ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

NCSR DEMOKRITOS
INSTITUTE OF INFORMATICS AND TELECOMMUNICATIONS

Machine Learning for Children's Music Emotion Recognition

by

Georgios C. Batsis

Submitted

in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

April 2024

Author . . . .

Georgios C. Batsis
II-MSc "Artificial Intelligence"
April 30, 2024

Certified by. . . . .

Theodoros Giannakopoulos
Principal Researcher
Thesis Supervisor

Certified by. . . . .

George Vouros
Professor
Member of Examination Committee

Certified by. . . . .

Efthymios Papatzikis
Associate Professor
Member of Examination Committee

Machine Learning for Children's Music Emotion Recognition

By

Georgios C. Batsis

Submitted to the II-MSc "Artificial Intelligence" on
April 30, 2024,
in partial fulfillment of the
requirements for the MSc degree

Abstract
This work focuses on the application of Machine Learning techniques for Music Emotion Recognition, particularly focusing on children's music. The first step was to create a specialized dataset for children's music, which includes songs of varied emotions and cultural backgrounds, annotated by experts in child psychology, education, and Machine Learning Engineers. A Support Vector Machine was employed as a baseline model for the prediction task, processing a range of handcrafted audio features. Concerning more advanced models, Convolutional Neural Networks and a Dual-Stream architecture model, integrating both Convolutional and attention-based Long Short-Term Memory networks were evaluated. This approach offers a comprehensive analysis of children's music by examining both spectrograms and music transcription sequences. Models were evaluated using the Probabilistic Emotion Alignment to compare model posteriors with the probability distribution of expert annotations. Moreover, models evaluated using the established Machine Learning metrics, indicating that different modalities are able to enhance the predictive capacity for emotion recognition.

Thesis Supervisor: Theodoros Giannakopoulos
Title: Principal Researcher

Acknowledgments

Θα ήθελα να ευχαριστήσω τον κ. Θεόδωρο Γιαννακόπουλο, επιβλέπων της διπλωματικής εργασίας, για την στήριξη και την καθοδήγησή του κατά την διάρκεια της εκπόνησης της εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω τον κ. Ευθύμιο Παπατζίκη για την άριστη συνεργασία που είχαμε και την μεγάλη του συνεισφορά στην συλλογή δεδομένων, την κατεύθυνση της μεθοδολογίας και την ερμηνεία των αποτελεσμάτων. Πρόσθετα θέλω να ευχαριστήσω τον κ. Γεώργιο Βούρο που αποδέχτηκε την πρόσκληση να συμμετέχει στην εξεταστική επιτροπή.

Ακόμη, θα ήθελα να ευχαριστήσω την ομάδα του κ. Γιαννακόπουλου, τα παιδιά από το εργαστήριο MagCIL, αλλά και τα παιδιά που συνεργάζονται με τον κ. Παπατζίκη που βοήθησαν στο annotation των δεδομένων.

Ένα μεγάλο ευχαριστώ στην οικογένεια μου, τον πατέρα μου Χρήστο, την μητέρα μου Αναστασία και την αδερφή μου Αργυρώ, για την απεριόριστη στήριξη κατά την διάρκεια των σπουδών μου στο μεταπτυχιακό πρόγραμμα και όχι μόνο.

Ένα τεράστιο ευχαριστώ στην Δέσποινα για την υποστήριξη τόσο σε ερευνητικό, όσο και σε προσωπικό επίπεδο.

Contents

List of Figures

List of Tables

1 Introduction

This thesis consists of an in-depth exploration of Deep Learning (DL) application techniques for Music Emotion Recognition (MER), with a specific focus on children's music. The research intersects the critical disciplines of musicology, cognitive science, and Artificial Intelligence (AI), extending its implications to educational and therapeutic contexts since this genre can potentially influence the cognitive and emotional development of children.

The initial phase of the proposed methodology was the creation of a specialized dataset for children's music, meticulously collected from a variety of sources, including educational and entertainment platforms, aiming to create an inclusive dataset that contains songs of different ambiance and culture. The track selection includes songs suitable for various age groups, belonging in several musical styles, and transmitting different emotions. Expert annotations for emotional content were provided by professionals in child psychology, education, and Machine Learning (ML) engineers with expertise in Music Information Retrieval (MIR). This innovative dataset forms the foundational component of the training and evaluation of the DL models.

In the preliminary phase, a Support Vector Machines (SVM) classifier was utilized as a baseline for the MER task, due to its high efficiency in managing complex and high-dimensional data. The classifier processed a wide range of handcrafted audio features, such as Mel-Frequency Cepstral Coefficients (MFCCs) and Chroma features. The next stage entailed the deployment of Convolutional Neural Networks (CNN) and Dual-Stream DL model, that were selected for their exceptional pattern recognition capabilities in image and audio processing. On one hand, the CNN models were specifically adapted to analyze the spectrogram representations of the audio tracks enabling the identification of audio signal properties, crucial for discerning the emotional context in children's music. The CNN models were then enhanced by implementing various Transfer Learning configurations, to optimize their effectiveness in classifying the emotional content of children's music. On the other hand, the Dual-Stream DL model combines both CNN and attention-based Long Short-Term Memory (LSTM) networks, to process both spectrograms and music transcription sequences. This approach provides a comprehensive perspective on the emotional aspects of children's music by capturing information from both data types.

A comparative analysis was also conducted among the SVM and DL models for the evaluation of their effectiveness in accurately identifying and categorizing the emotional content of children's music. Evaluation metrics, such as accuracy, precision, recall, and the F1 score, were used to provide a detailed assessment of each model's performance. Additionally, a novel metric named Probabilistic Emotion Alignment is proposed to compare the model's posteriors with the emotional assessments of expert annotators. This work significantly contributes to the creation of an interdisciplinary knowledge base of the domains of musicology, cognitive science, and AI, while leveraging advanced DL models to offer a new perspective on the emotional dimensions of children's music and thus potentially impacting the understanding of its effects on the developing brain.

1.1 Problem Statement

Children's music is an intersection of cultural anthropology, cognitive science, and educational theory and consequently plays a pivotal role in the cognitive, emotional, and cultural development of children [6, 7]. Nevertheless, the exploration of emotional content in children's music via advanced computational models remains underrepresented in cognitive neuroscience. This gap in scientific literature highlights a missed opportunity to harness the capabilities of Data Science and Machine Learning (ML) in addressing the complexity and variability of musical data. This thesis aims to fill this gap by exploring the patterns, structures, and attributes of children's songs using diverse datasets and sophisticated ML algorithms.

The emotional dimension of musical compositions is not limited to subjective interpretations, but is also a quantifiable attribute. Based on the nombrable nature of music, MER was developed as a specialized field dedicated to the detailed analysis and classification of emotions in musical tracks. The advancements in ML and DL have provoked significant changes in MER [8], since they are exceptional for discerning and interpreting the complex emotional signals in music. ML models are capable of extracting useful information from audio signals, like music or speech, using a comprehensive spectrum of audio features derived from both the time and frequency domains of an audio signal. These features serve as inputs of traditional ML classifiers and more advanced DL architectures, such as CNNs and Recurrent Neural Networks (RNNs) [9].

In general, the construction of a MER dataset involves overcoming numerous challenges, particularly due to the subjective nature of emotion recognition, and thus it requires a refined labeling process. The diversity within musical genres and cultural contexts further complicates the accurate capture of a song's emotional essence. During the labeling process, experts categorize each track based on two emotional dimensions, namely valence (i.e., happy, sad, neutral) and arousal (i.e., strong, weak or neutral). To ensure label accuracy, statistical aggregation methods were employed to the votes of multiple annotators to reduce individual bias and to generate discrete labels [10]. The criteria for including or excluding music samples were the clarity of emotional expression and the level of reliability or disagreement among annotators.

Traditional methods of emotion recognition in music often rely on limited sets of rules or human interpretation, which are not always fitting for capturing the full spectrum of emotional expressions. This work addresses the challenges in accurately recognizing emotions in music by leveraging more sophisticated methods, like ML and DL. Considering the challenges and domain-specific characteristics of MER, the major contribution of this work can be summarized as follows:

- Development of a specialized dataset for children's music while encompassing a broad spectrum of emotions and cultural diversity, based on multi-discipline expert annotations.
- Applications of CNN models in MER: Customization and training of CNN models, initially trained to recognize emotions in western music, using diverse Transfer Learning configurations.
- Introduction of an innovative Dual-Stream model: A dual-stream architecture combining CNN with attention-based LSTM networks, processing both spectrograms and symbolic music representation sequences as part of a holistic analysis.

1.2 Thesis structure

This thesis is structured as follows:

- The introduction in Section 1 focuses on the importance of MER in children's music. It outlines the problem statement and accurately describes the main objectives of this research. Furthermore, this section includes a brief overview of the proposed methodology and a review of related work.
- Section 2 focuses on audio data processing and ML principles, such as audio feature extraction as well as DL algorithms and architectures in order to provide further insight to the technical components of this work.
- The proposed method is explained in detail in Section 3. More precisely, the complete workflow of this work is analyzed. Since three algorithms were employed for this problem, each step of every algorithm is described using diagrams and code snippets.
- Section 4 focuses on *Music in the crib* dataset creation, starting from the data collection and the comprehensive description of data annotation steps for the MER task.
- The results presented in Section 5 incorporate the outcomes of the experiments, the evaluation metrics, the audio feature analysis, and all the performance results of the three ML models.
- Finally, the conclusions in Section 6 provide a summary of this work's findings, their implications, and suggestions for future research in the field of MER, especially in the field of in children's music.

1.3 Related work

Recently, the methods for MER have experienced significant advancements driven by the rapid development in AI technologies. This chapter encompasses different methodologies for MER, each providing this field with unique findings in the complex relationship between music and emotions. A critical component of the research done in said domain involves the construction of specialized datasets, which serve as the foundation for training and evaluating various MER models. Moreover, the application of ML and multimodal learning techniques in MER are vital for highlighting different features and modalities to enhance the recognition and classification of emotions in music.

1.3.1 Construction of Music Emotion Recognition datasets

The successful development of MER frameworks is fundamentally reliant on the creation of comprehensive and accurately annotated datasets for the facilitation of the training and validation of ML frameworks. This dataset construction process involves several steps, with the most essential being the compilation of music tracks, annotation of emotional content, and precise assignment of labels.

CAL500, developed by Turnbull et al., features a collection of popular western music spanning five decades. This dataset was annotaded by sixty-six undergraduate students, who divided them into 18 different categories based on emotion on a three-point scale [11]. In addition, the *MoodSwings* dataset by Kim et al., that is created from pop music tracks, is unique in its approach to studying time-varying emotional perceptions. Its annotation utilizes a game-like interface, enabling ”players” to mark emotions within a continuous arousal-valence space [12].

Focusing on film soundtracks, the *Soundtracks* dataset by Eerola and Vuoskoski aims to mitigate biases commonly associated with well-known songs. Its annotations, that were created by musicologists and university students, encompass both categorical and dimensional emotions [13]. Furthermore, the *AMG1608* dataset, which contains contemporary western music, offers annotations for arousal and valence, providing a comprehensive view of mood categorizations in western music [14].

Emotify, constructed by Aljanaki, covers a variety of genres including rock, classical, pop, and electronic. This dataset employs the GEMS scale for annotations, introducing nine distinct emotional categories [15].

A pioneering approach is seen in the *DEAP* dataset by Koelstra et al., which incorporates electroencephalography (EEG) and physiological signals in its annotations of YouTube and Last.FM videos. providing dimensional ratings of arousal, valence, and dominance [16]. *Moodo*, created by Pesek et al., gathers samples from electronic, ethno, and popular music, that are annotated using choosing colors that correspond to the emotional perception of the music [17].

CH818, curated by Hu and Yang, features Chinese Pop songs annotated for arousal and valence by experts with a Chinese cultural background [18]. Likewise, the *4Q Emotion Dataset* by Panda et al., derived from the AllMusic API, categorizes songs into four quadrants based on arousal and valence [19].

The *MediaEval Database* by Soleymani et al. utilizes royalty-free music across a variety of genres. Its crowd-sourced annotations provide time-continuous arousal and valence ratings [20]. Another simmilar dataset is *PMEmo* by Zhang et al., that comprises of songs from international music charts, including time-continuous arousal and valence ratings as well as physiological data [5].

Lastly, the *EMOPIA* dataset, introduced by Hung et al., is notable for its focus on symbolic-domain music analysis and generation. This multi-modal dataset, that incorporates both audio and MIDI formats, enriches vailable for music emotion analysis and generation resources a [21].

1.3.2 Music Emotion Recognition using Machine Learning

MER has experienced a paradigm shift with the integration of ML techniques. Initially, this task relied on rule-based and fuzzy systems, but the onset of ML has enhanced its capacity to interpret complex emotional cues in music [22]. The applications of ML in MER began with conventional classifiers like K-Nearest Neighbors (KNN) and SVM [23]. Such models require handcrafted audio features extracted from the time and frequency domains of an audio signal, such as zero-crossing rate, energy, Mel-Frequency Cepstral Coefficients (MFCCs), rhythm-based features, and harmony-based features, like Chroma. In addition, lyrics and symbolic features, i.e., midi information, can obtain significant representations as well. The selection of these features is critical in determining the accuracy of the emotion recognition process [9]. Li et al. developed a SVM classifier using timbre, rhythmic and pitch features [24]. According to their findings, this classifier is not confident for certain classes. One of the first multimodal approaches is proposed by Laurier et al., who fused audio and lyrics to train a SVM and a Random Forest (RF) classifier, implicating that feature fusion enhances MER performance [25]. A more advanced multimodal approach for SVM is proposed in [26], where features obtained by audio and lyrics are combined by different feature fusion methods, showing that Late Fusion by Subtask Merging (LFSM) was the most precise method. Liu et al. [27] extracted emotion similarity embeddings combined with calibrated label ranking (CLR) for MER. Apart from these methods, conventional ML regressors have aslo been used to implement dimentional MER [28, 29, 30, 31, 32].

The domain of MER has evolved significantly with the arival of Deep Learning DL. Traditional ML methods, while groundbreaking, faced challenges in handling the high dimensionality and subjective nature of music data. Moreover, the time complexity of handcrafted feature extraction methods posed additional difficulties. DL models, however, have revolutionized MER frameworks by offering sophisticated data representation and streamlining feature extraction processes. Recently, Liu et al. presented a novel CNN-based approach for emotion classification in music. They utilized spectrograms for visual representation, bypassing the complex feature extraction process and outperforming existing techniques on benchmark datasets like CAL500 [33]. Following this paradigm, Koops et al. demonstrated the incorporation of mid-level perceptual features into DL models, specifically the VGG architecture a few years later. This integration not only maintained predictive performance, but also enhanced the explainability of MER models [34]. The same year, Er and Aydilek explored the use of pre-trained CNN models, including AlexNet and VGG-16, for obtaining visual features, followed by classification through SVM and softmax classifiers [35]. Additionally, Dong et al. introduced a Bidirectional Convolutional Recurrent Sparse Network (BCRSN), combining CNNs with Recurrent Neural Networks (RNNs), achieving significant improvements in accuracy and efficiency [36].

He et al. developed an innovative end-to-end DL framework that utilized raw audio signals as input to multi-view CNNs and Bidirectional LSTM (Bi-LSTM) networks. This approach effectively captured dynamic emotional content in music [37]. Rajech et al. focused on the influence of instrument types in MER, combining time and frequency domain features with RNNs to enhance accuracy in emotion identification [38]. Sarkar et al. modified the VGGNet to work with short audio segments and introduced a novel post-processing technique for aggregated emotion recognition in track level [39]. Yang et al. presented an improved backpropagation neural network, significantly enhancing the model's accuracy and speed in recognizing complex emotional dimensions [40]. Hizlisoy et al. aimed to refine the MER task by applying a combined CNN-LSTM model, incorporating log-mel filterbank energies and MFCCs along with standard acoustic features [41]. The most recent work in this domain is the one by Gupta et al. that introduced a novel approach using L3-Net deep audio embeddings and an attention-based neural network model with positional encoding. This method achieved notable results, while demonstrating the efficacy of attention mechanisms in MER without extensive feature engineering [42].

Transfer learning has also played a pivotal role in the evolution of MER. CNN models that are pre-trained in large vision tasks like ImageNet have shown strong generalization capacity in audio tasks, including MER [43]. The versatility of CNNs that are initially trained for music tagging, enhance their application to a variety of other music-related tasks, resulting in outstanding performance compared to traditional methods, and thus offering broad spectrum applications in MER [44]. Furthermore, the use of adversarial architectures for data augmentation and transfer learning techniques from other music domains and genres has demonstrated the superiority of DL models over traditional ML methods in classifying multiple emotional dimensions in music [45].

2 Background: Audio data processing and Machine Learning

This section focuses on audio data processing and ML principles, which constitute the basic background of this thesis. At first, a diverse range of audio feature extraction methods are explored, indicating their significance in obtaining informative representations. Subsequently, the focus shifts to the ML domain, specifically unraveling the architectures and algorithms that form the core of the proposed MER system. This exploration covers both traditional ML approaches and advanced DL methods.

2.1 Audio Features

The process of extracting informative representations from audio signals stands as a fundamental aspect in the audio data analysis domain. The primary goal of this process is to convert raw audio waveforms to numerically efficient formats suitable for computer-based recognition. Audio feature extraction methods capture informative characteristics of the audio signal that are crucial for domain-specific computational tasks such as speech recognition, music genre classification, and audio event detection [46].

2.1.1 Low-level audio features

Low-level audio features serve as the foundational layer in audio signal processing pipelines. These features include descriptors in both the time and frequency domains, such as zero crossing rate, spectral centroid, and Mel-Frequency Cepstral Coefficients (MFCCs), among others.

Techniques for Temporal Segmentation in audio feature extraction [4]

The partitioning of audio signals into temporally delimited segments, typically ranging from 20 to 100 milliseconds in duration, is necessitated for short-term analysis. Within these segments, features are extracted across both time and frequency domains, resulting in a series of feature vectors. The segmentation strategy is determined by the specific requirements of the application: overlapping frames, where the step size is less than the frame duration, or non-overlapping frames, characterized by a step size equivalent to the frame length. For mid-term analysis, this method is extended by utilizing larger temporal windows, generally spanning 1 to 10 seconds. These larger segments incorporate multiple short-term frames. A composite feature vector for each segment is derived through statistical aggregation (e.g., mean and standard deviation) of the features calculated at the frame level. In contexts involving audio recordings of substantial duration, long-term statistical aggregation techniques are applied to generate comparably informative features. This involves averaging the aggregated statistics across all mid-term segments, effectively condensing the extensive audio data into a singular, representative feature vector. This process is illustrated graphically in Figure 1.

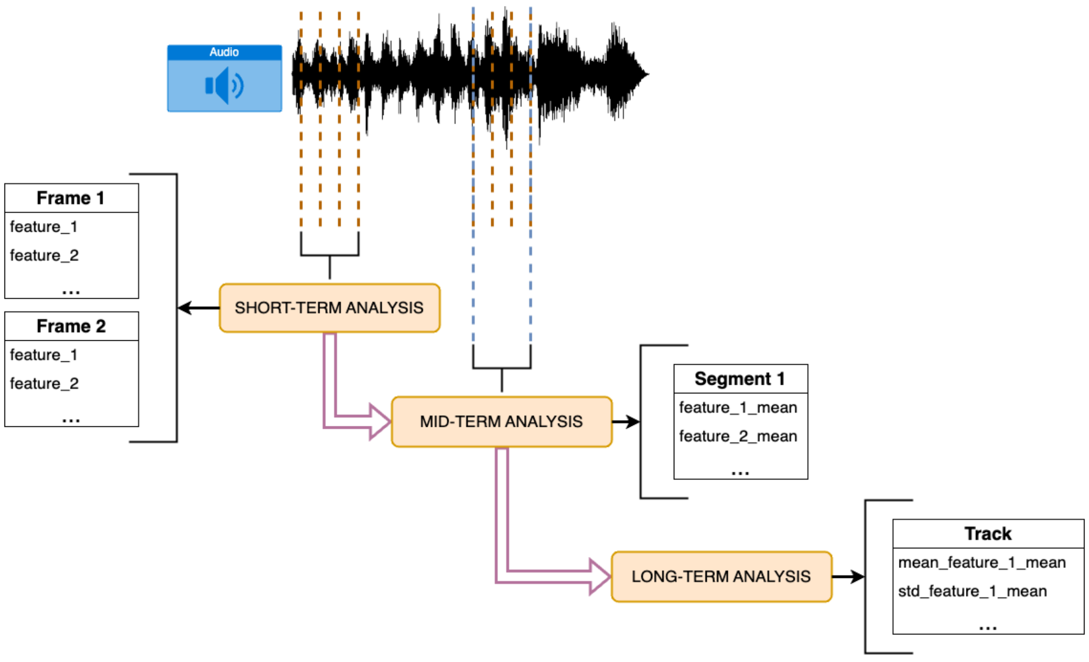


Figure 1: Temporal Segmentation and Aggregation in Audio Feature Extraction.

Time-domain features

Time-domain features provide a direct analysis of audio waveforms in the temporal axis. Such features are:

- *Zero-Crossing Rate* measures the sign changes rate of the waveform. In other words, this feature measures how quickly the signal changes from positive to negative amplitude and vice versa:

$$\mathrm{ZCR}(n) = \frac{1}{T-1}\sum_{t=1}^{T-1} |\mathrm{sgn}(x(t)) - \mathrm{sgn}(x(t-1))| \tag{1}$$

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

- *Energy* serves as an essential time-domain feature in audio signal processing, deeply rooted in the signal's physical characteristics. Formally defined, for a signal $x[n]$ with N samples in a given frame, the energy E can be articulated as:

$$E = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 \tag{2}$$

- *Energy Entropy* measures the measure of sadden changes in the signal's energy across different sub-frames within a given frame. For a given frame of a signal, it can be calculated by dividing the frame into M sub-frames, each with its own calculated energy E_i, and a cumulative frame energy E given by:

$$E = \sum_{i=1}^{M} E_i \tag{3}$$

  The Energy Entropy H is then defined as:

$$H = -\sum_{i=1}^{M} \left(\frac{E_i}{E} \log_2 \frac{E_i}{E} \right) \tag{4}$$

Frequency-domain features

Understanding audio signals necessitates the analysis of both the time and frequency domains. While time-domain features offer insights into the signal's temporal structure, frequency-domain features provide information about the spectral component. The transformation from time to frequency domain is most commonly facilitated by the Fast Fourier Transform (FFT). This algorithmic implementation of the Fourier Transform was specifically designed to ensure computational efficiency. The FFT algorithm transforms a sequence of N time-domain samples into an equivalent representation in the frequency domain. Mathematically, this transformation is expressed as:

$$X(f) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi fn/N} \tag{5}$$

The foundamental frequency-domain features in audio data processing are:

- *Spectral Centroid* is often called as the "center of mass" of the spectrum, and is defined as the weighted mean of the frequencies present in a signal. Mathematically, it is calculated as:

$$C = \frac{\sum_{n=0}^{N-1} f(n) \cdot |X(f(n))|}{\sum_{n=0}^{N-1} |X(f(n))|} \tag{6}$$

Where $f(n)$ is the frequency at bin n, and $X(f(n))$ is the FFT of the signal.

- *Spectral Spread* measures the bandwidth of the spectral content around its centroid, defined mathematically as:

$$\text{Spread} = \sqrt{\frac{\sum_{n=0}^{N-1} (f(n) - C)^2 \cdot |X(f(n))|}{\sum_{n=0}^{N-1} |X(f(n))|}} \tag{7}$$

- *Spectral Entropy* quantifies the amount of information contained in a spectrum and is defined as:

$$E = -\sum_{n=0}^{N-1} P(n) \log_2 P(n) \tag{8}$$

Where $P(n) = \frac{|X(f(n))|}{\sum_{n=0}^{N-1} |X(f(n))|}$ is the normalized spectral magnitude at bin n.

- *Spectral Flux* quantifies the rate of change in the spectral magnitude between two successive frames and is given by:

$$F = \sqrt{\sum_{n=0}^{N-1} (|X_t(f(n))| - |X_{t-1}(f(n))|)^2} \tag{9}$$

Where $X_t(f(n))$ and $X_{t-1}(f(n))$ are the magnitudes of the FFT in the current frame t and the previous frame $t-1$, respectively.

- *Spectral Rolloff* is the frequency below which a specified percentage of the total spectral energy falls. Mathematically, for a given percentage α, it is calculated as:

$$R = f(r) \quad \text{where} \quad \sum_{n=0}^{r} |X(f(n))| = \alpha \sum_{n=0}^{N-1} |X(f(n))| \tag{10}$$

Mel-Frequency Cepstral Coefficients (MFCCs)

Mel-Frequency Cepstral Coefficients (MFCCs) are a representation of the short-term power spectrum of sound. The process for calculating MFCCs consists of several key steps:

- Compute Discrete Fourier Transform (DFT) for each frame, given as:

$$X_t(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi kn}{N}} \tag{11}$$

  where $X_t(k)$ represents the k-th DFT coefficient and $x(n)$ is the time-domain signal.

- Mel-Scale Filter Bank Application The next step applies a bank of filters, typically 20-40, to the DFT coefficients. These filters are distributed across the frequency axis according to the Mel scale.

- Compute Filter Bank Output Power (Ok) of each filter using:

$$O_k = \sum_{f=0}^{N-1} |X_t(f)|^2 \cdot H_k(f) \tag{12}$$

  where $H_k(f)$ is the k-th Mel filter and $|X_t(f)|^2$ is the power spectral density.

- Discrete Cosine Transform: The next step is to compute the log power spectrum followed by the Discrete Cosine Transform (DCT):

$$c(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \log(O_k) \cos\left[\frac{\pi m(k-0.5)}{N}\right] \tag{13}$$

  Typically, the first 13 coefficients are chosen for further analysis.

- Generalized Cepstrum: The general cepstrum is obtained by applying an inverse FFT to the logarithm of the spectrum, given by:

$$c(n) = \mathcal{F}^{-1}\left\{\log\left(|\mathcal{F}\{x(t)\}|\right)\right\} \tag{14}$$

Chroma vector

The Chroma Vector serves as a 12-element frequency-domain representation designed to capture the 12 different pitch classes characteristic of ”Western Music”. A critical step in the calculation of the Chroma Vector involves the partitioning of the Discrete Fourier Transform (DFT) coefficients into 12 distinct bins. Each of these bins corresponds to one of the 12 equal-tempered pitch classes found in ”Western Music”. The mathematical representation for the binning process is:

$$B_k = \sum_{f \in S_k} |X(f)|, \tag{15}$$

where B_k is the amplitude of the k-th bin, and S_k is the set of frequencies grouped into that bin. The bins are spaced in semitone intervals, closely aligning with the frequency structure of western musical scales. S_k is defined as the set of frequencies for the k-th bin, and it represents specific DFT coefficients. The set S_k is essentially a gathering of all the DFT frequencies that fall under the k-th pitch class.

$$S_k = \{f | f \text{ is a DFT frequency and } f \text{ belongs to k-th pitch class}\} \tag{16}$$

The practical application of these low-level audio features in MER is significant for the identification of the emotional content embedded within musical compositions. For instance, the Zero-Crossing Rate, often reflective of the rhythm and texture of music, is utilized in distinguishing energetic tracks, thus contributing to the understanding of arousal in emotional contexts. Similarly, spectral features like the Spectral Centroid and Spectral Flux are employed to provide insights into the timbral quality of music, which is essential for the identification of emotions such as happiness or sadness. Furthermore, MFCCs and Chroma Vectors are instrumental in capturing the tonal characteristics of music, which are closely tied to the perception of mood and affect. By effectively analyzing these features, emotional states conveyed by music can be classified and predicted by MER systems, thereby offering invaluable tools for the corresponding applications.

2.1.2 High-level audio features

High-level audio features, such as Spectrograms and Mel-based features, provide intricate representations of audio data, crucial for advanced audio processing tasks. These features, offering detailed frequency-time representations, are pivotal in tasks requiring a deep analysis of audio signals, such as in MER. Their ability to visually represent a music signal makes them invaluable in Digital Image Processing and Computer Vision pipelines, where they assist in the informative interpretation of emotional content in music. Figure 2 illustrates an example of a spectrogram, showcasing how these features encapsulate the dynamic nature of audio signals.

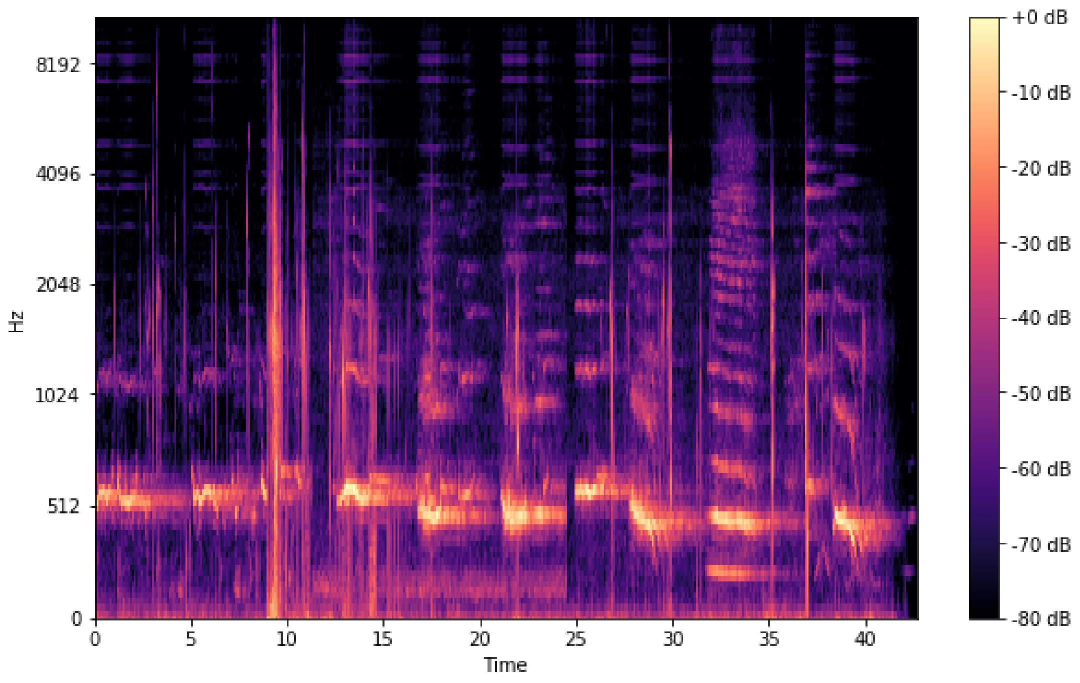


Figure 2: Spectogram example.

Spectrograms

Spectrogram serves as a visual representation of the spectrum of frequencies in an audio signal as they vary with time. As a result, x-axis is the corresponding time vector and in y-axis the frequency. Mathematically, a Spectrogram S is defined as the squared magnitude of the Short-Time Fourier Transform (STFT):

$$S(t, f) = |STFT(t, f)|^2 \tag{17}$$

Typically, Mel Spectrogram is used for more advanced DL applications. The conversion to Mel scale is accomplished through the application of a Mel filter bank on the Fourier Transform of the audio signal:

$$M(f) = 1127 \ln(1 + \frac{f}{700}) \tag{18}$$

While the Mel Spectrum provides a more perceptually relevant representation, taking the logarithm of it (Log - Mel Spectrum) further approximates the nonlinear attributes of loudness and pitch:

$$LogMel(f) = \log(1 + M(f)) \tag{19}$$

2.1.3 Symbolic music representation and transcription

Music transcription, an essential facet in the field of MIR, involves the complex process of converting raw audio signals into symbolic musical representation. This task aims to convert music tracks into a structured, digital format, mainly represented in the Musical Instrument Digital Interface (MIDI) protocol. MIDI is more than a digital representation; it's a language that encapsulates the essence of music in terms of discrete events, each event consisting of a note's starting time (s), pitch (p), and duration (d), formalized as:

$$\text{MIDI Event} = (s, p, d) \tag{20}$$

The journey of automatic music transcription is marked by the challenge of capturing the multifaceted nature of music, from the simplest melodies to the most complex harmonies. It employs advanced algorithms, such as Harmonic-Percussive Source Separation (HPSS) and Multiple Fundamental Frequency (F_0) Estimation, which dissect audio into its constituent components. These algorithms operate across both time and frequency domains, meticulously extracting details that make up the musical piece. Furthermore, advancements in DL have significantly enhance the capabilities of automatic music transcription, enhancing its accuracy and efficiency.

2.2 Machine Learning and Deep Neural Networks

Artificial Intelligence (AI) serves as the umbrella term encompassing techniques and algorithms aimed at enabling machines to mimic human-like cognitive functions. Within the domain of AI lies *Machine Learning (ML)*, a specialized subset that leverages statistical methods and principles of Computer Science to design models with the ability to *learn* correlations and patters from data [47]. As an evolution of ML, *Deep Learning (DL)* focuses on the simulation of human brain neural structure, offering advanced models like Deep Neural Networks for more complex problem-solving [48]. Figure 3 represents the hierarchical relationship among these three fields in the form of a Venn diagram.

Types of Learning in Machine Learning

Machine learning offers a diverse range of learning paradigms, each characterized by its unique attributes and applicability. The principal categories are Supervised, Unsupervised, Semi-supervised, Reinforcement, and Self-learning. Specialized techniques like Transfer Learning and Adversarial Learning are also noteworthy.

- Supervised learning algorithms aim to establish mathematical models incorporating both input features and corresponding output labels. In the realm of Super-

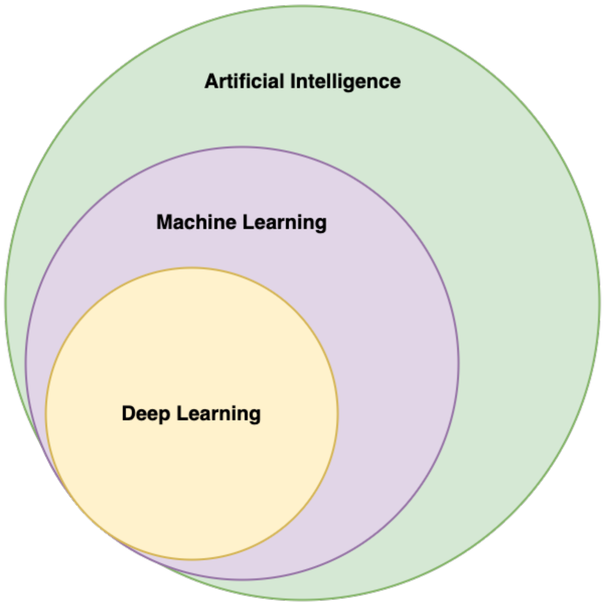


Figure 3: Venn diagram illustrating the relationship between AI, ML, and DL.

vised Learning, a Deep Learning (DL) model is trained using paired input-output data. The model's output can be either a continuous variable, typical for a Regression Task, or a discrete value representing the category to which the input data belongs, generally seen in Classification Tasks.

- Unsupervised Learning strives to discover patterns and relationships within unlabeled data, with no requirement for supervision.
- Semi-supervised Learning is an integration of both supervised and unsupervised methodologies, benefiting from a smaller labeled dataset alongside a larger unlabeled dataset to enhance model efficiency.
- Reinforcement Learning involves training agents to make decisions that maximize a cumulative reward function, generally in an interactive environment. Such a type of learning is commonly used in robotic systems.
- Self-supervised learning is a paradigm where a model learns representations from the data itself without requiring supervision. Instead, the model generates labels or targets from the input data, enabling it to learn meaningful representations through tasks such as pretext tasks or generative modeling.

2.2.1 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning algorithm extensively utilized for classification and regression tasks. It operates by identifying a hyperplane in an N-dimensional space (N being the number of features) that distinctly separates different classes of data points. Given a set of labeled data $(x_i, y_i), i = 1, \ldots, n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, the optimization problem for SVM can be solved as:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } y_i(w \cdot x_i + b) \geq 1,\ i = 1, \ldots, n \end{aligned} \tag{21}$$

In scenarios where the data is not linearly separable, SVM utilizes the kernel trick to map input vectors to a higher-dimensional space. A kernel function $K(x, z)$ is responsible to perform this type of mapping without loosing discriminative information and without having to compute the coordinates in the higher-dimensional space. Among the variety of kernel functions such as linear, polynomial, and sigmoid kernels, the Radial Basis Function (RBF) kernel stands out for its efficiency in handling non-linear data. It is mathematically defined as:

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|^2) \tag{22}$$

Where γ is a hyperparameter which can be adjusted to control the complexity of the model, and $\|.\|$ denotes the Euclidean norm.

2.2.2 Perceptron and Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models used for pattern recognition, whose structure and functionality resemble that of a biological neural network. They consist of a collection of neurons, each of which undergoes a specific computational process [49]. A neuron in the human brain receives electrical signals through dendrites, which are transmitted within the cell body. The signals that emanate from the cell body pass through the axon and propagate to other neurons through synapses. The structure and function of the Perceptron neuron, as depicted in Figure 4, are inspired by this particular procedure.

Its input X_n carries numerical data to the central body after being multiplied by certain weights W_n and the summation is calculated in the cell body, as follows:

Summation step:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b \tag{23}$$

Activation step:

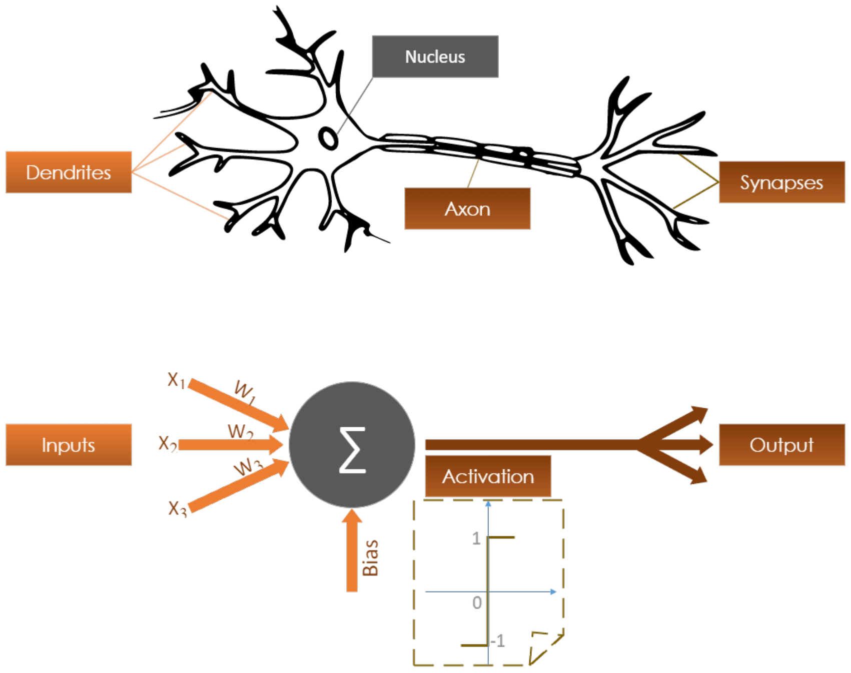


Figure 4: Perceptron functionality.

$$a = \begin{cases} 1 & \text{z} \geq \mathit{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

In the summation step, z represents the weighted sum of inputs x_i with corresponding weights w_i, plus a bias term b. The activation step applies a threshold function to the summation result z, where a is the activation output. If z is greater than or equal to a specified threshold, the activation output is 1; otherwise, it is 0. Considering these computational steps, the final version of Eq. 23 is:

$$z = f(\sum_{i=1}^{n} w_i \cdot x_i + b) \tag{25}$$

The Perceptron is often considered the progenitor of Artificial Neural Networks (ANNs) and serves as a fundamental building block of a key topology known as the Multilayer Perceptron (MLP). MLP is a subclass of Feed-Forward Neural Networks (FFNNs) and comprises Perceptron neurons organized into three distinct layers: the Input Layer, Hidden Layer, and Output Layer, as illustrated in Figure 5. Contemporary ANNs are characterized by increased computational complexity. This entails the incorporation of additional layers in the Hidden Layer equipped with nonlinear Activation Functions, as well as a larger volume of training data. These enhancements significantly elevate performance metrics, enabling ANNs to learn more intricate patterns [50]. Deep Neural Networks (DNNs), an extension of this paradigm, are employed for a myriad of applications including image and speech recognition, Music Information Retrieval (MIR) and Natural Language Processing (NLP).

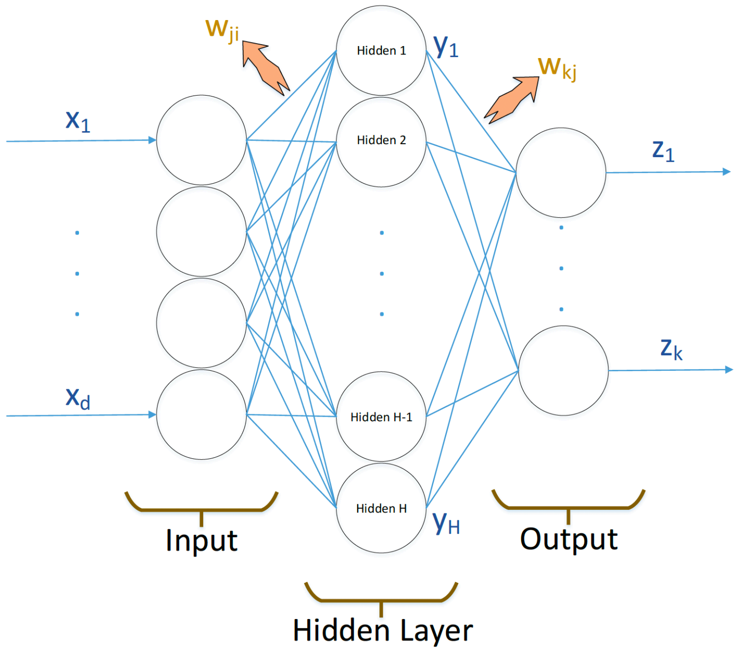


Figure 5: Basic MLP architecture.

2.2.3 Training Strategies in Deep Neural Networks: Backpropagation and Optimization

Deep Neural Networks (DNNs) have become instrumental in a myriad of applications, where the linchpin of their performance is the training process. This training paradigm is fundamentally bifurcated into two phases: Forward Propagation and Backward Propagation [51, 48].

1. *Forward Propagation:* This phase involves the feed-forward mechanism of the neural network, where the network processes the input data to generate predictions. The mathematical formulation can be represented by the equations:

$$n_j = \sum_i x_i w_{ji} \tag{26}$$

$$y_j = f(n_j) \tag{27}$$

Where x_i are the inputs, w_{ji} are the weights, and y_j is the output of hidden layers. Further, extending this to the output layer can be denoted as:

$$z_k = f\left(\sum_j w_{kj} f\left(\sum_i x_i w_{ji}\right)\right) \tag{28}$$

2. *Backward Propagation and Optimization:* The end objective is the minimization of a designated cost function $E(w, b)$, which is inherently non-linear and dependent on the network parameters w and b. This is achieved using optimization techniques like Gradient Descent, expressed mathematically as:

$$\Delta w = -\eta \nabla E(w) = -\eta \frac{\partial E(w, b)}{\partial w} \tag{29}$$

Here, η represents the learning rate, and $\nabla E(w)$ is the gradient of the cost function with respect to w.

The gradients are calculated using the Chain Rule of Calculus, and in the case of a multi-layered network, can be generalized as:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial n_k}\frac{\partial n_k}{\partial w_{kj}} \tag{30}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial n_j}\frac{\partial n_j}{\partial w_{ji}} \tag{31}$$

In practice, numerous variants of the basic Gradient Descent algorithm have been devised to address its limitations, such as susceptibility to local minima and inefficient learning rates [52, 53]. Prominent among these are Stochastic Gradient Descent (SGD) and its momentum-enhanced version, as well as adaptive methods like Adagrad [54], AdaDelta [55], and Adam [56]. These advanced algorithms leverage techniques such as adaptive learning rates and first-order and second-order moments to facilitate more effective and efficient optimization:

1. *Stochastic Gradient Descent (SGD):* This extension of SGD lies in the utilization of a single data point, or mini-batch, to calculate the gradient at each iteration [57]. This introduces a level of stochasticity that allows the algorithm to escape local minima, at the expense of a noisier convergence path. Mathematically, SGD is represented as:

$$w_{t+1} = w_t - \eta \nabla E(w_t; x_i) \tag{32}$$

where x_i is a randomly chosen data point or mini-batch.

2. *Momentum-based Methods:* To alleviate the oscillations caused by SGD, momentum-based variants such as SGD with momentum introduce a velocity component [58]. This allows the algorithm to build up "momentum" in directions with consistent gradients, leading to faster convergence. The update equation becomes:

$$v_{t+1} = \mu v_t + \eta \nabla E(w_t) \tag{33}$$

$$w_{t+1} = w_t - v_{t+1} \tag{34}$$

   where μ is the momentum coefficient.

3. *Adaptive Methods:* Algorithms like Adagrad [54], Adadelta [55], and Adam [56] extend the basic Gradient Descent by incorporating adaptive learning rates for each parameter. Adam, for instance, combines the benefits of both momentum and adaptive learning rates:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla E(w_t) \tag{35}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\nabla E(w_t)^2 \tag{36}$$

$$w_{t+1} = w_t - \frac{\eta m_t}{\sqrt{v_t} + \epsilon} \tag{37}$$

   Here, β_1 and β_2 are hyperparameters that control the decay rates of first and second moment estimates, respectively.

Each of these variants contributes unique attributes that make them more suited for specific types of optimization landscapes. In particular, momentum and adaptive methods often outperform basic Gradient Descent and SGD in non-convex, high-dimensional optimization problems commonly encountered in Deep Learning [59]. Empirically, Adam has shown superior performance across a wide range of ANN architectures and their applications. Its ability to efficiently adapt makes it particularly useful for scenarios requiring optimization on large-scale training datasets.

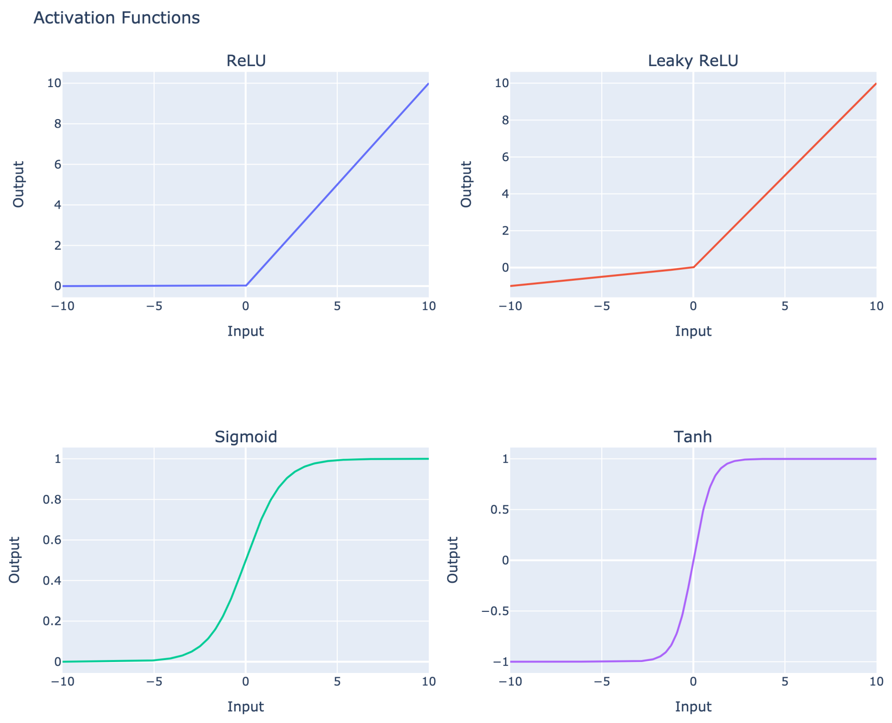


Figure 6: The fundamental activation functions for DNNs

2.2.4 Activation Functions

Activation functions are a basic component DNNs and play an important role by introducing non-linearity, enabling the model to learn from the error and make adjustments. Figure 6 depicts the fundamental activation functions commonly employed in DL models.

1. *Rectified Linear Unit (ReLU):* ReLU is among the most widely used activation functions due to its computational efficiency and capability to combat the vanishing gradient problem [60]. Mathematically, it is defined as:

$$f(x) = \max(0, x) \tag{38}$$

   While effective, it suffers where neurons become inactive and no longer update during training if their output is zero.

2. *Leaky Rectified Linear Unit (Leaky ReLU):* To address the limitations of ReLU, Leaky ReLU was introduced [61]. It allows a small, non-zero gradient when the input is less than zero.

$$f(x) = \max(\alpha x, x) \tag{39}$$

where α is a small constant.

3. *Sigmoid Function:* The sigmoid activation function has historical significance but is less frequently used in contemporary models due to the vanishing gradient problem [62]. However, this function is used in the end of a DNN in order to map the output vector to probabilities. It is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{40}$$

4. *Tanh Function:* Similar to the sigmoid but rescaled to range between -1 and 1, the tanh function offers steeper gradients, facilitating backpropagation.

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{41}$$

5. *Soft-max Function:* Soft-max is widely used in multi-class classification problems. It takes a vector of real numbers and transforms it into a probability distribution.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{42}$$

where **z** is the input vector and K is the number of classes.

6. *Swish Function:* Introduced as a self-gated activation function, Swish offers the benefits of ReLU and sigmoid and mitigates their individual limitations [63].

$$f(x) = x \cdot \sigma(x) \tag{43}$$

Swish has been shown to outperform ReLU in deeper architectures due to its smoother gradient, which helps in backpropagation.

The choice of an activation function can dramatically influence the performance and training dynamics of a neural network. ReLU and its variants are often preferable for tasks where computational efficiency is crucial, while sigmoid and tanh may find specific applications in architectures requiring bounded activations [48].

2.2.5 Loss Function in Neural Networks

Loss function is as a quantitative measure for evaluating the deviation of predicted values from the actual ground truth during the training stage of a DNN. Network's optimization process strongly depends on this function and as a consequence it influences the model's overall accuracy. The choice of this function is equivalent to the task the model is designed to perform. More precisely, these are the most common loss functions for classification tasks:

- `Binary Cross-Entropy` quantifies the difference between two probability distributions:

$$\mathrm{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{44}$$

- `Categorical Cross-Entropy` can be used when the classification task is not binary:

$$\mathrm{CCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{45}$$

On the other hand, the most common Loss Functions for Regression tasks are:

- `Mean Squared Error Loss` computes the average squared difference between the predicted and ground-truth output values:

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{46}$$

- `Mean Absolute Error` calculates the average absolute difference between the predicted and actual values:

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{47}$$

2.2.6 Overfitting and Regularization Techniques

Overfitting occurs when a ML model a model performs well on the training data but poorly on unseen or test data. This phenomenon typically indicates a failure in the model's generalization capability, leading to higher variance and lower bias. There are certain methodologies applied to avoid overfitting during model development and training.

Dropout Dropout is an effective and widely used regularization technique. Dropout operates by stochastically setting a fraction p of the neurons to zero in each training iteration for a given layer. As a result, Dropout enforces the remaining neurons to adapt more robustly to the input, reducing the model's sensitivity to overfitting.

L1 and L2 Regularization L1 and L2 are regularization techniques that add a penalty term to the loss function. For a given weight matrix **W**, the L1 and L2 penalties are given by:

$$\text{L1 Penalty} = \lambda \sum_i |w_i| \tag{48}$$

$$\text{L2 Penalty} = \lambda \sum_i w_i^2 \tag{49}$$

where λ is the regularization coefficient. L1 regularization tends to produce sparse weight matrices, while L2 forcing weights to be small but not zero.

Early Stopping Early Stopping involves monitoring the validation loss or metric during training. The training process is terminated once the selected validation loss or metric stops improving, typically after a certain number of epochs. The latter number is called *patience*.

Data Augmentation Data Augmentation expands the training dataset through modality-based transformations. This increases the diversity of the training data and allows the model to generalize better to unseen data. For instance, rotation, scaling, or cropping are typical transformations applied to image data.

Normalization Methods like Batch Normalization are also employed to avoid overfitting. These techniques normalize the inputs of a layer, making the model more stable and faster.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a specialized architecture within the Deep Neural Networks (DNNs) designed primarily for image recognition and classification tasks [64]. Analogous to how ANNs emulate a network of biological neurons in their computational process, CNNs draw inspiration from the functioning of the human visual cortex in information acquisition and subsequent recognition [65]. Consequently, CNN architectures can be broadly partitioned into two main components. The first component focuses on feature extraction and learning in images through the convolution operation employing specialized filters. Following this, the spatial information gleaned is passed to the second structural component, which consists of a set of fully connected neurons responsible for the final image classification task.

2.3.1 Convolution operation and basic hyperparameters

The convolution operation represents a specialized application of spatial filters on images. In this operation, a filter or mask is defined and then moved across the image. The sum of the products between this filter and the pixel values for each localized region is computed [66], as illustrated in Figure 7

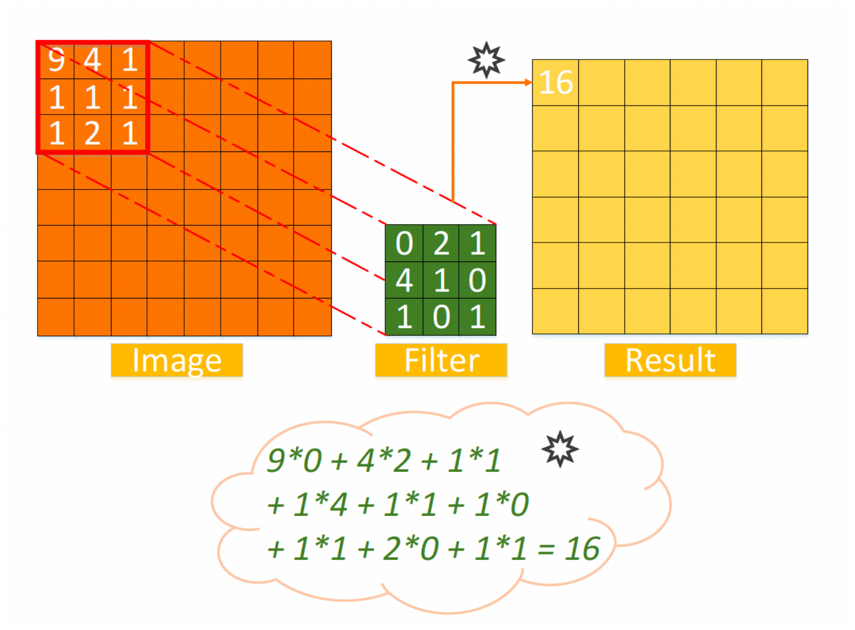


Figure 7: Convolution operation

It is universally acknowledged that a majority of real-world problems demand images with a greater number of channels than a singular unit. For this reason, the filter is configured such that its depth matches the corresponding depth of the image. In this manner, convolution is executed independently for each dimension, and the final outcomes are subsequently aggregated as depicted in Figure 8.

Understanding and defining the hyperparameters of a Convolutional Neural Network (CNN) are critically pivotal for their effective construction and optimization. Initially, the *number of filters* in a convolutional layer holds considerable importance, primarily because an increase in the filter count directly corresponds to an elevation in the neuron count. This, in turn, enhances the network's capacity for detecting more intricate patterns. Furthermore, the weights of these filters, as determined during the training phase, signify the salience of the features they capture in the output image. For this

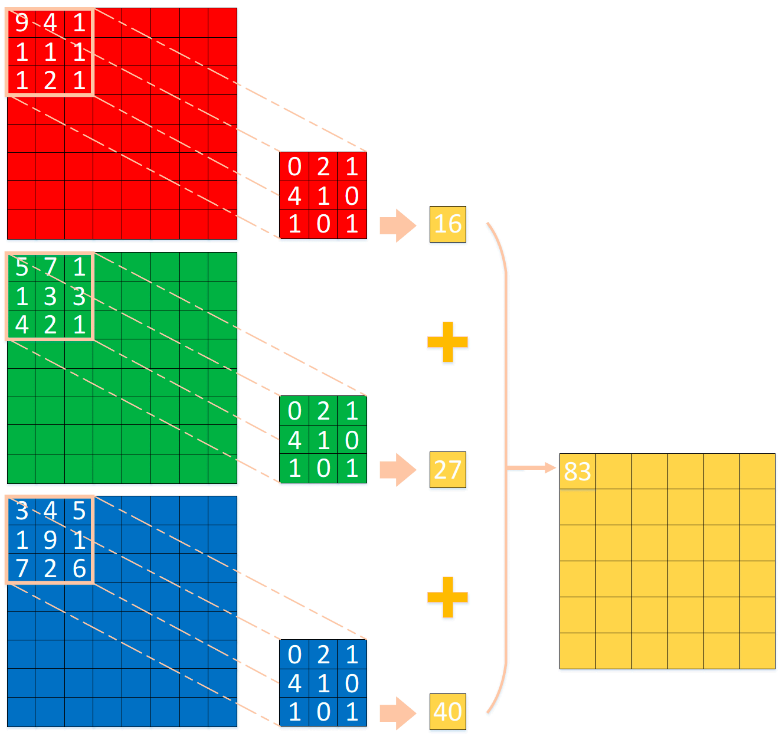


Figure 8: Convolution operation in an RGB image.

reason, the *dimensions* of the filters are of paramount importance. Commonly, these dimensions are square, typically 3x3 or 5x5, and in rarer instances, 7x7. In a generalized context, smaller-sized filters are adept at capturing more localized information, thereby detailing finer aspects of the image. In contrast, filters with larger dimensions offer a more generalized, albeit less detailed, view of the image.

Two additional fundamental parameters in the convolution operation within CNNs are *Padding* and *Stride*. Padding refers to the practice of augmenting the image perimeter with zero-values. This is generally employed to prevent the drastic reduction of output dimensions following the convolution operation. Additionally, padding ensures that the center of the filter traverses the edge pixels of the image, thereby enabling the capture of complete spatial features [67]. On the other hand, Stride represents the step size or the distance between two successive positions of the filter as it moves across the image. Stride is instrumental for the effective subsampling of features, adjusting the granularity of the feature mapping. The Stride parameter is critical in determining the spatial dimensions of the output, given the dimensions of the input image and the applied filter. Consequently, if the image has dimensions $d_i \times d_i$ and the filter has dimensions $d_f \times d_f$, the resulting convolutional output will have dimensions:

$$\frac{d_i - d_f + 2 \times padding}{stride} + 1 \tag{50}$$

The strategic incorporation of padding and stride parameters serves as a pivotal lever for controlling the spatial dimensions of convolutional outputs, thereby profoundly affecting the feature representation and computational efficiency of a CNN. Simultaneously, the selection of other hyperparameters such as the number of filters count dimensions fundamentally determines the CNN's capability to generalize and adapt to increasingly complex pattern recognition tasks. Given the high-stakes impact of these hyperparameters, leveraging advanced optimization techniques like grid search or Bayesian optimization becomes imperative for empirically determining the most efficacious settings. This confluence of thoughtful hyperparameter tuning is indispensable for constructing robust and efficient CNN architectures, positioning them as viable solutions for a diverse array of machine learning challenges.

2.3.2 Spatial pooling operations

In pursuit of effective sampling while simultaneously reducing the size of the Convolution Output, often referred to as the Feature Map, the technique of Spatial Pooling becomes imperative. This entails a moving window that traverses the Feature Map in both dimensions—length and width—to selectively extract specific spatial information, which varies according to the type of pooling employed. An example of Spatial Pooling operations is presented in Figure 9. The most popular methods are:

- *Max Pooling* filter slides across the Feature Map and selects the maximum element in each step while the remaining information is eliminated. This method provides a form of invariance to minor changes, captures the most salient features and extracts a downscaled feature map. The max pooling layer also involves hyperparameters, in this case the dimensions and stride of the pooling window. The dimensions of the extracted feature map can be determined by setting $stride = 0$ in Eq. 50.

- *Average Pooling* calculates the mean of all elements in the selected area, which tends to smooth out the Feature Map and offers a generalized representation.

- *Sum Pooling* computes the sum of all elements in the chosen window, offering yet another form of aggregating spatial information, although it is less commonly employed compared to Max and Average Pooling.

- *Global Average Pooling* performs average pooling over the entire feature map, reducing it to a single value per channel. This operation captures the global context of the feature map, providing a highly compressed representation that is often used before the classification layer in various architectures.

- *Global Max Pooling* operates similarly but selects the maximum value from the entire feature map for each channel. Like Global Average Pooling, this offers a global context but focuses on capturing the most salient features across the entire spatial extent.

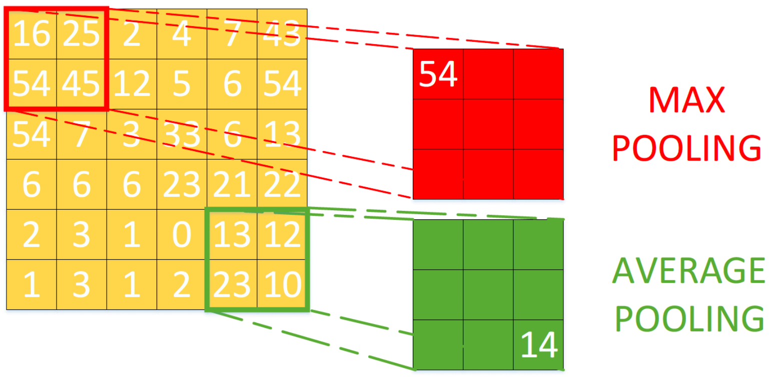


Figure 9: Spatial pooling operations example: Max and Average pooling.

2.3.3 Convolutional block and CNN Architecture

The architecture of a CNN intrinsically involves Convolutional Layers as its basic structural component. A typical Convolutional Layer is comprised of multiple Feature Maps to facilitate effective feature learning from the input image. An example of a Convolutional block is presented in Fig 10. The basic CNN components are:

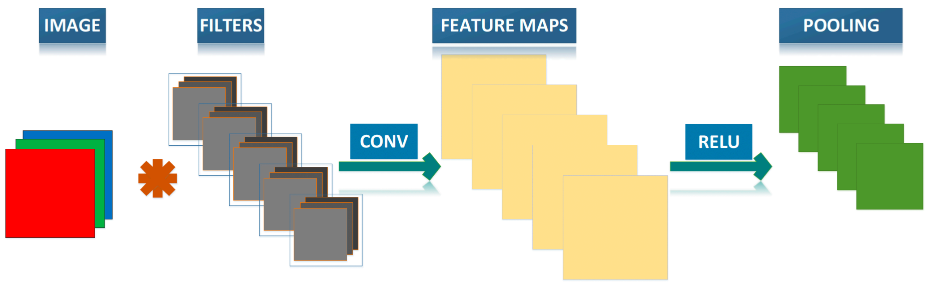


Figure 10: Convolutional layer example.

- *Filters and Kernels:* These are initialized based on certain pre-defined patterns and adapt during the training phase via backpropagation [48].

$$W' = W - \alpha \frac{\partial L}{\partial W} \tag{51}$$

- *Non-linearity:* Activation functions are applied post-convolution to introduce non-linearity into the model. The ReLU activation function is commonly employed [60].
- *Pooling:* Following activation, pooling techniques are often used to reduce the dimensionality of the Feature Maps, enhancing the network's ability to generalize.
- *Flattening and Fully Connected Layers:* As the CNN architecture deepens, the Feature Maps are flattened to a one-dimensional array, which is then fed into Fully Connected Layers for final feature extraction and classification [68].

A simple example of the CNN architecture, which contains all the basic components mentioned above, is illustrated in Figure 11

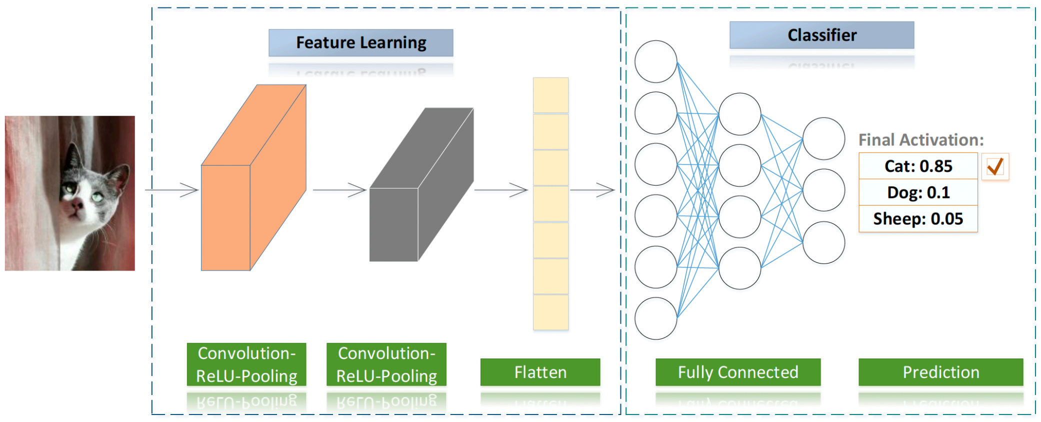


Figure 11: A simple example of CNN architecture.

2.3.4 Established Convolutional architectures

Yann LeCun's development of LeNet in 1998 marked a significant milestone in the domains of DL and computer vision[69]. As one of the CNNs successfully applied to digital image processing, particularly in digit recognition, LeNet paved the way for the widespread adoption of CNNs in various domains. In addition, LeNet delineated the basic CNN structure and thus configurations and variations implemented to develop a series of established architectures.

AlexNet, developed by Krizhevsky et al. in 2012, emerged as a groundbreaking architecture in the field of DL [68]. The remarkable performance of this model in the ImageNet Large Scale Visual Recognition Challenge demonstrated the superiority of convolutional models over traditional image processing approaches and as a result revitalized the research in Neural Networks. AlexNet's architecture introduced several innovative concepts that were critical in its performance. These include five convolutional layers with varying filter sizes and depths, the introduction of the ReLU activation function that was important for the reduction of the training time, Local Response Normalization (LRN) which enhanced model generalization, three Fully Connected Layers, a dropout rate of 50% to mitigate overfitting and the final soft-max layer for classification into 1000 distinct classes. A diagrammatic representation of the architecture can be seen in Figure 12.

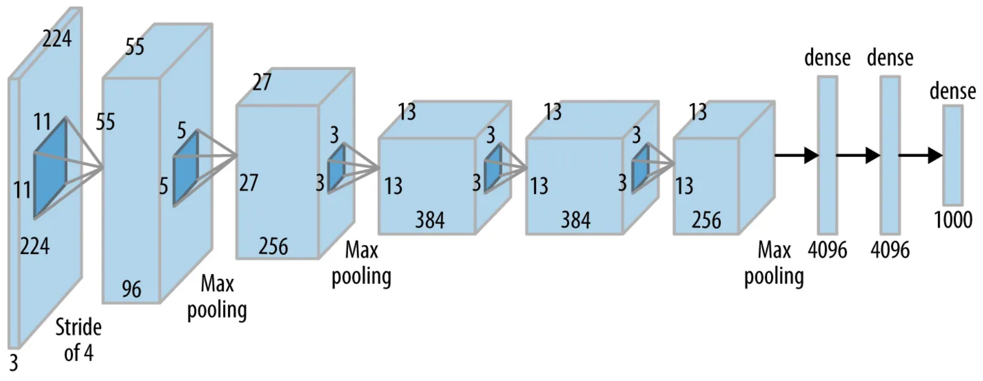


Figure 12: The architecture of AlexNet.

The VGG (Visual Geometry Group) models, developed by Simonyan and Zisserman from the University of Oxford, represent a significant milestone in the evolution of CNN architectures. Introduced in 2014, these models are renowned for their deep yet systematically structured layers, which have set a new benchmark in image classification tasks, particularly in the ImageNet challenge. The VGG family primarily includes two models: VGG-16 and VGG-19, differentiated by their depth.

Both VGG-16 and VGG-19 employ layers of 3×3 convolutional filters with a stride of 1, uniformly throughout the network. This consistency in filter size simplifies the network design and improves feature learning capabilities. VGG-16 comprises 16 layers, while VGG-19 extends to 19 layers. These models showcased that increased depth, with small and consistent filter sizes, can effectively enhance the network's learning capacity. Each model concludes with three fully connected layers, where the first two consist of 4096 neurons each, and the third performs the final classification. Similar to AlexNet, VGG models employ ReLU activation functions after each convolutional layer to introduce non-linearity and accelerate training. VGG's primary innovation was the application of the Batch Normalization technique prior to layer input, enhancing model stability. In addition, they established design principles for deep architectures and influenced the development of subsequent CNN models, including those designed for tasks other than image classification. In many applications, pretrained VGG models can be used for efficient feature extraction. A diagrammatic representation of VGG-16 and VGG-19 can be illustrated in Figure 13.

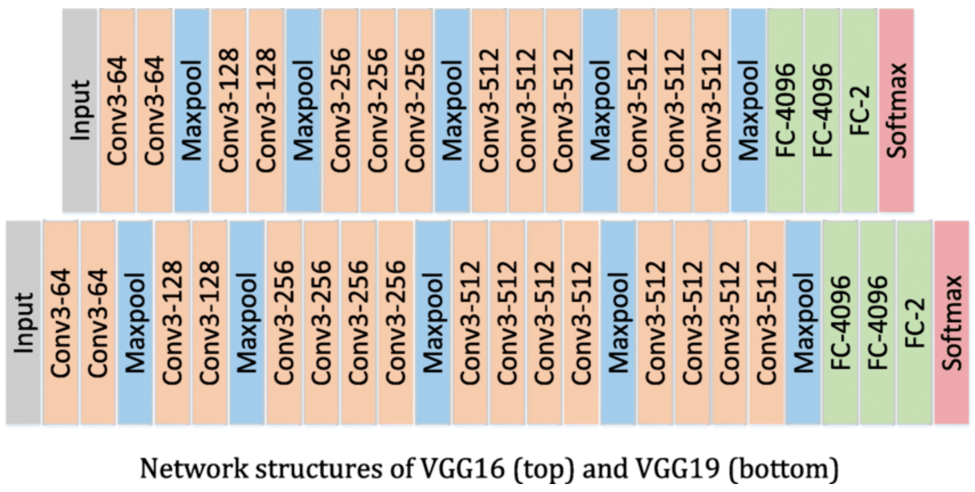


Figure 13: The architecture of VGG-16 (top) and VGG-19 (bottom). Adopted from [1].

ResNet, introduced by He et al. in 2015, represented a significant leap forward in the field of DL, particularly in addressing the challenges associated with training very deep models [70]. This architecture introduced residual learning technique and enabled the development of networks that were substantially deeper than previous architectures. Residual blocks incorporated shortcut connections that skip one or more layers. These connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. The primary advantage of this design is that it allows the network to learn residual functions with reference to the layer inputs, thereby facilitating the training of deeper networks. ResNet models come in various sizes, the most common being ResNet-50, ResNet-101, and ResNet-152, referring to the number of layers in each. ResNet achieved remarkable success, most notably winning the ImageNet Large Scale Visual Recognition Challenge in 2015. The introduction of residual learning not only solved the vanishing gradient problem but also set a new standard for the depth of networks, influencing a wide array of subsequent CNN models. A diagrammatic representation of the ResNet architecture, showcasing the skip connections, can be found in Figure 14.

Inception Net, initially known as GoogLeNet, introduced by Szegedy et al. in 2014, marked a significant advancement in the field of deep CNNs [71]. The primary goal of

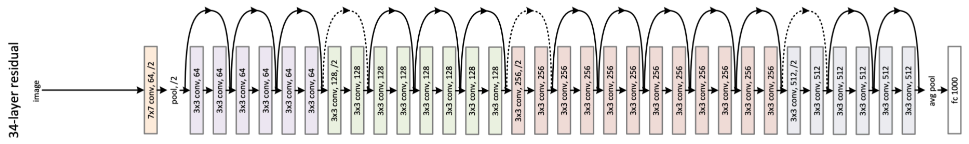


Figure 14: The architecture of ResNet.

Inception Net was to optimize the utilization of computing resources within the network, allowing for more efficient and deeper architectures. Model consists of several Inception modules, a novel design that allowed for parallel processing through multiple filter sizes within the same layer. This design not only improved the network's ability to capture information at various scales but also significantly reduced computational cost due to the fact that the usage of 1x1 convolutions for dimensionality reduction enhanced computational efficiency. Beside that improvements, Auxiliary Classifiers was introduced in intermediate layers to combat the vanishing gradient problem. The result of them was contributing to the final loss function. Over time, the Inception architecture evolved through multiple iterations, from Inception V1 to the more sophisticated Inception V4 and Inception-ResNet combinations. An illustration of the Inception module and the evolution of Inception Net versions is provided in Figures 15 and 16.

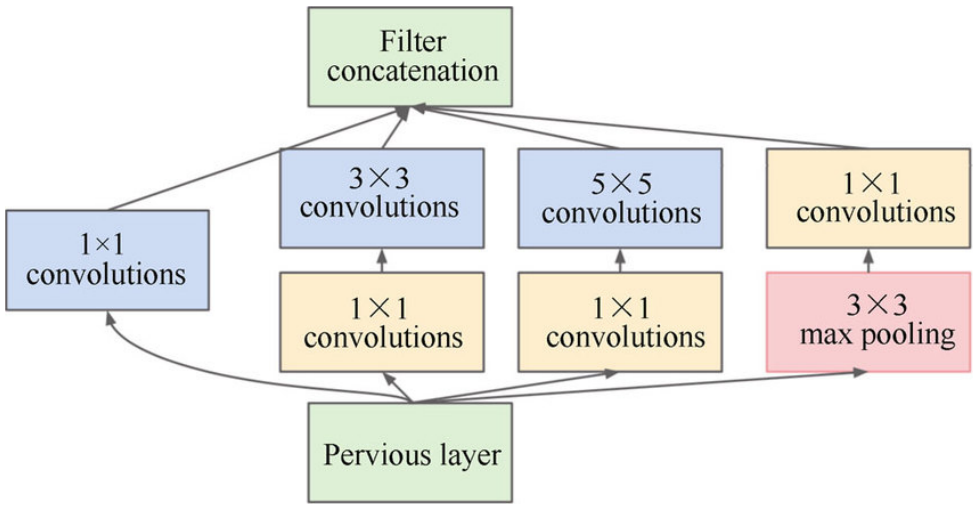
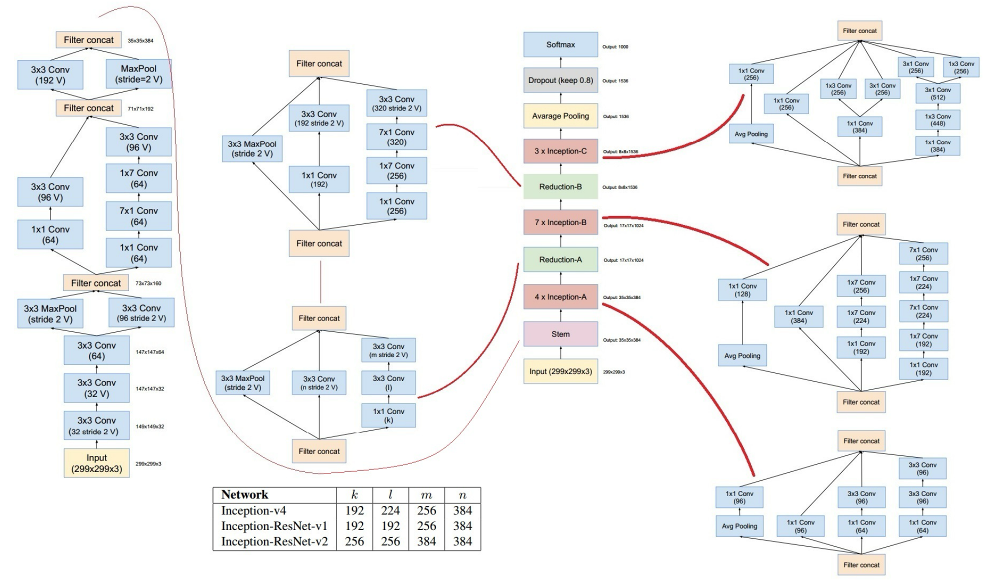


Figure 15: An Inception module in the Inception network.

The first component of Inspection v4 is the Stem Block. The input stem of Inception v4 designed to reduce the grid size while maintaining a rich feature representation. It consists of a series of convolutions, max-pooling, and average pooling layers. This idea paved the way for the development of more accurate object detection models too. The core of Inception v4 consists of several updated Inception modules, which include a



Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

Figure 16: Inception v4 architecture

variety of convolutions with different filter sizes to capture diverse feature representations. These modules are categorized into Inception-A, Inception-B, and Inception-C blocks, each with a unique arrangement of convolutional layers. Transitioning between different Inception blocks, reduction blocks are used to reduce the grid size of the feature maps. These blocks apply a combination of convolutional and pooling layers to efficiently downsample the feature maps. Inception v4 integrates residual connections to facilitate training of deeper networks by addressing the vanishing gradient problem.

MobileNet, introduced by Howard et al., is a series of CNN architectures specifically designed for mobile and edge devices, emphasizing efficiency and compactness [72]. The central innovation of MobileNet lies in its ability to maintain a balance between accuracy and computational cost, making it suitable for applications where resources are limited. At the core of MobileNet's architecture are depthwise separable convolutions. This technique divides a standard convolution into two parts: a depthwise convolution and a pointwise convolution. This split significantly reduces the computational load and model size without a substantial decrease in performance. Such operations are depicted in Figure 17. MobileNet has evolved through various versions, i.e., MobileNetV2, and MobileNetV3, each introducing improvements in efficiency and performance. MobileNetV3 introduces the use of Squeeze-and-Excitation (SE) blocks, a form of lightweight attention mechanism that enhances the representational power of the network. Another significant innovation is the use of platform-aware Neural Architecture Search (NAS), optimizing the network's structure for different hardware constraints. SE block architecture is presented in Figure 18.

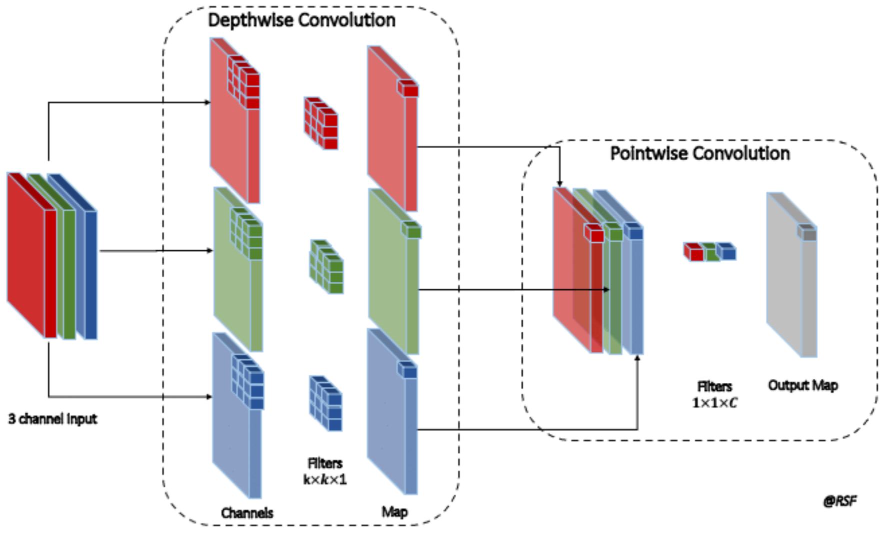


Figure 17: Depthwise separable convolution in MobileNet [2].

Each of these architectures not only advanced the technical capabilities of DNNs but also broadened their applicability across a spectrum of real-world scenarios. The continuous evolution of CNNs illustrates the dynamic nature of this field, wherein each innovation builds upon the last, pushing the boundaries of what is possible in ML. The exploration of these architectures sets the stage for further research and development, encouraging the pursuit of more optimized, effective, and innovative solutions in DL domain.

2.4 Sequential models and Attention mechanism

Sequential models are important for analyzing and interpreting data that is inherently ordered, such as time series, speech, or textual data. These models, capable of capturing temporal structure, have been fundamental in numerous applications across various domains. The advent of attention mechanisms has further revolutionized this field, enhancing the capability of sequential models to focus on relevant parts of the input sequence, thereby improving their performance, especially in complex tasks. This section focuses on the principles of Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and Long Short-Term Memory (LSTM) Networks, each representing a fundamental architecture in the development of sequential models. Additionally, the section unfolds the concept and impact of the attention mechanism, a breakthrough that has significantly influenced the implementation of DL models.

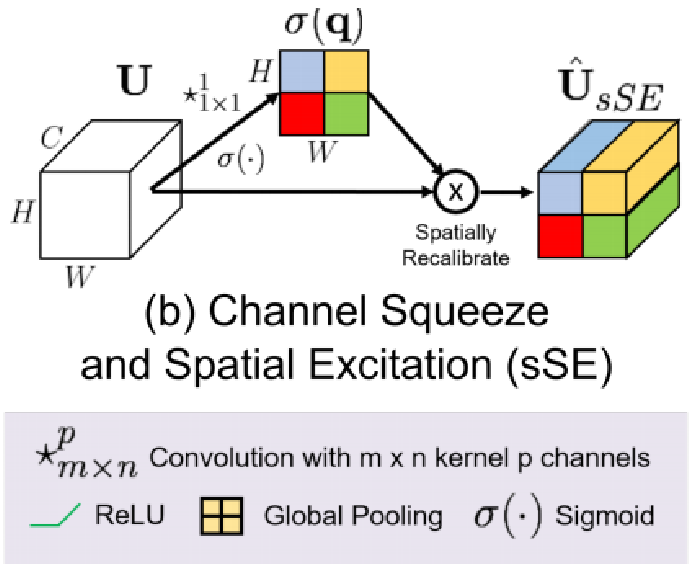


Figure 18: Squeeze-and-Excitation (SE) block structure. Adopted from [3].

2.4.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) designed to process sequences of inputs by incorporating memory elements to capture temporal dynamics in data, a feature absent in traditional feedforward networks [73]. At their core, RNNs consist of a network of neuron-like units, each passing a sequence of inputs through a looped network architecture. This design enables them to retain information over time. Mathematically, an RNN can be described as:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t), \tag{52}$$

where h_t is the hidden state at time t, x_t is the input at time t, and W are the weights. A simple example of RNN layer is presented in Figure 19

In RNNs, forward propagation involves processing sequences where the output from previous steps is fed into the current step. This process effectively captures temporal dependencies in the data. Each time step's output is determined by both the current input and the previously hidden state. The loss in RNNs is computed at each time step, reflecting the prediction error for that particular instance in the sequence. For tasks like sequence generation, the loss is often accumulated over all time steps to gauge the model's overall performance on the sequence. Backpropagation Through Time involves unrolling the RNN through time and applying the chain rule to compute gradients. However, it often encounters challenges like vanishing or exploding gradients due to the repeated multiplication of gradients through time.

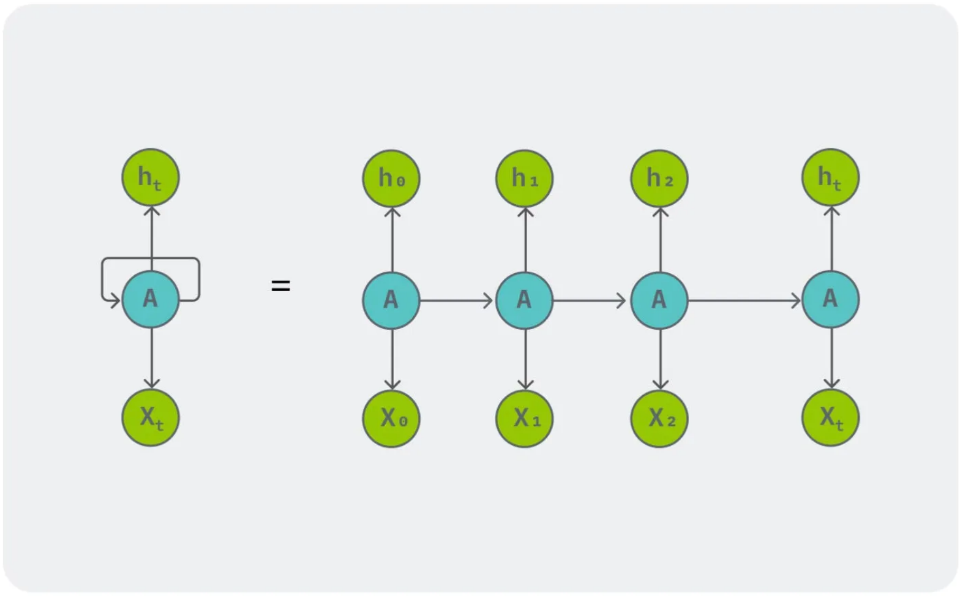


Figure 19: A simple example of RNN layer. Adopted from this link.

RNNs can be categorized based on their input-output structure: One-to-Many for scenarios like image captioning (single input, sequence output), Many-to-One for tasks like sentiment analysis (sequence input, single output), and Many-to-Many for applications like machine translation (sequence input, sequence output). RNNs have evolved into various forms to overcome challenges like vanishing gradients and to enhance performance. Bidirectional RNNs process data in both forward and backward directions, providing a richer context. In addition deep RNN architectures, with multiple hidden layers, capture more complex features. An example of a Bidirectional layer is presented in Figure 20.

2.4.2 Gated Recurrent Units

Gated Recurrent Units (GRUs) are an advanced variant of the basic RNN architecture, introduced to mitigate the vanishing gradient problem existed in traditional RNNs. They achieve this by employing a gating mechanism that regulates the flow of information. GRUs utilize two mechanisms, known as update and reset gates. The update gate's role is to determine the extent to which information from previous time steps should be carried forward. Conversely, the reset gate is responsible for deciding the proportion of past information that should be disregarded or forgotten. These gates are crucial in handling large amount of information contain in large sequences.

The operations of a GRU can be represented by the following equations:

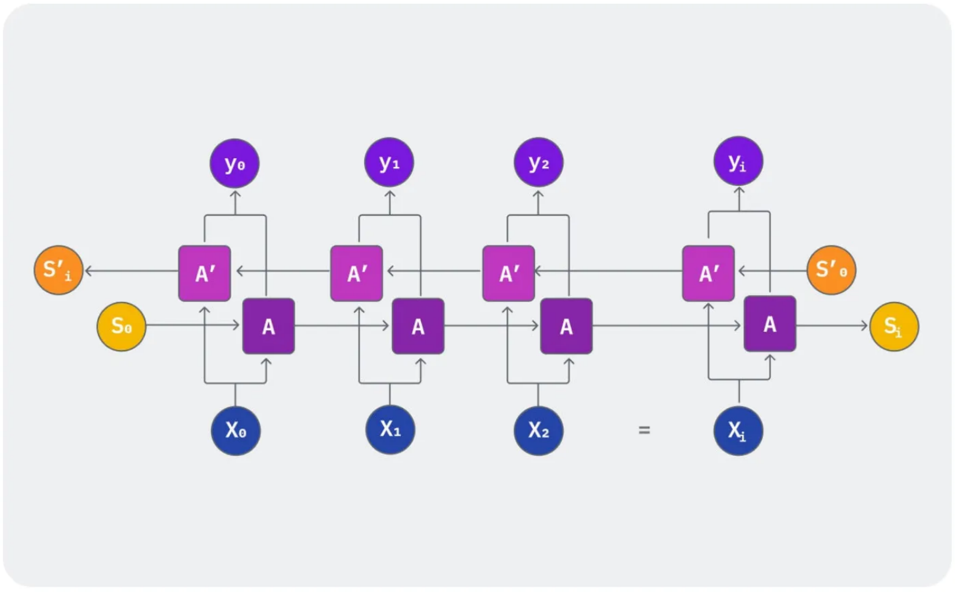


Figure 20: A simple example of Bidirectional RNN layer. Adopted from this link.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{53}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \tag{54}$$

$$\tilde{h}_t = \text{tanh}(W \cdot [r_t * h_{t-1}, x_t]), \tag{55}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \tag{56}$$

where z_t and r_t are the update and reset gates respectively, $\tilde{h}_t$ is the candidate hidden state, h_t is the final hidden state, σ represents the sigmoid function, and W denotes the weight matrices.

2.4.3 Long Short-Term Memory Networks

Long Short-Term Memory Networks (LSTMs) are an advanced type of RNNs specifically designed to overcome the limitations of traditional RNNs, particularly the vanishing gradient problem and to enhance the effectiveness of GRUs [74]. LSTMs comprising different gates: the input gate, the forget gate, and the output gate, alongside a cell state. The later stores relevant information during the sequence passing and gates control the flow of information into and out of the cell state. Thus, network handles information in a dynamic manner and can select of discard parts of sequences.

The operation mentioned above can be described as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{57}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{58}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{59}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{60}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{61}$$

$$h_t = o_t * \tanh(C_t), \tag{62}$$

where:

- Forget gate: f_t
- Input gate: i_t
- Cell state update: $\tilde{C}_t$
- Final cell state: C_t
- Output gate: o_t
- Final hidden state: h_t
- Sigmoid function: σ
- Weights and biases:W and b

A visual representation of both LSTM and GRU cells is presented in Figure 21. LSTMs are particularly adept at capturing long-range dependencies in data, a capability that is crucial for tasks involving complex sequences. They effectively address both vanishing and exploding gradients, which enhances their training efficiency and model performance. LSTMs have a wide range of applications, including but not limited to, language modeling and translation in NLP, speech recognition and synthesis, anomaly detection in time series data, image captioning and video data processing.

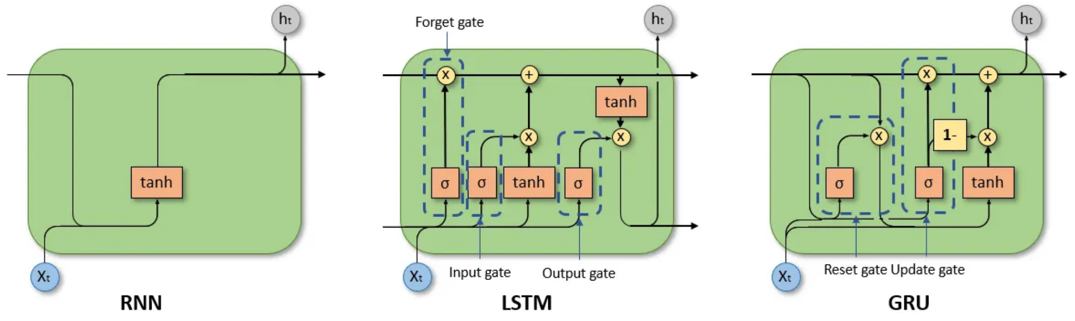


Figure 21: Visual representation of reccurent, GRU and LSTM units. Adopted from this link.

2.4.4 Attention mechanism

Attention mechanisms have emerged as a groundbreaking concept in the realm of neural networks, particularly revolutionizing sequence modeling tasks [75]. They were developed as a response to the limitations of traditional models like RNNs and LSTMs in processing long sequences. At its core, attention is a technique that enables models to focus selectively on parts of the input sequence that are more relevant to desired task.

In practice, attention mechanisms are integrated into neural network architectures to dynamically weigh the significance of different inputs. They have been notably successful when combined with models like RNNs, LSTMs, and particularly Transformers, where attention serves as the backbone of the architecture.

The essence of attention can be captured in the equation:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{63}$$

, where Q, K, and V represent queries, keys, and values, respectively, and d_k is the dimension of the keys. In this thesis, a type of type of self-attention used to handle sequential data. Given an input sequence $\mathbf{h} \in \mathbb{R}^{n \times d}$, where n is the number of tokens in the sequence and d is the dimensionality of each token:

1. *First Linear Transformation*: The first operation in the self-attention mechanism is a linear transformation of the input $\mathbf{h}$ using weight matrix $\mathbf{W}_{s1} \in \mathbb{R}^{d \times da}$.

$$\mathbf{h}' = \mathbf{h}\mathbf{W}_{s1}$$

This results in a transformed input $\mathbf{h}' \in \mathbb{R}^{n \times da}$.

2. *Activation Function*: The transformed input $\mathbf{h}'$ is then passed through the tanh activation function element-wise:

$$\mathbf{a} = \tanh(\mathbf{h}')$$

3. *Second Linear Transformation*: Next, the activated sequence $\mathbf{a}$ undergoes another linear transformation using weight matrix $\mathbf{W}_{s2} \in \mathbb{R}^{da \times r}$:

$$\mathbf{a}' = \mathbf{a}\mathbf{W}_{s2}$$

   This produces $\mathbf{a}' \in \mathbb{R}^{n \times r}$, where r is typically the number of desired attention heads or contexts.

4. *Softmax Activation*: The Softmax function is applied along the first dimension (axis=1) to normalize the weights of the sequence, ensuring they sum up to 1:

$$\mathbf{A} = \text{softmax}(\mathbf{a}')$$

   Here, $\mathbf{A} \in \mathbb{R}^{n \times r}$ represents the attention scores for each token in the sequence.

The applications of attention mechanisms are diverse, including natural language processing tasks like translation and summarization, speech recognition and image captioning. In addition, this method is a core component of advanced multimodal and Large Language models.

2.5 Transfer Learning

Transfer Learning is a powerful technique in ML domain, where a model developed for one task is reused as the starting point for a model on a second task [76]. This approach is particularly beneficial in scenarios where labeled data is scarce or when training a large model from scratch is computationally intensive.

Transfer learning typically involves using pre-trained models that have been trained on large datasets. These models can either be used as they are (feature extraction) or can be fine-tuned on a new task with some additional layers being trained. Transfer learning has found extensive applications in various domains, notably in NLP for tasks like sentiment analysis and language translation, in computer vision for image classification and object detection, and even in areas like medical image and audio data analysis where labeled data is limited.

3 Methodology

DL technologies have undergone significant advancements, offering effective solutions across diverse domains, including audio data processing, medical applications, and natural language understanding [77]. The rapid progress of DL models is attributed to their capacity to acquire hierarchical features from data, leading to state-of-the-art performance on several tasks.

The first step of the proposed method delves into the development of a SVM model for recognising emotions in children's music, specifically focusing on arousal and valence. Both classification tasks employed time and frequency domain handcrafted audio features. These models provided a foundational comprehension of the complexities inherent in such tasks, serving as a strong baseline for the subsequent DL algorithms.

Transitioning from SVMs, the following experiments focus on CNNs combined with Transfer Learning techniques for feature extraction and subsequent classification tasks [78]. Among the range of audio data representation techniques, Log - Mel spectrograms emerged as the most informative representation of audio signals, offering an efficient alternative feature set suitable for CNN input [79]. Particular emphasis was given to varying layer freezing configurations, investigating the impact of different transfer learning strategies on model efficiency and generalisation capabilities [80].

Further exploration of DL applications involved the assessment of LSTM networks [81], augmented by an attention mechanism, for handling music transcription sequences. This model was fused with the previously discussed CNN within a Dual - Stream architecture to evaluate the contribution of music transcription information to the emotion classification task [82]. In summary, this work aims to evaluate the efficiency, advantages, and limitations of each model, supported by a series of experimental configurations and evaluations.

3.1 Support Vector Machines for Music Emotion Recognition

The decision to utilise Support Vector Machines (SVM) for audio classification in this thesis was predicated on its efficiency to handle non - linear dependencies among features, particularly within audio analysis pipelines. The primary objective encompasses the classification of emotional expressions in children's music. The experimental framework, illustrated in Figure 22, provides a methodical approach to these objectives, thereby enabling a discernible baseline for potential deep learning methodologies in future investigations.

The feature extraction phase played an important role in enhancing the efficiency of the Support Vector Machine (SVM) classifier. In this context, handcrafted audio features were extracted from mono-WAV files by leveraging the functionality provided by the *pyAudioAnalysis* library [4]. The selection of these features was supported by

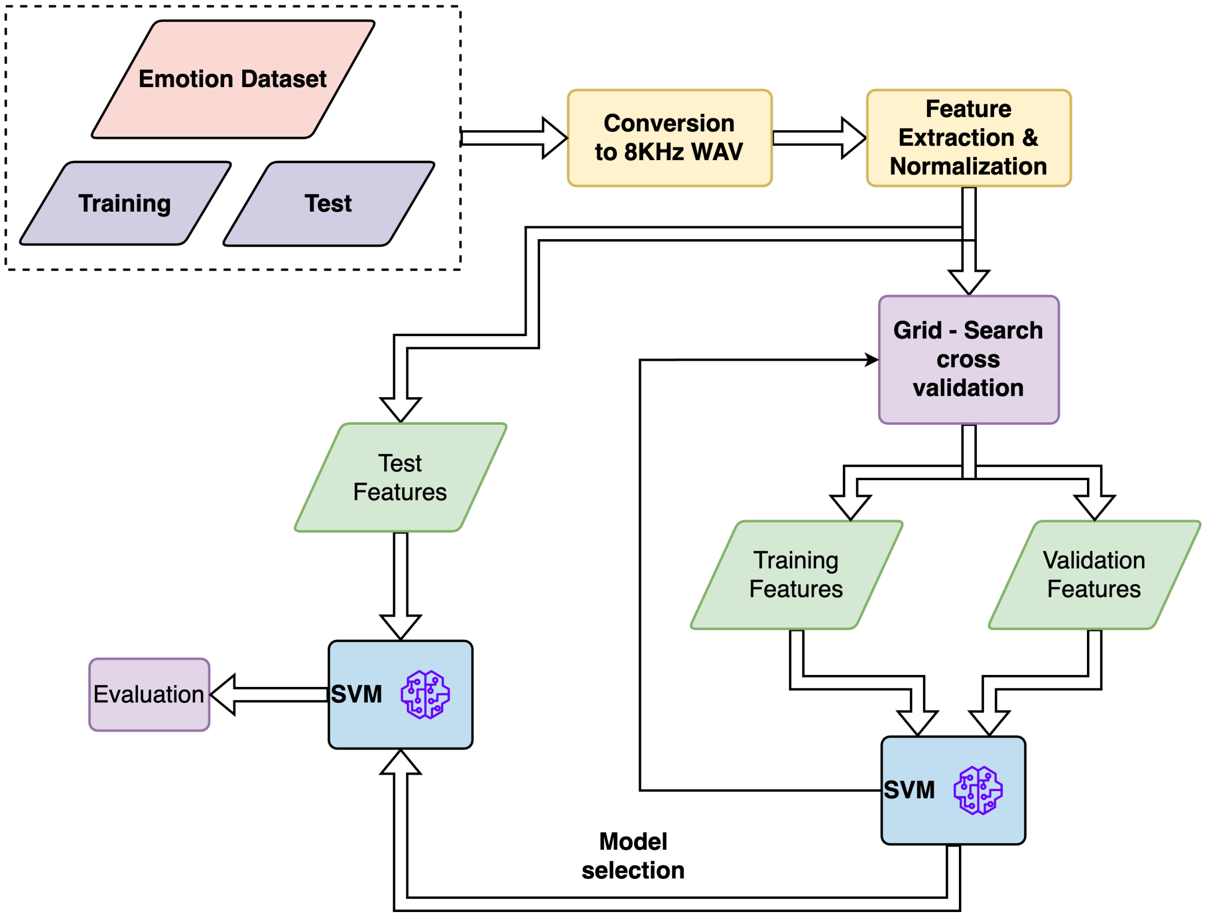


Figure 22: Schematic representation of the SVM model development.

their capability to capture both spectral and temporal characteristics of audio samples. Among these features are the Mel-Frequency Cepstral Coefficients (MFCCs), Chroma Vector, and an array of Spectral features. A comprehensive breakdown of them is presented in Table 1. Extracted features were standardized before the training step to enhance computational efficiency. Such transformation was implemented by employing the following formula:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma} \tag{64}$$

In this equation, X represents the original feature vector, μ signifies the mean, and σ denotes the standard deviation of the feature vector. This standardization process was exclusively applied to the training subset of the dataset, whereby μ and σ were computed and subsequently employed for normalizing the validation and test subsets.

The SVM model was trained using a Radial Basis Function (RBF) kernel. This ker-

Table 1: Comprehensive list of audio features extracted using the pyAudioAnalysis library [4].

Index	*Name*	*Description*
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9–21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
22–33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

nel trick has been widely used in audio classification tasks [83], primarily due to its capability in nonlinear mapping of feature spaces, which ensures the recognition of informative discrimination patterns. The hyperparameters of the model, including the regularisation parameter C and the kernel parameter γ, were optimised using a grid search approach and a validation set. The selected model was re-trained with the most optimal hyperparameters and was used to predict the classes of unknown music samples.

The previously discussed workflow was implemented using the functionality provided by *pyAudioAnalysis* library. In order to provide a comprehensive perspective, we've detailed the specific implementation in the code snippet below. The code outlines two primary phases. The first phase entails the extraction of features from the audio data, model development, and tuning, while in the subsequent phase, the model was used for predictions and evaluation on the test portion of the dataset.

```

```

```
from pyAudioAnalysis import audioTrainTest as aT
...

aT.extract_features_and_train(train_dir_list), 1.0, 1.0, aT.
shortTermWindow, aT.shortTermStep, "svm_rbf", model_path, False)

...

for k,v in self.testDF.iterrows():
    filePath = ...

    class_id, probability, classes = aT.file_classification(filePath,
self.model_path, "svm")
```

3.2 Convolutional Neural Networks for Emotion Recognition in Children's Music

Over the past two decades, CNNs have emerged as the key player for image recognition tasks. They are applied across a broad spectrum of audio signal processing tasks, achieving competitive results in speech recognition, music genre classification, and audio event detection [84]. More specifically, the capability of CNNs to recognise patterns within spectrogram representations of audio signals paved the way for their adoption in music emotion recognition. This thesis aims to utilise CNNs for emotion recognition, specifically within a collection of children's music tracks. The research goal was to modify CNN architectures previously optimised for emotion recognition in popular Western music genres and to assess their performance and adaptability within a collection of children's music songs.

3.2.1 Workflow and model architecture

Elaborating on the methodology represented in Figure 23, the research process was initiated with a series of audio pre-processing steps. The selected audio samples, primarily sourced in MP3 format, were converted to 8KHz mono WAV files. In order to ensure pre-trained model adaptability and dataset invariability, music tracks were segmented into consistent 10-second non-overlapping audio segments. The segmentation method was applied to both training and test subsets.

The input feature of CNN was the Mel-scale spectrogram, a visual representation of the audio signal that closely aligns with human auditory perception [79]. During the data loading process, spectrograms were generated with respect to pre-trained model sampling parameters and image dimensions. Thus, the resulting spectrograms were forwarded in the DL model as images with a dimensionality of 128×51.

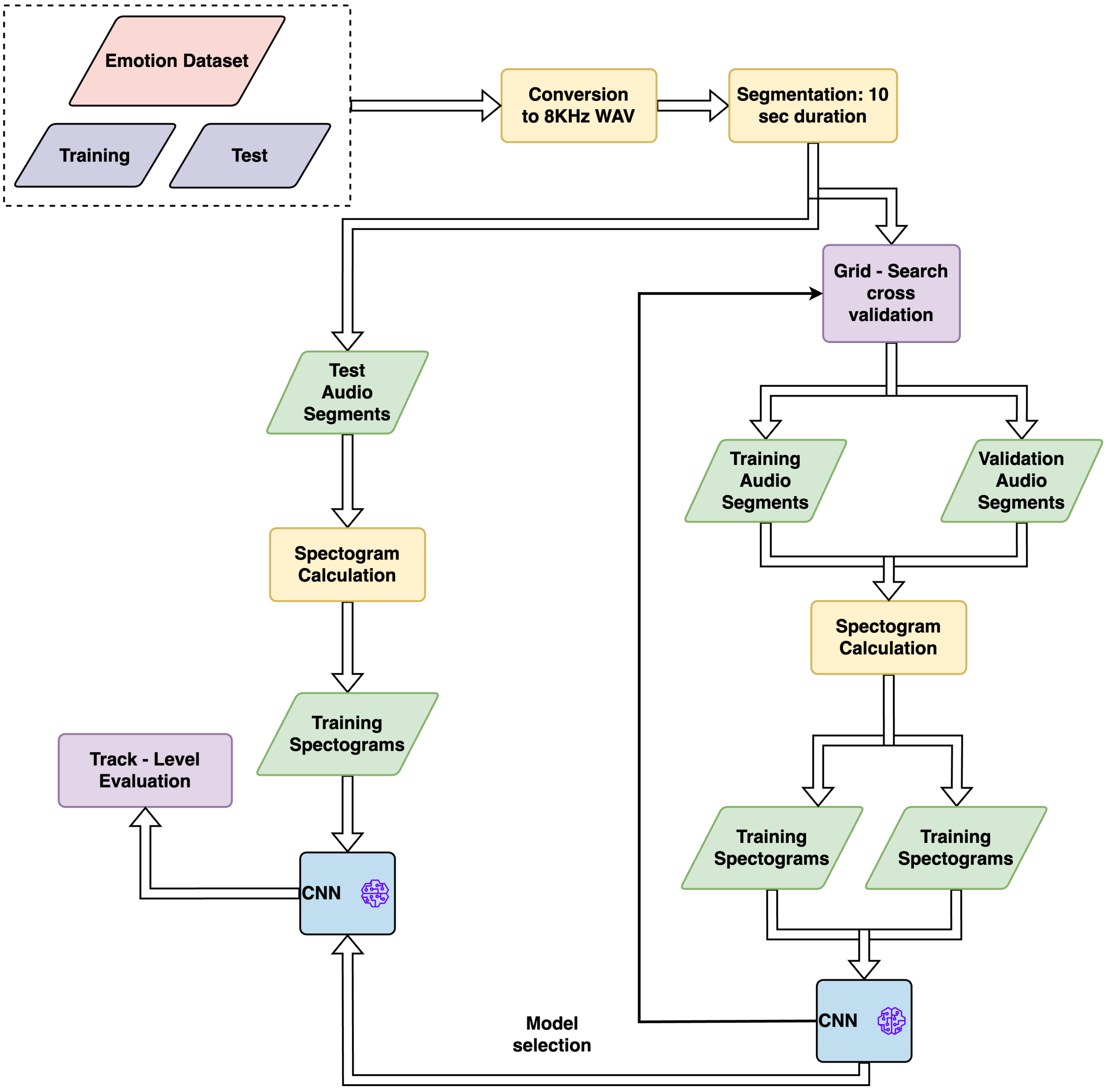


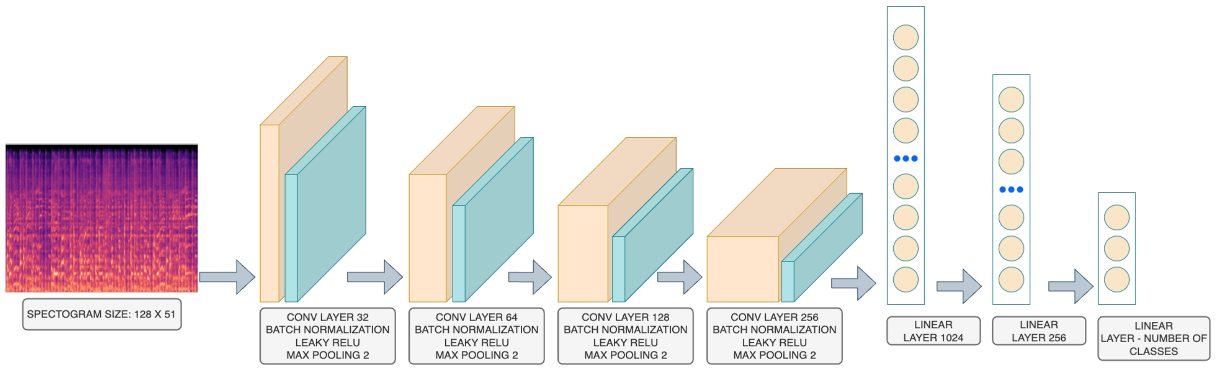
Figure 23: Illustrative depiction of the CNN model development workflow.

The code base utilised for these steps was largely derived from the *deep audio features* Python library, and feature extraction was performed via the *librosa* package [85]. Below is an excerpt of the code for Mel-spectrogram extraction within data loaders:

```

WINDOW_LENGTH = 50 * 1e-3
HOP_LENGTH = 50 * 1e-3

class FeatureExtractorDataset(Dataset):
```

```python
    ...
    feature = sound_processing.get_melspectrogram(
        signal, fs=fs, n_fft=int(WINDOW_LENGTH * fs),
        hop_length=int(HOP_LENGTH * fs))
    ...
```

```python

def get_melspectrogram(...):
    ...
    spectrogram = librosa.feature.melspectrogram(y=x, sr=fs,
        n_fft=n_fft, hop_length=hop_length)

    spectrogram_dB = librosa.power_to_db(spectrogram, ref=np.max)
    ...
```



SPECTOGRAM SIZE: 128 X 51
CONV LAYER 32 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 64 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 128 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 256 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
LINEAR LAYER 1024
LINEAR LAYER 256
LINEAR LAYER - NUMBER OF CLASSES

Figure 24: A detailed exposition of the CNN architecture adopted in this thesis.

The architectural backbone of the CNN, demonstrated in Figure 24, comprises four convolutional blocks. Each is accompanied by Batch Normalization and Leaky ReLU activation, followed by a 2×2 Max-Pooling operation. The channel depth escalates exponentially (32, 64, 128, 256) across the layers. A fully connected (FC) block, comprising three layers, performs the final classification, outputting the probability distribution over the emotion categories. The output dimensions for these FC layers are systematically configured as 1024, 256, and the number of target emotion classes.

CNN developed using the following code section:

```python
class CNN(nn.Module):
        def __init__(self, height, width, output_dim, first_channels=32,
    kernel_size=5, stride=1, padding=2):

                super(CNN, self).__init__()
                self.num_cnn_layers = 4
                self.cnn_channels = 2


                height = int(self.height / (2 ** (self.num_cnn_layers)))
                width = int(self.width / (2 ** (self.num_cnn_layers)))
                kernels = (self.cnn_channels ** (self.num_cnn_layers - 1)) *\
                    self.first_channels

                flatten_dim = kernels * height * width

                self.conv_layer1 = nn.Sequential(
                    nn.Conv2d(1, first_channels, kernel_size=kernel_size,
                              stride=stride, padding=padding),
                    nn.BatchNorm2d(first_channels),
                    nn.LeakyReLU(),
                    nn.MaxPool2d(kernel_size=2))

                self.conv_layer2 = nn.Sequential(
                    nn.Conv2d(first_channels, self.cnn_channels*first_channels,
                              kernel_size=kernel_size, stride=stride, padding=
    padding),
                    nn.BatchNorm2d(self.cnn_channels*first_channels),
                    nn.LeakyReLU(),
                    nn.MaxPool2d(kernel_size=2))

                self.conv_layer3 = nn.Sequential(
                    nn.Conv2d(self.cnn_channels * first_channels,
                              (self.cnn_channels ** 2) * first_channels,
                              kernel_size=kernel_size, stride=stride, padding=
    padding),
                    nn.BatchNorm2d((self.cnn_channels ** 2) * first_channels),
                    nn.LeakyReLU(),
                    nn.MaxPool2d(kernel_size=2))

                self.conv_layer4 = nn.Sequential(
                    nn.Conv2d( (self.cnn_channels**2) * first_channels, (self.
    cnn_channels**3) * first_channels, kernel_size=kernel_size, stride=
    stride, padding=padding),
                    nn.BatchNorm2d((self.cnn_channels ** 3) *
    first_channels),
                    nn.LeakyReLU(),
                    nn.MaxPool2d(kernel_size=2)
                    )

                self.linear1 = nn.Sequential(
                    nn.Dropout(0.75),
                    nn.Linear(flatten_dim, 1024),
                    nn.LeakyReLU()
                )

                self.linear2 = nn.Sequential(
                    nn.Dropout(0.5),
                    nn.Linear(1024, 256),
                    nn.LeakyReLU()
                )
                self.linear3 = nn.Sequential(
                    nn.Dropout(0.2),
                    nn.Linear(256, output_dim),
                    nn.LeakyReLU()
                )

            def forward(self, x):
                out = self.conv_layer1(x)
                out = self.conv_layer2(out)
                out = self.conv_layer3(out)
                out = self.conv_layer4(out)

                out = out.view(out.size(0), -1)
                out = self.linear1(out)
                out = self.linear2(out)
                out = self.linear3(out)

                return out
```

Dropout layers [86] were strategically placed after each Max Pooling operation as a regularisation technique to reinforce model generalisation capacity. The optimisation objective for the CNN model was the Cross Entropy Loss combined with the corresponding class weights to handle imbalance. This loss function measures the dissimilarity between the predicted emotion probability distributions and the true labels [48]. To dynamically adjust the learning rate during training, `ReduceLROnPlateau` strategy was employed. If the model's validation performance plateaus for a specified number of epochs, the learning rate is reduced by a factor ensuring efficient training and better convergence rates [87]. Another technique integrated into the training process was Early Stopping. This approach monitors a chosen metric and terminates training if it observes no improvement over a predefined number of epochs, reducing the risk of overfitting, as the model stops training before it starts memorising the training data [88].

3.2.2 Model Adaptation and Hyperparameter Optimization

In the pursuit of optimizing model performance for the MER task, a systematic approach was undertaken through the application of grid search cross-validation, employing a k-fold method where $k = 5$. In each fold, the dataset was split among full tracks and not in segments, with the aim of reducing model bias. This methodological framework was designed to meticulously determine the optimal settings for batch size, learning rate, and specific layer-freezing scenarios within the pre-trained CNN models.

The configurations for layer-freezing scenarios were presented as follows:

- **Scenario 0 (S0):** It was determined that all layers within the model should remain trainable, thus allowing the entire network to adapt its weights in response to the unique characteristics of the dataset, as identified.
- **Scenario 1 (S1):** Attention was exclusively directed towards the linear layers of the model by freezing the Convolutional blocks. The objective was to refine the decision-making aspects of the network, while the integrity of the feature extraction layers was maintained.
- **Scenarios 2 to 4 (S2-S4):** A progressive freezing strategy was implemented for the initial one to three layers, thereby restricting adjustments in the early layers. This approach was based on the assumption that the foundational feature detection capabilities, acquired through previous training, were universally applicable, including for the task of identifying emotional cues within music.

Hyperparameter selection was comprised of batch sizes of 16, 32, and 64, coupled with learning rates of 0.001 and 0.002. This strategy was carefully crafted with the aim of achieving a consistent trade-off between the rate of model learning and computational efficiency. The main goal was identified as enhancing model performance and robustness, while simultaneously averting the risks associated with overfitting.

3.3 Children Music emotion recognition with a Dual-Stream architecture

3.3.1 Workflow and model architecture

The goal of this model was to build a unified model that can process and combine information from multiple types of data. In this project, a Dual-Stream architecture that combines CNNs and LSTM networks with attention mechanisms is proposed for the task of children's MER. The CNN is responsible for processing spectrogram images, while the attention-based LSTM, handles the sequential aspects of symbolic music representation sequences. Total workflow is presented in Fig. 25.

Music transcription information obtained by the Basic Pitch tool [89] proposed by Spotify researchers. Basic pitch is a lightweight Automatic Music Transcription (AMT) model designed to transcribe polyphonic recordings from a single class of instruments, such as a solo piano, an ensemble of violins, and vocals, among others. Model trained to predict frame-level onset, multi-pitch, and note posteriorgrams. The input audio is first transformed into a Constant-Q Transform (CQT) representation with 3 bins per semitone and a hop size of approximately 11 ms. The Harmonic CQT (HCQT) is then computed to align harmonically related frequencies along a third dimension, thereby

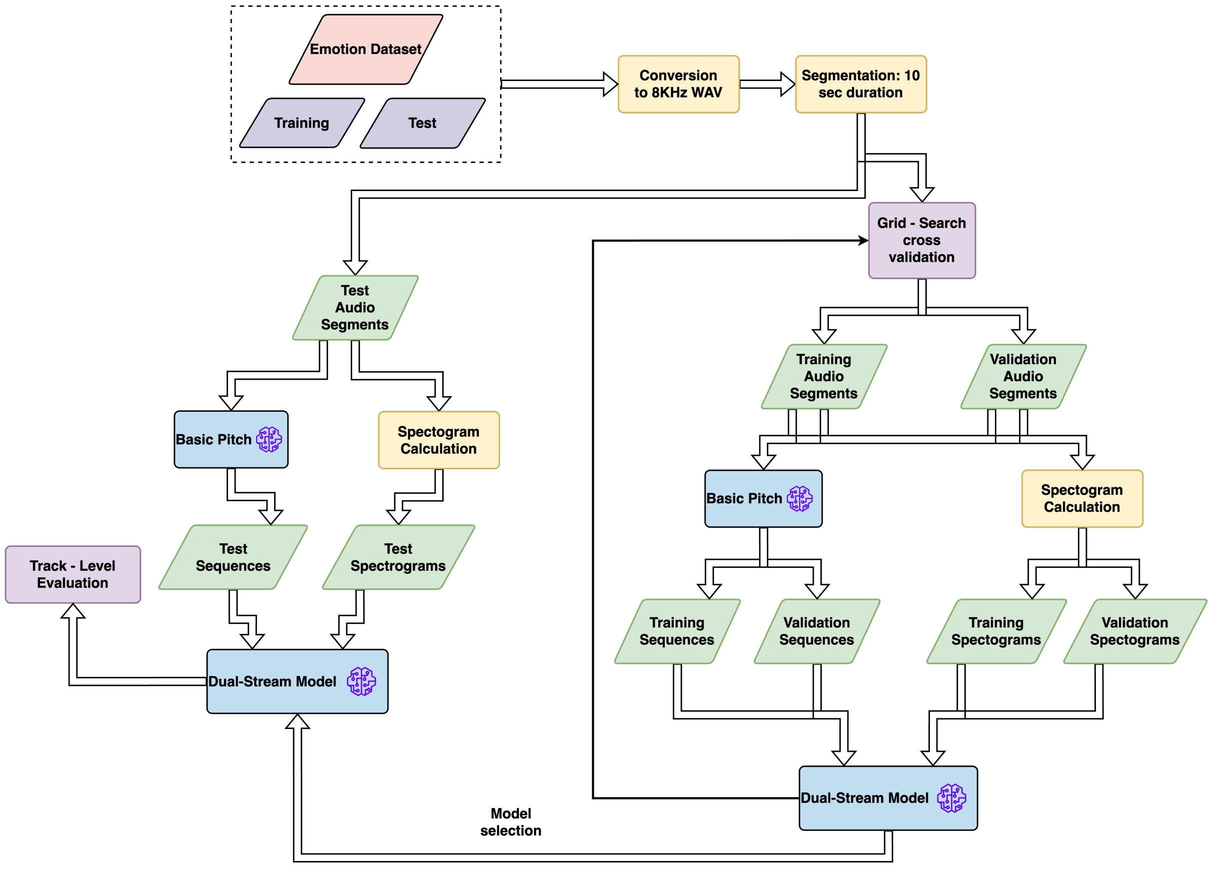


Figure 25: Illustrative depiction of the Dual-Stream model development workflow.

enabling the use of small convolutional kernels to capture this information. The model architecture, as illustrated in Fig. 26, comprises a total of 16,782 parameters. It employs a fully convolutional network that generates three distinct posteriorgrams: the onset of a note (Y_o), a note that is active (Y_n), and a pitch that is active (Y_p). These are binary matrices used as targets during training, generated from note and pitch annotations. The posteriorgrams are post-processed to create note events, which are then used for MIDI information extraction. These features are also extracted in the form of multivariate time series for onset, contour, and note events.

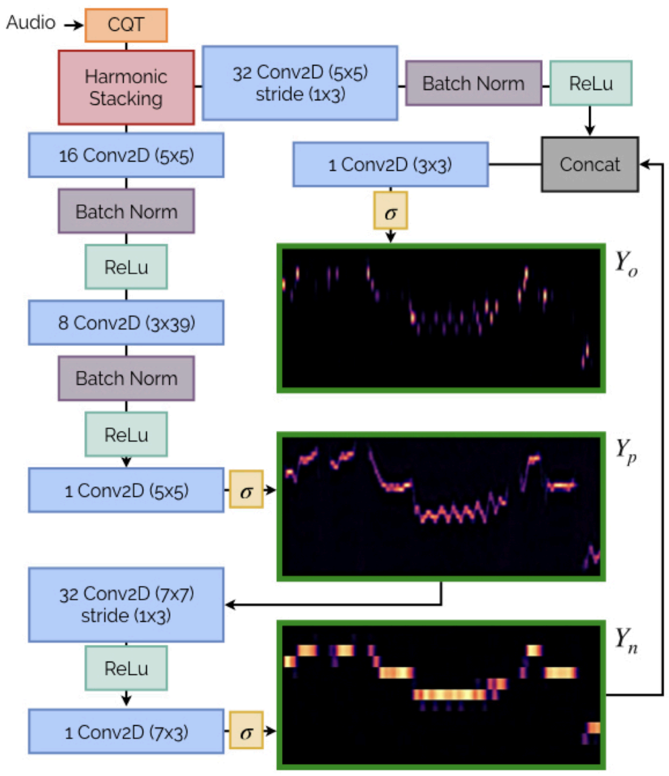


Figure 26: Basic pitch architecture.

As described in Sec. 3.2, all audio tracks from the dataset were converted to a standard format of 8 kHz WAV and segmented into 10-second duration tracks. Two primary features were extracted from the segmented tracks:

1. *Spectrogram Calculation:* Each segment underwent a spectrogram calculation, resulting in time-frequency representations that capture the energy distribution over frequencies.

2. *Basic Pitch Transcription Sequences:* This entails deriving sequences that represent the onset, contour, and note information, which are significant for capturing melodic patterns. The Basic Pitch tool provides the essential programming interface for ATM task:

```
...
from basic_pitch.inference import predict_and_save

predict_and_save(
    segments_list,
    targetDir,
    model_or_model_path = ICASSP_2022_MODEL_PATH,
    save_midi=False,
    save_model_outputs=True,
    save_notes=False,
    sonify_midi=False
)
...
```

The proposed model comprises two main components: a CNN for processing spectrogram data and a sequential model that combines onset, contour, and note features through a Long Short-Term Memory (LSTM) network coupled with a self-attention mechanism. The outputs of these networks are subsequently combined and passed through a dense classification layer. The spectrogram recognition branch is the CNN described in Section 3.2 without using the last two linear layers. The sequential model receives as input three vectors with the following dimensions: 858x88 for onset and note and 858x264 for contour, with the first dimension representing the time step. Transcription branch processes combine onset, contour, and note features using a bidirectional LSTM cell with a hidden state dimension of 128. A self-attention mechanism was integrated after LSTM to weigh the sequence of outputs, allowing the model to focus on the most relevant temporal patterns for emotion recognition. The operational characteristics as well as the mathematical implementation of this mechanism were described in Section 2.4.4. This information passes through the final linear layer to result in the transcription embedding vector. The attention-based LSTM model was implemented using the following code snippet:

```
class SelfAttention(nn.Module):
    def __init__(self, input_dim, attn_hidden, heads):
        super(SelfAttention, self).__init__()
        self.fc1 = nn.Linear(input_dim, attn_hidden, bias=False)
        self.fc2 = nn.Linear(attn_hidden, heads, bias=False)

    def forward(self, h: torch.Tensor) -> torch.Tensor:
        attn_mat = F.softmax(self.fc2(torch.tanh(self.fc1(h))), dim=1)
        return attn_mat.permute(0, 2, 1)

class Seq_Model(nn.Module):
    def __init__(self, heads=8, input_size_onset=88, input_size_contour
   =264, input_size_note=88,
                 lstm_hidden=128,
                 hidden = 256):

        super(Seq_Model, self).__init__()
        input_dim = input_size_onset + input_size_contour + input_size_note
        self.hidden = hidden
        self.emb_dim = lstm_hidden*heads*2
        # LSTM layer to process the combined input vector
        self.lstm = nn.LSTM(input_size=input_dim, hidden_size=lstm_hidden,
    batch_first=True, bidirectional=True)

        # Self-attention layer
        self.attention = SelfAttention(input_dim=lstm_hidden*2, attn_hidden
    =128, heads=heads)

        self.linear1 = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(lstm_hidden*heads*2, hidden),
            nn.LeakyReLU(),
        )

    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        features = self.attention(lstm_out)
        weighted_features = torch.bmm(features, lstm_out)
        weighted_features = weighted_features.view(weighted_features.size
    (0), -1)
        output = self.linear1(weighted_features)
        return output
```

Representations resulted by CNN and the attention-based LSTM model were fused and then passed into the final classification fully connected network, as illustrated in Figure 27. This fusion scheme enables the model to extract a combined representation after modality-specific processing. The final classification is performed using two fully connected layers with output dimensions of 128 and the number of classes, respectively.

Dropout layers [86] were incorporated after each Max Pooling step and after attention output as a regularization strategy to enhance the model's generalization capability. The optimisation goal for the Dual - Stream model was set to minimize the Cross Entropy Loss, which was adjusted for class imbalance by integrating corresponding class weights. This loss function quantifies the divergence between the predicted probability distributions of emotions and the actual labels [48]. To dynamically adjust the learning rate during the training process, the `ReduceLROnPlateau` strategy was employed. This approach reduces the learning rate by a factor when the validation performance of the model plateaus for a predefined number of epochs, ensuring more efficient training and improved convergence rates [87]. Additionally, the training protocol incorporated the Early Stopping technique, which terminates training when no improvement is observed on a monitored metric over a specified number of epochs. This strategy reduces the risk of overfitting by preventing the model from memorising the training data [88].

3.3.2 Model Adaptation and Hyperparameter Optimization

The optimization of the Dual-Stream model commenced with an essential phase in which grid search cross-validation was deployed to determine the optimal number of

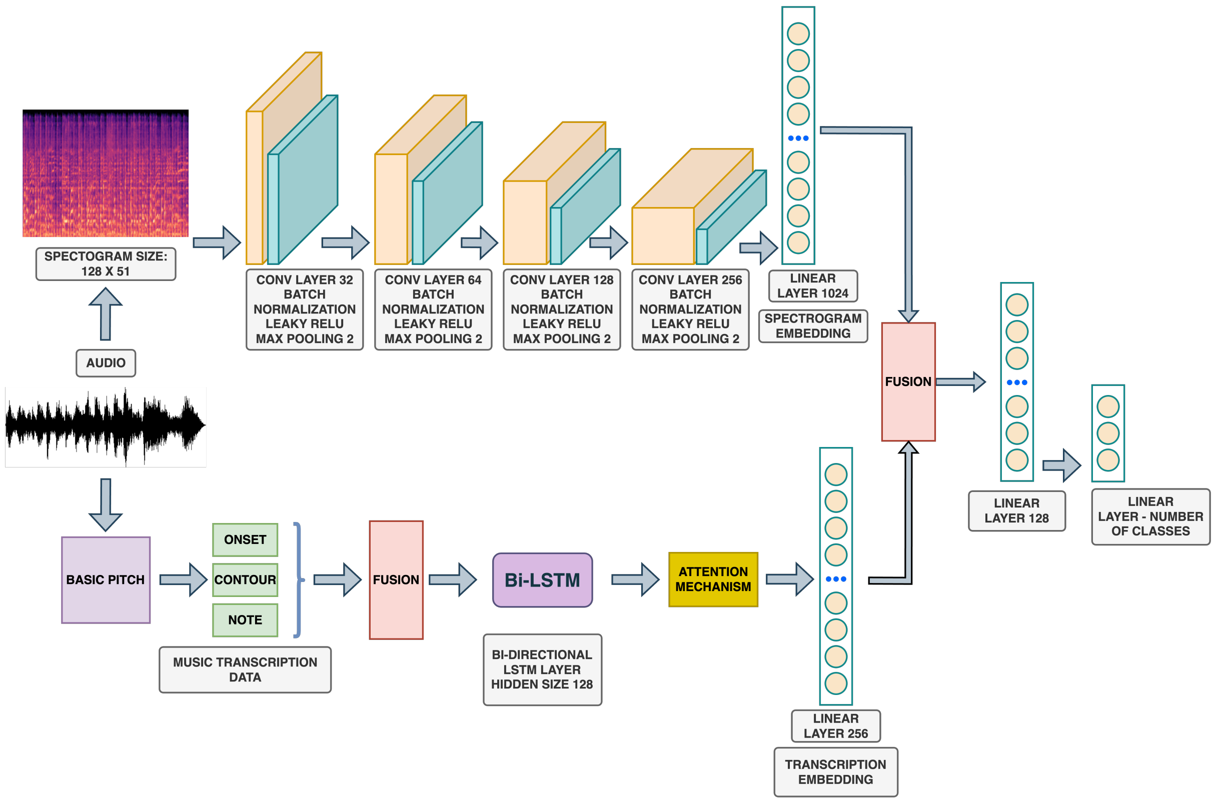

SPECTOGRAM SIZE: 128 X 51
AUDIO
CONV LAYER 32 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 64 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 128 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
CONV LAYER 256 BATCH NORMALIZATION LEAKY RELU MAX POOLING 2
LINEAR LAYER 1024
SPECTROGRAM EMBEDDING
FUSION
LINEAR LAYER 128
LINEAR LAYER - NUMBER OF CLASSES
BASIC PITCH
ONSET
CONTOUR
NOTE
MUSIC TRANSCRIPTION DATA
FUSION
Bi-LSTM
BI-DIRECTIONAL LSTM LAYER HIDDEN SIZE 128
ATTENTION MECHANISM
LINEAR LAYER 256
TRANSCRIPTION EMBEDDING

Figure 27: Dual-Stream Deep Learning model architecture.

attention heads alongside foundational hyperparameters. The k-fold method, with k = 5, facilitated this phase, ensuring that in each fold, the dataset was divided among full tracks rather than segments to mitigate model bias. This initial stage involved a rigorous assessment of the impact that various quantities of attention heads, specifically, 2, 4, 8, and 16, have on the model's efficacy. Additionally, this examination was extended to incorporate the evaluation of different hyperparameter configurations, namely batch sizes of 16 and 32, and learning rates of 0.001 and 0.002. The aim was to ascertain the combination that most significantly improves the model's overall performance. In alignment with the practices established for the Convolutional branch of the architecture, this process included the application of pre-trained weights and the most effective fine-tuning strategies previously identified during the CNN model's development, as documented in Section 3.2.

4 Music in the crib dataset

In this section, the steps of constructing the *Music in the Crib* dataset are described in detail. The proposed dataset is a comprehensive collection designed specifically for the analysis of children's music. The following sections offer a comprehensive view of the dataset's creation, ranging from data collection strategies to the annotation process for the MER task.

4.1 Data Collection

To address the challenges of children's music analysis, a dataset comprising 5055 songs with a focus on both genre diversity and data integrity was constructed. The data collection process was organised into three strategic phases, each designed to contribute distinct attributes to the composite dataset:

- *Phase A - Expert-Labelled Corpus:* This phase utilised a carefully curated dataset of children's songs, collected and annotated in terms of sub-genre by child psychology researchers from OsloMet University. Each entry was characterised across multiple musical dimensions, such as melody, tempo, and content, to produce a multi-faceted feature set. Special attention was given to mitigating the subjectivity associated with certain labels, by employing a consensus approach among domain experts.

- *Phase B - Platform-Sourced Data:* This phase incorporated curated Spotify playlists selected by experts in child education and musicology. Each playlist served as a distinct category, enabling a rich, context-aware feature space that extended beyond traditional musical attributes.

- *Phase C - Dynamic Data Ingestion:* In this final phase, an incoming data repository was established to facilitate continual updates throughout the project duration.

Understanding the structure of the dataset serves as a first step towards its analysis. The song categories within the music corpus provide essential insights into its organization. Table 2 enumerates the frequency distribution of these categories:

- *Lullabies:* With 2683 entries, the prevalence of lullabies underscores their enduring cultural and cognitive importance. Historically utilised in bedtime routines, the lullabies genre is both a comfort representative and an introduction to cultural auditory landscapes.

Table 2: Distribution of songs across various categories.

Category	*Count*
Lullaby	2683
Nonverbal_Songs	650
Classical	601
Jazz	267
Rhymes	226
Relaxing	47
Verbal_Songs	44
Songs_for_Babies	39
Movement	32
Songs_for_Toddlers	23
Total	*5055*

- *Instrumental against Verbal Songs:* The dataset reveals a preference for non-verbal compositions, as evident from the 650 entries under Nonverbal Songs. When coupled with other instrumental categories, such as rhymes and classical songs, this bias is amplified. This suggests that the absence of lyrics may offer universal accessibility.

- *Genre Diversity:* The classical and jazz genres demonstrate the dataset's variety, with 601 and 267 songs, respectively. Classical compositions often provide foundational musical exposure, while the nature of jazz engages and stimulates children's creativity.

- *Functional Categories:* Categories like Relaxing, Movement, Songs for Babies, and Toddlers depict the role of music in children's developmental stages. They highlight the intricate ways music is adapted to various needs, moods, and developmental milestones.

This categorical distribution allows for fine - grained analysis, accommodating a plethora of interpretations towards the cognitive, emotional, and physical dimensions of children's development.

4.2 Music emotion recognition task

Music has the unique ability to elicit a wide range of emotions. The objective of this phase was to identify the emotional patterns within the corpus of children's songs. To accomplish this, a MER dataset that utilizes two predominant emotional dimensions was developed. Each emotion dimension, i.e., arousal and valence, is classified using a 3-class approach as described by Posner et al. [90]:

- Arousal: Categorised as Low (Weak), Medium (Neutral), and High (Strong).
- Valence: Classified into Negative, Neutral, and Positive.

These dimensions were chosen for their robustness and applicability in capturing a broad spectrum of emotional states, especially beneficial in the context of children's music. To carry out the MER task, 447 songs were selected from the complete dataset as introduced in Section 4.1, hereafter referred to as the Complete Collection (CC). This subset was selected to capture a diverse range of musical elements, genres, and moods. This strategy aimed to provide a comprehensive view of the emotional spectrum embedded in these songs.

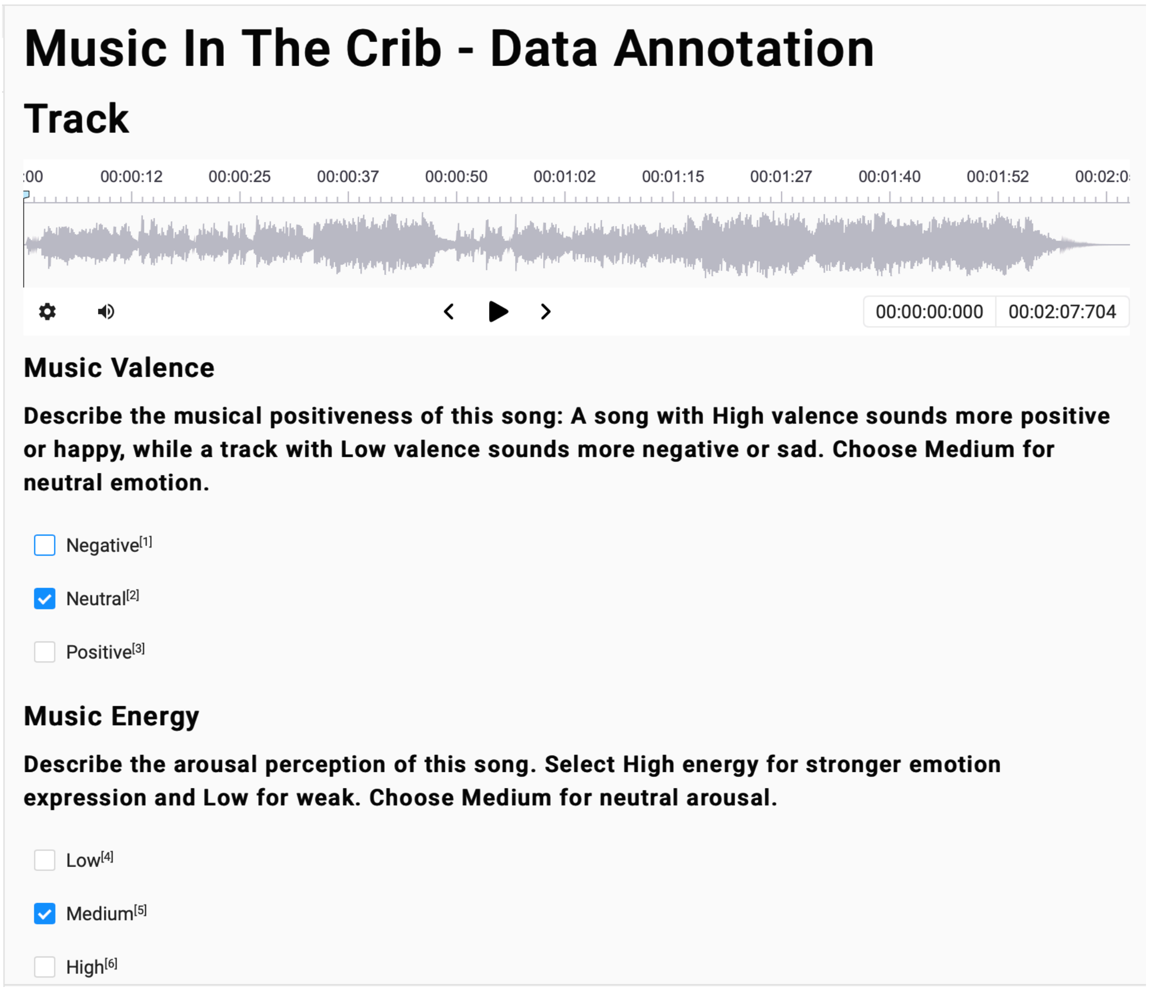
4.2.1 Annotation process

To guarantee the robustness and diversity of the annotations, a heterogeneous group of 10 annotators was engaged. Their demographic information is summarized in Table 3. An aspect worth-noting is the intentional variability in the volume of songs assigned to each annotator. While three of them undertook the comprehensive task of evaluating all 447 songs, the remaining seven were assigned approximately 50 songs each, chosen at random. This stratified approach aimed to mitigate potential bias and offer a more expansive emotional assessment.

Table 3: Information about annotators.

Annotator ID	*Age*	*Nationality*	*Area of Expertise*
0	18-25	Europe	Formal Science
1	18-25	Europe	Formal Science
2	25-30	Europe	Other
3	Over 30	Europe	Formal Science
4	Over 30	Europe	Formal Science
5	25-30	Europe	Formal Science
6	25-30	Asia	Natural Science
7	25-30	Europe	Other
8	Over 30	Europe	Other
9	25-30	Europe	Formal Science

The foundation for any data-driven research lies in a structured and efficient annotation process. In this study, Label Studio [91] was employed to create an intuitive and user-friendly annotation environment, as illustrated in Figure 28. This platform was selected for its flexibility and ease of customisation, allowing annotators to listen to selected song snippets and to assign appropriate labels for each emotional dimension.


Music In The Crib - Data Annotation

Track

Music Valence

Describe the musical positiveness of this song: A song with High valence sounds more positive or happy, while a track with Low valence sounds more negative or sad. Choose Medium for neutral emotion.

- ☐ Negative[1]
- ☑ Neutral[2]
- ☐ Positive[3]

Music Energy

Describe the arousal perception of this song. Select High energy for stronger emotion expression and Low for weak. Choose Medium for neutral arousal.

- ☐ Low[4]
- ☑ Medium[5]
- ☐ High[6]

Figure 28: Annotation environment created using Label studio.

4.2.2 Annotations aggregation process

Despite the fact that categorical labels were crucial in guiding the annotators, for the sake of computational approaches, emotion classes were mapped to numerical labels according to the guidelines of Table 4. This conversion facilitated the subsequent stages of data analysis, especially in computing the level of disagreement among annotators and in aggregating annotations for final label assignment.

The fundamental post-annotation step was to adopt a specific statistical approach in order to find the final emotion labels and assess annotator agreement. Firstly, the mean and standard deviation for each song were calculated. The final label assignment was performed by mapping the continuous mean value into a discrete category. On the other hand, the computed standard deviation provided insights into the reliability of

Table 4: Music emotion classes: Categorical to numerical mapping.

Arousal	*Valence*	*Numerical Label*
Weak/Low	Negative	1
Neutral/Medium	Neutral	2
Strong/High	Positive	3

the final label.

For the annotation aggregation process in our MER task, mean values were compared with two predefined label decision thresholds, and each song was classified into one of the three emotion categories per dimension. These threshold values were chosen based on careful analysis and consideration of the domain-specific emotional distribution. Thresholds were determined as follows:

- If $\mu < T_1 = 1.66$, it's classified as "Weak" for arousal or "Negative" for valence.
- If $T_1 \leq \mu \leq T_2 = 2.33$, it's classified as "Neutral".
- If $\mu > T_2$, it's classified as "Strong" for arousal or "Positive" for valence.

One of the most important steps of the post-annotation procedure was to capture the level of disagreement among the annotators. This is essential to understanding the consistency and reliability of the annotations. Songs that displayed high deviations in labels, indicating a significant disagreement among annotators, were excluded. As a consequence, along with the average values of arousal and valence annotations used for assigning the final labels through the previously mentioned thresholding process, a deviation threshold is also applied to eliminate annotations that have high variability. In this context, for each sample, the Mean Absolute Deviation (MAD) of the annotations is determined for both tasks using the formula:

$$MAD_s = E\left(|X_{s,\alpha} - E(X_{s,\alpha})|\right) \tag{65}$$

To further refine annotations, the following elimination strategy was used by comparing MAD with these thresholds:

- For Arousal: If $MAD_s > T_{mad_1} = 0.7$.
- For Valence: If $MAD_s > T_{mad_2} = 0.57$.

To measure the inter-annotator agreement, consider the annotation of the i-th sample by annotator a, represented by $A_{i,a}$. This annotation value lies within the range of 1 to 3. Let M be the total number of annotators. The deviation of $A_{i,a}$ from the mean annotation value for the same sample is computed as the absolute difference:

$$u_{i,a} = \left| A_{i,a} - \frac{1}{M} \sum_{j=1}^{M} A_{i,j} \right| \tag{66}$$

Annotation aggregation details are depicted in Table 5. Upon applying the 0.7 deviation threshold, 362 samples were retained from the initial 447 for the arousal task. Conversely, the threshold of 0.59 yielded 381 valid samples for valence. The division into training and test samples was also influenced by this thresholding, ensuring balanced representations in both datasets. It's noteworthy that the average disagreement, post-thresholding, was slightly lower (0.39) for Valence task. On the other hand, the average disagreement on the arousal task was 0.43.

Table 5: Annotation aggregation details

	Arousal	*Valence*
Deviation Threshold Value	0.7	0.59
Total number of samples before deviation thresholding	447	447
Total number of samples after deviation thresholding	362	381
Training samples	189	214
Test samples	173	167
Average disaggrement	0.43	0.39

In addition, annotation aggregation results for each emotional dimension and the corresponding class are presented in Table 6 for Arousal and in Table 7 for Valence.

Table 6: Annotation Aggregation: Per class details for arousal

Emotional Arousal	*Mean Threshold Value*	*Samples per Class*
Weak	$\mu < 1.66$	153
Neutral	$1.66 \leq \mu \leq 2.33$	168
Strong	$\mu > 2.33$	41

Ensuring the robustness of the test set was of paramount importance. Therefore, all songs that received more than four annotations were auto-included in the test subset. This criteria ensured that the selected songs had undergone a comprehensive emotional assessment. To sum up, the annotations, when processed, would not only help in understanding the emotional landscape of the chosen songs but also serve as a crucial

Table 7: Annotation Aggregation: Per class details for valence

Emotional Valence	*Mean Threshold Value*	*Samples per Class*
Negative	$\mu < 1.66$	85
Neutral	$1.66 \leq \mu \leq 2.33$	170
Positive	$\mu > 2.33$	126

component for subsequent modeling and prediction tasks. The diverse set of annotators, with their varied backgrounds, promises a rich and multidimensional perspective on the emotional content of the songs.

5 Results

In this section of the thesis, the detailed analysis and outcomes of the experiments conducted are presented. Subsequent subsections delve into the experimental hardware setup and the evaluation metrics used to assess model performance, providing insights into the criteria for success and areas for improvement. The analysis of audio features follows, highlighting the key characteristics that distinguish children's music and its emotional content. The chapter progresses to discuss the results of various ML models, including SVM and CNN-based models applied to the MER framework. Each model's performance is meticulously evaluated, with emphasis on its strengths and limitations. This comprehensive presentation of results not only validates the research methodology but also sets the stage for potential future advancements in the domain.

5.1 Codebase instructions

For the successful completion of the MSc thesis on the "Music in the Crib" project, a structured approach in managing the project's codebase has been adhered to. The code used in this thesis is available in this Github Repository. It is ensured that Python 3.8 is installed, as it is the primary programming language for this project. All necessary dependencies have been installed by executing the command `pip install -r requirements.txt` in the terminal. This step prepares the environment with all the libraries needed for the project. You can exclude pytorch and Tensorflow to ensure the compatibility with the correct version which will match for your system. Additionally, FFmpeg, which is crucial for handling audio files, has been installed.

Once the environment setup has been completed, focus has been shifted to the dataset construction and manipulation for the research. The directory '.data_project' has been navigated to, where the data, including MP3 files, annotations, and metadata, are organized. Links to download data are available in repository. The command `python data_project/constructor.py` has been executed to process these files into a structured format suitable for analysis. This script consolidates the data into a new directory structure under 'mic_dataset', which includes WAV files, extracted audio features, and formatted annotations essential for music emotion recognition tasks. Particular attention has been paid to the exploratory data analysis (EDA), conducted using the `eda.ipynb` notebook to understand data distributions and potential biases, which is crucial for preliminary data analysis. Additionally, scripts like `src/utils/preprocessing.py` have been provided to prepare the data for deep learning models, which involve segmenting audio files and organizing them by emotional valence and energy levels, essential for training the models effectively. Consistency in data handling and preprocessing has been maintained to ensure the reliability of the machine learning models, which is pivotal for the empirical section of the thesis.

Script `src/ml_pipeline.py` is used as to run the workflow for the training and evaluation of ML models, including the SVM baseline, the CNN and the Dual - Stream model. Core functions are presented bellow:

```

# SVM

from trainers.svm_trainer import SVM_Trainer
from evaluation.test_model import SVM_Tester

trainer = SVM_Trainer(task, results_dir, wav_dir, infoFile)
trainer.train()

tester = SVM_Tester(task, results_dir, wav_dir, infoFile, full_annot_file)
cm_df, perClassDF, roc_curve, pr_curve, overall = tester.test()
```

```

# CNN

from trainers.cnn_trainer import CNN_Trainer

from evaluation.evaluate_cv import evaluate_deep_model_cv
from evaluation.test_model import Deep_Tester

# Cross validation
run_cnn_cv(task, scenarios, hyps, cnn_cv_results_file, ptModelPath,
    results_dir, train_dirs, test_dirs, wav_dir, infoFile)
cnn_cv_results_df = evaluate_deep_model_cv(cnn_cv_results_file)

# Train selected model
cnn_trainer = CNN_Trainer(task=task,
                    scenario=scenarios[scenario],
                    batch_size=batch_size,
                    learning_rate=learning_rate,
                    ptModelPath=ptModelPath,
                    resultsDir=results_dir,
                    trainDirs=train_dirs,
                    testDirs=test_dirs,
                    wavDir=wav_dir,
                    infoFile=infoFile)

res = cnn_trainer.train_model()

print("Model trained successfully:")
print(res)

m_idx = input("Select model index: ")

cnn_trainer.select_model(m_idx)
```

```

# Test model
cm_df, perClassDF, roc_curve, pr_curve, overall = Deep_Tester(task=task,
                                        results_dir=results_dir,
                                        model_name="cnn",
                                        test_dirs=test_dirs,
                                        wav_dir=wav_dir,
                                        info_file=infoFile,
                                        full_annot_file=full_annot_file).
    test_model()
```

```

# Dual - Stream model

from trainers.ds_trainer import DS_Trainer

from evaluation.evaluate_cv import evaluate_deep_model_cv
from evaluation.test_model import Deep_Tester


if task == "music_energy":
        cnn_conf = {
            "strategy":None,
            "layers_freezed":3
        }
elif task == "music_valence":
    cnn_conf = {
        "strategy":None,
        "layers_freezed":0
    }

# Cross validation
run_ds_cv(task, cnn_conf, heads, hyps, ds_cv_results_file, ptModelPath,
    results_dir, train_dirs, test_dirs, wav_dir, infoFile,midi_dir)
ds_cv_results_df = evaluate_deep_model_cv(ds_cv_results_file, mode="ds")

# Train selected model
ds_trainer = DS_Trainer(task=task,
                    cnn_conf=cnn_conf,
                    head=head,
                    batch_size=batch_size,
                    learning_rate=learning_rate,
                    ptModelPath=ptModelPath,
                    resultsDir=results_dir,
                    trainDirs=train_dirs,
                    testDirs=test_dirs,
                    wavDir=wav_dir,
                    infoFile=infoFile,
                    midi_dir=midi_dir)
```

```

res = ds_trainer.train_model()

print("Model trained successfully:")
print(res)

m_idx = input("Select model index: ")

ds_trainer.select_model(m_idx)

# Test model
cm_df, perClassDF, roc_curve, pr_curve, overall = Deep_Tester(task=task,
                                            results_dir=results_dir,
                                            model_name="ds",
                                            test_dirs=test_dirs,
                                            wav_dir=wav_dir,
                                            info_file=infoFile,
                                            full_annot_file=full_annot_file
    ).test_model()
```

5.2 Experimental Hardware Setup

The experiments were conducted on two distinct hardware configurations. System 1 was a Mac system running macOS, equipped with an M1 Pro processor and 16 GB of RAM, optimised for efficient CPU-based computations. System 2 was an Ubuntu 20 machine with an AMD 8-core CPU, 16 GB of RAM, and an NVIDIA RTX 2060 TI with 6 GB of GPU memory, providing a suitable environment for tasks that required both CPU and GPU acceleration. Both systems were rigorously maintained to ensure uniformity in the software environment, which included library dependencies and operating system parameters.

5.3 Evaluation metrics

5.3.1 Metrics for the evaluation of Machine Learning models

It was essential to establish a robust evaluation framework in order to assess the classification of emotional attributes within the proposed dataset. The evaluative framework encompassed basic error metrics, composite metrics, and methodologies for segment-level and full track-level evaluations. The basic building blocks of performance evaluation are the number of correctly classified samples as well as Type I and Type II errors [92]: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These metrics were derived from the comparison of the model's predictions against the ground truth labels.

- *True Positives (TP):* Correctly identified positive instances.
- *True Negatives (TN):* Correctly identified negative instances.
- *False Positives (FP):* Incorrectly identified positive instances.
- *False Negatives (FN):* Incorrectly identified negative instances.

The Confusion Matrix encapsulates the distribution of error metrics, offering an intuitive visualization of a model's performance:

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Derived metrics such as Accuracy, Precision, Recall, and F1 Score are significant for a comprehensive assessment of a model's performance:

- *Accuracy:* The ratio of correctly classified instances to the total instances.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision:* The ratio of correctly predicted positive observations to the total predicted positives.

$$Precision = \frac{TP}{TP + FP}$$

- *Recall:* The ratio of correctly predicted positive observations to the all observations in actual class.

$$Recall = \frac{TP}{TP + FN}$$

- *F1 Score:* The harmonic mean of Precision and Recall.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the prediction capacity of a model by depicting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The Area Under the Curve (AUC) score quantifies the overall ability of the model to discriminate between positive and negative classes. A higher AUC score indicates better model performance as well as a higher generalisation capability.

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

These metrics, evaluated at both segment-level and full track-level, provide an enriched assessment of models in the MER task. Transitioning from segment-level to full track-level evaluation entails aggregating segment-level predictions to obtain the emotional attributes of the entire track. A probabilistic aggregation method was employed to obtain full track-level predictions.

5.3.2 Permutation tests for assessing classifier significance

Permutation tests, a non-parametric statistical tool, offer a robust framework for evaluating ML models by assessing the significance of features in model predictions without making assumptions about the underlying data distribution [93]. This method is particularly suited for comparing the empirical distribution of model accuracies under the null hypothesis, which posits no difference in performance between models.

Given two models, A and B, with observed accuracy sets $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ respectively, where n represents the number of measurements (e.g., accuracy scores from cross-validation folds), the permutation test evaluates the null hypothesis $H_0 : \mu_X = \mu_Y$ against the alternative $H_a : \mu_X \neq \mu_Y$ without assuming normal distributions of accuracies. The test statistic used is the difference in means between the two sets, $\Delta = |\bar{x} - \bar{y}|$, where $\bar{x}$ and $\bar{y}$ are the sample means of X and Y, respectively.

The permutation test involves randomly shuffling the combined dataset $Z = X \cup Y$ and reallocating it into two new sets X' and Y' of sizes equal to X and Y. This resampling process is repeated M times (e.g., 10,000 permutations), each time calculating the difference in means $\Delta_m = |\bar{x'} - \bar{y'}|$ for the permuted sets. The p-value is then computed as the proportion of permutations where $\Delta_m \geq \Delta$, providing a measure of the probability of observing a test statistic as extreme as, or more extreme than, the observed under the null hypothesis.

This method offers a robust framework for comparing classifiers by effectively controlling for Type I error rates without reliance on the assumptions required by traditional parametric tests. By employing permutation tests, the significance of performance differences between ML models can be assessed by providing a solid statistical foundation for our comparative analysis.

5.3.3 Probabilistic Emotion Alignment

The universal emotional impact of music poses a substantial challenge in quantification, necessitating robust approaches for emotion recognition. Conventional methods often rely on direct label comparisons against true labels, as seen in the preceding sections.

However, this section introduces an alternative probabilistic framework for model assessment. Specifically, rather than utilizing aggregated annotations for songs, individual annotations are re-collected and processed to construct probability distributions that represent the emotional content within musical tracks.

By leveraging the annotation distributions associated with each music sample, a comparative analysis is facilitated, contrasting them with the probability distributions derived from the model's posterior predictions through the adoption of a divergence-based similarity metric. A higher similarity score indicates a higher alignment between the model's predictions and the human emotional perception in musical tracks. The Jensen-Shannon divergence, a symmetrized and smoothed variant of the Kullback-Leibler divergence, is employed to quantify the dissimilarity between probability distributions. The Jensen-Shannon divergence (JSD) is defined between two probability distributions P and Q as:

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \tag{67}$$

where $M = \frac{1}{2}(P + Q)$, and $D(P||M)$ represents the Kullback-Leibler divergence between P and M. The JSD is inherently symmetric and possesses a finite value. The square root of JSD defines a metric known as the Jensen-Shannon distance.

To transform the Jensen-Shannon divergence into a similarity score, an exponential function is employed:

$$S_{JS}(P, Q) = e^{-JSD(P||Q)} \tag{68}$$

This transformation ensures that the similarity score $S_{JS}(P, Q)$ lies within the range of 0 to 1, where 1 denotes identical distributions and 0 signifies maximal dissimilarity. To derive a single similarity measure across the dataset, individual similarity scores are aggregated using the arithmetic mean:

$$\bar{S}_{JS} = \frac{1}{N}\sum_{i=1}^{N} S_{JS}(P_i, Q_i) \tag{69}$$

Here, N denotes the number of tracks in the dataset, while P_i and Q_i represent the model and annotator probability distributions for the i-th track, respectively. The mean similarity score $\bar{S}_{JS}$ provides a holistic assessment of the extent to which the model's emotion predictions align with human annotations, on average.

5.4 Analysis of audio features

In the ever-evolving domain of audio signal processing, the extraction and interpretation of informative features serve as significant steps for a range of applications, such as music genre classification and emotion recognition. Information encoded within audio signals demands an exhaustive analysis of features that can effectively represent this complex data. This work aims to unravel the discrimination capacity of audio features in two specific tasks:

- Discrimination between children's and non-children's audio data.
- Emotion recognition within children's audio data.

The proposed methodological approach relies on a phased strategy. Initially, Analysis of Variance (ANOVA) [94] was employed to investigate the distribution of each feature across the different classes for both tasks. This statistical technique affords us the capability of understanding which features are statistically significant in separating the classes. Subsequently, feature selection procedures are undertaken, with a preliminary consideration given to retaining the top 10 features based on their F-values in the ANOVA test. Nevertheless, the number of features to retain (k) remains an open question and will be revisited in the context of potential ML model performance.

5.4.1 Audio Features for Genre Classification

Within the domain of music genre classification, the task of discriminating between children's and non-children's audio data presents notable challenges. The study and interpretation of the relevant audio features not only contribute to the present analysis but also carry substantial implications for future explorations within the children's audio data corpus. Utilizing the Analysis of Variance (ANOVA) as the principal metric, a set of statistically significant features was identified. Each feature captured significant discriminative information between the two classes. To augment the robustness of the statistical analysis, box plots of these select features for each class are presented in Figure 29.

To begin with, the mean spectral spread is higher for non-children's music in comparison with children's music, and their boxes have a significant discrimination margin. Spectral spread measures the distribution and spread of energy around the mean frequency of an audio signal. The slight elevation in the spectral spread mean for non-children's music indicates a more expansive frequency distribution due to the complex instrumentation and richer harmonics typically found in non-children's music. On the other hand, the distribution of this feature in children's music is equivalent to the simple and repetitive melody patterns found in such music.

In terms of other frequency domain features, there are notable differences in the distributions of spectral entropy and roll-off. Spectral entropy measures the randomness of a frequency spectrum. The standard deviation of spectral entropy is lower for children's music, implying that there is less variability in spectral complexity across different children's songs. However, these results were anticipated due to the genre diversity within the Western music dataset. Spectral rolloff is the frequency below which a specified percentage of the total spectral energy lies. The standard deviation of this feature is significantly higher for non-children's music, suggesting major variances in frequency distribution cutoffs. As a consequence, the delta standard deviations of these two features are informative for music genre discrimination. More particularly, the variability in the changes in spectral entropy and roll-off is more noticeable for non-children's music.

Zero Crossing Rate (ZCR) measures the rate at which a signal changes from positive to negative or vice versa. The delta standard deviation of ZCR emerges as a significant descriptor for rhythmic dynamics, effectively capturing rapid changes in the audio signal. The variability in ZCR changes (delta) appears to be more pronounced in non-children's music, indicating that it may have more rapid fluctuations in sound energy.

As far as Mel Frequency Cepstral Coefficients (MFCCs) are concerned, the mean of MFCC-3 and standard deviation of MFCC-9 represent a distinct separation between children and non-children music. The third MFCC coefficient has higher values for non-children's music, and this difference suggests variations in the timbral texture between the two categories. However, the range of children's music box plots is significantly higher, indicating texture diversity. On the other hand, the ninth MFCC coefficient's standard deviation is higher for children's music. This suggests more variability in the spectral shape of children's music, potentially hinting at the diverse range of sounds within children's songs. Finally, the standard deviations of Delta Chroma 3 and Delta Chroma 7 serve as primary indicators for variations in the 3rd and 7th chroma bands, closely related to harmonic complexities and key modulations. Their mean values and the associated range of values are higher for Western songs.

5.4.2 Audio Features for children music emotion recognition

In the domain of emotion recognition within children's audio data, highlighting features that accurately capture the essence of various emotional dimensions stands as a crucial task. In this step, analysis focuses on two specific emotional dimensions: music energy/arousal and music valence. Selected using the ANOVA method, these features have shown significant discrimination capacity for classifying music emotions.

The features imperative for discerning music arousal encompass various spectral attributes and their derivatives, according to Figure 30. Mean of Spectral Spread, Standard Deviation of Spectral Entropy, and Standard Deviation of Spectral Rolloff serve as robust indicators of spectral complexity and the spread of energy across frequency bands. Additionally, the Delta Standard Deviations of Zero Crossing Rate (ZCR), Spectral Entropy, and Spectral Rolloff provide insights into the rhythmic and spectral transitions, essential for understanding energetic attributes. Unique to this task are the Delta Standard Deviations of the 10th through 13th Mel-frequency Cepstral Coefficients

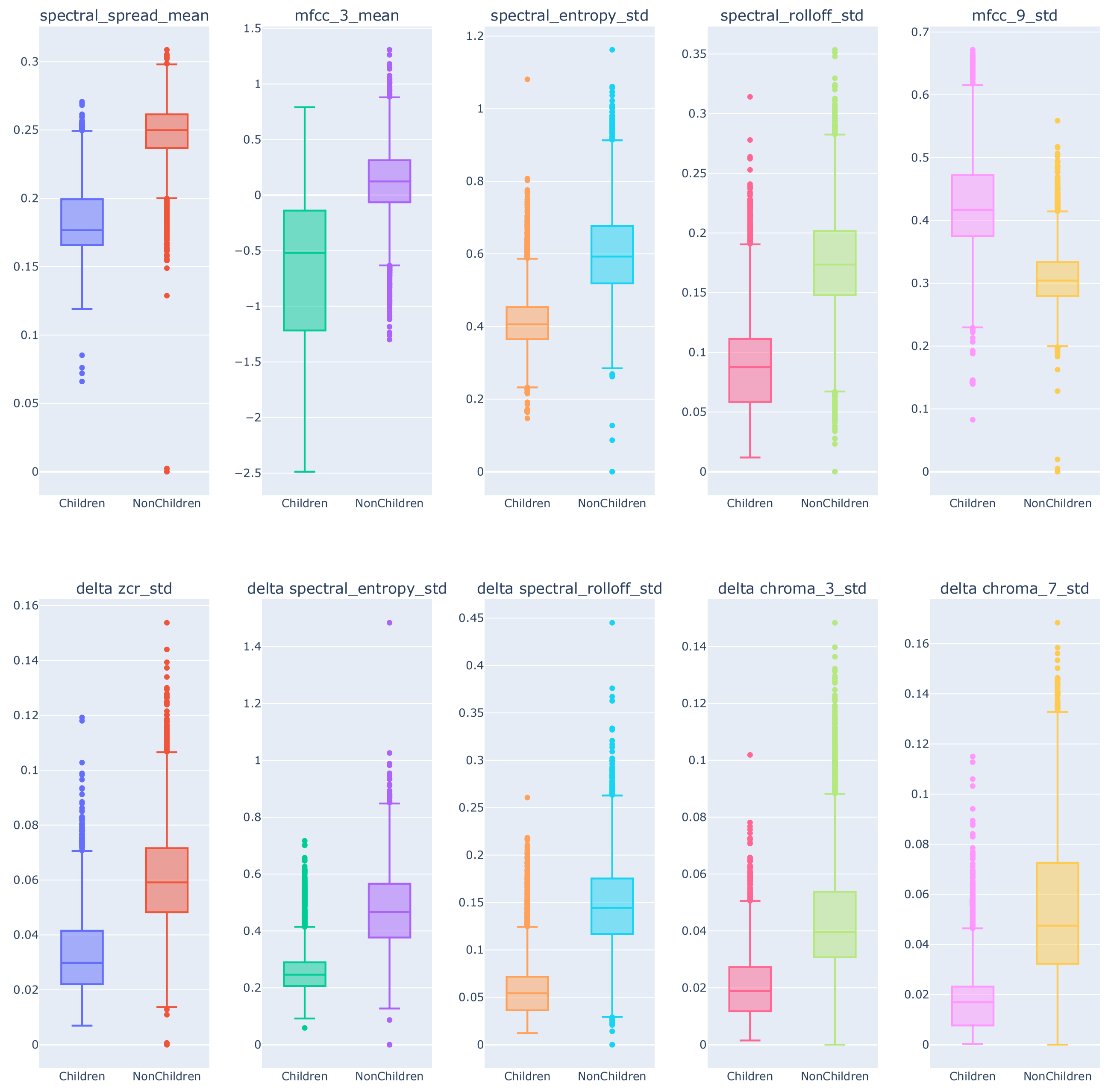


Figure 29: Box plot of the 10 most important features for genre classification: Children or Non-children music.

(MFCCs), which capture intricate timbral and textural nuances pivotal for differentiating energy levels.

Concerning music valence classification, the feature set partially overlaps with that of music energy, emphasising the interconnections of these emotional dimensions. The Mean of Energy Entropy and Mean of Spectral Spread, along with the Standard Deviations of Spectral Entropy and Spectral Roll-off, are retained for their capacity to depict the spectral landscape and energy distribution. Unique to this task are the Delta Standard Deviations of Spectral Centroid and the 11th and 13th MFCCs, which capture timbral information and spectral shifts crucial for the assessment of valence. These results can be evaluated in the box plots in Figure 31.

5.5 Machine Learning models results

5.5.1 Support Vector Machine results

In this study, the utilities of SVM were explored to establish a baseline algorithm for MER task, focusing on two key emotional dimensions: arousal and valence. The initial step involved employing SVM without parameter optimization to establish a foundational performance benchmark. Recognizing the importance of model tuning, a grid search technique was applied to ascertain the optimal regularization parameter C, which resulted in a selection of $C = 5$ for both arousal and valence recognition tasks. This parameter tuning is crucial as it balances the trade-off between model complexity and the degree of deviations for individual data points.

For a more granular understanding of model performance, predictions on the test dataset were evaluated through Confusion Matrices. The matrices for arousal and valence (Tables 8 and 9, respectively) offer insights into the model's predictive capabilities across different emotional intensities. For arousal, the model showed a tendency to confuse neutral arousal with weak, highlighting potential areas for model refinement. In contrast, SVM miss-classified a significant percentage of positive valence songs as neutral. Moreover, a considerable proportion of neutral samples classified as negative.

Table 8: Confusion Matrix of SVM in Music Arousal Recognition.

	Predicted Label		
Actual Label	*Weak*	*Neutral*	*Strong*
Weak	69	9	0
Neutral	23	50	5
Strong	3	9	5

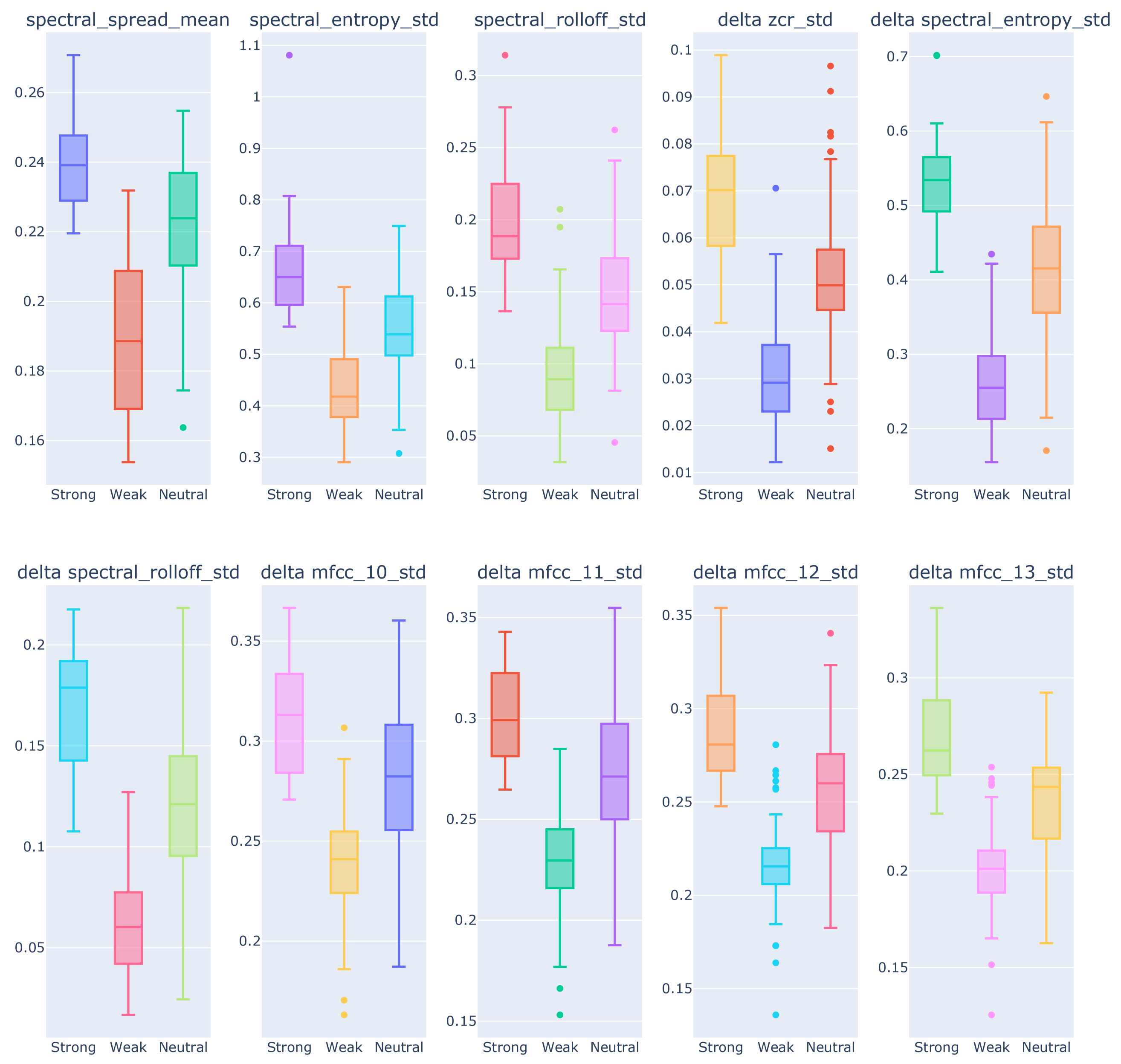


Figure 30: Box plot of the 10 most important features for children music emotion classification: Music energy/arousal.

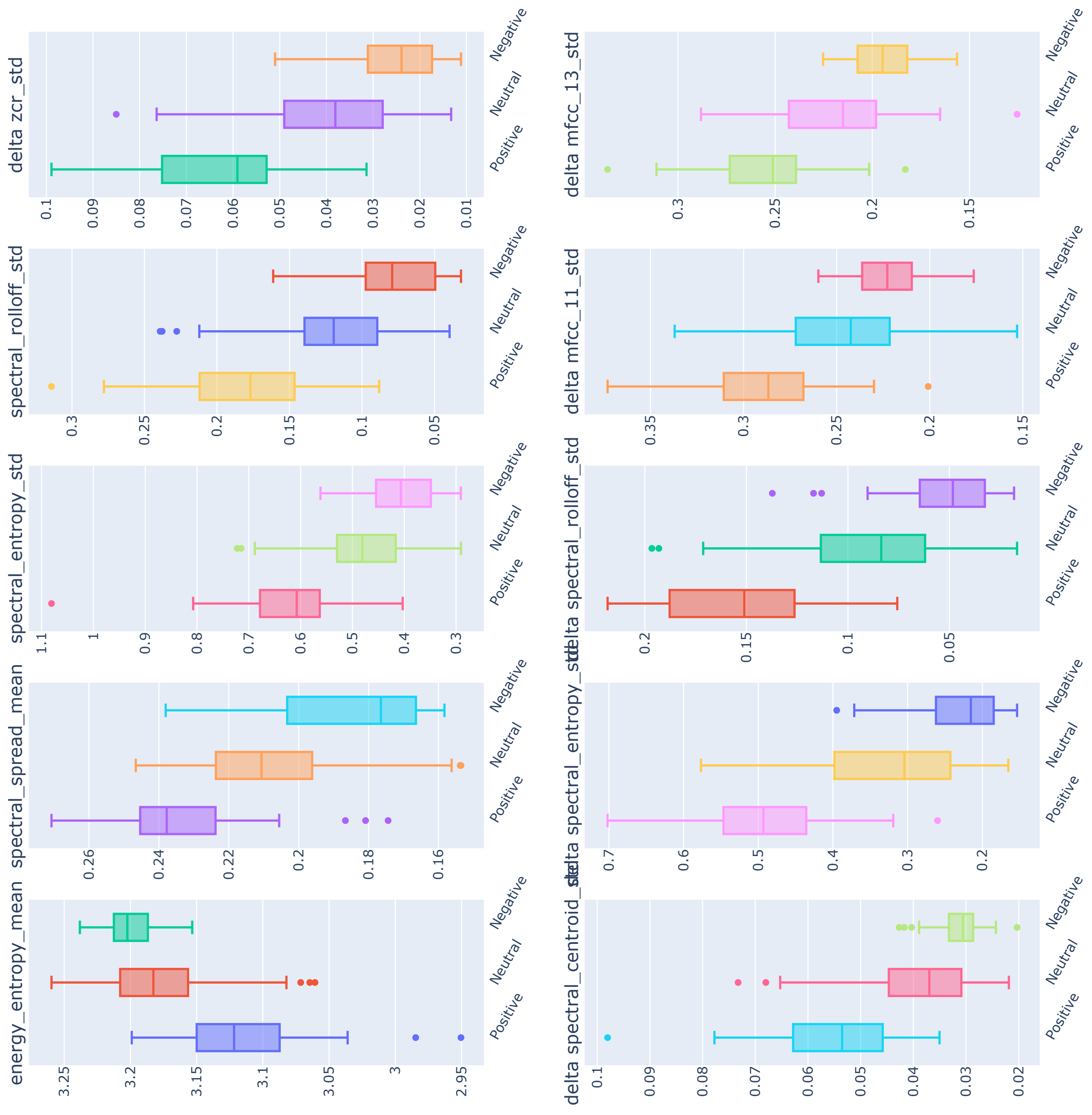


Figure 31: Box plot of the 10 most important features for children music emotion classification: Music valence.

Table 9: Confusion Matrix of SVM in Music Valence Recognition.

	Predicted Label		
Actual Label	*Negative*	*Neutral*	*Positive*
Negative	31	10	0
Neutral	23	35	4
Positive	0	33	31

The strengths and weaknesses of the model were further elucidated by class-wise performance metrics. In the domain of music arousal recognition, high precision, recall, and F1-score for the weak class were observed, indicating the model's proficiency in identifying less intense emotional states. Conversely, lower scores were recorded for the strong class, highlighting a potential area for enhancement in recognizing high arousal levels, as indicated in Table 10. With regard to valence recognition, the challenges outlined in the analysis of Confusion Matrices were corroborated by the class-wise performance metrics in Table 11. Although a high precision rate for the positive class was achieved, a low recall was observed in the recognition of this class. The recall and F1 score for the negative class suggest that improvements can be made for the model to consistently identify this specific class.

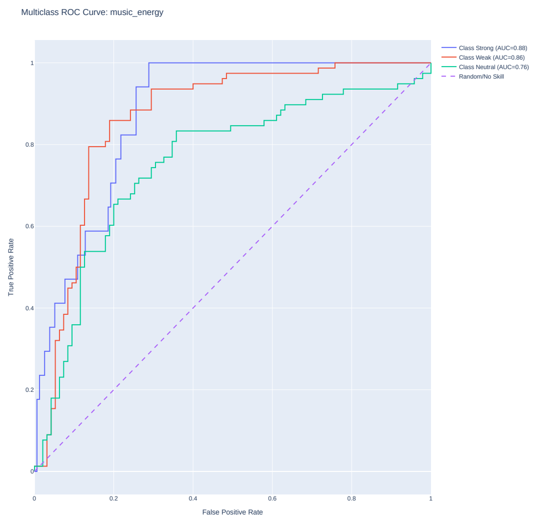
Table 10: Class-wise Performance Metrics for SVM in Music Arousal Recognition.

Class	**Precision**	**Recall**	**F1-Score**
Weak	0.726	0.885	0.798
Neutral	0.735	0.641	0.685
Strong	0.5	0.294	0.370

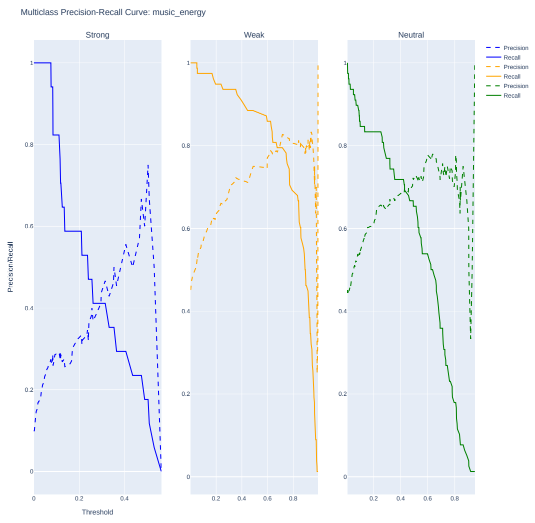
Table 11: Class-wise Performance Metrics for SVM in Music Valence Recognition.

Class	**Precision**	**Recall**	**F1-Score**
Negative	0.574	0.756	0.653
Neutral	0.449	0.565	0.500
Positive	0.886	0.484	0.626

For the task of music emotion recognition with respect to arousal, the analysis of the ROC-AUC curve reveals varying degrees of performance across different classes. More specifically, AUC scores of 0.88, 0.87, and 0.76 were observed for the strong, weak, and neutral classes, respectively. These scores indicate a generally good ability of the classifier to distinguish between classes, especially for weak arousal levels, while its performance on the neutral class is notably lower. Conversely, the analysis of Precision-Recall curves indicates a need for improvement in the recognition of strong arousal levels. Recalibration of the decision threshold could potentially enhance the classifier's performance metrics and ensuring a more balanced trade-off between sensitivity and specificity. Both sets of curves are presented in Figure 32.



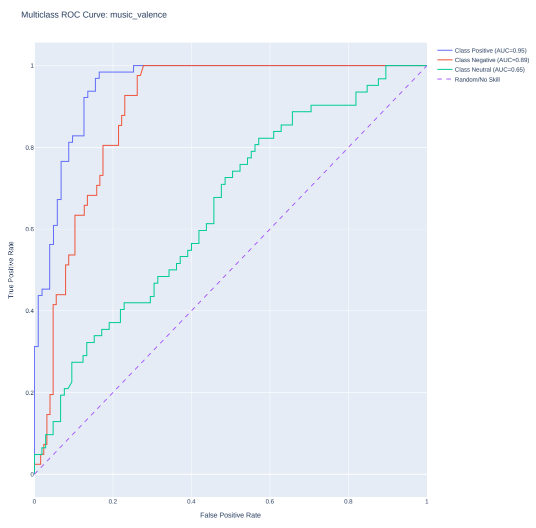
(a) ROC Curve



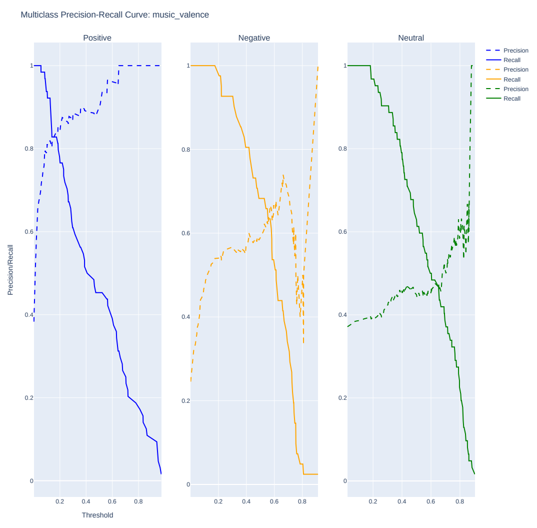
(b) Precision/Recall vs Threshold

Figure 32: Evaluation curves for SVM in Music Arousal Recognition.

In the evaluation of valence, as illustrated by the curves in Figure 33, it is shown that the classifier possesses a pronounced ability to differentiate between classes, with AUC values recorded at 0.95 for positive, 0.89 for negative, and 0.65 for neutral. The high AUC value for positive valence indicates an excellent capability of the classifier in identifying positive emotions in music, which significantly surpasses its performance in identifying neutral emotions, the latter being the area with the lowest performance. This conclusion is further supported by the analysis of Precision-Recall curves, particularly for the neutral class.



(a) ROC Curve



(b) Precision/Recall vs Threshold

Figure 33: Evaluation curves for SVM in Music Valence Recognition.

The overall performance metrics, summarized in Table 12, reveal an accuracy of 71.7% and an F1-score of 61.8% for the arousal task, and 58.1% accuracy with a 59.3% F1-score for valence. Moreover, statistical significance of the SVM model against a random classifier were calculated using permutation tests. The results demonstrate a marked superiority of the SVM model over the random classifier, with p-values of 0.0001 for both music arousal and valence tasks. These results affirm the potential of SVM in music emotion recognition but also underscore the necessity for further optimizations to enhance its predictive accuracy, especially in distinguishing between nuanced emotional states.

Table 12: Overall performance of SVM classifier.

Task	**F1**	**Accuracy**	**AUC**	**Precision**	**Recall**	**Random Classifier**	**Random Classifier F1**	**Statistical Significance** p_{value}
Arousal	0.618	0.717	0.812	0.654	0.607	0.44	0.38	0.0001
Valence	0.593	0.581	0.838	0.636	0.602	0.32	0.30	0.0001

5.5.2 Convolutional Neural Networks Results

Model selection process

In an effort to ascertain the most efficacious model for the MER task, a meticulous cross-validation strategy was implemented. The analysis encompassed 30 unique combinations of hyperparameters along with various numbers of non-trainable (frozen) Convolutional layers. In this section, merely the configurations demonstrating superior performance in terms of arousal and valence within musical contexts are delineated. The exhaustive outcomes derived from the grid search cross-validation for both aforementioned dimensions are systematically catalogued in Appendix A (Tables 30 and 31). With respect to arousal recognition within music, the models selected achieved notable results in the selected evaluation metrics. The results from the cross-validation of these models are delineated in Table 13. Following a strict evaluative process, Model 'S4', which was configured with a batch size of 64 and a learning rate of 0.002, was adjudged as the most optimal. This determination was based on its consistent superiority across a spectrum of metrics, encompassing average segment accuracy, segment F1 score, average track accuracy, and the range of confidence interval for the F1 score. Model fine-tuning configurations and their naming convention are reported in Section 3.2.

Table 13: Cross validation results of CNN for Music Arousal

Model Configuration	Mean Segment Accuracy	Segment Accuracy Std	Mean Segment F1	Segment F1 Std	Lower Confidence Interval for Segment F1	Upper Confidence Interval for Segment F1	Mean Track Accuracy	Track Accuracy Std	Mean Track F1	Track F1 Std	Lower Confidence Interval for Track F1	Upper Confidence Interval for Track F1	Mean Epoch
S4 Batch size: 64 LR: 0.002	0.763	0.024	0.732	0.033	0.666	0.798	0.836	0.029	0.808	0.037	0.734	0.882	23
S0 Batch size: 32 LR: 0.002	0.774	0.021	0.756	0.041	0.674	0.838	0.847	0.034	0.829	0.045	0.739	0.919	24
S2 Batch size: 16 LR: 0.002	0.799	0.009	0.757	0.024	0.709	0.805	0.847	0.034	0.815	0.054	0.707	0.923	24
S3 Batch size: 16 LR: 0.002	0.799	0.015	0.77	0.028	0.714	0.826	0.831	0.063	0.813	0.089	0.635	0.991	22

Similarly, the Valence recognition task was subjected to a rigorous model selection process. Table 14 provides a summary of the top-performing cross-validation outcomes. Model 'S0', with a batch size of 64 and a learning rate of 0.002, was elected as the optimal configuration due to its high mean segment accuracy and F1 score, coupled with a notable mean track accuracy and low standard deviation of track F1, indicating a robust capacity for valence prediction. In both tasks, the models selected not only excelled in their predictive capabilities but also demonstrated stability and reliability, crucial for the accurate classification of emotional content in children's music.

Table 14: Cross validation results of CNN for Music Valence

Model Configuration	**Mean Segment Accuracy**	**Segment Accuracy Std**	**Mean Segment F1**	**Segment F1 Std**	**Lower Confidence Interval for Segment F1**	**Upper Confidence Interval for Segment F1**	**Mean Track Accuracy**	**Track Accuracy Std**	**Mean Track F1**	**Track F1 Std**	**Lower Confidence Interval for Track F1**	**Upper Confidence Interval for Track F1**	**Mean Epoch**
S0 Batch size: 64 LR: 0.002	0.747	0.029	0.779	0.025	0.729	0.829	0.846	0.03	0.836	0.02	0.796	0.876	22
S2 Batch size: 16 LR: 0.002	0.761	0.026	0.778	0.031	0.716	0.84	0.809	0.041	0.803	0.038	0.727	0.879	22
S2 Batch size: 64 LR: 0.001	0.736	0.024	0.771	0.017	0.737	0.805	0.804	0.034	0.803	0.032	0.739	0.867	26
S3 Batch size: 64 LR: 0.002	0.74	0.031	0.773	0.027	0.719	0.827	0.827	0.05	0.825	0.045	0.735	0.915	29

Model Testing and Performance Evaluation

After the selection of the optimal model configurations, a comprehensive test was conducted to assess the performance of the CNN classifier in MER, focusing on arousal and valence dimensions. This section details the evaluation methodology, per-class metrics, overall performance, and analysis of ROC-AUC and Precision-Recall curves. The performance assessment of the CNN model necessitated a comparison of actual labels against the model's predictions using confusion matrices for both arousal and valence tasks, as depicted in Tables 15 and 16. As far as music Arousal recognition is concerned, the model's varied performance across the different energy levels, with a particularly strong performance in identifying weak arousal levels. Nevertheless, FN and FP values for neutral class can be improved, as a significant number of neutral song samples were classified as weak. Regarding Valence task, model is particularly accurate in predicting positive valence, with a high number of true positives (47) and few misclassifications. Moreover, model manages to correctly identify a number of negative instances, though there's some confusion with the neutral category. Neutral class was the most challenging for the classifier, with a relatively higher number of instances being misclassified as negative.

Table 15: Confusion Matrix of CNN in Music Arousal Recognition.

	Predicted Label		
Actual Label	*Weak*	*Neutral*	*Strong*
Weak	70	7	1
Neutral	21	51	6
Strong	0	4	13

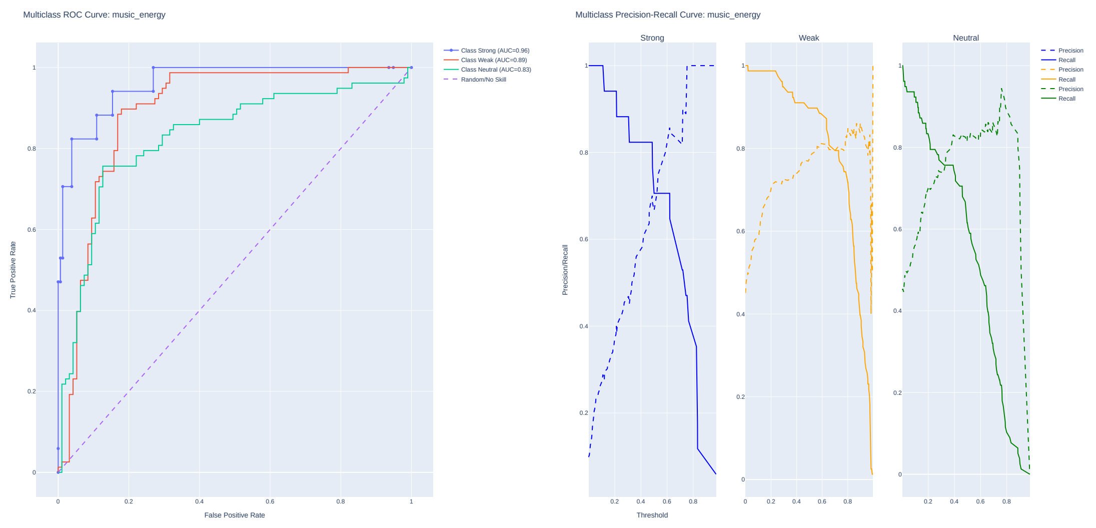
Table 16: Confusion Matrix of CNN in Music Valence Recognition.

	Predicted Label		
Actual Label	*Negative*	*Neutral*	*Positive*
Negative	29	12	0
Neutral	22	34	6
Positive	0	17	47

The class-wise metrics provide insightful observations regarding the CNN model's capability to identify and classify different emotional states within music tracks. As indicated in Table 17, model performance in identifying emotional states within music tracks varied across arousal categories. Notably, it excelled in identifying tracks with weak arousal, demonstrating a precision of 0.769, a recall of 0.897, and an F1-score of 0.828. Conversely, for tracks with strong arousal, while recall was high at 0.765, precision was comparatively lower at 0.65, resulting in an F1-score of 0.703. The model achieved the highest precision for neutral tracks, with an F1-score of 0.729. The evaluation of discrimination ability is comprehensively detailed through the analysis of ROC-AUC and Precision-Recall curves, as visualized in Figure 34. ROC curves revealed excellent discrimination between classes, particularly for strong arousal (AUC = 0.96). Precision-recall curves indicated the model's effectiveness in maintaining high precision as recall increased, especially for strong and weak categories. These findings highlight the model's capability in identifying emotional states within music tracks, with room for improvement in certain areas.

Table 17: Class-wise Performance for CNN in Music Arousal Recognition

	Precision	**Recall**	**F1**
Weak	0.769	0.897	0.828
Neutral	0.823	0.654	0.729
Strong	0.650	0.765	0.703



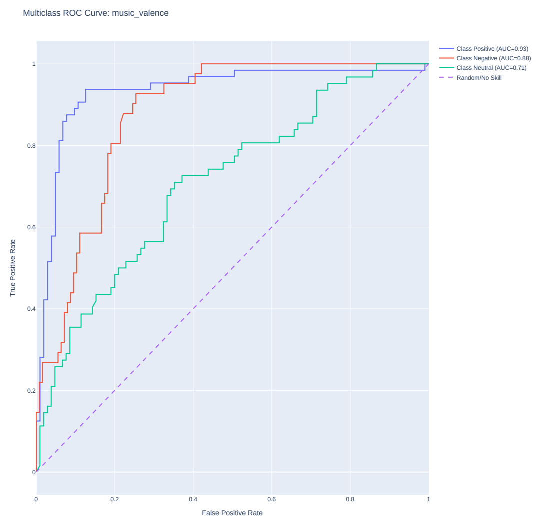
(a) ROC Curve (b) Precision/Recall vs Threshold

Figure 34: Evaluation curves of CNN for Music arousal.

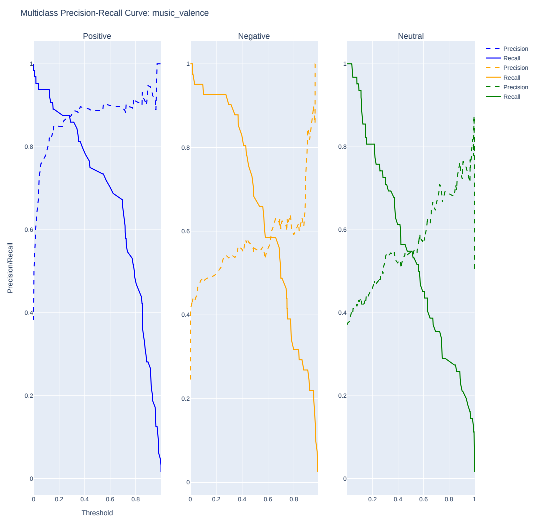
The CNN model's performance in identifying emotional valence within music tracks varied across classes, as observed in Table 18. The 'Positive' class stood out with high precision (0.887) and an F1-score of 0.803, indicating the model's proficiency in identifying positive tracks. Conversely, the 'Negative' and 'Neutral' classes exhibited moderate performance, with F1-scores of 0.63 and 0.544, respectively, suggesting challenges in distinguishing between negative and neutral valences. Evaluation through ROC curves revealed AUC values of 0.93, 0.88, and 0.71 for the 'Positive', 'Negative', and 'Neutral' classes, respectively, indicating the model's effectiveness in distinguishing emotional expressions in music. Precision-recall curves similarly demonstrated high precision and recall for 'Positive' and 'Negative' classes, affirming the model's capability in accurately classifying tracks with clear emotional valence. However, performance on the 'Neutral' class underscored difficulties in precisely identifying music with a neutral emotional tone, reflecting the challenge in interpreting nuanced emotional content. Evaluation curves for Valence recognition are presented in Figure 35.

Table 18: Per Class Metrics for Music Valence.

	Precision	**Recall**	**F1**
Negative	0.569	0.707	0.630
Neutral	0.540	0.548	0.544
Positive	0.887	0.734	0.803



(a) ROC Curve



(b) Precision/Recall vs Threshold

Figure 35: Evaluation curves of CNN for Music valence.

The aggregate performance metrics, as presented in Table 19, offer a holistic view of the model's efficacy. For the arousal task, the model achieves an AUC of 0.894, an F1-score of 0.753, and an accuracy of 0.775, underscoring its robust capability to differentiate between arousal states in music. The precision and recall values of 0.747 and 0.772, respectively, further attest to the model's balanced performance in correctly identifying and classifying arousal levels. In contrast, the valence task reveals a slightly lower performance, with an F1-score of 0.659 and an accuracy of 0.659. The precision and recall both stand at 0.665 and 0.663, respectively, accompanied by an AUC of 0.841. These metrics suggest that while the model is effective in recognizing valence in music, there is a notable margin for enhancement, particularly in improving the balance between precision and recall. Both models achieved remarkable performance in comparison with the Random Classifier. Moreover, their performance stood out significantly when set against the SVM, with their respective p-values (0.0001 for Arousal and 0.045 for Valence) underscoring their statistical significance.

Table 19: Overall performance of CNN model.

Task	**F1**	**Accuracy**	**AUC**	**Precision**	**Recall**	**Random Classifier Accuracy**	**Random Classifier F1**	**Statistical Significance SVM** p_{value}
Arousal	0.753	0.775	0.894	0.747	0.772	0.44	0.38	0.0001
Valence	0.659	0.659	0.841	0.665	0.663	0.32	0.30	0.045

5.5.3 Dual-stream model

Model selection process

The initial phase in the development of the Dual-stream model entailed the execution of a grid search cross-validation procedure, specifically tailored to optimize the architecture of the Sequential component and to obtain the optimal set of hyperparameters. This optimization process was guided by an evaluation of varying configurations of attention heads, enumerated as 2, 4, 8, and 16, alongside variations in batch sizes and learning rates to identify the architecture that yields the highest performance. The validation results of this process, presenting the top four model configurations based on their performance, are detailed in Tables 20 and 21. For an exhaustive overview of all model configurations explored during this phase, refer to Appendix A (Tables 32 and 33), which provides a complete compilation of the grid search cross-validation results.

For the Music Arousal task, the optimal model configuration was identified with 16 attention heads, a batch size of 32 and a learning rate of 0.002. This configuration exhibited superior performance, characterized by a Mean Segment Accuracy of 0.771, Mean Segment F1 of 0.746, and Mean Track F1 of 0.839. These metrics not only underscore the model's proficiency in identifying arousal-related emotional cues within music but also its precision and reliability at both the segment and track levels. Additionally, the standard deviations of these metrics, along with the lower and upper bounds for the F1 score's confidence interval, underscore the stability and dependability of the selected classifier.

Table 20: Cross validation results of Dual-Stream model for Music Arousal Recognition.

Model Configuration	Mean Segment Accuracy	Segment Accuracy Std	Mean Segment F1	Segment F1 Std	Lower Confidence Interval for Segment F1	Upper Confidence Interval for Segment F1	Mean Track Accuracy	Track Accuracy Std	Mean Track F1	Track F1 Std	Lower Confidence Interval for Track F1	Upper Confidence Interval for Track F1	Mean Epoch
Heads: 8 Batch size: 32 LR: 0.001	0.767	0.034	0.739	0.047	0.645	0.833	0.836	0.05	0.813	0.053	0.707	0.919	29
Heads: 16 Batch size: 32 LR: 0.002	0.771	0.029	0.746	0.033	0.68	0.812	0.862	0.034	0.839	0.039	0.761	0.917	24
Heads: 8 Batch size: 16 LR: 0.002	0.795	0.027	0.753	0.035	0.683	0.823	0.831	0.047	0.779	0.083	0.613	0.945	20
Heads: 2 Batch size: 16 LR: 0.002	0.795	0.025	0.755	0.032	0.691	0.819	0.857	0.047	0.808	0.091	0.626	0.99	28

Conversely, for the music Valence recognition task, a parallel configuration was employed, solidifying the selection of 2 attention heads, a batch size of 16 and a learning rate of 0.002. This model configuration produced commendable results, as evidenced by a Mean Segment Accuracy of 0.77, a Mean Segment F1 of 0.781, and a Mean Track F1 of 0.791. The achievements in model performance thereby substantiate the rigor of the architectural and optimization decisions made during the research process. Despite the fact that other model achieved higher F1 score in both segment and track levels, selected model achieved the lowest Track F1 standard deviation and the highest Lower confidence intervals of this metric, indicating model stability and generalization. The integration of optimal fine-tuning scenarios—S4 for arousal and S0 for valence—within the CNN part of the Dual-stream framework not only elevates the model's performance but also contributes to the broader discourse on computational approaches to understanding emotional expressions in music.

Table 21: Cross validation results of Dual-Stream model for Music Valence Recognition.

Model Configuration	Mean Segment Accuracy	Segment Accuracy Std	Mean Segment F1	Segment F1 Std	Lower Confidence Interval for Segment F1	Upper Confidence Interval for Segment F1	Mean Track Accuracy	Track Accuracy Std	Mean Track F1	Track F1 Std	Lower Confidence Interval for Track F1	Upper Confidence Interval for Track F1	Mean Epoch
Heads: 16 Batch size: 16 LR: 0.002	0.767	0.022	0.781	0.027	0.727	0.835	0.785	0.052	0.778	0.048	0.682	0.874	28
Heads: 8 Batch size: 16 LR: 0.001	0.773	0.031	0.788	0.033	0.722	0.854	0.804	0.044	0.8	0.044	0.712	0.888	25
Heads: 2 Batch size: 32 LR: 0.001	0.763	0.032	0.789	0.036	0.717	0.861	0.827	0.053	0.821	0.049	0.723	0.919	22
Heads: 2 Batch size: 16 LR: 0.002	0.77	0.037	0.781	0.038	0.705	0.857	0.804	0.034	0.791	0.022	0.747	0.835	32

Model Testing and Performance Evaluation

Upon evaluating the Dual-Stream Model's performance on the test set, the confusion matrices for both music Arousal and music Valence tasks reveal insightful details about the model's predictive accuracy and its nuanced understanding of emotional expressions in music. As far as, music Arousal recognition is concerned, the confusion matrix (Table 22) highlights the model's robust ability to classify weak energy levels accurately, with 71 true positives out of 78 predictions. This signifies the model's precision in identifying lower energy levels within music tracks. However, the classification of strong and neutral energy levels presents a challenge, as evidenced by a substantial number of neutral labels being misclassified as weak. Despite these challenges, the model demonstrates a commendable proficiency in distinguishing between the various energy levels.

Table 22: Confusion Matrix of Dual-Stream model in Music Arousal Recognition.

	Predicted Label		
Actual Label	*Weak*	*Neutral*	*Strong*
Weak	71	7	0
Neutral	23	54	1
Strong	0	6	11

In contrast, the Music Valence task showcases a different aspect of the model's capability (Table 23). The model exhibits a strong performance in correctly predicting positive valence, with 52 out of 64 instances accurately classified. This underscores the model's effectiveness in recognizing the positive emotional spectrum in music. However, the neutral valence category, with a significant portion of misclassifications, points to the inherent difficulty in distinguishing neutral from negative, reflecting the complex and subjective nature of music valence perception.

Table 23: Confusion Matrix of Dual-Stream model in Music Valence Recognition.

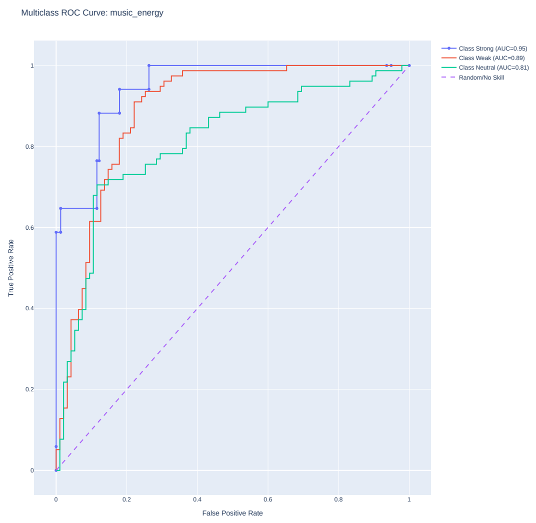
	Predicted Label		
Actual Label	*Negative*	*Neutral*	*Positive*
Negative	32	9	0
Neutral	17	37	8
Positive	0	12	52

The class-wise metrics and evaluation curves collectively articulate the Dual-Stream Model's adeptness in discerning music arousal levels, as delineated in Table 24 and Figure 36. The model exhibits remarkable precision, 0.917, and an F1 score of 0.759 for strong arousal, affirming its capability to accurately identify high-energy tracks. This proficiency extends to weak arousal, where an F1 score of 0.826 underscores effective discrimination of lower-energy tracks. The neutral category, despite a commendable F1 score of 0.745, reveals an area for enhancement in precision akin to the extreme arousal states.

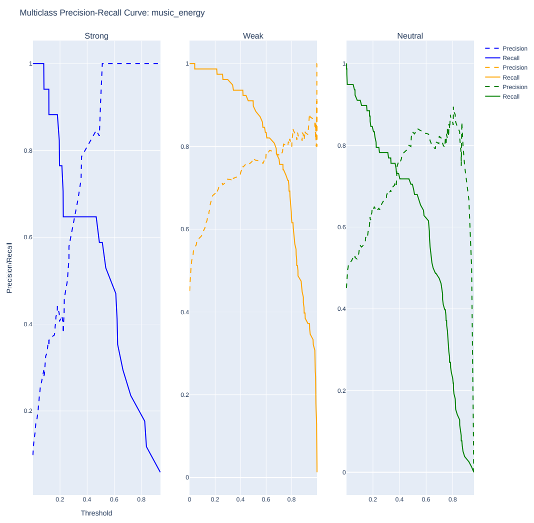
The ROC and Precision-Recall Curves further highlight the model's discrimination capability, with AUC values of 0.95, 0.89, and 0.81 for strong, weak, and neutral arousal levels, respectively. These metrics showcase the model's confidence, particularly in distinguishing strong arousal tracks. The performance on neutral arousal, while slightly lesser, still demonstrates robust differentiation between arousal levels. The model's efficacy is particularly evident in maintaining high precision across increasing recall levels for strong and weak categories, suggesting a well-balanced approach in handling arousal distinctions in music. This comprehensive analysis not only affirms the model's current strengths in classifying arousal levels but also points towards potential refinements for enhancing its granularity in arousal perception.

Table 24: Class-wise Performance of Dual-Stream model in Music Arousal Recognition

	Precision	**Recall**	**F1**
Weak	0.755	0.910	0.826
Neutral	0.806	0.692	0.745
Strong	0.917	0.647	0.759



(a) ROC Curve



(b) Precision/Recall vs Threshold

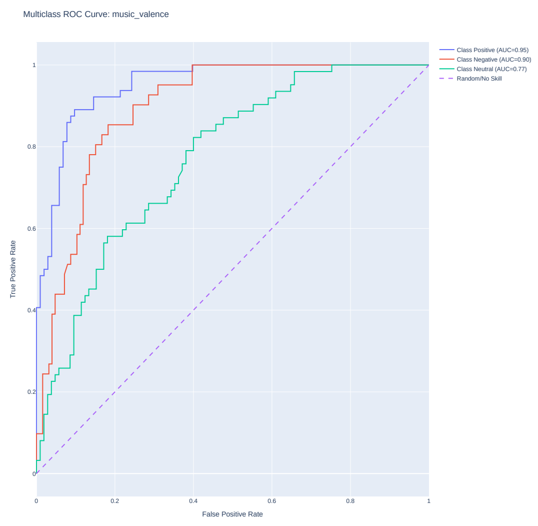
Figure 36: Evaluation curves of Dual-stream model for Music Arousal.

The Dual-Stream Model's nuanced performance on the music valence task is comprehensively depicted through class-specific metrics and evaluation curves, highlighting its strengths and areas for improvement in discerning emotional valence within music. Detailed in Table 25, the model excels in recognizing positive valence, achieving an impressive F1 score of 0.839, supported by high precision. Regarding negative valence, an F1 score of 0.711 illustrates the model's moderate ability to differentiate negative emotional content. Conversely, the neutral category poses a challenge, evidenced by a lower F1 score of 0.617, signaling a critical need for refinement in classifying neutrally toned tracks.

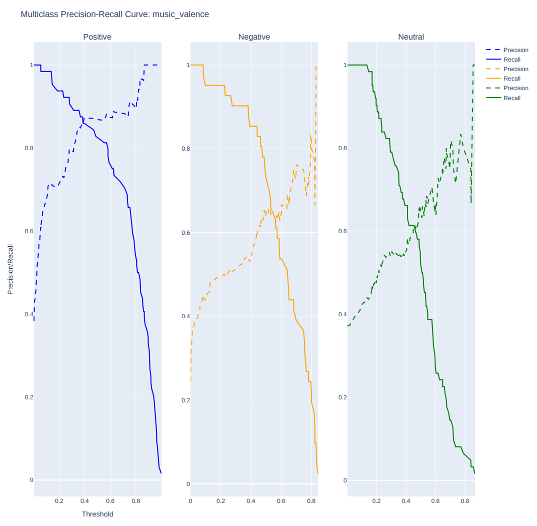
Evaluation curves, as depicted in Figure 37, reinforce these findings. ROC curve analysis demonstrates the model's classification capabilities with AUC values of 0.95 for positive and 0.89 for negative valences, indicating a robust capacity to distinguish these emotional expressions. However, the neutral class's AUC of 0.73 reveals a struggle in accurately classifying such an emotional content, underscoring the necessity for enhancing model sensitivity. The Precision-Recall curve analysis depicts the model's proficiency with positive and negative classes but also accentuates the difficulty in precisely identifying neutral valence, spotlighting the intricate challenges.

Table 25: Class-wise Performance for Dual-Stream model in Music Valence Recognition

	Precision	**Recall**	**F1**
Negative	0.653	0.780	0.711
Neutral	0.638	0.597	0.617
Positive	0.867	0.812	0.839



(a) ROC Curve



(b) Precision/Recall vs Threshold

Figure 37: Evaluation curves of Dual-stream model for Music Valence.

The Dual-Stream Model's overall performance, as encapsulated in Table 26, underscores its efficacy and competitive edge in the domain of MER. Across both arousal and valence tasks, the model not only demonstrates commendable accuracy, precision, and recall but also surpasses the benchmarks set by a random classifier, as evidenced by F1 scores significantly higher than those of a chance-level baseline. Specifically, for the arousal task, the model achieves an AUC of 0.868 and an F1 score of 0.776, indicating a high degree of accuracy in distinguishing between different levels of musical energy. Similarly, in the valence task, the model records an AUC of 0.87 and an F1 score of 0.722, showcasing its ability to discern the nuanced spectrums of emotional valence in music with substantial reliability.

Notably, the model's performance is statistically significant when compared against the CNN model, as reflected by the p-values (0.29 for arousal and 0.013 for valence), affirming its superior predictive capabilities. While the comparison with CNN models, especially in the arousal task, indicates room for improvement, the overall metrics present a compelling case for the Dual-Stream Model's advanced comprehension of complex emotional expressions in music.

Table 26: Overall performance of Dual-Stream model.

Task	**F1**	**Accuracy**	**AUC**	**Precision**	**Recall**	**Random Classifier Accuracy**	**Random Classifier F1**	**Statistical Significance CNN** p_{value}
Arousal	0.776	0.786	0.868	0.826	0.75	0.44	0.38	0.296
Valence	0.722	0.725	0.87	0.719	0.73	0.32	0.30	0.0133

5.5.4 Comparisons, ablation study and emotion alignment

In the evaluation of emotional recognition models on the PMEmo dataset [5], distinct approaches have been benchmarked to ascertain their efficacy in predicting arousal and valence metrics. The proposed model, i.e., the CNN approach, with statistical aggregation on 10-second segments showcased superior performance, achieving the highest recorded accuracies of 87.2% for arousal and 87.8% for valence. The F1 scores of this model were also impressive at 86.5 and 87.1 respectively, indicating a robust ability to balance precision and recall effectively. Comparative analysis with other methodologies, such as the SVM approach by Yin et al. [95] and the CNN combined with a BiLSTM by He et al. [96], demonstrates that the proposed model not only excels in accuracy but also maintains high F1 scores. The CNN-BiLSTM configuration by He et al. achieved notable scores, particularly an F1 score of 86.52 for arousal, yet the proposed model's holistic performance remains unmatched in both metrics under study. Comparisons are presented in Table 27.

Table 27: Comparisons with other model using *PMEmo* Dataset [5].

			Arousal		**Valence**	
Method	**Model**	**Type**	**Accuracy**	**F1**	**Accuracy**	**F1**
Yin et al. [95]	SVM	Audio features on Full-tracks	71.49	76.36	70.43	75.32
He et al. [96]	CNN + BiLSTM	10 sec segments + Deep aggregation	82.79	85.17	77.44	81.91
He et al. [96]	CNN-based autoencoder + BiLSTM	10 sec segments + Deep aggregation	83.62	**86.52**	79.01	83.2
Proposed model	CNN	10 sec segments +Statistical aggregation	**87.2**	86.5	**87.8**	**87.1**

The ablation study, as summarized in Table 28, was meticulously designed to evaluate the performance of various models in the tasks of music arousal and valence recognition. For the arousal task, the Dual-Stream model demonstrated superior performance across most metrics, with an F1 score of 0.776, accuracy of 0.786, and a precision of 0.826, although its AUC of 0.868 was slightly lower than that achieved by the CNN model (0.894). In contrast, the Attention-based LSTM model exhibited lower metrics in comparison, notably in F1 score and accuracy, underscoring the enhanced efficacy of the Dual-Stream model in capturing the complexity of music arousal.

Similarly, in the music valence task, the Dual-Stream model outperformed the CNN and Attention-based LSTM models, achieving the highest F1 score of 0.722, accuracy of 0.725, and an AUC of 0.870. The consistency in the performance of the Dual-Stream model across both tasks highlights its robustness and adaptability in recognizing nuanced emotional components in music. These results underline the importance of leveraging diverse architectures to enhance model performance in music emotion recognition tasks.

Regarding the emotion alignment of model predictions in music arousal, all models exhibited comparable performance. Notably, the Attention-based LSTM model slightly outperformed its counterparts in terms of this similarity metric. Nevertheless, its predictions were deemed insufficiently effective. The analysis reveals that the posterior distributions of all models align with the annotated distribution to a moderate extent. For the recognition of music valence, the Dual-Stream model demonstrated superior performance in emotion alignment, achieving an impressive similarity score of 0.789. This indicates that the Dual-Stream model is notably proficient at capturing valence-related emotional content in music, with its posterior distributions showing a high degree of alignment with the human annotations. This proficiency underscores the model's effectiveness in reflecting human perceptual evaluations of music emotion.

Table 28: Ablation Study Results for Music Arousal and Valence Recognition

Task	**Model**	**F1**	**Accuracy**	**AUC**	**Precision**	**Recall**	**Probabilistic Emotion Alignment**
Arousal	CNN	0.753	0.775	0.894	0.747	0.772	0.741
	Attention-based LSTM	0.648	0.717	0.834	0.660	0.653	0.744
	Dual-Stream	0.776	0.786	0.868	0.826	0.750	0.739
Valence	CNN	0.659	0.659	0.841	0.665	0.663	0.761
	Attention-based LSTM	0.656	0.665	0.824	0.654	0.676	0.775
	Dual-Stream	0.722	0.725	0.870	0.719	0.730	0.786

6 Conclusions

This study has made significant contributions to the domain of MER, with a specific focus on children's music. The primary objective was the development of a specialized music dataset tailored to young listeners, combined with the application of advanced DL algorithms for the analysis of emotional content. The secondary aim encompassed the identification of important features and modalities affecting emotional dimensions and categories. The research methodology employed in this work spans from the initial data collection phase to the implementation of DL architectures. A distinctive aspect of this approach lies in the creation of a unique dataset and its associated annotation strategy, incorporating insights derived from both child psychology and music theory. Furthermore, the adaptation and utilization of CNNs and LSTMs within a dual-stream model capable of handling both spectrograms and music transcription sequences highlights the innovative nature of this work. This approach not only enhanced the accuracy of emotion recognition but also afforded deeper insights into the nuanced patterns of emotional expression within children's music. The use of these diverse techniques represents a significant advancement in the application of AI to investigate cognitive and emotional processes in children.

Table 29: Aggregated results of Machine Learning models.

Task	**Class**	**SVM F1**	**Probabilistic Emotion Alignment SVM**	**CNN F1**	**Probabilistic Emotion Alignment CNN**	**Dual-Stream F1**	**Probabilistic Emotion Alignment Dual-Stream**
Arousal	Weak	0.798	-	0.828	-	0.826	-
	Neutral	0.685	-	0.729	-	0.745	-
	Strong	0.370	-	0.703	-	0.759	-
	Overall	0.618	0.724	0.753	0.741	0.776	0.739
Valence	Negative	0.653	-	0.630	-	0.711	-
	Neutral	0.500	-	0.544	-	0.617	-
	Positive	0.626	-	0.803	-	0.839	-
	Overall	0.593	0.763	0.659	0.761	0.722	0.786

The results summarized in Table 29 provide an overview of the F1 scores and Probabilistic Emotion Alignment scores attained by the SVM, CNN, and Dual-Stream architecture models across various scenarios. The Dual-Stream model, in particular, showcased superior performance in tasks related to both arousal and valence, affirming its robustness and capability in capturing the complex emotional content within music. The Probabilistic Emotion Alignment, offering an additional dimension of analysis, compares the distributions of annotations against the predictions made by models. In this regard, the Dual-Stream model exhibited significant alignment, particularly in the valence category, indicating its proficiency in discerning the nuanced emotional aspects of music samples.

The unimodal CNN and the CNN-LSTM models also demonstrated commendable performance in the recognition of musical emotions. However, the Dual-Stream architecture surpassed the CNN model's achievements. In the context of arousal classification, the CNN-LSTM model was noted for its enhanced precision and F1 scores across all emotional categories, highlighting its effectiveness in differentiating various energy levels in music. Although the CNN model showed competitive performance, it encountered more pronounced challenges within the neutral emotion category across both tasks of emotion recognition. Thus, the use of spectrograms and music transcription data was beneficial in improving the differentiation capabilities for arousal levels. Similarly, for valence recognition, the Dual-Stream Neural Network excelled over the CNN model across several metrics, including precision, recall, F1-score, AUC, and accuracy, showcasing a strong capacity to identify positive and negative valence in children's music, with areas for growth in the recognition of neutral valence.

Despite the advancements presented in this study, it is crucial to recognize the encountered challenges and limitations, particularly regarding dataset diversity. The complex nature of children's emotional reactions to music, varying across cultural contexts and developmental stages, necessitates further exploration into the models' generalization capabilities to new and varied musical compositions. These challenges offer opportunities for future research, stressing the importance of advanced data collection methods and the development of more sophisticated models to cover a wider array of emotional expressions and musical genres.

Future research avenues are vast, including the exploration of culturally diverse datasets to broaden the models' interpretability of emotional expressions across different cultural backgrounds. Investigating alternative DL architectures could improve model accuracy and generalization capabilities. Additionally, a deeper analysis of music's impact on various facets of child development, such as cognitive growth, emotional intelligence, and social skills, would contribute significantly to developmental psychology and musicology. Advancements in DL models, potentially developed through self-supervised techniques like Contrastive Learning and augmented by Explainable AI principles, could further the interpretability of these models, making their predictions and insights more accessible to a wider audience, including educators and therapists. Such approaches underscore the commitment to ethical AI use, laying the groundwork for technology's responsible application in sensitive domains like child development and education. The overarching aim is to leverage music's transformative power to enhance child development, utilizing AI's potential to deepen the understanding of the complex interplay between music and human emotions.

7 References

[1] W. Wang, J. Tian, C. Zhang, Y. Luo, X. Wang, and J. Li, "An improved deep learning approach and its applications on colonic polyp images detection," *BMC Medical Imaging*, vol. 20, pp. 1–14, 2020.

[2] C. Cheuque, M. Querales, R. León, R. Salas, and R. Torres, "An efficient multi-level convolutional neural network approach for white blood cells classification," *Diagnostics*, vol. 12, no. 2, p. 248, 2022.

[3] A. G. Roy, N. Navab, and C. Wachinger, "Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 540–549, 2018.

[4] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, p. e0144610, 2015.

[5] K. Zhang, H. Zhang, S. Li, C. Yang, and L. Sun, "The pmemo dataset for music emotion recognition," in *Proceedings of the 2018 acm on international conference on multimedia retrieval*, 2018, pp. 135–142.

[6] E. Papatzikis, C. Svec, and N. Tsakmakidou, "Studying neural correlates of music features in the early years education and development process: a preliminary understanding based on a taxonomical classification and logistic regression analysis," in *Proceedings of the Conference Abstract: 4th International Conference on Educational Neuroscience, Abu Dhabi, United Arab Emirates*, 2019, pp. 10–11.

[7] K. L. Hyde, J. Lerch, A. Norton, M. Forgeard, E. Winner, A. C. Evans, and G. Schlaug, "Musical training shapes structural brain development," *Journal of Neuroscience*, vol. 29, no. 10, pp. 3019–3025, 2009.

[8] D. Han, Y. Kong, J. Han, and G. Wang, "A survey of music emotion recognition," *Frontiers of Computer Science*, vol. 16, no. 6, p. 166335, 2022.

[9] R. Panda, R. M. Malheiro, and R. P. Paiva, "Audio features for music emotion recognition: a survey," *IEEE Transactions on Affective Computing*, 2020.

[10] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, pp. 335–359, 2008.

[11] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, “Towards musical query-by-semantic-description using the cal500 data set,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 439–446.

[12] Y. E. Kim, E. M. Schmidt, and L. Emelle, “Moodswings: A collaborative game for music mood label collection.” in *Ismir*, vol. 8, 2008, pp. 231–236.

[13] T. Eerola, “Are the emotions expressed in music genre-specific? an audio-based evaluation of datasets spanning classical, film, pop and mixed genres,” *Journal of New Music Research*, vol. 40, no. 4, pp. 349–366, 2011.

[14] Y.-A. Chen, Y.-H. Yang, J.-C. Wang, and H. Chen, “The amg1608 dataset for music emotion recognition,” in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 693–697.

[15] A. Aljanaki *et al.*, “Emotion in music: representation and computational modeling,” Ph.D. dissertation, Utrecht University, 2016.

[16] S. Koelstra, C. Muhl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras, “Deap: A database for emotion analysis; using physiological signals,” *IEEE transactions on affective computing*, vol. 3, no. 1, pp. 18–31, 2011.

[17] M. Pesek, G. Strle, A. Kavčič, and M. Marolt, “The moodo dataset: Integrating user context with emotional and color perception of music for affective music information retrieval,” *Journal of New Music Research*, vol. 46, no. 3, pp. 246–260, 2017.

[18] X. Hu and Y.-H. Yang, “Cross-dataset and cross-cultural music mood prediction: A case on western and chinese pop songs,” *IEEE Transactions on Affective Computing*, vol. 8, no. 2, pp. 228–240, 2017.

[19] R. Panda, R. Malheiro, and R. P. Paiva, “Novel audio features for music emotion recognition,” *IEEE Transactions on Affective Computing*, vol. 11, no. 4, pp. 614–626, 2018.

[20] A. Aljanaki, Y.-H. Yang, and M. Soleymani, “Developing a benchmark for emotional analysis of music,” *PloS one*, vol. 12, no. 3, p. e0173392, 2017.

[21] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y.-H. Yang, “Emopia: A multimodal pop piano dataset for emotion recognition and emotion-based music generation,” *arXiv preprint arXiv:2108.01374*, 2021.

[22] Y.-H. Yang, C.-C. Liu, and H. H. Chen, "Music emotion classification: A fuzzy approach," in *Proceedings of the 14th ACM international conference on Multimedia*, 2006, pp. 81–84.

[23] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *Proc. ismir*, vol. 86, 2010, pp. 937–952.

[24] T. Li and M. Ogihara, "Detecting emotion in music," 2003.

[25] C. Laurier, J. Grivolla, and P. Herrera, "Multimodal music mood classification using audio and lyrics," in *2008 seventh international conference on machine learning and applications*. IEEE, 2008, pp. 688–693.

[26] Y.-H. Yang, Y.-C. Lin, H.-T. Cheng, I.-B. Liao, Y.-C. Ho, and H. H. Chen, "Toward multi-modal music emotion classification," in *Advances in Multimedia Information Processing-PCM 2008: 9th Pacific Rim Conference on Multimedia, Tainan, Taiwan, December 9-13, 2008. Proceedings 9*. Springer, 2008, pp. 70–79.

[27] Y. Liu, Y. Liu, Y. Zhao, and K. A. Hua, "What strikes the strings of your heart?—feature mining for music emotion analysis," *IEEE TRANSACTIONS on Affective computing*, vol. 6, no. 3, pp. 247–260, 2015.

[28] J.-C. Wang, Y.-H. Yang, H.-M. Wang, and S.-K. Jeng, "The acoustic emotion gaussians model for emotion-based music annotation and retrieval," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 89–98.

[29] Y.-A. Chen, J.-C. Wang, Y.-H. Yang, and H. H. Chen, "Component tying for mixture model adaptation in personalization of music emotion recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1409–1420, 2017.

[30] Y.-A. Chen, J.-C. Wang, Y.-H. Yang, and H. Chen, "Linear regression-based adaptation of music emotion recognition models for personalization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 2149–2153.

[31] S. Fukayama and M. Goto, "Music emotion recognition with adaptive aggregation of gaussian process regressors," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 71–75.

[32] M. Soleymani, A. Aljanaki, Y.-H. Yang, M. N. Caro, F. Eyben, K. Markov, B. W. Schuller, R. Veltkamp, F. Weninger, and F. Wiering, "Emotional analysis of music: A comparison of methods," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1161–1164.

[33] X. Liu, Q. Chen, X. Wu, Y. Liu, and Y. Liu, “Cnn based music emotion classification,” *arXiv preprint arXiv:1704.05665*, 2017.

[34] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, “Towards explainable music emotion recognition: The route via mid-level features,” *arXiv preprint arXiv:1907.03572*, 2019.

[35] M. B. Er and I. B. Aydilek, “Music emotion recognition by using chroma spectrogram and deep visual features,” *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1622–1634, 2019.

[36] Y. Dong, X. Yang, X. Zhao, and J. Li, “Bidirectional convolutional recurrent sparse network (bcrsn): an efficient model for music emotion recognition,” *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3150–3163, 2019.

[37] N. He and S. Ferguson, “Multi-view neural networks for raw audio-based music emotion recognition,” in *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2020, pp. 168–172.

[38] S. Rajesh and N. Nalini, “Musical instrument emotion recognition using deep recurrent neural network,” *Procedia Computer Science*, vol. 167, pp. 16–25, 2020.

[39] R. Sarkar, S. Choudhury, S. Dutta, A. Roy, and S. K. Saha, “Recognition of emotion in music based on deep convolutional neural network,” *Multimedia Tools and Applications*, vol. 79, pp. 765–783, 2020.

[40] J. Yang, “A novel music emotion recognition model using neural network technology,” *Frontiers in Psychology*, vol. 12, p. 760060, 2021.

[41] S. Hizlisoy, S. Yildirim, and Z. Tufekci, “Music emotion recognition using convolutional long short term memory deep neural networks,” *Engineering Science and Technology, an International Journal*, vol. 24, no. 3, pp. 760–767, 2021.

[42] S. Gupta, “Deep audio embeddings and attention based music emotion recognition,” in *2023 15th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2023, pp. 357–362.

[43] K. Palanisamy, D. Singhania, and A. Yao, “Rethinking cnn models for audio classification,” *arXiv preprint arXiv:2007.11154*, 2020.

[44] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Transfer learning for music classification and regression tasks,” *arXiv preprint arXiv:1703.09179*, 2017.

[45] A. Geroulanos and T. Giannakopoulos, “Emotion recognition in music using deep neural networks,” in *Advances in Speech and Music Technology: Computational Aspects and Applications*. Springer, 2022, pp. 193–213.

[46] G. Sharma, K. Umapathy, and S. Krishnan, “Trends in audio signal feature extraction methods,” *Applied Acoustics*, vol. 158, p. 107020, 2020.

[47] T. Panch, P. Szolovits, and R. Atun, “Artificial intelligence, machine learning and health systems,” *Journal of global health*, vol. 8, no. 2, 2018.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[49] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[50] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[51] R. Duda, P. Hart, D. Stork, and A. Ionescu, “Pattern classification, chapter non-parametric techniques,” 2000.

[52] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” *Advances in neural information processing systems*, vol. 27, 2014.

[53] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[54] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.” *Journal of machine learning research*, vol. 12, no. 7, 2011.

[55] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.

[56] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[57] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.

[58] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*. PMLR, 2013, pp. 1139–1147.

[59] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[60] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[61] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

[62] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks*. Springer, 1995, pp. 195–201.

[63] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[64] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[65] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?" *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[66] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[67] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[71] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[72] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[73] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[74] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[76] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[77] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[78] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[79] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1d & 2d cnn lstm networks," *Biomedical signal processing and control*, vol. 47, pp. 312–323, 2019.

[80] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[81] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[82] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.

[83] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam, "Classification of audio signals using svm and rbfnn," *Expert systems with applications*, vol. 36, no. 3, pp. 6069–6075, 2009.

[84] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.

[85] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python." in *SciPy*, 2015, pp. 18–24.

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[87] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[88] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural networks*, vol. 11, no. 4, pp. 761–767, 1998.

[89] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, 2022.

[90] J. Posner, J. A. Russell, and B. S. Peterson, "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology," *Development and psychopathology*, vol. 17, no. 3, pp. 715–734, 2005.

[91] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov, "Label Studio: Data labeling software," 2020-2022, open source software available from https://github.com/heartexlabs/label-studio. [Online]. Available: https://github.com/heartexlabs/label-studio

[92] A. Banerjee, U. Chitnis, S. Jadhav, J. Bhawalkar, and S. Chaudhury, "Hypothesis testing, type i and type ii errors," *Industrial psychiatry journal*, vol. 18, no. 2, p. 127, 2009.

[93] M. Ojala and G. C. Garriga, "Permutation tests for studying classifier performance." *Journal of machine learning research*, vol. 11, no. 6, 2010.

[94] L. St, S. Wold *et al.*, "Analysis of variance (anova)," *Chemometrics and intelligent laboratory systems*, vol. 6, no. 4, pp. 259–272, 1989.

[95] G. Yin, S. Sun, H. Zhang, D. Yu, C. Li, K. Zhang, and N. Zou, "User independent emotion recognition with residual signal-image network," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3277–3281.

[96] N. He and S. Ferguson, “Music emotion recognition based on segment-level two-stage learning,” *International Journal of Multimedia Information Retrieval*, vol. 11, no. 3, pp. 383–394, 2022.

Appendices

Appendix A

As far as grid search cross validation results are concerned, only the top four performing models are presented in Section 5. These outcomes are derived from the following grid search cross-validation tables for both Arousal and Valence Recognition. Specifically, Tables 30 and 31 pertain to Music Arousal and Valence recognition for the CNN model. Additionally, Tables 32 and 33 present the grid search cross-validation results for the Dual-Stream model.

Table 30: Cross validation results of CNN for Music Arousal Recognition

Model Configuration	Mean Segment Accuracy	Segment Accuracy Std	Mean Segment F1	Segment F1 Std	Lower Confidence Interval for Segment F1	Upper Confidence Interval for Segment F1	Mean Track Accuracy	Track Accuracy Std	Mean Track F1	Track F1 Std	Lower Confidence Interval for Track F1	Upper Confidence Interval for Track F1	Mean Epoch
S4 Batch size: 64 LR: 0.002	0.763	0.024	0.732	0.033	0.666	0.798	0.836	0.029	0.808	0.037	0.734	0.882	23
S1 Batch size: 64 LR: 0.002	0.76	0.024	0.733	0.035	0.663	0.803	0.826	0.029	0.799	0.043	0.713	0.885	28
S0 Batch size: 32 LR: 0.002	0.774	0.021	0.756	0.041	0.674	0.838	0.847	0.034	0.829	0.045	0.739	0.919	24
S2 Batch size: 32 LR: 0.002	0.781	0.013	0.759	0.025	0.709	0.809	0.831	0.03	0.801	0.045	0.711	0.891	25
S0 Batch size: 32 LR: 0.001	0.778	0.031	0.757	0.044	0.669	0.845	0.815	0.032	0.8	0.054	0.692	0.908	25
S2 Batch size: 16 LR: 0.002	0.799	0.009	0.757	0.024	0.709	0.805	0.847	0.034	0.815	0.054	0.707	0.923	24
S4 Batch size: 64 LR: 0.001	0.753	0.032	0.733	0.045	0.643	0.823	0.815	0.049	0.803	0.057	0.689	0.917	22
S2 Batch size: 16 LR: 0.001	0.789	0.023	0.754	0.026	0.702	0.806	0.857	0.044	0.839	0.059	0.721	0.957	24
S3 Batch size: 32 LR: 0.002	0.782	0.015	0.756	0.019	0.718	0.794	0.815	0.032	0.775	0.06	0.655	0.895	27
S2 Batch size: 64 LR: 0.002	0.78	0.027	0.755	0.038	0.679	0.831	0.82	0.034	0.796	0.061	0.674	0.918	20
S2 Batch size: 32 LR: 0.001	0.764	0.025	0.736	0.045	0.646	0.826	0.836	0.042	0.795	0.062	0.671	0.919	20
S1 Batch size: 64 LR: 0.001	0.746	0.02	0.721	0.039	0.643	0.799	0.831	0.039	0.812	0.063	0.686	0.938	24
S4 Batch size: 32 LR: 0.001	0.761	0.034	0.734	0.055	0.624	0.844	0.831	0.047	0.814	0.065	0.684	0.944	24
S0 Batch size: 16 LR: 0.002	0.788	0.016	0.747	0.04	0.667	0.827	0.836	0.057	0.806	0.066	0.674	0.938	26
S3 Batch size: 64 LR: 0.002	0.767	0.027	0.741	0.042	0.657	0.825	0.831	0.039	0.802	0.066	0.67	0.934	28
S0 Batch size: 16 LR: 0.001	0.799	0.027	0.758	0.042	0.674	0.842	0.82	0.053	0.767	0.072	0.623	0.911	21
S4 Batch size: 16 LR: 0.002	0.794	0.023	0.759	0.03	0.699	0.819	0.836	0.047	0.807	0.073	0.661	0.953	20
S1 Batch size: 32 LR: 0.001	0.749	0.039	0.724	0.046	0.632	0.816	0.799	0.062	0.77	0.074	0.622	0.918	22
S4 Batch size: 32 LR: 0.002	0.776	0.022	0.749	0.027	0.695	0.803	0.852	0.043	0.818	0.074	0.67	0.966	24
S0 Batch size: 64 LR: 0.002	0.779	0.022	0.753	0.036	0.681	0.825	0.831	0.054	0.802	0.081	0.64	0.964	29
S0 Batch size: 64 LR: 0.001	0.758	0.015	0.742	0.04	0.662	0.822	0.794	0.056	0.772	0.082	0.608	0.936	28
S1 Batch size: 16 LR: 0.001	0.771	0.02	0.732	0.036	0.66	0.804	0.82	0.065	0.786	0.084	0.618	0.954	25
S1 Batch size: 32 LR: 0.002	0.767	0.03	0.741	0.035	0.671	0.811	0.836	0.073	0.817	0.087	0.643	0.991	22
S3 Batch size: 16 LR: 0.002	0.799	0.015	0.77	0.028	0.714	0.826	0.831	0.063	0.813	0.089	0.635	0.991	22
S1 Batch size: 16 LR: 0.002	0.784	0.03	0.743	0.04	0.663	0.823	0.852	0.06	0.825	0.09	0.645	1.005	25
S3 Batch size: 64 LR: 0.001	0.749	0.037	0.722	0.055	0.612	0.832	0.81	0.059	0.783	0.091	0.601	0.965	26
S3 Batch size: 16 LR: 0.001	0.791	0.017	0.755	0.031	0.693	0.817	0.836	0.073	0.81	0.092	0.626	0.994	31
S2 Batch size: 64 LR: 0.001	0.751	0.03	0.726	0.046	0.634	0.818	0.804	0.088	0.77	0.094	0.582	0.958	29
S3 Batch size: 32 LR: 0.001	0.766	0.021	0.739	0.034	0.671	0.807	0.836	0.082	0.801	0.103	0.595	1.007	31
S4 Batch size: 16 LR: 0.001	0.783	0.024	0.74	0.041	0.658	0.822	0.826	0.086	0.787	0.104	0.579	0.995	24

Table 31: Cross validation results of CNN for Music Valence Recongition

Model Configuration	**Mean Segment Accuracy**	**Segment Accuracy Std**	**Mean Segment F1**	**Segment F1 Std**	**Lower Confidence Interval for Segment F1**	**Upper Confidence Interval for Segment F1**	**Mean Track Accuracy**	**Track Accuracy Std**	**Mean Track F1**	**Track F1 Std**	**Lower Confidence Interval for Track F1**	**Upper Confidence Interval for Track F1**	**Mean Epoch**
S0 Batch size: 16 LR: 0.001	0.758	0.035	0.776	0.035	0.706	0.846	0.809	0.033	0.801	0.031	0.739	0.863	28
S0 Batch size: 16 LR: 0.002	0.753	0.031	0.767	0.034	0.699	0.835	0.804	0.047	0.796	0.036	0.724	0.868	25
S0 Batch size: 32 LR: 0.001	0.745	0.023	0.769	0.027	0.715	0.823	0.818	0.05	0.816	0.05	0.716	0.916	25
S0 Batch size: 32 LR: 0.002	0.747	0.013	0.774	0.022	0.73	0.818	0.809	0.049	0.805	0.047	0.711	0.899	27
S0 Batch size: 64 LR: 0.001	0.735	0.031	0.768	0.026	0.716	0.82	0.776	0.047	0.774	0.041	0.692	0.856	30
S0 Batch size: 64 LR: 0.002	0.747	0.029	0.779	0.025	0.729	0.829	0.846	0.03	0.836	0.02	0.796	0.876	22
S1 Batch size: 16 LR: 0.001	0.756	0.023	0.773	0.033	0.707	0.839	0.799	0.057	0.799	0.055	0.689	0.909	25
S1 Batch size: 16 LR: 0.002	0.752	0.024	0.769	0.029	0.711	0.827	0.79	0.042	0.788	0.038	0.712	0.864	25
S1 Batch size: 32 LR: 0.001	0.731	0.021	0.759	0.03	0.699	0.819	0.795	0.059	0.789	0.059	0.671	0.907	24
S1 Batch size: 32 LR: 0.002	0.751	0.025	0.774	0.031	0.712	0.836	0.804	0.062	0.8	0.057	0.686	0.914	20
S1 Batch size: 64 LR: 0.001	0.724	0.034	0.757	0.033	0.691	0.823	0.771	0.064	0.769	0.062	0.645	0.893	23
S1 Batch size: 64 LR: 0.002	0.742	0.032	0.774	0.034	0.706	0.842	0.823	0.072	0.82	0.075	0.67	0.97	20
S2 Batch size: 16 LR: 0.001	0.753	0.023	0.772	0.025	0.722	0.822	0.818	0.052	0.815	0.054	0.707	0.923	31
S2 Batch size: 16 LR: 0.002	0.761	0.026	0.778	0.031	0.716	0.84	0.809	0.041	0.803	0.038	0.727	0.879	22
S2 Batch size: 32 LR: 0.001	0.755	0.029	0.782	0.033	0.716	0.848	0.799	0.052	0.79	0.052	0.686	0.894	29
S2 Batch size: 32 LR: 0.002	0.755	0.024	0.781	0.027	0.727	0.835	0.813	0.045	0.805	0.038	0.729	0.881	28
S2 Batch size: 64 LR: 0.001	0.736	0.024	0.771	0.017	0.737	0.805	0.804	0.034	0.803	0.032	0.739	0.867	26
S2 Batch size: 64 LR: 0.002	0.748	0.033	0.78	0.031	0.718	0.842	0.837	0.043	0.834	0.041	0.752	0.916	41
S3 Batch size: 16 LR: 0.001	0.766	0.032	0.781	0.033	0.715	0.847	0.804	0.044	0.797	0.042	0.713	0.881	26
S3 Batch size: 16 LR: 0.002	0.758	0.02	0.777	0.027	0.723	0.831	0.795	0.057	0.792	0.054	0.684	0.9	28
S3 Batch size: 32 LR: 0.001	0.751	0.022	0.777	0.029	0.719	0.835	0.818	0.03	0.814	0.025	0.764	0.864	26
S3 Batch size: 32 LR: 0.002	0.754	0.015	0.779	0.021	0.737	0.821	0.814	0.058	0.809	0.053	0.703	0.915	21
S3 Batch size: 64 LR: 0.001	0.729	0.031	0.761	0.022	0.717	0.805	0.781	0.05	0.78	0.046	0.688	0.872	24
S3 Batch size: 64 LR: 0.002	0.74	0.031	0.773	0.027	0.719	0.827	0.827	0.05	0.825	0.045	0.735	0.915	29
S4 Batch size: 16 LR: 0.001	0.753	0.018	0.771	0.026	0.719	0.823	0.781	0.052	0.779	0.053	0.673	0.885	28
S4 Batch size: 16 LR: 0.002	0.749	0.02	0.768	0.028	0.712	0.824	0.799	0.052	0.796	0.051	0.694	0.898	22
S4 Batch size: 32 LR: 0.001	0.736	0.016	0.764	0.022	0.72	0.808	0.79	0.048	0.788	0.048	0.692	0.884	29
S4 Batch size: 32 LR: 0.002	0.742	0.021	0.769	0.027	0.715	0.823	0.781	0.047	0.779	0.046	0.687	0.871	26
S4 Batch size: 64 LR: 0.001	0.726	0.038	0.759	0.032	0.695	0.823	0.781	0.062	0.779	0.06	0.659	0.899	24
S4 Batch size: 64 LR: 0.002	0.739	0.028	0.772	0.024	0.724	0.82	0.804	0.053	0.801	0.053	0.695	0.907	26

Table 32: Cross validation results of Dual - Stream model for Music Arousal Recognition

Model Configuration	Mean Segment Accuracy	Segment Accuracy Std	Mean Segment F1	Segment F1 Std	Lower Confidence Interval for Segment F1	Upper Confidence Interval for Segment F1	Mean Track Accuracy	Track Accuracy Std	Mean Track F1	Track F1 Std	Lower Confidence Interval for Track F1	Upper Confidence Interval for Track F1	Mean Epoch
Heads: 2 Batch size: 16 LR: 0.001	0.78	0.029	0.737	0.042	0.653	0.821	0.847	0.056	0.809	0.065	0.679	0.939	23
Heads: 2 Batch size: 16 LR: 0.002	0.795	0.025	0.755	0.032	0.691	0.819	0.857	0.047	0.808	0.091	0.626	0.99	28
Heads: 2 Batch size: 32 LR: 0.001	0.763	0.032	0.732	0.043	0.646	0.818	0.847	0.068	0.822	0.068	0.686	0.958	29
Heads: 2 Batch size: 32 LR: 0.002	0.766	0.029	0.741	0.035	0.671	0.811	0.836	0.05	0.807	0.06	0.687	0.927	25
Heads: 4 Batch size: 16 LR: 0.001	0.784	0.024	0.745	0.041	0.663	0.827	0.836	0.05	0.785	0.092	0.601	0.969	26
Heads: 4 Batch size: 16 LR: 0.002	0.791	0.025	0.75	0.039	0.672	0.828	0.842	0.058	0.797	0.077	0.643	0.951	25
Heads: 4 Batch size: 32 LR: 0.001	0.758	0.03	0.737	0.039	0.659	0.815	0.826	0.039	0.802	0.051	0.7	0.904	25
Heads: 4 Batch size: 32 LR: 0.002	0.773	0.023	0.745	0.034	0.677	0.813	0.831	0.065	0.805	0.062	0.681	0.929	23
Heads: 8 Batch size: 16 LR: 0.001	0.78	0.022	0.738	0.04	0.658	0.818	0.82	0.046	0.771	0.089	0.593	0.949	24
Heads: 8 Batch size: 16 LR: 0.002	0.795	0.027	0.753	0.035	0.683	0.823	0.831	0.047	0.779	0.083	0.613	0.945	20
Heads: 8 Batch size: 32 LR: 0.001	0.767	0.034	0.739	0.047	0.645	0.833	0.836	0.05	0.813	0.053	0.707	0.919	29
Heads: 8 Batch size: 32 LR: 0.002	0.777	0.022	0.75	0.035	0.68	0.82	0.826	0.047	0.778	0.069	0.64	0.916	23
Heads: 16 Batch size: 16 LR: 0.001	0.783	0.026	0.745	0.036	0.673	0.817	0.842	0.058	0.805	0.069	0.667	0.943	25
Heads: 16 Batch size: 16 LR: 0.002	0.787	0.025	0.748	0.038	0.672	0.824	0.842	0.048	0.799	0.073	0.653	0.945	30
Heads: 16 Batch size: 32 LR: 0.001	0.762	0.029	0.732	0.043	0.646	0.818	0.836	0.065	0.793	0.089	0.615	0.971	25
Heads: 16 Batch size: 32 LR: 0.002	0.771	0.029	0.746	0.033	0.68	0.812	0.862	0.034	0.839	0.039	0.761	0.917	24

Table 33: Cross validation results of Dual - Stream model for Music Valence Recognition

Model Configuration	**Mean Segment Accuracy**	**Segment Accuracy Std**	**Mean Segment F1**	**Segment F1 Std**	**Lower Confidence Interval for Segment F1**	**Upper Confidence Interval for Segment F1**	**Mean Track Accuracy**	**Track Accuracy Std**	**Mean Track F1**	**Track F1 Std**	**Lower Confidence Interval for Track F1**	**Upper Confidence Interval for Track F1**	**Mean Epoch**
Heads: 2 [Batch size: 16] LR: 0.001	0.762	0.028	0.78	0.033	0.714	0.846	0.809	0.05	0.806	0.044	0.718	0.894	26
Heads: 2 [Batch size: 16] LR: 0.002	0.77	0.037	0.781	0.038	0.705	0.857	0.804	0.034	0.791	0.022	0.747	0.835	28
Heads: 2 [Batch size: 32] LR: 0.001	0.763	0.032	0.789	0.036	0.717	0.861	0.827	0.053	0.821	0.049	0.723	0.919	22
Heads: 2 [Batch size: 32] LR: 0.002	0.757	0.039	0.784	0.04	0.704	0.864	0.804	0.053	0.793	0.048	0.697	0.889	27
Heads: 4 [Batch size: 16] LR: 0.001	0.767	0.03	0.783	0.033	0.717	0.849	0.799	0.044	0.791	0.041	0.709	0.873	24
Heads: 4 [Batch size: 16] LR: 0.002	0.761	0.029	0.777	0.035	0.707	0.847	0.813	0.043	0.805	0.044	0.717	0.893	24
Heads: 4 [Batch size: 32] LR: 0.001	0.754	0.033	0.781	0.036	0.709	0.853	0.818	0.055	0.81	0.048	0.714	0.906	27
Heads: 4 [Batch size: 32] LR: 0.002	0.759	0.025	0.784	0.028	0.728	0.84	0.795	0.068	0.787	0.063	0.661	0.913	26
Heads: 8 [Batch size: 16] LR: 0.001	0.773	0.031	0.788	0.033	0.722	0.854	0.804	0.044	0.8	0.044	0.712	0.888	25
Heads: 8 [Batch size: 16] LR: 0.002	0.757	0.045	0.774	0.047	0.68	0.868	0.781	0.081	0.775	0.069	0.637	0.913	24
Heads: 8 [Batch size: 32] LR: 0.001	0.75	0.039	0.776	0.041	0.694	0.858	0.804	0.076	0.802	0.072	0.658	0.946	27
Heads: 8 [Batch size: 32] LR: 0.002	0.749	0.042	0.774	0.044	0.686	0.862	0.818	0.038	0.805	0.031	0.743	0.867	24
Heads: 16 [Batch size: 16] LR: 0.001	0.756	0.041	0.775	0.042	0.691	0.859	0.804	0.061	0.805	0.05	0.705	0.905	27
Heads: 16 [Batch size: 16] LR: 0.002	0.767	0.022	0.781	0.027	0.727	0.835	0.785	0.052	0.778	0.048	0.682	0.874	28
Heads: 16 [Batch size: 32] LR: 0.001	0.755	0.031	0.78	0.031	0.718	0.842	0.809	0.077	0.805	0.066	0.673	0.937	28
Heads: 16 [Batch size: 32] LR: 0.002	0.753	0.032	0.778	0.035	0.708	0.848	0.79	0.073	0.783	0.069	0.645	0.921	23