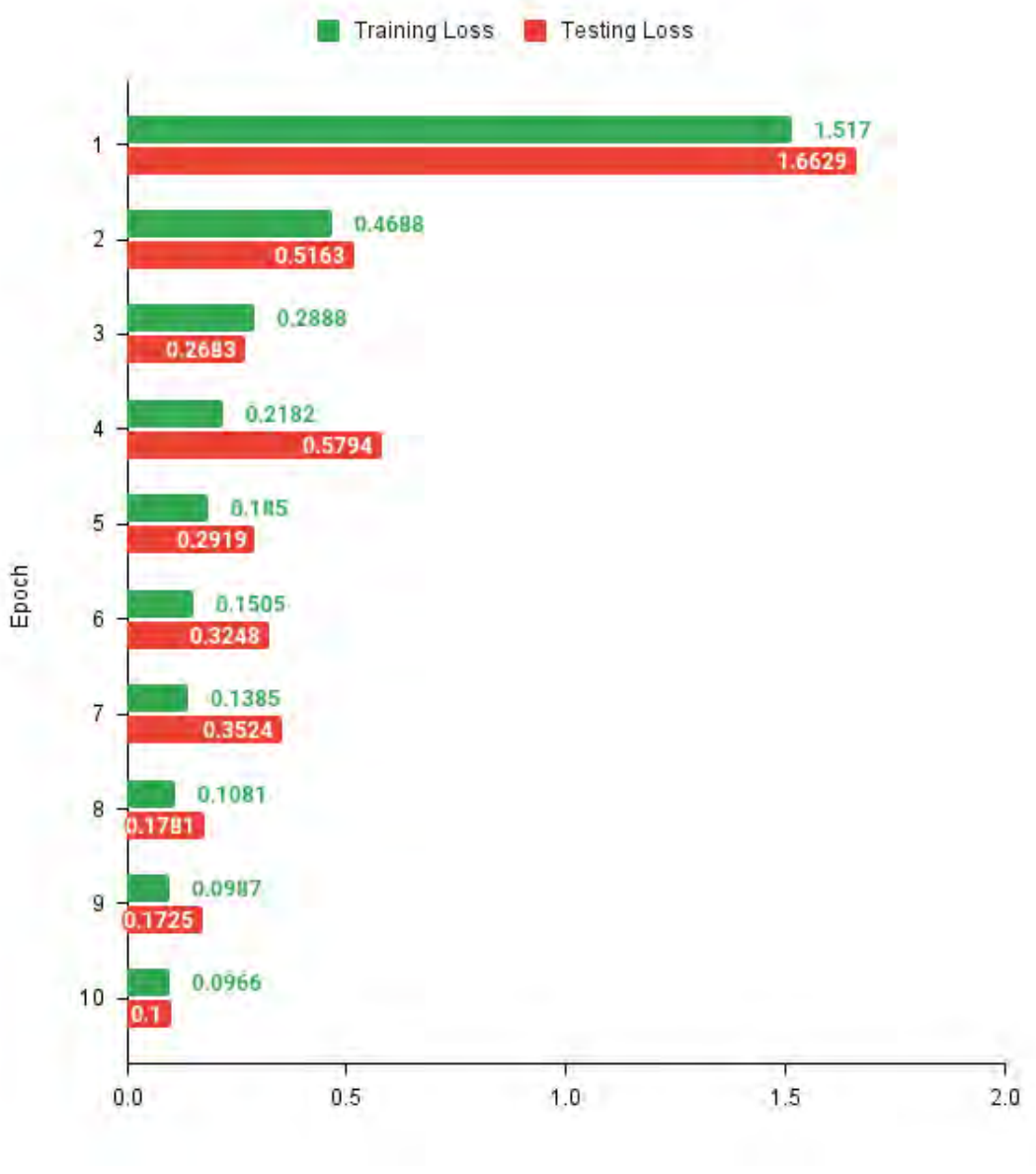

Figure 6.15: Spectro SETNet Train and Test Loss Bar Chart

As Figure 6.16 describes in the graph, there are three bar charts depicting the precision, recall, and F1 score of the Spectro SETNet model. The precision which is marked at 0.970 stands for the ratio of true positives to all positive predictions. Recall that has a value of 0.974, which represents the number of relevant instances that were retrieved divided by the number of relevant instances in the dataset. Lastly, the F1 score that is equal to 0.975 is a measure that combines two of the most commonly used classification metrics such as precision and recall into one metric value that gives us a good overview of how well our model performs for prediction class tasks.

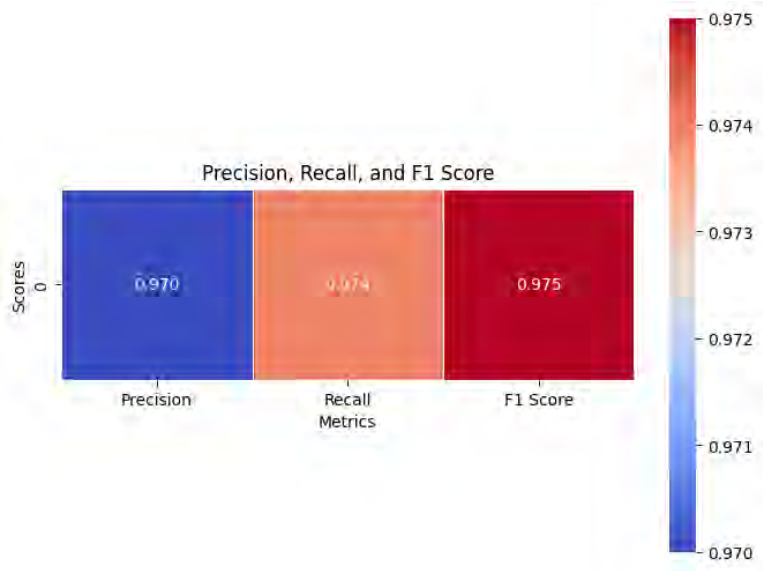

Figure 6.16: Spectro SETNet Precision, F1, Recall Score

Figure 6.17 shows Spectro SETNet's class predictions compared to true labels which demonstrate accurate predictions for various classes.

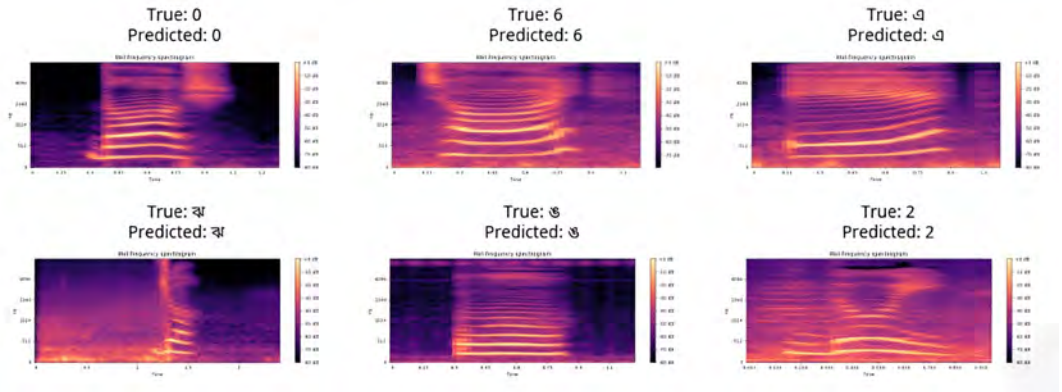

Figure 6.17: Spectro SETNet Class Prediction

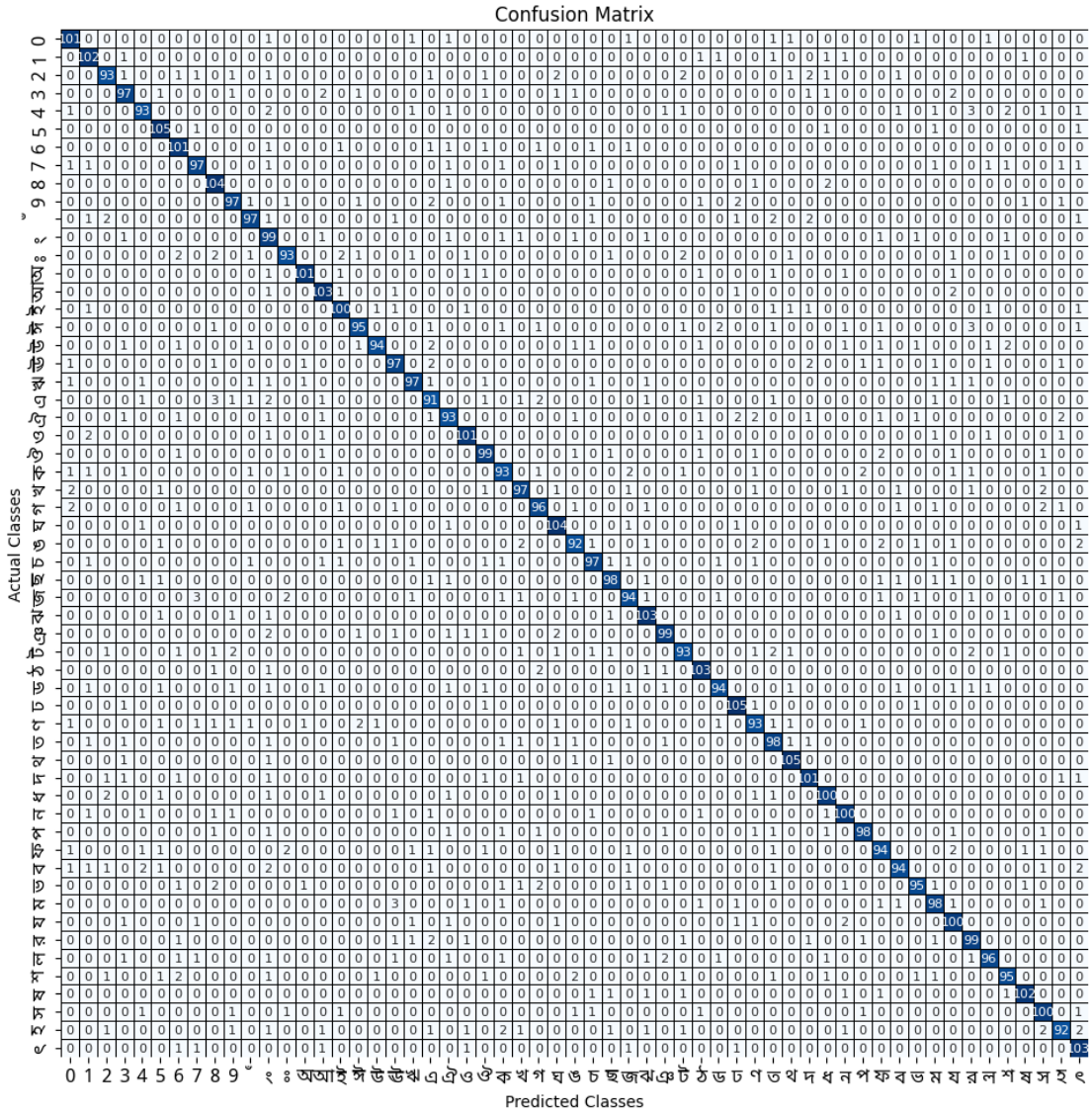

Figure 6.18: Spectro SETNet Confusion Matrix

6.3 Discussion

6.3.1 Comparative Analysis of the Performance of Models

In comparing the neural network architectures from the dataset, we observe significant differences in performance across models:

Highest Performer - Spectro SETNet (Proposed Model): With a test accuracy of 97%, achieved in just 10 epochs, Spectro SETNet tops the list. Its high accuracy and efficiency, even with complex data, mark a significant advancement in neural network capabilities, distinguishing it from all other models.

Notable High Performer - SENet: SENet also shows exceptional results with a test accuracy of 96.89%, demonstrating its effectiveness in handling feature interdependencies through its specialized attention mechanism.

Lowest Performer - Inception V3: In contrast, Inception V3 trails significantly with a test accuracy of 57%. This lower performance might be due to its complex module-based architecture, which can pose challenges in optimizing for specific datasets.

General Performance Overview - The remaining models exhibit a range of performances. CNN, a foundational architecture, demonstrates robustness with an accuracy of 84.86%, making it a reliable choice for various applications. ResNet50, with its deep residual learning, achieves a solid accuracy of 88.70%, effectively addressing deeper network challenges. VGG16 and VGG19, with accuracies of 79.61% and 80.96% respectively, offer dependable results but are somewhat overshadowed by newer, more efficient architectures. DenseNet stands out with an 85% accuracy due to its innovative feature reuse and reduction of overfitting. EfficientNet, balancing model complexity and computational resources, reaches an 80% accuracy. Lastly, MobileNet V2, optimized for mobile devices, scores 75.67%, highlighting its utility in resource-constrained environments.

Model	Epoch	Training Acc.	Test Acc.
CNN	20	84.87%	84.86%
VGG16	20	80.81%	79.61%
VGG19	20	87.87%	80.96%
ResNet18	20	83.33%	83.27%
ResNet50	20	88.70%	88.70%
DenseNet	20	86.37%	85%
Inception V3	20	69.75%	57%
EfficientNet	20	83.47%	80%
MobileNet V2	20	78.34%	75.67%
SENet	20	96.89%	96.89%
Spectro SETNet [Proposed]	**10**	**97.50%**	**97%**

Table 6.1: Accuracy Comparison Table of individual model

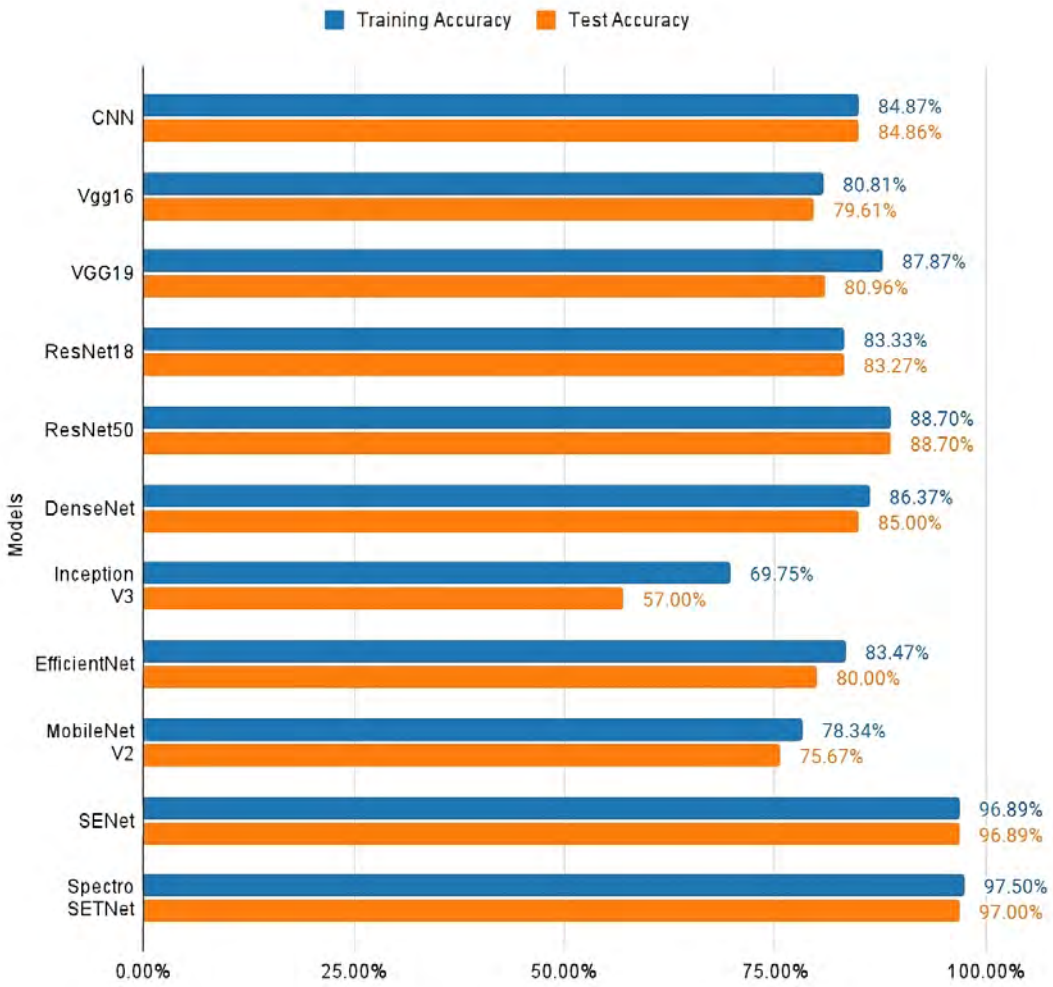

Figure 6.19: Model Accuracy Comparison

6.3.2 Collaborative Performance and Strengths

Our proposed, Spectro SETNet emerges as the top performer, with a precision of 97%, recall of 97.47%, and an F1-Score of 97.50%. These outstanding figures suggest that Spectro SETNet excels in both accurately identifying relevant data points and minimizing false positives and negatives, making it an exceptional model for critical tasks where high precision and recall are paramount. SENet also stands out for its exceptional accuracy, achieving a precision of 96.92%, recall of 96.89%, and an F1-Score of 96.89%. This model's high scores across all metrics indicate its ability to consistently and accurately process and classify data inputs with minimal error, positioning it as a preferred choice for applications where reliability is critical. ResNet50 shows strong performance with a precision of 88%, recall of 88.70%, and an F1-Score of 88.70%. These results highlight ResNet50's capability to effectively identify relevant features without significant false detections, making it suitable for tasks that require high sensitivity. VGG19 demonstrates solid consistency with precision and recall both at 87.69% and an F1-Score slightly higher at 87.87%. While it does not surpass SENet or ResNet50, its balanced performance makes it a reliable option for various general-purpose tasks. DenseNet performs commendably with a precision of 86.37%, recall of 85%, and an F1-Score of 85%. Its architecture, which emphasizes feature reuse, makes it effective in handling complex datasets with efficiency. EfficientNet, designed for efficiency, shows a precision of 83.04%, recall of 83.47%, and an F1-Score of 83.22%, making it a good choice for environments where computational resources are limited. CNN, although not the top performer, presents a decent balance with a precision of 84.86%, recall of 84%, and an F1-Score of 84%. It is robust and versatile, suitable for a broad range of tasks, though surpassed by more specialized models in this comparison. MobileNet V2 and Inception V3 are on the lower end, with MobileNet V2 scoring a precision of 78%, recall of 75.45%, and an F1-Score of 75.66%, and Inception V3 even lower with a precision of 57%, recall of 67%, and an F1-Score of 69%. These models are tailored for specific use cases, such as mobile applications and high-recall scenarios, respectively.

Model Name	**Precision**	**Recall**	**F1-Score**
CNN	84.86%	84.00%	84%
VGG16	79.61%	79.00%	80.56%
VGG19	87.69%	87.69%	87.87%
ResNet18	83.33%	83.20%	83.33%
ResNet50	88.00%	88.70%	88.70%
DenseNet	86.37%	85.00%	85.00%
Inception V3	57.00%	67.00%	69.00%
EfficientNet	83.04%	83.47%	83.22%
MobileNet V2	78.00%	75.45%	75.66%
SENet	96.92%	96.89%	96.89%
Spectro SETNet [Proposed]	**97.00%**	**97.47%**	**97.50%**

Table 6.2: Performance Comparison Table of Individual Models

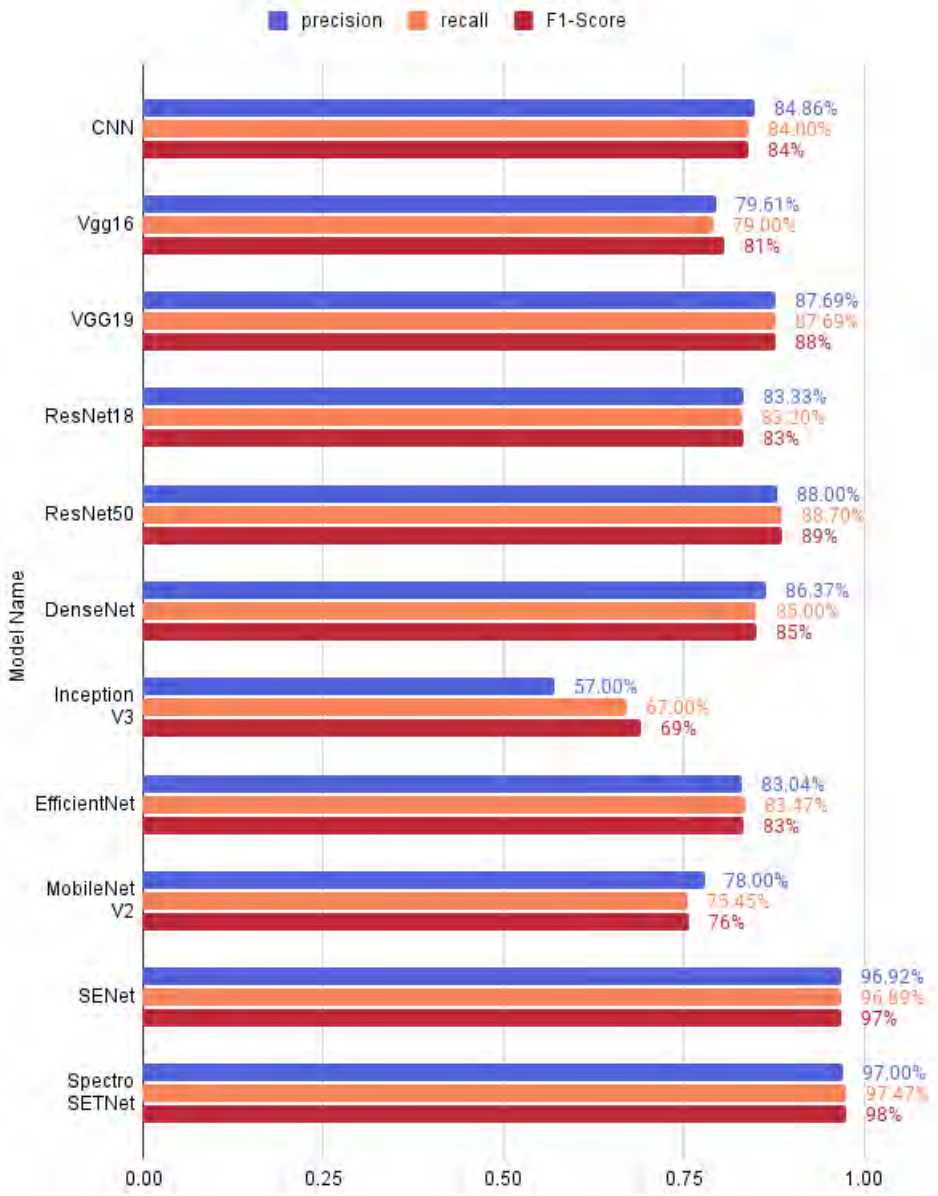

Figure 6.20: Performance Comparison of individual model

6.3.3 Findings

CNN, with its generalist approach, provides strong baseline capabilities in image processing, offering robustness across a broad range of tasks. Its uniform treatment of input data can sometimes hinder performance in applications requiring precise spatial recognition, highlighting the need for targeted enhancements in feature discrimination.

VGG16's deep architecture with small, repetitive filters excels at extracting fine details but may incur high computational costs and overfitting, suggesting a need for refined training approaches. VGG19 is well-suited for detailed image analysis due to its deep structure and fine filters, facilitating meticulous pattern recognition. However, the depth also introduces challenges in computational efficiency and potential overfitting, necessitating careful handling and optimization strategies to improve its adaptability.

ResNet18 leverages a residual learning framework to alleviate the vanishing gradient problem, showing a good balance between training and testing accuracy, though it could be optimized for handling complex data. ResNet50 employs deep residual networks to simplify the training of very deep architectures, effectively preventing the vanishing gradient problem and ensuring robust performance across varied tasks. Its ability to maintain high accuracy reflects well-balanced training and testing performance, although fine-tuning may enhance its capabilities in processing exceptionally complex data.

DenseNet uses densely connected layers for efficient feature usage, reducing overfitting risks and enhancing feature propagation, but may experience computational slowdowns.

Inception V3, with its inception modules, captures multi-scale information effectively, though it struggles with high precision tasks, indicating a need for architectural optimization.

EfficientNet achieves balanced scaling across different dimensions, enhancing adaptability and efficiency, yet fine-tuning is essential to avoid unnecessary complexity.

MobileNet V2, designed for mobile and edge computing, uses depthwise separable convolutions for efficiency but may lack in handling complex tasks due to its streamlined architecture.

SENet leverages Squeeze-and-Excitation blocks to enhance feature relevance selectively, significantly improving accuracy and efficiency in recognizing complex patterns. This model excels in contexts requiring high precision, though its specialized focus might limit broader generalization across diverse datasets.

The proposed Spectro SETNet shows remarkable accuracy and efficiency, suggesting that its innovative techniques significantly enhance complex task handling. Further architectural exploration could extend these benefits to other models.

6.3.4 Drawbacks in Predictive Accuracy

CNN is generally effective but occasionally misclassifies phonetically similar signs. This issue points to a need for improved feature extraction capabilities within the model to better distinguish between characters that sound alike but differ visually, thus enhancing the model's overall predictive accuracy.

VGG19 shows competency across multiple categories but struggles with characters like (গ, ঘ) and (চ, ছ). This pattern of misclassification could be indicative of insufficient training data for these particular characters, or it may stem from the model's learning mechanisms, which could be blending characters that share visual similarities.

ResNet50 generally performs well, but it encounters difficulties with specific characters such as (প and ফ). These issues suggest that despite its overall reliability, ResNet50 might benefit from adjustments in its approach to handling rare or visually similar characters that are underrepresented in the training data.

DenseNet and EfficientNet, while noted for their efficiency and effectiveness in various scenarios, might also suffer from similar issues of generalization to less common or more complex characters. This could be due to their dense and scalable architectures respectively, which, while powerful, may not capture all nuances without further tuning.

Inception V3 and MobileNet V2, designed for high efficiency and low-resource use, may not achieve the same level of accuracy in complex character recognition tasks. Their limitations may become evident when dealing with a broad diversity of character sets, particularly in environments requiring detailed character distinctions.

SENet, though highly effective for a broad range of characters, may still face limitations when applied to datasets with unique, less frequent characters or nuanced variations. This indicates a potential overfitting to the specific characteristics of the training dataset, which may not generalize well to new, unseen data sets or rare character types.

Spectro SETNet achieves remarkable predictive accuracy, but it may still encounter challenges with highly irregular or outlier data not well-represented in its training set, potentially indicating overfitting. This specialization, while yielding high performance, could impair its ability to generalize across diverse, unseen datasets. Moreover, the complexity and computational demands of Spectro SETNet might hinder its application in resource-constrained environments where processing efficiency is critical. It is essential to rigorously test and validate the model across varied datasets to ensure its robustness and versatility in real-world scenarios, mitigating any limitations in scalability and generalization.

6.4 Additional Experiments

6.4.1 Batch Size

Small batches can help improve a model's accuracy. They can dodge local minima, enhancing its general use. But, there's a drawback. The gradient estimates may be noisy, making learning rough and exploratory. This is just one way batch sizes impact image categorization. Big batches need large memory and might result in poorer applicability. Because they could meet sharp minima. They can, though, speed up training due to better utilization of computing resources, and gradient estimates are more consistent.

6.4.2 Data Enrichment

The training data is subjected to various data augmentation techniques, such as rotation, flipping, and random cropping. We use these tactics to boost how well the model adapts. That made the validation accuracy lift up a little but still, it's noteworthy.

6.5 Challenges

6.5.1 Dataset Collection

There were many difficulties in gathering the dataset, mostly because we had to begin from scratch in the absence of any previous data. It was especially difficult to record characters from children between the ages of two and five. Dealing with the unpredictable nature of young children and the challenges of obtaining clear and consistent recordings was part of working with them. Every character sample needed to be carefully converted into the WAV audio format after it was recorded, which called for close attention to detail.

Using Audacity to divide these recordings into thirty segments was yet another significant challenge. To guarantee that each character was precisely recorded, this step requires a high degree of precision in addition to being time-consuming. Furthermore, there were additional difficulties involved in eliminating background noise from these recordings, again with Audacity. It took dexterity and keen hearing to ensure that the recordings were clear while removing unwanted background noise.

Organizing the enormous amount of data was another major difficulty. The process frequently required multiple recording sessions because it involved datasets from 21 children and voice recordings from 57 classes. In addition to being labor intensive, managing, chopping, and editing such a large amount of data also required a high level of technical skill and patience. All things considered, the entire process from recording to final data preparation highlighted the difficulties and resource requirements involved in building a specialized dataset from the ground up that is of a high caliber.

6.5.2 Module Import Errors

We frequently had to import different versions for four different models due to the conflicting versions of our many models, which presented the issue of version mismatching. There were many challenges at first, particularly with module import issues. Inadequate Python packages and incorrect environment setups resulted in a number of issues that affected data processing and model training. Solving these issues was essential to maintaining the validity of the research and ensuring that the chosen machine learning libraries performed as intended.

6.5.3 GPU Error

We ran CNN, ResNet50, and VGG19 on our GPU, which had 4 GB of VRAM; however, we used the main system RAM for SENET. Despite having a potent NVIDIA GeForce GTX 1650 GPU at its disposal, preliminary tests showed that TensorFlow wasn't operating to its full capacity. This issue turned into a significant constraint, particularly for resource-intensive models such as VGG19, ResNet50, SENet, and CNN. A crucial aspect of the project was modifying the code and GPU configurations to guarantee optimal GPU utilization.

6.5.4 Memory Overuse

Our main system RAM was 16GB, which allowed us to run Vgg19, ResNet50, and CNN but not SENet, so we reduced the batch size. In a voice-to-image generation, we split the data and worked with four children. Then, since the RAM was unable to accommodate five children, we created the image from the WAV files. Implementing the SENet model often resulted in resource-exhausted errors, primarily because of its extensive parameter count and comprehensive architecture. Several tactics were used to get around this. Reducing the batch size was a crucial strategy for better controlling the model's resource requirements. We also looked into mixed-precision training for better computer performance. But it did bring up fresh issues for the SENet model's deployment which had to be managed carefully.

6.6 Experimental Setup

Experimental Setup	
Operating System	Windows 11
GPU Accelerators	NVIDIA® GeForce RTX 4060
CPU	Intel^{T} Core^{TM} i7-9750H
RAM	32GB
Python	3.10.11
DL Framework	TensorFlow 2.11.0
Batch Size	32
Image Size	224 x 224

Table 6.3: Summary of Experimental Setup

6.7 Testbed Implementation

Figure 6.21, shows the home page of our website dedicated to Bengali Phonemic understanding for child speech featuring two options for classification, image and (.wav) file classification.

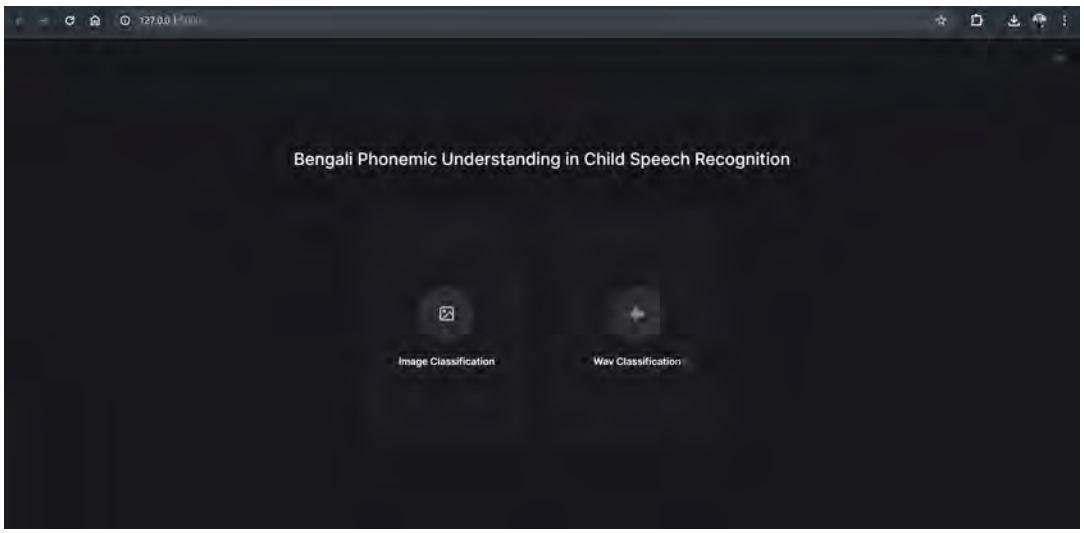

Figure 6.21: Home Page of Our Website

Figure 6.22 shows the image upload page of our website. Here users can select and upload an image for classification, followed by clicking the "Predict" button to predict the class of the uploaded spectrogram image.

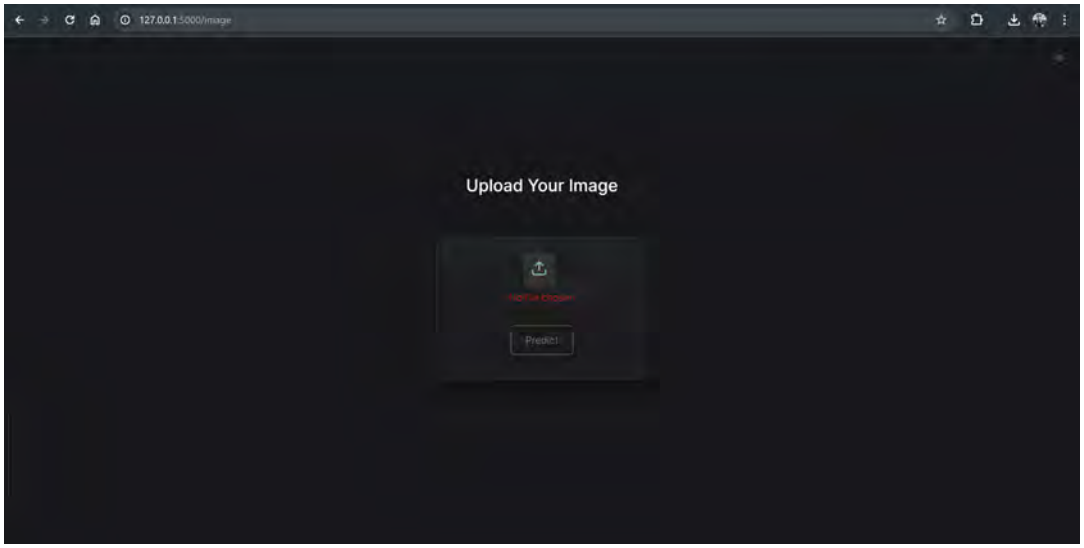

Figure 6.22: Image Upload Page

Figure 6.23 illustrates the image classification page. Here an uploaded image is being predicted with the class and is displayed along with its corresponding Mel-frequency spectrogram.

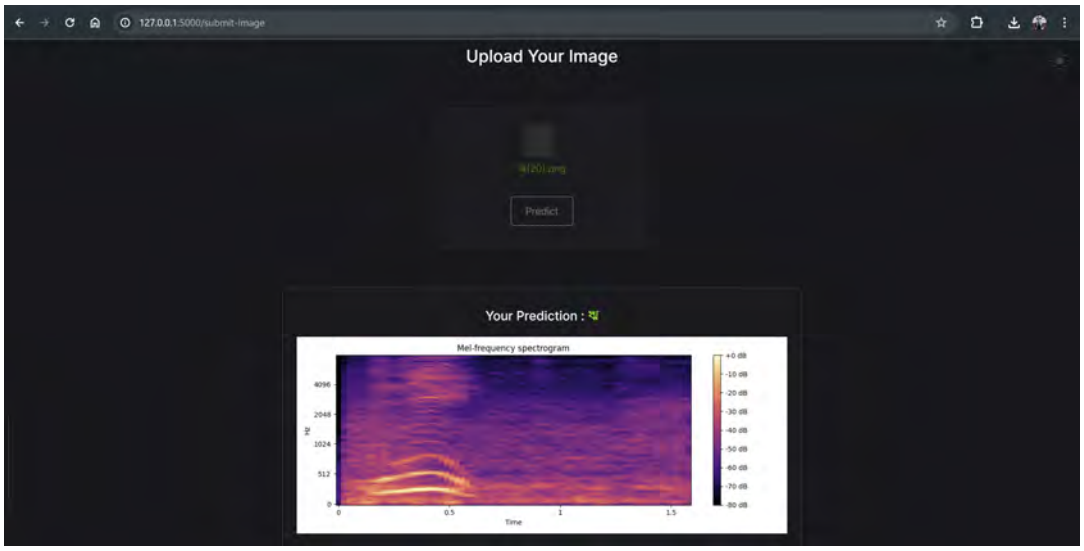

Figure 6.23: Image Classification Page

Figure 6.24 shows the image upload page of our website. Here users can select and upload an image for classification, followed by clicking the ”Predict” button to predict the class of the uploaded spectrogram image.

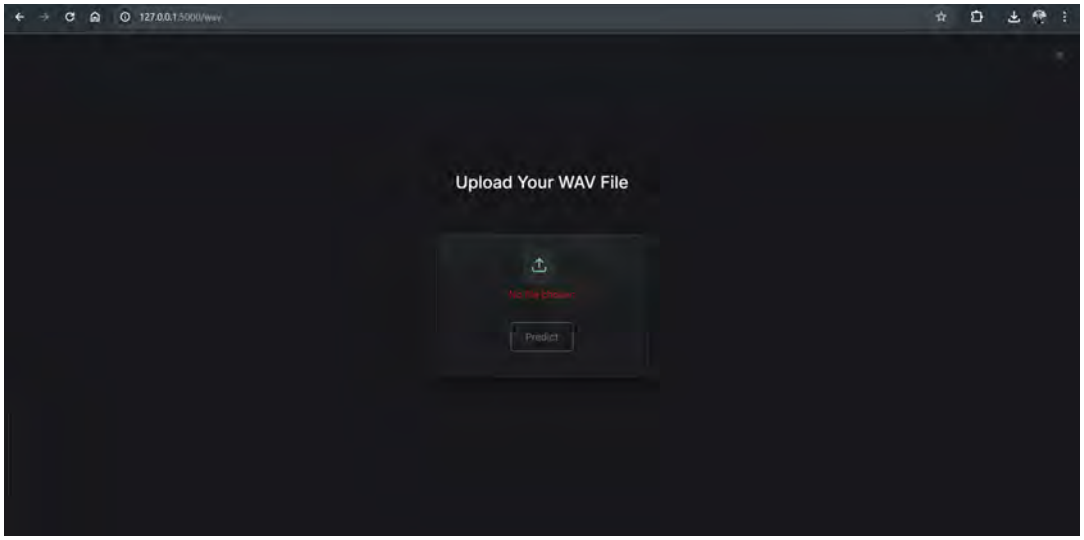

Figure 6.24: (.wav) File Upload Page

Figure 6.25 illustrates the wav classification page. Here the uploaded (.wav) file firstly being converted into mel frequency spectrogram image. Then the file is predicted with the class and is displayed along with its corresponding Mel-frequency spectrogram.

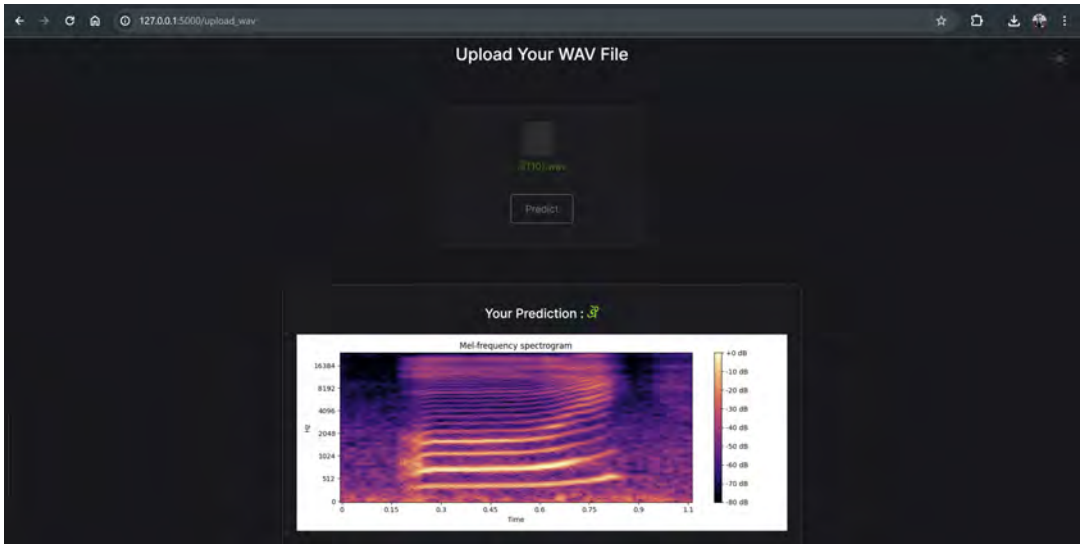

Figure 6.25: (.wav) File Classification Page

Chapter 7

Conclusion

In the thesis "Comprehensive Analysis and Development of Deep Learning Models for Bengali Character's Spectrogram Image Classification in Child Speech: Introduction of Spectro SETNet" we undertook a groundbreaking journey to enhance the field of character recognition in Bangla, focusing on the distinct linguistic patterns of children. Recognizing the significant gap in research on Bangla character' spectrogram image recognition, especially for children's speech, this study aimed to fill this void with a robust and innovative approach. We meticulously gathered a comprehensive dataset of 31147 audio, comprising vowels, consonants, and numerical vocals from young Bangla speakers, transforming these audio recordings into mel frequency spectrogram images. This critical step allowed us to leverage the potential of recent deep learning models, namely VGG16, VGG19, ResNet18, ResNet50, DenseNet, EfficientNet, Inception V3, MobileNet, SENet and our proposed model Spectro SETNet, to analyze and interpret these complex visual representations of sound. This research addresses the significant challenge of accurately identifying characters in children's speech spectrograms—a task previously underexplored. Our proposed model, Spectro SETNet, demonstrates exceptional proficiency in deciphering nuanced auditory cues within these spectrograms, significantly enhancing character recognition accuracy with reduced computational demands. This thesis contributes substantially to the advancement of linguistic technology and the development of educational tools for Bangla-speaking children. By implementing the Spectro SETNet model, we have made significant advancements in spectrogram image recognition for Bengali, effectively filling the dataset gap for this demographic and introducing a more efficient architectural approach to image recognition. The improvements in accuracy and efficiency not only enhance the model's real-world applicability but also lay a robust foundation for future research in this crucial area.

7.1 Future Work

Firstly, we need to collect more data to fill gaps in our Bangla character dataset of children. Having varied datasets can enhance outcomes. Getting plenty of diverse child voice samples in Bangla, for example, is key. By filling gaps in our data, make our system more effective, and provide more learning experiences for our models. This results in better performance on new tasks.

Secondly, another extension will focus on Bangla sentence recognition, moving beyond phonemic analysis to full sentence processing. This will help in achieving more advanced language understanding and application in real-world scenarios.

Lastly, to make an interactive online platform for children by which to teach Bengali children how to pronounce the Bangla character correctly.

Bibliography

[1] J. F. Werker and R. C. Tees, "Speech perception as a window for understanding plasticity and commitment in language systems of the brain.," *Developmental psychobiology*, vol. 46 3, pp. 233–51, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:9839518.

[2] M. M. Hasan, F. Hassan, G. M. M. Islam, *et al.*, "Bangla triphone hmm based word recognition," in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, 2010, pp. 883–886. DOI: 10.1109/APCCAS.2010.5775010.

[3] F. Hassan, M. S. A. Khan, M. R. A. Kotwal, and M. N. Huda, "Gender independent bangla automatic speech recognition," in *2012 International Conference on Informatics, Electronics Vision (ICIEV)*, 2012, pp. 144–148. DOI: 10.1109/ICIEV.2012.6317500.

[4] F. Hassan, M. R. A. Kotwal, and M. N. Huda, "Bangla phonetic feature table construction for automatic speech recognition," in *16th Int'l Conf. Computer and Information Technology*, 2014, pp. 51–55. DOI: 10.1109/ICCITechn.2014.6997376.

[5] S. B. Zinnat, R. M. A. Siddique, M. I. Hossain, D. M. Abdullah, and M. N. Huda, "Automatic word recognition for bangla spoken language," in *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, 2014, pp. 470–475. DOI: 10.1109/ICSPCT.2014.6884886.

[6] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, Nov. 2015.

[7] S. M. Rasel Kabir, F. Hassan, F. Ahamed, K. Mamun, M. N. Huda, and F. Nusrat, "Phonetic features enhancement for bangla automatic speech recognition," in *2015 International Conference on Computer and Information Engineering (ICCIE)*, 2015, pp. 25–28. DOI: 10.1109/CCIE.2015.7399309.

[8] S. T. Swarna, S. Ehsan, M. S. Islam, and Marium-E-Jannat, "A comprehensive survey on bengali phoneme recognition," *ArXiv*, vol. abs/1701.08156, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:10907786.

[9] S. Ahmed Sumon, J. Chowdhury, S. Debnath, N. Mohammed, and S. Momen, "Bangla short speech commands recognition using convolutional neural networks," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 2018, pp. 1–6. DOI: 10.1109/ICBSLP.2018.8554395.

[10] M. Mateen, J. Wen, D. Nasrullah, S. Song, and Z. Huang, "Fundus image classification using vgg-19 architecture with pca and svd," *Symmetry*, vol. 11, p. 1, Dec. 2018. DOI: 10.3390/sym11010001.

[11] M. M. Rahman, D. Roy Dipta, and M. M. Hasan, "Dynamic time warping assisted svm classifier for bangla speech recognition," in *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*, 2018, pp. 1–6. DOI: 10.1109/IC4ME2.2018.8465640.

[12] M. A. A. Amin, M. T. Islam, S. Kibria, and M. S. Rahman, "Continuous bengali speech recognition based on deep neural network," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–6. DOI: 10.1109/ECACE.2019.8679341.

[13] J. Islam, M. Mubassira, M. R. Islam, and A. K. Das, "A speech recognition system for bengali language using recurrent neural network," in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, 2019, pp. 73–76. DOI: 10.1109/CCOMS.2019.8821629.

[14] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP)*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:209080281.

[15] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," vol. 9, Oct. 2019, p9420. DOI: 10.29322/IJSRP.9.10.2019.p9420.

[16] C. Wang, D. Chen, H. Lin, *et al.*, "Pulmonary image classification based on inception-v3 transfer learning model," *IEEE Access*, vol. PP, pp. 1–1, Oct. 2019. DOI: 10.1109/ACCESS.2019.2946000.

[17] K. Dong, C. Zhou, R. Yihan, and Y. Li, "Mobilenetv2 model for image classification," Dec. 2020, pp. 476–480. DOI: 10.1109/ITCA52113.2020.00106.

[18] J. Liang, "Image classification based on resnet," *Journal of Physics: Conference Series*, vol. 1634, p. 012 110, Sep. 2020. DOI: 10.1088/1742-6596/1634/1/012110.

[19] J. Liang, "Image classification based on resnet," *Journal of Physics: Conference Series*, vol. 1634, p. 012 110, Sep. 2020. DOI: 10.1088/1742-6596/1634/1/012110.

[20] E. Rocha, L. Rodrigues Moreira, and J. Mari, "Maize leaf disease classification using convolutional neural networks and hyperparameter optimization," Oct. 2020. DOI: 10.5753/wvc.2020.13489.

[21] D. Eberhard, G. Simons, and C. Fennig, *Ethnologue: Languages of the World, 24th Edition*. Feb. 2021.

[22] S. Mascarenhas and M. Agarwal, "A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification," in *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, vol. 1, 2021, pp. 96–99. DOI: 10.1109/CENTCON52345.2021.9687944.

[23] O. Sen, M. Fuad, M. N. Islam, *et al.*, "Bangla natural language processing: A comprehensive review of classical, machine learning, and deep learning based methods," *ArXiv*, vol. abs/2105.14875, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235253826.

[24] G. Wang, Z. Guo, X. Wan, and X. Zheng, "Study on image classification algorithm based on improved densenet," *Journal of Physics: Conference Series*, vol. 1952, p. 022 011, Jun. 2021. DOI: 10.1088/1742-6596/1952/2/022011.

[25] R. Ayon, M. Rabbi, U. Habiba, and M. Hasana, "Bangla speech emotion detection using machine learning ensemble methods," *Advances in Science Technology and Engineering Systems Journal*, vol. 7, pp. 70–76, Nov. 2022. DOI: 10.25046/aj070608.

[26] M. Rakib, M. I. Hossain, N. Mohammed, and F. Rahman, "Bangla-wave: Improving bangla automatic speech recognition utilizing n-gram language models," *Proceedings of the 2023 12th International Conference on Software and Computer Applications*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252531829.

[27] A. V. Sai Abhishek, "Resnet18 model with sequential layer for computing accuracy on image classification dataset," vol. 10, pp. 2320–2882, Jul. 2022.

[28] K. Shaheed, A. Mao, I. Qureshi, M. Kumar, S. Hussain, and X. Zhang, "Recent advancements in finger vein recognition technology: Methodology, challenges and opportunities," *Information Fusion*, vol. 79, pp. 84–109, 2022, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.10.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521002025.

[29] H. Shahgir, K. Sayeed, and T. Zaman, *Applying wav2vec2 for speech recognition on bengali common voices dataset*, Sep. 2022.

[30] Y. Zhou, H. Chang, X. Lu, and Y. Lu, "Denseunet: Improved image classification method using standard convolution and dense transposed convolution," *Knowledge-Based Systems*, vol. 254, p. 109 658, 2022, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.109658. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122008395.