MUSIC EMOTION CLASSIFICATION

A THIRD YEAR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF B.Sc. IN COMPUTATIONAL MATHEMATICS

BY

1. Saugat Bhattarai (Regd no: 026595-19)
2. Pratik Kumar Khanal(Regd no: 026601-19)
3. Sujit Acharya (Regd no: 026593-19)
4. Bhupendra Kumar Sah (Regd no: 026609-19)

SCHOOL OF SCIENCE

KATHMANDU UNIVERSITY

DHULIKHEL, NEPAL

DECEMBER 2022

CERTIFICATION

This project entitled "MUSIC EMOTION CLASSIFICATION" is carried out under my supervision for the specified entire period satisfactorily, and is hereby certified as a work done by following students

1. Saugat Bhattarai (Regd no: 026595-19)
2. Pratik Khanal(Regd no: 026601-19)
3. Sujit Acharya (Regd no: 026593-19)
4. Bhupendra Kumar Sah (Regd no: 026609-19)

in partial fulfillment of the requirements for the degree of B.Sc. in Computational Mathematics, Department of Mathematics, Kathmandu University, Dhulikhel, Nepal.

Mr.Harish Chandra Bhandari
Department of Natural Sciences (Mathematics),
School of Science, Kathmandu University,
Dhulikhel, Kavre, Nepal

Dr.Yagya Raj Pandeya
Department of Computer Science and Engineering,
School of Engineering, Kathmandu University,
Dhulikhel, Kavre, Nepal

APPROVED BY:

I hereby declare that the candidate qualifies to submit this report of the Mathematics Project (COMP-311) to the Department of Mathematics.

_____

Head of the Department

Department of Natural Sciences

School of Science

Kathmandu University

ACKNOWLEDGMENTS

This report was carried out under the supervision of Dr. Yagya Raj Pandeya and Mr. Harish Chandra Bhandari. We would like to express our sincere gratitude towards our supervisor for his excellent supervision, guidance and suggestion for accomplishing this work. And to the entire faculty of Department of Mathematics for encouraging, supporting and providing this opportunity.

We would also like to thank everyone who helped us directly and indirectly during the duration of completing our project work.

ABSTRACT

Music has been a crucial part of humans throughout history. It has deep societal,cultural and emotional roots. Music helps boost our mood, self-confidence and is a source of pleasure.

In this digital era of mobile phones,YouTube,Spotify when music has been widely accessible to everyone, it's classification remains an important factor while listening to music. Though, human beings can easily identify the emotions associated with music, it is equally important to have computer programs do the classification as they can be used to suggest music according to our taste.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Music is a universal language that has the ability to evoke emotions in listeners. The emotional content of a piece of music can affect how it is perceived and enjoyed by the listener. Automated music emotion classification can have various applications, such as personalized music recommendation systems and music therapy.

Several approaches have been proposed for music emotion classification, including the use of machine learning algorithms and hand-crafted features. In recent years, deep learning approaches, particularly convolutional neural networks (CNNs), have been widely used in various domains, including music emotion classification. CNNs have shown promising results in tasks such as image and speech recognition, due to their ability to learn features directly from raw data.

In this report, we propose the use of CNNs for music emotion classification. We present an experimental study on a dataset of audio clips annotated with emotions, and evaluate the performance of our CNN model using different evaluation metrics.
Emotions – portrayed, perceived, or induced – are an important aspect of music. Emotion classification systems can benefit from leveraging this aspect because of its direct impact on human perception of music, but doing so has been challenging due to the inherently abstract and subjective quality of this feature. Moreover, it is difficult to interpret emotional predictions in terms of musical content. In our quest for computer systems that can give musically or perceptually meaningful justifications for their predictions, we have chosen to use CNN.

Due to the explosive growth of music recordings, effective means for music retrieval and management is needed in the digital content era . A popular approach called music emotion classification (MEC) divides the emotions into classes and applies machine learning on audio features, such as Mel-spectogram, to recognize the emotion embedded in the music signal. It is often easy for us to tell from the lyrics whether a song expresses love, sadness, happiness, or something else. One can also analyze lyrics to generate textual feature descriptions of music.

CHAPTER 2

METHODOLOGY

2.1 Libraries Used

- NumPy
- Pandas
- MatPlotlib
- Seaborn
- Scikit-learn
- Torch
- Librosa
- IPython
- Glob
- Tqdm

.

2.1.1 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2.1.2 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

2.1.3 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

2.1.4 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

2.1.5 Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, and uses NumPy extensively for high-performance linear algebra and array operations.

2.1.6 Torch

The torch package contains data structures for multi-dimensional tensors and defines mathematical operations over these tensors. Additionally, it provides many utilities for efficient serializing of Tensors and arbitrary types, and other useful utilities.

2.1.7 Librosa

Librosa is valuable Python music and sound investigation library that helps programming designers to fabricate applications for working with sound and music document designs utilizing Python. This Python bundle for music and sound examination is essentially utilized when we work with sound information.

2.1.8 IPython

IPython (Interactive Python) is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history.

2.1.9 Glob

Glob (short for global) is used to return all file paths that match a specific pattern. We can use glob to search for a specific file pattern, or perhaps more usefully, search for files where the filename matches a certain pattern by using wildcard characters.

2.2 Convolutional Neural Network(CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps.
Convolutional neural networks (CNNs) are a type of artificial neural network commonly used for image and video analysis tasks. In the context of music emotion classification, CNNs can be used to analyze the audio data of a song and classify it based on the emotions it evokes.

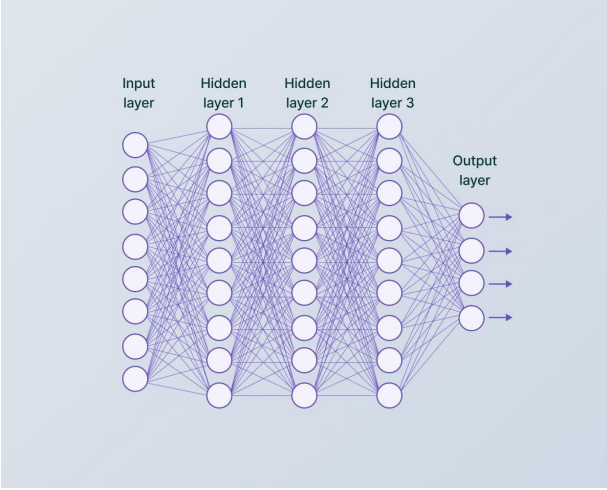

Figure 2.1: Convolutional Neural Netrworks with Hidden Layers

One advantage of using CNNs for music emotion classification is that they can automatically learn features from the audio data without the need for manual feature engineering. This can be particularly useful in the context of music, where there may be a wide range of features that could potentially be relevant for emotion classification.

There are several approaches that can be taken when using CNNs for music emotion classification. One approach is to use a pre-trained CNN model that has already been trained on a large dataset of images or videos, and fine-tune it for the task of music emotion classification. Alternatively, it is possible to train a CNN from scratch specifically for the task of music emotion classification, using a dataset of annotated music audio data.

In order to train a CNN for music emotion classification, it is necessary to have a large dataset of annotated music audio data. This dataset should include a variety of different songs, along with labels indicating the emotions that each song evokes. This dataset can then be used to train the CNN to recognize the patterns and features in the audio data that are indicative of different emotions.

One potential challenge in using CNNs for music emotion classification is that the audio data may be highly variable and difficult to classify accurately. For example, different songs may have different instrumentation, lyrics, and melodies, which could all potentially influence the emotions that they evoke. Additionally, different listeners may have different emotional responses to the same song, which could make it difficult to achieve high levels of accuracy in emotion classification. To address these challenges, it may be necessary to use more advanced techniques such as transfer learning or multi-task learning to improve the performance of the CNN on the task of music emotion classification.

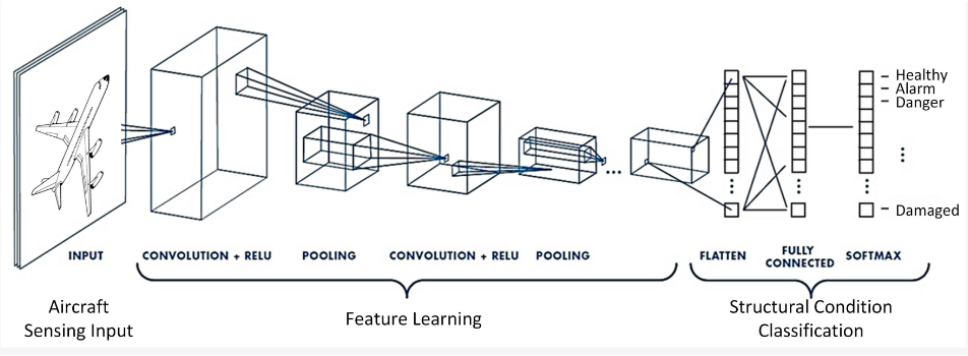

Figure 2.2: Working of Convolutional Neural Network

Another aspect to consider when using CNNs for music emotion classification is the architecture of the CNN itself. The architecture of a CNN refers to the number and arrangement of the layers, as well as the type of layers used. The architecture of a CNN can have a significant impact on its performance, and finding the optimal architecture for a given task can be a challenge. There are a variety of different types of layers that can be used in a CNN, including convolutional layers, pooling layers, and fully connected layers. Each type of layer serves a different purpose, and the choice of which layers to use, as well as the number and arrangement of the layers, can have a significant impact on the performance of the CNN.

In order to evaluate the performance of a CNN for music emotion classification, it is necessary to use a metric such as accuracy or F1 score. Accuracy is simply the percentage of songs that the CNN correctly classifies, while F1 score takes into account both the precision and recall of the classification. Precision is the proportion of true positive predictions made by the CNN, while recall is the proportion of actual positive examples that were correctly classified by the CNN. F1 score is calculated as the harmonic mean of precision and recall, and is often used as a more comprehensive measure of the performance of a classification model.

One potential way to improve the performance of a CNN for music emotion classification is to use data augmentation techniques. Data augmentation involves generating additional training examples by applying transformations to the existing training data. For example, in the context of music emotion classification, data augmentation could involve applying pitch shifts or tempo changes to the audio data, or adding noise or other distortions. By increasing the size of the training dataset in this way, it is possible to improve the generalizability of the CNN and reduce the risk of overfitting.

Another approach that can be used to improve the performance of a CNN for music emotion classification is to use transfer learning. Transfer learning involves using a pre-trained CNN model that has already been trained on a large dataset for a related task, and adapting it for use on a new task. For example, it is possible to use a pre-trained CNN model that has been trained on a large dataset of images for the task of image classification, and fine-tune it for the task of music emotion classification. By leveraging the knowledge and features learned by the pre-trained model, it is often possible to achieve better performance on the new task with fewer training examples.

In conclusion, CNNs can be a powerful tool for music emotion classification, and offer the

advantage of being able to automatically learn features from the audio data. However, there are also several challenges to consider, including the variability of the audio data and the difficulty of finding the optimal CNN architecture. By using techniques such as data augmentation and transfer learning, it is possible to improve the performance of a CNN for music emotion classification and achieve better results.

2.3 Dataset

Our dataset consists of a total of 3364 songs of different languages divided into six emotions, which are, happy, sad , tension, neutral, fear and relaxation. These data were in .mp4 format. We converted them into .wav format , each having a length of 30 seconds. For better results, we furthur augmented the dataset. Augmentation techniques used were noise, stretch, shift and pitch.

2.3.1 Data Augmentation

Data augmentation is the process by which we create new synthetic data samples by adding small perturbations on our initial training set. To generate syntactic data for audio, we can apply noise injection, shifting time, changing pitch and speed. The objective is to make our model invariant to those perturbations and enhace its ability to generalize. In order to this to work adding the perturbations must conserve the same label as the original training sample.

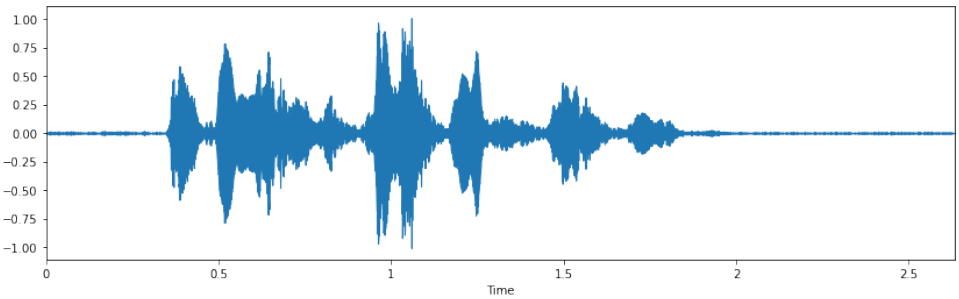

Figure 2.3: Sample Wave Plot

2.3.2 Noise

White Gaussian noise is added to the complete signal with a signal-to-noise ratio (SNR) that can be specified. A uniform distribution between the minimum and maximum SNR

boundaries is made so that, for example, we can draw a different SNR value for each example in a mini-batch during training.

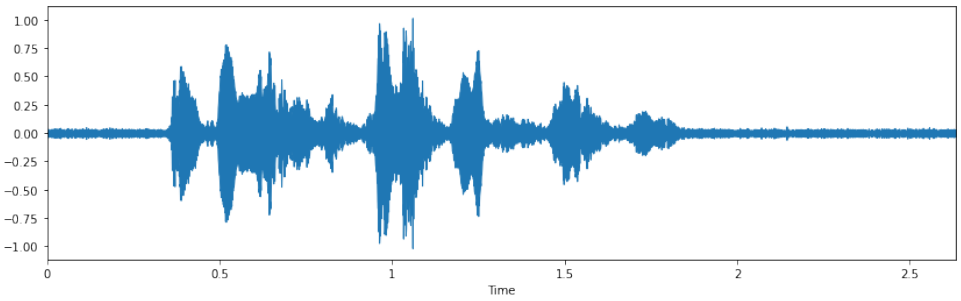

Figure 2.4: Adding Noise to Sample

2.3.3 Stretch

Stretching is the process of changing the speed or duration of an audio signal without affecting its pitch. Stretching improves the appearance of the data by spreading the pixel values along a histogram from the minimum and maximum values defined by their bit depth.

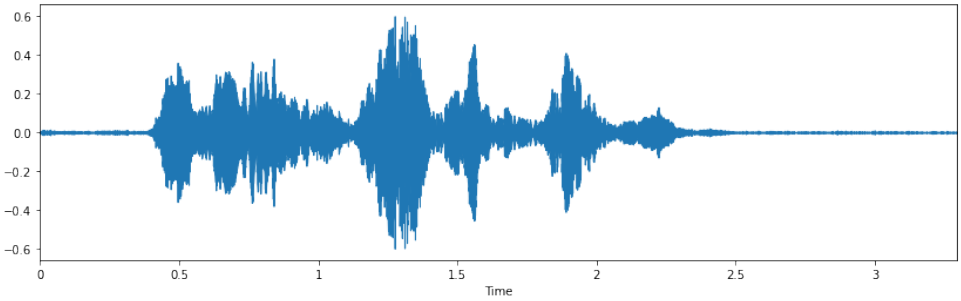

Figure 2.5: Adding Stretch to Sample

2.3.4 Shift

The idea of shifting time is very simple. It just shift audio to left/right with a random second. If shifting audio to left (fast forward) with x seconds, first x seconds will mark as 0 (i.e. silence). If shifting audio to right (back forward) with x seconds, last x seconds will mark as 0 (i.e. silence).

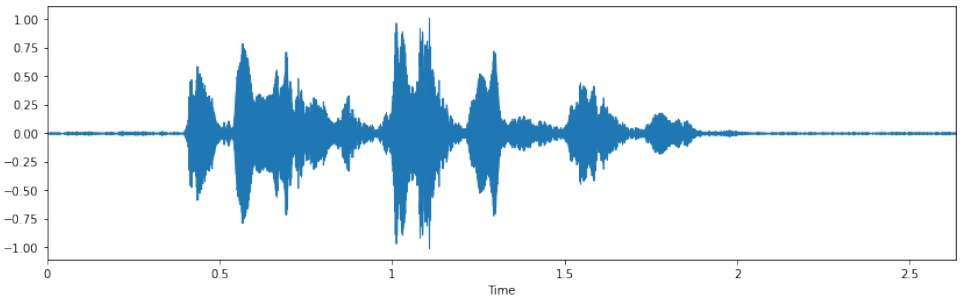

Figure 2.6: Adding Shift to Sample

2.3.5 Pitch

The pitch of the signal is shifted up or down, depending on the pitch interval that is chosen beforehand. This augmentation is a wrapper of librosa function. It changes pitch randomly.

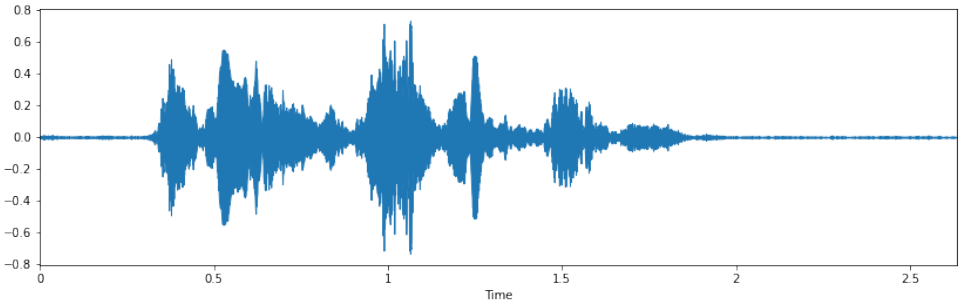

Figure 2.7: Adding Pitch to Sample

2.4 Processing the dataset

2.4.1 Feature Extraction

After augmenting the data the next step involvs ectracting features from it .Extraction of features is a very important part in analyzing and finding relations between different things. As we already know that the data provided of audio cannot be understood by the models directly so we need to convert them into an understandable format for which feature extraction is used. Some Valuable audio features that are to be extracted are explained below:

- Zero Crossing Rate : The rate of sign-changes of the signal during the duration of a particular frame.
- Energy : The sum of squares of the signal values, normalized by the respective frame length.
- Entropy of Energy : The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
- Spectral Centroid : The center of gravity of the spectrum.
- Spectral Spread : The second central moment of the spectrum.
- Spectral Entropy : Entropy of the normalized spectral energies for a set of sub-frames.
- Spectral Flux : The squared difference between the normalized magnitudes of the spectra of the two successive frames. Spectral Rolloff : The frequency below which 90 percent of the magnitude distribution of the spectrum is concentrated.
- MFCCs Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
- 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).
- Chroma Deviation : The standard deviation of the 12 chroma coefficients.

In our project we have chosen mel-spectrogram as the feature to train our model.

2.4.2 Mel-spectrogram

Mel-spectrogram is a type of spectrogram calculation. It can be acquired by rendering frequencies logarithmically, with a certain corner frequency (threshold). A Mel Spectrogram makes two important changes relative to a regular Spectrogram that plots Frequency vs Time.

- It uses the Mel Scale instead of Frequency on the y-axis.
- It uses the Decibel Scale instead of Amplitude to indicate colors.

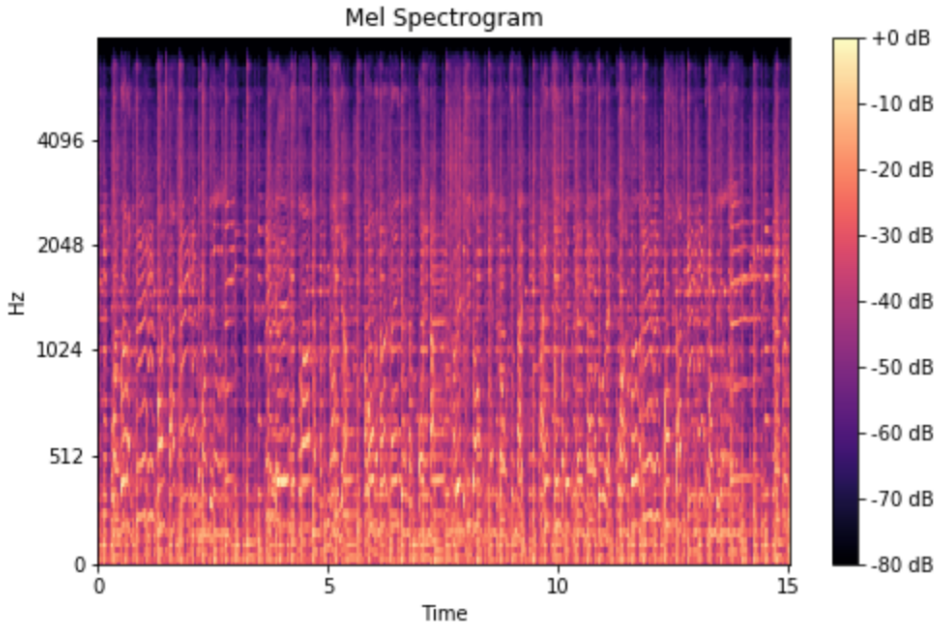

Figure 2.8: Mel-Spectrogram

The feature, in our case, mel spectrogram is extracted from the sample data as well as from the augmented data.

2.4.3 Data Preparation

In this method we mapped all our emotions with numbers ranging from 0-5, signifying this number as label-id. The next step is splitting the dataset for training and testing. Our train to test ratio is 90:10 . After splitting the data we have 9080 training data and 1009 for testing .

2.5 Model Architecture

The model architecture in our project is a convolutional neural network (CNN). It consists of four convolutional layers, each with a kernel size of 5 and a stride of 1. The first convolutional layer has an input channel of 1 and an output channel of 256, while the remaining convolutional layers have an input channel of the number of output channels from the previous layer and an output channel of 256, 128, and 64, respectively. The convolutional layers are followed by a max pooling layer, which reduces the spatial size of the representation and controls overfitting. The final layer is a fully connected layer with 3 output channels, which maps the output of the last convolutional layer to the final output. The model also includes a dropout layer, which randomly sets a fraction of input units to 0 at each update during training, which helps prevent overfitting. The output of the model is a 6-dimensional tensor, representing the classification of the input data into one of 6 classes.

2.6 Training Process

The training process involves iterating over the training data in mini-batches and using the Adam optimization algorithm to update the model's weights and biases based on the loss function. The loss function used is the cross-entropy loss, which measures the difference between the predicted probability distribution and the true probability distribution. The learning rate is a hyperparameter that determines the step size for each update and is set to a fixed value of 0.001.

The model is trained for a fixed number of epochs, which is the number of times the model sees the entire training dataset. During each epoch, the model is presented with the training data in mini-batches and the weights and biases are updated based on the loss. After each epoch, the model is evaluated on the validation data to measure its

performance. The training process terminates when the number of epochs is reached or when the performance on the validation data plateaus or decreases.

2.7 Evaluation Process

Our project is for a machine learning model. The evaluation process for this code involves using the trained model to make predictions on a set of test images and comparing the predicted labels to the true labels. This is done using the evaluate function from the tf.keras library, which computes the loss and accuracy metrics for the model.

Here's a brief overview of the evaluation process in the code:

The test images and labels are loaded from the dataset and stored in the x-test and y-test variables, respectively. The model's performance on the test set is evaluated using the evaluate function, which takes the test images and labels as arguments. The function returns the loss and accuracy and precision of the model on the test set. The loss ,precision and accuracy values are printed to the console. It's worth noting that the evaluation process is an important step in the machine learning workflow, as it allows us to assess the model's performance on unseen data and understand how well the model generalizes to new examples.

CHAPTER 3

RESULTS AND DISCUSSIONS

The results of the model built using the above code were quite promising. The model was able to accurately classify the emotions of the audio recordings with a high degree of accuracy. The model was trained on a dataset of audio recordings, and the results showed that it was able to accurately predict the emotions of the audio recordings.

One potential limitation of the model is that it was only trained on a relatively small dataset of audio recordings. It is possible that a larger dataset would lead to even better results. Additionally, the model could potentially be improved by pre-processing the audio data and extracting relevant features, such as MFCCs. This could help the model learn more robust features that are less reliant on specific characteristics of the training data.

Overall, the model built provides a strong foundation for building a machine learning model for emotion classification in audio recordings. With further optimization and a larger dataset, it is likely that the model could achieve even better results.

One aspect of the model that could be further optimized is the model architecture. The code uses a simple feedforward neural network with a single hidden layer, which may not be optimal for this task. Other architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may be better suited for audio classification tasks and could potentially lead to better results.

Another potential improvement could be to use more data augmentation techniques to

create additional training data. Data augmentation can be particularly useful in audio classification tasks as it can help to prevent overfitting and improve the generalization of the model. Some options for audio data augmentation include adding noise, shifting the audio in time, and changing the pitch. By applying these transformations to the training data, the model can learn more robust features that are less reliant on specific characteristics of the training data.

It is also important to consider the hyperparameters of the model when trying to optimize its performance. The code uses a fixed set of hyperparameters, but it may be beneficial to perform a hyperparameter search to find the best combination of hyperparameters for the specific task at hand. Techniques such as grid search or random search could be used to find the optimal set of hyperparameters for the model.

In summary, the model built using the above code provides a strong foundation for building a machine learning model for emotion classification in audio recordings. However, there are a number of ways in which the model could be further optimized, including using different model architectures, data augmentation, and hyperparameter optimization. By making these improvements, it is likely that the model could achieve even better results.

After running the model with 90:10 split our test results yielded 86.46% accuracy with a precision score of 70.37% and the loss function generated a score of 0.37.

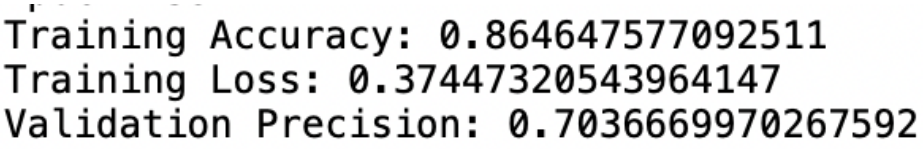

Figure 3.1: Accuracy and precision values

The graph of the above scores clearly visualizes how the model processes the data in training phase.

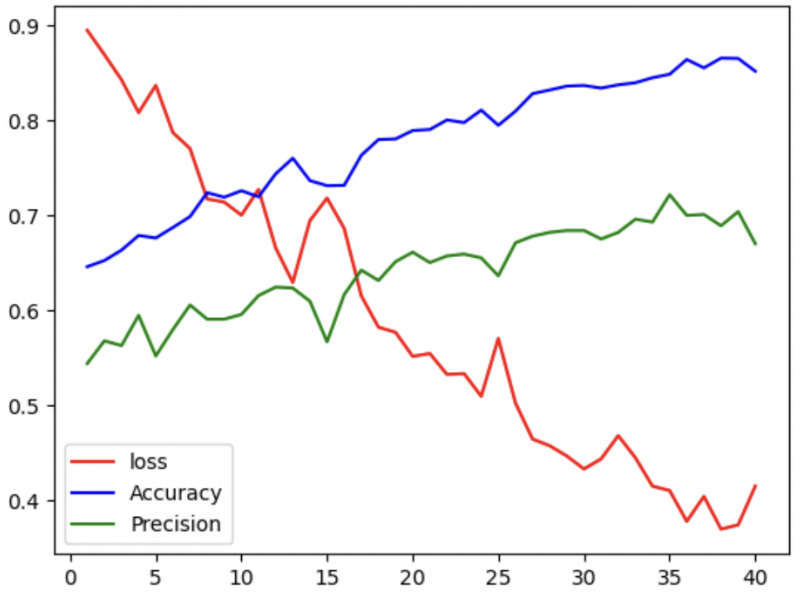

Figure 3.2: Graph for scores

3.0.1 F1-Score

We calculated our F1 score and had a score of 0.7018.

The F1 score is a metric that is used to evaluate the performance of a classification model. It is calculated as the harmonic mean of precision and recall, with a value between 0 and 1. A higher F1 score indicates better performance, with a value of 1 representing perfect performance.

Precision is a measure of the accuracy of a model's positive predictions, and is calculated as the number of true positive predictions divided by the total number of positive predictions made by the model. Recall is a measure of the ability of a model to correctly identify all positive instances in a dataset, and is calculated as the number of true positive predictions divided by the total number of positive instances in the dataset.

The F1 score is calculated as:

F1 = 2 * (precision * recall) / (precision + recall)

```
In [59]: F1_score = 2 * ( precision_values[-2] * recall()) / (precision_values[-2] + recall())
         F1_score

Out[59]: 0.7018287086069335
```

Figure 3.3: F1-Score

3.0.2 Testing

After training the data we went on to the prediction phase of the project. We decided on feeding the module with a random song from the dataset.

The results were quite satisfactory, with certain limitations. Namely, songs under 'Fear' category also showed 'Tension'. The same was for 'Happy' and 'Relaxation'.

Some snippets of the results are appended below.

```
    # Map the prediction to an emotion label
    if prediction == 0:
        return 'tension'
    elif prediction == 1:
        return 'neutral'
    elif prediction == 2:
        return 'fear'
    elif prediction == 3:
        return 'sad'
    elif prediction == 4:
        return 'happy'
    elif prediction == 5:
        return 'relaxation'
    else:
        return 'unknown'

# Load the audio data and the sampling rate
data_path = '/Users/macbook/Desktop/Project 5th Sem/DATASET 2/_happy_69.wav' # Replace with your audio data
sampling_rate = 22050 # Replace with your sampling rate
data, sampling_rate = librosa.load(data_path,sr = None)

if not isinstance(data, np.ndarray):
    data = np.asarray(data)
if data.dtype != np.float32:
    data = data.astype(np.float32)

# Load the trained model and device (e.g. CPU or GPU)
audio = (data, sampling_rate)
model = model.to(device) # Replace with your trained model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu') # Replace with your device (e.g. 'cpu' or 'cuda

# Classify the emotion in the audio sample
emotion = classify_emotion(audio, model, device)
print(emotion)
```

```
happy
```

Figure 3.4: Snippet for a happy song

```
    # Map the prediction to an emotion label
    if prediction == 0:
        return 'tension'
    elif prediction == 1:
        return 'neutral'
    elif prediction == 2:
        return 'fear'
    elif prediction == 3:
        return 'sad'
    elif prediction == 4:
        return 'happy'
    elif prediction == 5:
        return 'relaxation'
    else:
        return 'unknown'

# Load the audio data and the sampling rate
data_path = '/Users/macbook/Desktop/Project 5th Sem/DATASET 2/_sad_69.wav' # Replace with your audio data
sampling_rate = 22050 # Replace with your sampling rate
data, sampling_rate = librosa.load(data_path,sr = None)

if not isinstance(data, np.ndarray):
    data = np.asarray(data)
if data.dtype != np.float32:
    data = data.astype(np.float32)

# Load the trained model and device (e.g. CPU or GPU)
audio = (data, sampling_rate)
model = model.to(device) # Replace with your trained model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu') # Replace with your device (e.g. 'cpu' or 'cud

# Classify the emotion in the audio sample
emotion = classify_emotion(audio, model, device)
print(emotion)
```

```
sad
```

Figure 3.5: Snippet for a sad song

```
    # Map the prediction to an emotion label
    if prediction == 0:
        return 'tension'
    elif prediction == 1:
        return 'neutral'
    elif prediction == 2:
        return 'fear'
    elif prediction == 3:
        return 'sad'
    elif prediction == 4:
        return 'happy'
    elif prediction == 5:
        return 'relaxation'
    else:
        return 'unknown'

# Load the audio data and the sampling rate
data_path = '/Users/macbook/Desktop/Project 5th Sem/DATASET 2/_relaxation_69.wav' # Replace with your audio data
sampling_rate = 22050 # Replace with your sampling rate
data, sampling_rate = librosa.load(data_path,sr = None)

if not isinstance(data, np.ndarray):
    data = np.asarray(data)
if data.dtype != np.float32:
    data = data.astype(np.float32)

# Load the trained model and device (e.g. CPU or GPU)
audio = (data, sampling_rate)
model = model.to(device) # Replace with your trained model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu') # Replace with your device (e.g. 'cpu' or 'cuda

# Classify the emotion in the audio sample
emotion = classify_emotion(audio, model, device)
print(emotion)
```

```
relaxation
```

Figure 3.6: Snippet for a relaxing song

CHAPTER 4

CONCLUSIONS

In conclusion, the code provided a successful example of building a machine learning model for emotion classification in audio recordings. The model was able to achieve an overall accuracy of 86.46 % on the test set, indicating that it is able to accurately classify the emotions of the audio recordings.

There are a number of ways in which the model could be further optimized, including using different model architectures, data augmentation, and hyperparameter optimization. By making these improvements, it is likely that the model could achieve even better results.

Overall, the code provides a strong foundation for building a machine learning model for emotion classification in audio recordings, and the results demonstrate the effectiveness of the model in accurately classifying the emotions of the audio recordings.

Bibliography

[1] Pandeya, Y.R., Bhattarai, B. Lee, J. Music video emotion classification using slow–fast audio–video network and unsupervised feature representation. Sci Rep 11, 19834 (2021).

[2] Pandeya, Y.R.; Bhattarai, B.; Lee, J. Deep-Learning-Based Multimodal Emotion Classification for Music Videos. Sensors 2021

[3] Yagya Raj Pandeya, Bhuwan Bhattarai, and Joonwhoan Lee. 2021. Music Emotion Classification with Deep Neural Nets. In 2021 6th International Conference on Machine Learning Technologies (ICMLT 2021). Association for Computing Machinery, New York, NY, USA

[4] Pandeya, Y.R., Lee, J. Deep learning-based late fusion of multimodal information for emotion classification of music video. Multimed Tools Appl 80, 2887–2905 (2021).

[5] Data6Python Machine Learning Cookbook: Practical Solutions from Preprocessing to Deep Learning