Applied Deep Learning Project

Music for Mental Health

Team 6	
Name	**Student ID**
Yang Ziyi	1006253
Christopher Lim Kai En	1006361
Wang Zixuan	1006391

Table of contents

1. Introduction

A. Background

Mental Health as the Biggest health Problem
According to Global Health Service Monitor[1] , mental health is now the number one concern when Singaporeans and people around the world are asked about the main healthcare problems facing their country.

Close to half (46%) of Singaporeans place mental health as the biggest health problem facing the country today, followed by cancer (38%) and stress (35%) [2]. In addition, close to half of Singaporeans (49%) reported feeling depressed to the point that they felt sad or hopeless almost every day for weeks at a time.

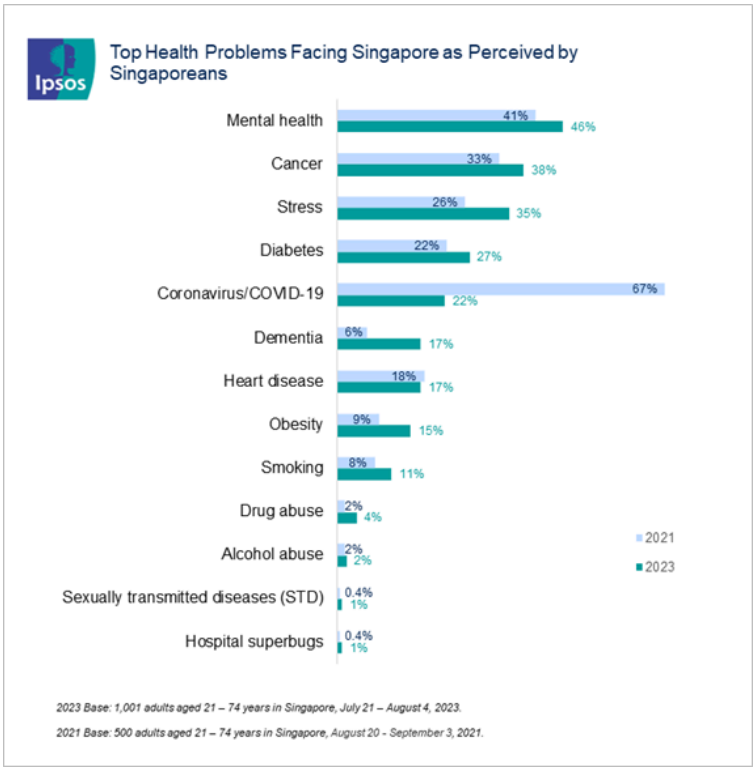


Figure1: Top Health Problems Facing Singapore as Perceived by Singaporeans

Music for Mental Health
For centuries, music has been a powerful tool that has profound effects on mental health and well-being through its ability to influence emotions and behaviours.

Music has a profound impact on mental health, leveraging its ability to influence emotions and behaviours through neurological engagement. Here's a condensed overview of its effects:

Historical and Therapeutic Use

Historically, music has been used for healing across many cultures, reflecting its deep-rooted connection to emotional and psychological well-being. Modern music therapy, now a formalised discipline, uses these principles to address physical, emotional, cognitive, and social needs in various settings.

Psychological and Neurological Benefits
Music can stimulate the brain's limbic system, influencing mood and emotional states. It enhances the release of neurotransmitters like dopamine and serotonin, which help elevate mood and reduce stress. It also affects the autonomic nervous system, lowering heart rate and blood pressure.

Current Trends and Future Research
With technological advancements, personalised music interventions are increasingly used in mental health treatments. Research continues to explore how different music types can specifically affect brain regions and optimise therapeutic outcomes.

This summary highlights the essential role of music in mental health, demonstrating its potential as a powerful therapeutic tool. “Furthermore, there are numerous reviews on the effect of music interventions, both active (e.g. performing) and passive (e.g. listening), on individuals in clinical settings”[3].

B. Introduction

Problem Statement
How might we design and train a deep learning model that provides users with information on their music preferences with relation to their mental health.

Objectives
To explore how different music genres and listening habits can affect a person suffering from various mental health issues from the dataset. Based on the user’s mental issues, and demographics, train and test a model that informs the user if their favourite music genre is improving their mental health issues.

Scope
Analyse and clean the data provided before exploring the possible deep learning models which are available to us and suitable for this project. The next task is to create and train the model, running tests to see its accuracy. Lastly, evaluating its success and looking for possible future improvements.

2. Data Collection and Preprocessing

The dataset used in this project consists of survey results related to music and mental health, obtained from Kaggle. It features a wide range of variables, each representing a question for the respondents to answer. Below is a description of each variable in this dataset:

1. Demographic and General Information

- **Timestamp:** Date and time when each response was submitted.
- **Age:** Respondent's age, providing context to the generational music preferences and mental health impact.

2. Music Listening Habits

- **Primary Streaming Service:** Identifies the main platform used for music streaming, such as Spotify, YouTube Music, or others.
- **Hours per Day:** Captures the average number of hours respondents listen to music daily.
- **While Working:** Indicates whether the respondent listens to music during work or study periods.
- **Instrumentalist:** Notes if the respondent regularly plays any musical instruments.
- **Composer:** Identifies if the respondent engages in composing music.
- **Fav Genre:** Records the respondent's favourite or most preferred music genre.
- **Exploratory:** Checks if the respondent actively explores new artists and genres.
- **Foreign Languages:** States whether the respondent regularly listens to music with lyrics in languages they do not fluently speak.

3. Specific Musical Preferences

- **BPM:** Beats per minute of the respondent's favourite genre, providing a quantitative measure of music tempo preference.
- **Frequency [Genre]:** A series of questions asking how frequently the respondent listens to various genres: Classical, Country, EDM, Folk, Gospel, Hip hop, Jazz, K-pop, Latin, Lofi, Metal, Pop, R&B, Rap, Rock, and Video game music.

4. Mental Health Metrics

- **Anxiety:** Self-reported level of anxiety on a scale of 0-10.
- **Depression:** Self-reported level of depression on a scale of 0-10.
- **Insomnia:** Self-reported level of insomnia on a scale of 0-10.
- **OCD:** Self-reported level of obsessive-compulsive disorder on a scale of 0-10.

5. Impact of Music on Mental Health

- **Music Effects:** An assessment of whether music generally improves or worsens the respondent's mental health conditions.

The data collected provides a comprehensive view of the intersection between music consumption habits and mental health, facilitating an analysis aimed at understanding how different musical elements and listening behaviours could potentially influence mental health outcomes. This detailed dataset allows for a multifaceted analysis of the role of music in therapeutic and everyday contexts.

A. Data Cleaning

When preparing our dataset for our analysis. We identified the need for essential cleaning of the data that kept the integrity and usability to use for the model:

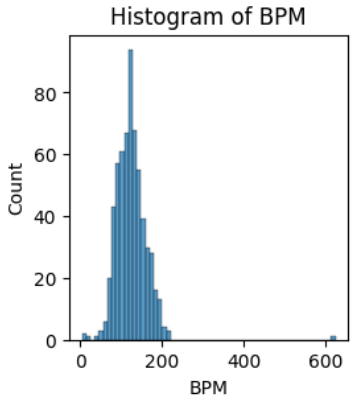
Steps taken in Data Cleaning

First, we removed two columns - ‘Timestamp’ and ‘Permission’ - as they contain information about the survey that is not necessary for our project.

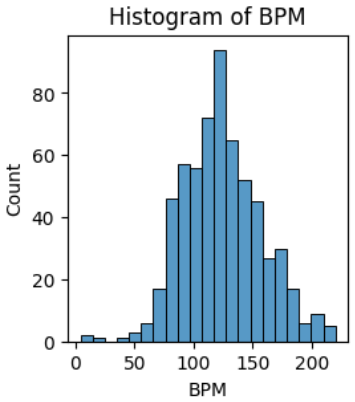
Second, during a summary analysis of the BPM variable, we observed that the maximum BPM value was extremely high, exceeding the normal BPM range, and the minimum BPM value 0, which does not make sense. After researching the current highest BPM value, which is 1015, we set a threshold and removed rows with BPM values exceeding this threshold. Additionally, we replaced the 0 values with NA, treating them as missing values. Following the removal of outliers, we plotted a histogram and identified a remaining outlier. We then conducted a second round of outlier removal.

```
count    6.160000e+02
mean     1.623500e+06
std      4.029114e+07
min      0.000000e+00
25%      1.000000e+02
50%      1.200000e+02
75%      1.440000e+02
max      1.000000e+09
Name: BPM, dtype: float64
```

Original Summary Analysis



First round of removing outliers



Second round of removing outliers

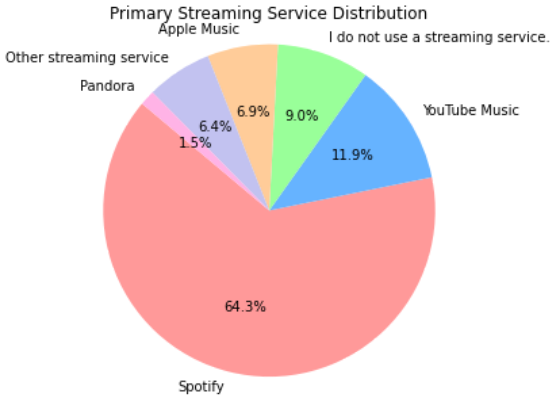
Last, we noted a small number of missing values, fewer than 10 in each, in the ‘Age’, ‘Primary streaming service’, ‘While working’, ‘Instrumentalist’, ‘Composer’, ‘Foreign languages’ and ‘Music effects’ columns. We removed rows with missing values in these columns. However, this approach could not be applied to the ‘BPM’ column due to a large number of missing values, which would shrink our dataset size if dropped. Instead, we decided to replace missing values with the median. While considering other options such as replacing missing value with 0 or ‘NA’ to maintain numeric data type, we opted for the median to avoid introducing inaccuracies into the data. After examining the data distribution, we chose the median over the mean, as median is still a safer option and more representative of the central data trend, reducing the risk of the replaced values being influenced by any remaining outliers or data irregularities.

B. Data Visualization

Pie charts

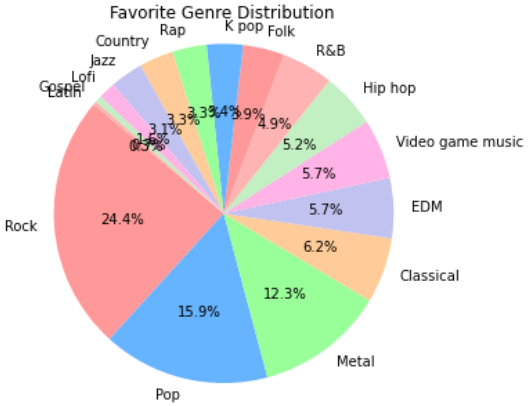
Primary Streaming Service Distribution

Visualise the distribution of respondents based on their primary streaming service.



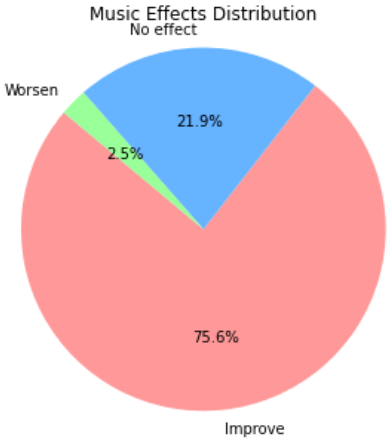
Fav Genre Distribution

Visualise the distribution of respondents' favourite genres.



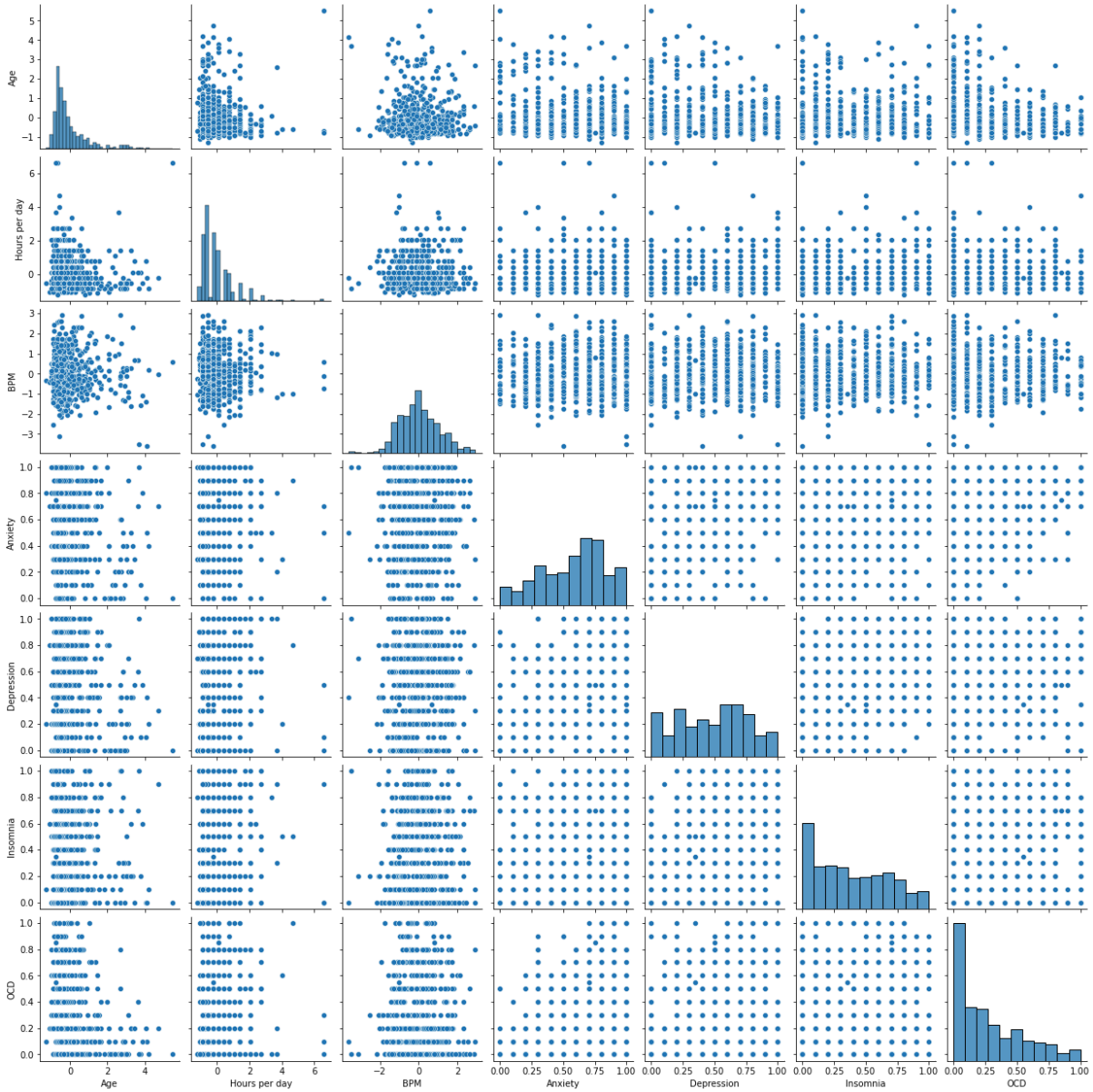
Music Effects Distribution

Visualise the distribution of different music effects reported by respondents.



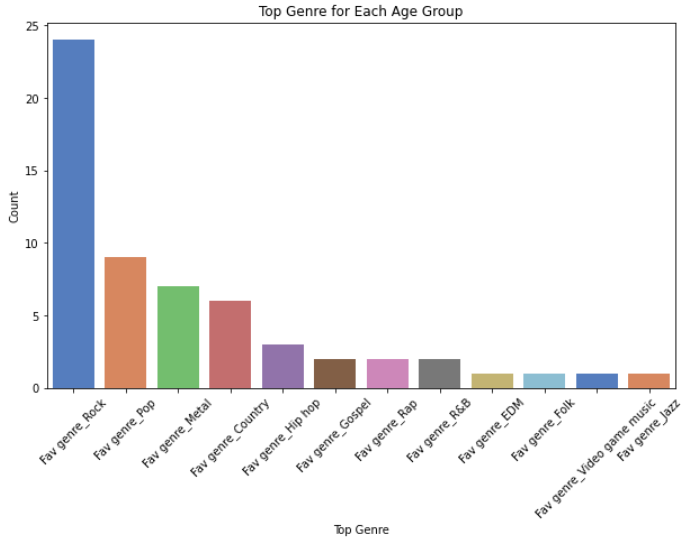
Pairplot

We created a pairplot to visualise pairwise relationships between different numerical features in the dataset.



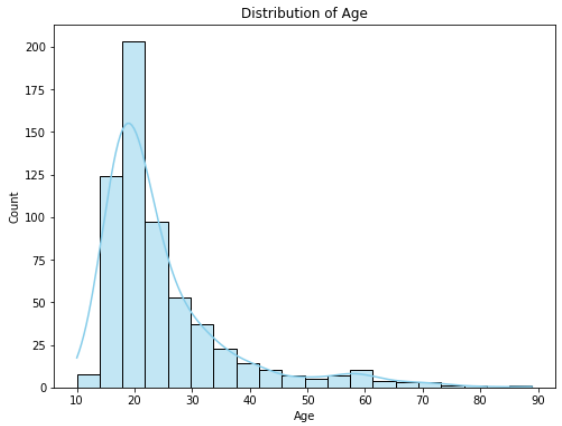
Countplot of Music Preferences

To visualise the frequency of different music genres among respondents.



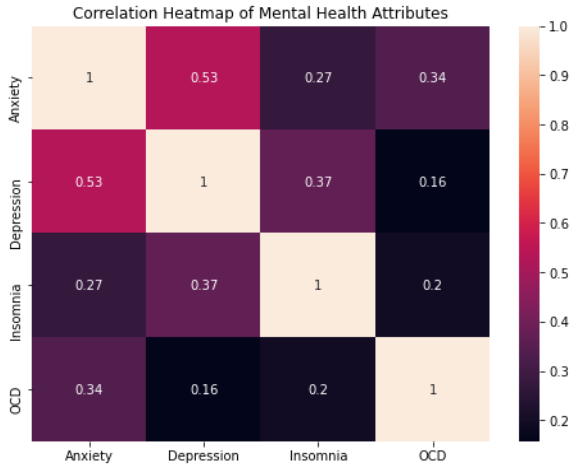
Distribution of Age

Plotted a histogram to visualise the distribution of ages in the dataset.



Correlation heatmap of Mental Health Attributes

Create a boxplot to compare the distribution of anxiety, depression, insomnia, and OCD scores.



C. Data preprocessing for model

In order to conduct the model , the current dataset was not readily available as it contains a mixture of numeric, boolean and object data types. Pre-processing of the data is needed, with different approach used for each type of data, as outlined below:

Numeric Data Normalisation

Numeric data such as the 'Age', 'Hours per day', and 'BPM' columns are normalised using Standard Scaling. This method is chosen because it effectively handles skewness and reduces the impact of any extreme values that may present.

```
# Normalize numerical data through Standard Scaling
stanard_scaler = StandardScaler()

numeric_columns = ['Age', 'Hours per day', 'BPM']
df_process[numeric_columns] = stanard_scaler.fit_transform(df_process[numeric_columns])
```

Scaling Ordinal Mental Health Metrics

On the other hand, numeric data such as the 'Anxiety', 'Depression', 'Insomnia', and 'OCD' columns represent ordinal data with values ranging from 0 to 10. The distribution of these variables does not indicate heavy skewness or outliers within a normal range of these scales. Therefore, Min-Max Scaling is deemed more suitable in this case, as it transforms the scores into a range from 0 to 1, preserving the ordinal nature of data while ensuring consistency across the variables.

```
# Normalize numerical data through Min-Max Scaling
min_max_scaler = MinMaxScaler()

mental_columns = ['Anxiety', 'Depression', 'Insomnia', 'OCD']
df_process[mental_columns] = min_max_scaler.fit_transform(df_process[mental_columns])
```

Encoding Frequency of Genre Columns

All the frequency [Genre] columns are ordinal data, with categorical values of 'Never', 'Rarely', 'Sometimes', and 'Very frequently'. These categories can be encoded into numerical values while preserving their order, ranging from 0 to 3.

```
# Encode categories into numerical value from 0-3
mapping = {'Never': 0, 'Rarely': 1, 'Sometimes': 2, 'Very frequently': 3}
genre_freq_columns = ['Frequency [Classical]', 'Frequency [Country]', 'Frequency [EDM]',

for column in genre_freq_columns:
    df_process[column] = df_process[column].replace(mapping)
```

Binary Data Transformation

Similarly, the columns 'While working', 'Instrumentalist', 'Composer', 'Exploratory' and 'Foreign languages' contain binary categorical data with values of 'Yes' and 'No'. These values can be encoded as 0 and 1, respectively.

```
# Encode binary categorical data
mapping = {'No': 0, 'Yes': 1}
binary_categorical_columns = ['While working', 'Instrumentalist', 'Composer', 'Exploratory', 'Foreign languages'

for column in binary_categorical_columns:
    df_process[column] = df_process[column].replace(mapping)
```

Handling Nominal Categorical Data

Lastly, for the remaining columns ‘Primary streaming service’, ‘Fav genre’ and ‘Music effects’, they represent nominal categorical data where there is no ordinal relationship among the values. To handle these categorical variables, the one-hot encoding method is applied. This method creates a new binary column for each category of the variable, effectively converting categorical data into numerical format suitable for our models.

Data columns	Data Type	Encoding Method
Age, Hours per day, BPM	Numeric	Standard Scaling
Anxiety, Depression, Insomnia, OCD	Numeric	Min-Max Scaling
Frequency [Genre]	Ordinal	Ordinal Encoding
While working, instrumentalist, composer, exploratory, foreign languages	Binary categorical	Binary Encoding
Primary streaming service, Fav genre, Music effects	Nominal	One-Hot Encoding

The pre-processed data:

	Age	Hours per day	BPM	Frequency [Classical]	Frequency [Country]	Frequency [EDM]	Frequency [Folk]	Frequency [Gospel]	Frequency [Hip hop]	Frequency [Jazz]	...	Fav genre_Rap
2	-0.581585	0.095959	0.261316	0	0	3	0	0	1	1	...	0
3	3.101224	-0.391818	-1.191827	2	0	0	1	2	0	3	...	0
4	-0.581585	0.095959	-0.495529	0	0	1	0	1	3	0	...	0
5	-0.581585	0.421143	-1.131279	1	2	0	0	0	2	3	...	0
6	-0.581585	-0.229226	-1.736755	2	0	1	2	1	1	2	...	0

Fav genre_Rock	Fav genre_Video game music	Exploratory_No	Exploratory_Yes	Foreign languages_No	Foreign languages_Yes	Music effects_Improve	Music effects_No effect	Music effects_Worsen
0	1	1	0	0	1	0	1	0
0	0	0	1	0	1	1	0	0
0	0	0	1	1	0	1	0	0
0	0	0	1	0	1	1	0	0
0	1	0	1	0	1	1	0	0

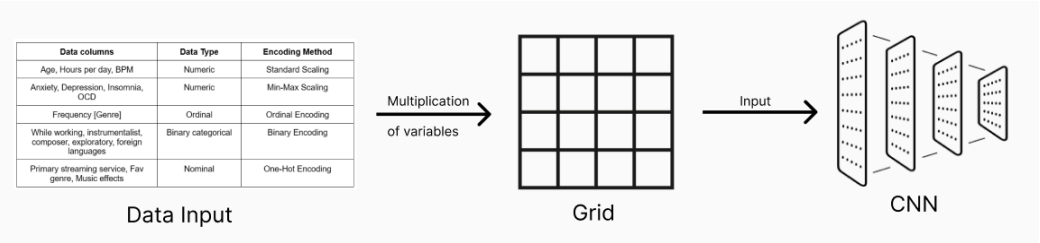
D. Model Exploration 1: Convolutional Neural Network (CNN)

1. Overview of the Approach

In our exploration of deep learning models to analyse the impact of music genres on mental health, we initially experimented with an unconventional approach by employing Convolutional Neural Networks (CNNs). CNNs are typically known for their performance in image processing and analysis tasks, where they excel in interpreting the spatial hierarchy in data.

Rationale Behind Using CNNs

The primary motivation for using CNNs was their proficiency in handling data in a grid format, where the relationship between adjacent data points is crucial. CNNs operate by assessing not just individual data values but also by considering the context provided by neighbouring values. This capability made them a compelling choice for attempting to recognise patterns and relationships within our music and mental health dataset. This could prove to be beneficial as we take into consideration how music genres are not completely distinct from one another and often overlap.



Architecture of the CNN model

2. Inputs and Outputs

Inputs

The inputs include all the columns from the dataset in these 4 categories:

1. Demographic and General Information
2. Music Listening Habits
3. Specific Musical Preferences
4. Mental Health Metrics

The complete inputs are:

```
'Age', 'Hours per day', 'BPM', 'Frequency [Classical]',
'Frequency [Country]', 'Frequency [EDM]', 'Frequency [Folk]',
'Frequency [Gospel]', 'Frequency [Hip hop]', 'Frequency [Jazz]',
'Frequency [K pop]', 'Frequency [Latin]', 'Frequency [Lofi]',
'Frequency [Metal]', 'Frequency [Pop]', 'Frequency [R&B]',
'Frequency [Rap]', 'Frequency [Rock]', 'Frequency [Video game music]',
'Anxiety', 'Depression', 'Insomnia', 'OCD',
'Primary streaming service_Apple Music',
'Primary streaming service_I do not use a streaming service.',
'Primary streaming service_Other streaming service',
'Primary streaming service_Pandora',
'Primary streaming service_Spotify',
'Primary streaming service_YouTube Music', 'While working_No',
'While working_Yes', 'Instrumentalist_No', 'Instrumentalist_Yes',
'Composer_No', 'Composer_Yes', 'Fav genre_Classical',
'Fav genre_Country', 'Fav genre_EDM', 'Fav genre_Folk',
'Fav genre_Gospel', 'Fav genre_Hip hop', 'Fav genre_Jazz',
'Fav genre_K pop', 'Fav genre_Latin', 'Fav genre_Lofi',
'Fav genre_Metal', 'Fav genre_Pop', 'Fav genre_R&B', 'Fav genre_Rap',
'Fav genre_Rock', 'Fav genre_Video game music', 'Exploratory_No',
'Exploratory_Yes', 'Foreign languages_No', 'Foreign languages_Yes',
```

Outputs

How does a person's favourite genre affect their mental health: No effect, Improvement, or Worsening.

3.Transposing Data into a Matrix

To adapt our dataset for a CNN, we transposed the data into a matrix format. This transformation was crucial as it allowed us to treat the dataset similarly to an image, where each element's position in the matrix could help the model understand the data's structure and underlying patterns.

Use the processed data :df_encoded

```
df_cnn = pd.read_csv('encoded.csv')
```

Constructs a feature grid:

```
feature_grid = pd.DataFrame(new_columns)

for col in target_columns:
    feature_grid[col] = target[col]
```

[5 rows x 515 columns]
Total rows in feature grid: 611

Calculates the suggested dimensions for reshaping

```
def find_factors_close_to_square(n):
    factors = [(i, n // i) for i in range(1, int(n**0.5) + 1) if n % i == 0]
    return sorted(factors, key=lambda x: abs(x[0]-x[1]))[0]

grid_depth = len(other_columns.columns)

if 512 % grid_depth == 0:
    sub_features = 512 // grid_depth
    height, width = find_factors_close_to_square(sub_features)
```

Suggested grid dimensions: Depth=32, Height=4, Width=4

Reshape Data

```
X_reshaped = X.reshape(-1, 32, 4, 4)
```

Reshapes the feature grid (X) into a 4D tensor with dimensions (-1, 32, 4, 4), presumably to fit the expected input shape of a convolutional neural network (CNN).

Original X shape: (611, 512)
After Data Reshape: torch.Size([64, 32, 4, 4])

4. Implementing the Model

Our CNN model consists of two convolutional layers followed by two fully connected layers.

```
def __init__(self):
    super(MusicEffectCNN, self).__init__()
    self.layer1 = nn.Sequential(
        nn.Conv2d(32, 64, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2)
    )
    self.flatten = nn.Flatten()
    self.fc1 = nn.Linear(64 * 2 * 2, 128)
    self.fc2 = nn.Linear(128, 3)
```

We applied a 2D convolutional operation with a kernel size of 3x3, processing 32 input channels and producing 64 output channels. Padding is applied to maintain the spatial dimensions of the input.

Here are the “Base” hyperparameter values we used for initial modelling:

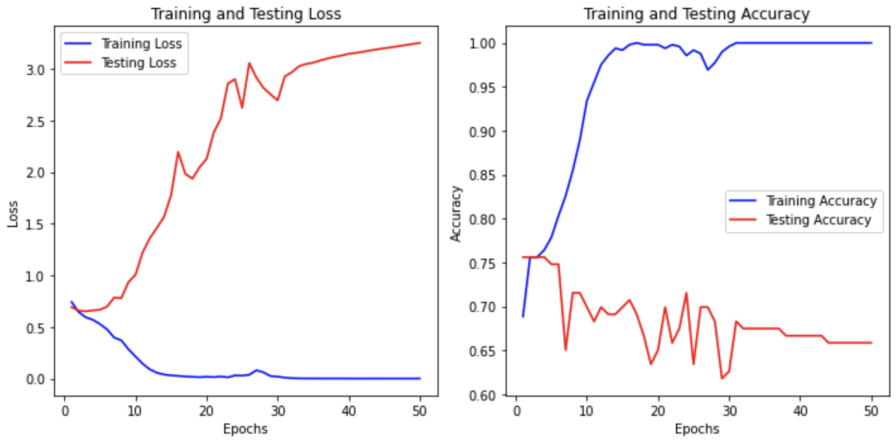
- Activation Function: ReLu
- Pooling Operation: 2x2 max pooling
- Loss Function: Cross-entropy loss
- Number of Epochs: 50
- Optimizer: Adam
- Learning Rate: 0.001

In addition, we split the data into an 80-20 train-test ratio:

```
X_train, X_test, y_train, y_test = train_test_split(X_tensor, y_tensor, test_size=0.2, random_state=42)
```

5. Initial model performance

Epoch [50/50], Train Loss: 0.0003, Train Acc: 1.0000, Test Loss: 3.2527, Test Acc: 0.6585



It's evident that the model is overfitting, resulting in an increase in loss for the test data and a significant gap between the training and test accuracies. The accuracy for the training data reaches 100% after 30 epochs, indicating that the model has essentially memorized the training data and is unable to generalize well to unseen examples.

6.Hyperparameters Tuning

To address the above issue, firstly, we attempted to adjust the hyperparameters.

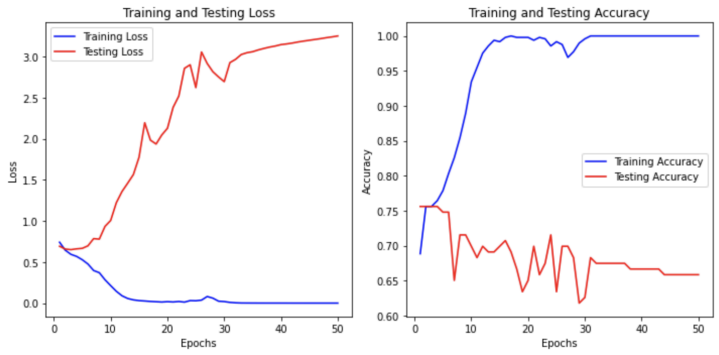
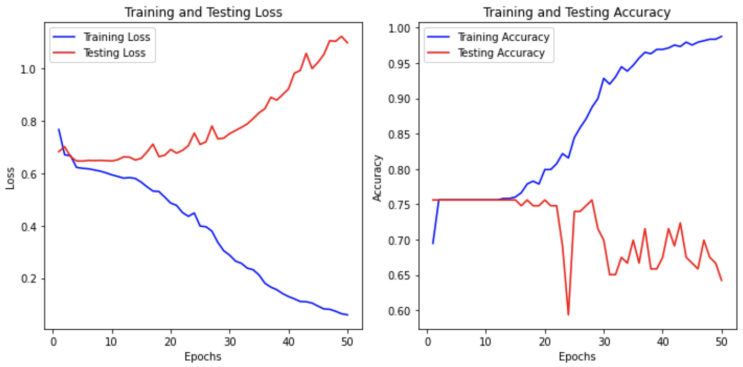
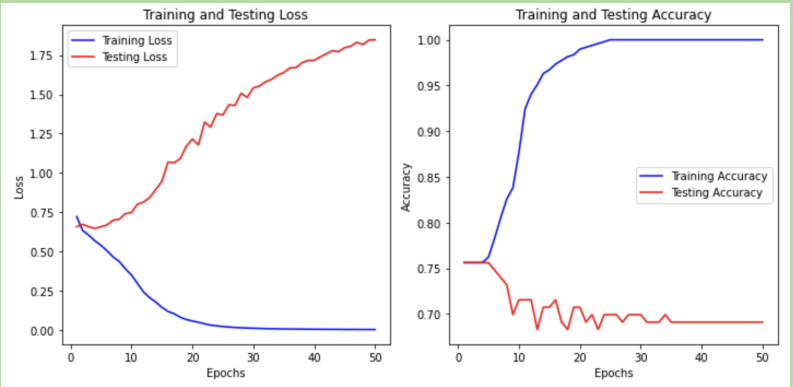
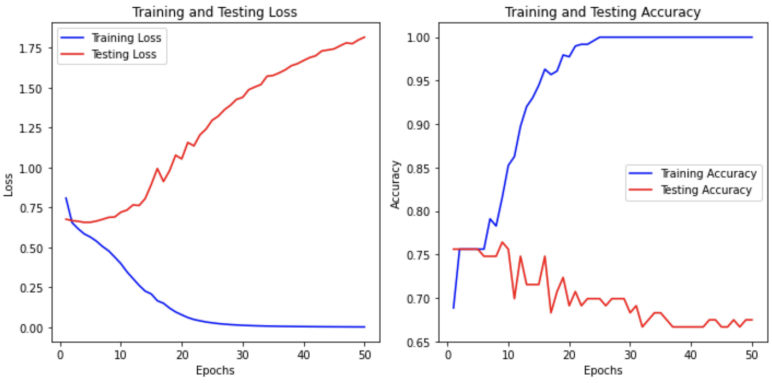
Here are the "Base" hyperparameter values we used for initial modelling:

- Activation Function: ReLu
- Pooling Operation: 2x2 max pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

Activation Function Tuning:

- Activation Function: ReLu
- Pooling Operation: 2x2 max pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

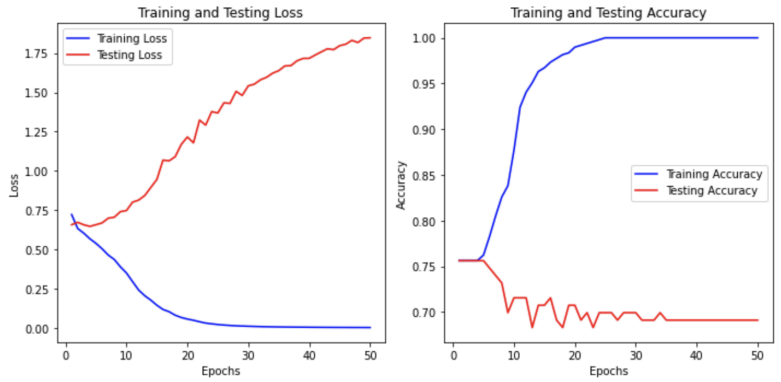
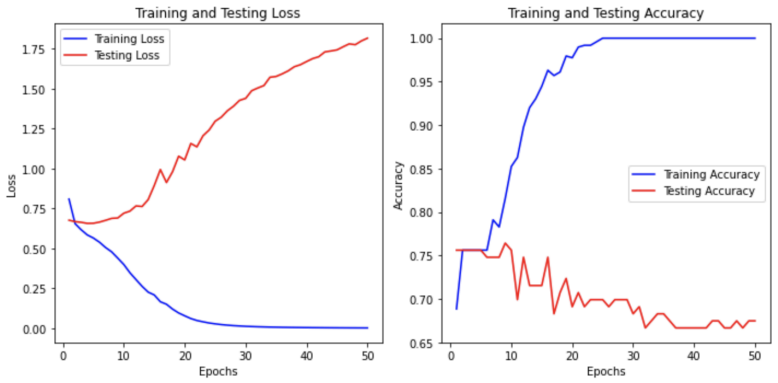
Activation Function	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs

ReLu	1.0000	0.6585	
Sigmoid	0.9877	0.6423	
Tanh	1.0000	0.6911	
LeakyReLu	1.0000	0.6748	

After the experiment, we chose Tanh because of the highest test accuracy.

Pooling Operation Tuning:

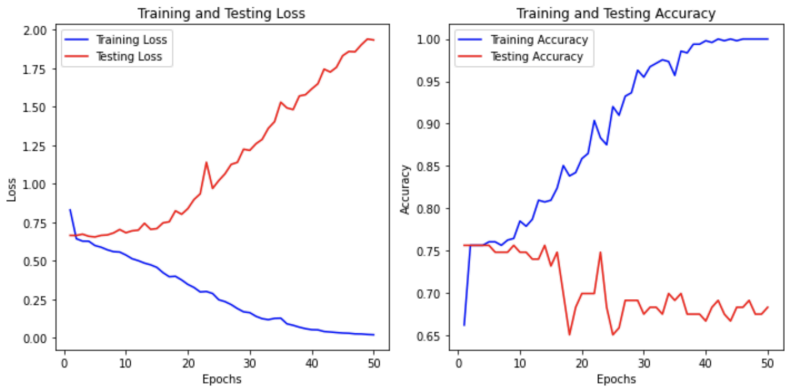
- Activation Function: Tanh
- Pooling Operation: 2x2 max pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

Pooling Operation	**Train Accuracy**	**Test Accuracy**	**Loss vs. Epochs & Accuracy vs. Epochs**
Max pooling	1.0000	0.6911	
Average Polling	1.0000	0.7073	
Global Polling	1.0000	0.7236	

After the experiment, we chose global polling because of the highest test accuracy.

Pooling Operation Tuning:

- Activation Function: Tanh
- Pooling Operation: Global pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

Loss Function	**Train Accuracy**	**Test Accuracy**	**Loss vs. Epochs & Accuracy vs. Epochs**
Cross-entropy loss	1.0000	0.7236	
Weighted Loss	1.0000	0.6748	
Focus Loss	1.0000	0.6829	

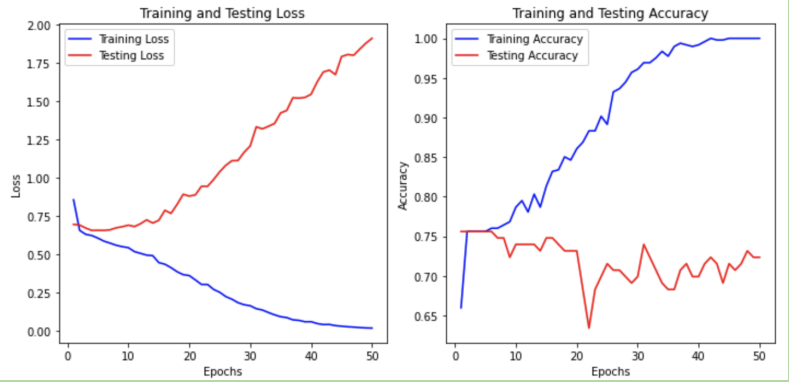
```python
class WeightedCrossEntropyLoss(nn.Module):
    def __init__(self, weights):
        super(WeightedCrossEntropyLoss, self).__init__()
        self.weights = weights

    def forward(self, inputs, targets):
        ce_loss = nn.CrossEntropyLoss(reduction='none')(inputs, targets)
        weighted_ce_loss = torch.mean(ce_loss * self.weights)
        return weighted_ce_loss
```

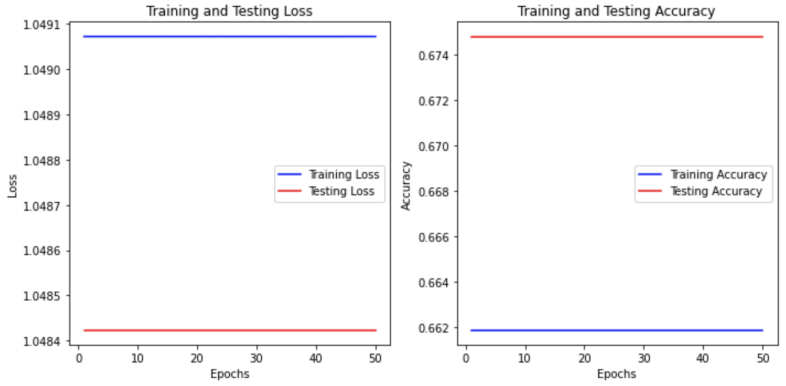
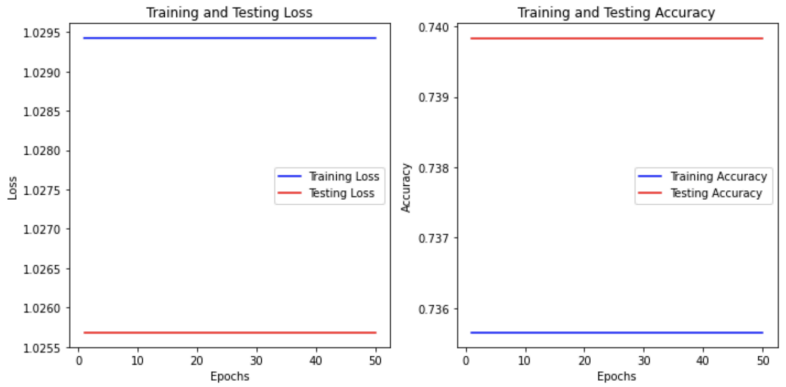
```python
class FocalLoss(nn.Module):
    def __init__(self, gamma=2, alpha=None):
        super(FocalLoss, self).__init__()
        self.gamma = gamma
        self.alpha = alpha

    def forward(self, inputs, targets):
        ce_loss = nn.CrossEntropyLoss(reduction='none')(inputs, targets)
        pt = torch.exp(-ce_loss)
        focal_loss = ((1 - pt) ** self.gamma * ce_loss)
        if self.alpha is not None:
            focal_loss = self.alpha * focal_loss
        return torch.mean(focal_loss)
```

After the experiment, we still chose Cross-entropy loss because of the highest test accuracy.

Pooling Operation Tuning:

- Activation Function: Tanh
- Pooling Operation: Global pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

Optimizer	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
Adam	1.0000	0.7236	

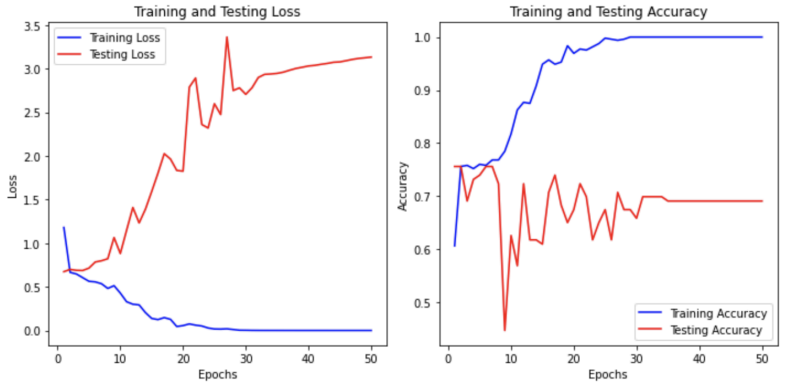
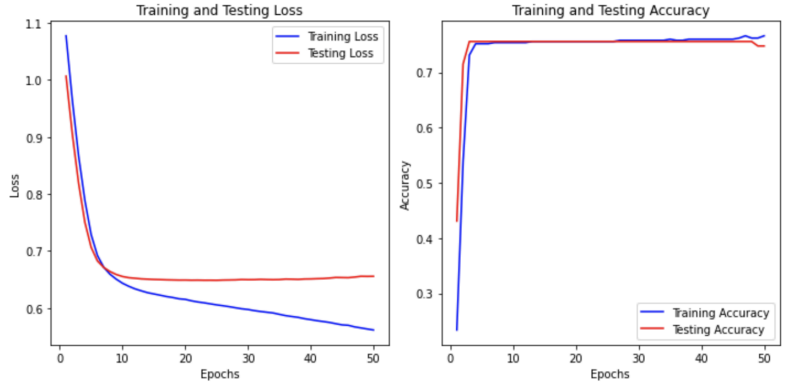
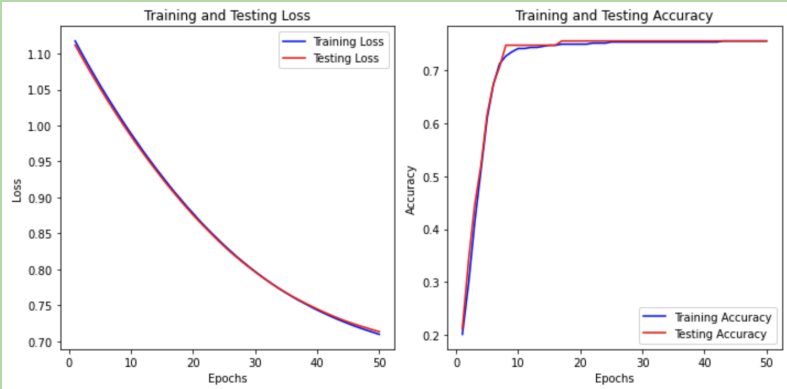
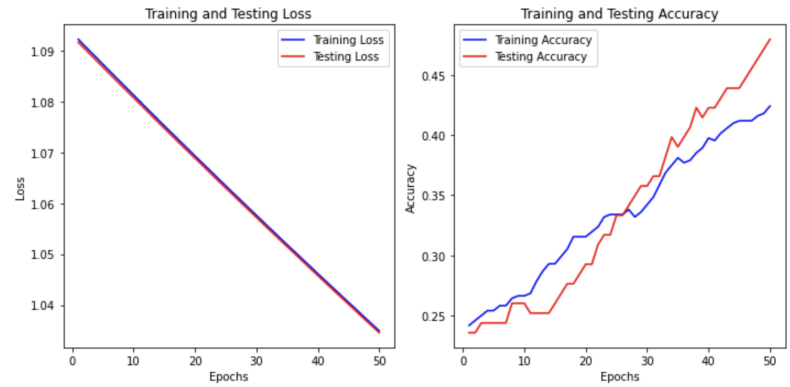
SGD	1.0000	0.6748	
RMSProp	1.0000	0.7398	

After the experiment, we still chose Adam because of the training progress.

Pooling Operation Tuning:

- Activation Function: Tanh
- Pooling Operation: Global pooling
- Loss Function: Cross-entropy loss
- Optimizer: Adam
- Learning Rate: 0.001

Learning Rate	**Train Accuracy**	**Test Accuracy**	**Loss vs. Epochs & Accuracy vs. Epochs**
0.001	1.0000	0.7236	

0.01	1.0000	0.6911	
0.0001	0.7664	0.7480	
0.00001	0.7561	0.7561	
0.000001	0.4242	0.4797	

After conducting the experiment, we chose lr=0.00001, because both the training accuracy and test accuracy remained consistent, with the test accuracy being the highest.

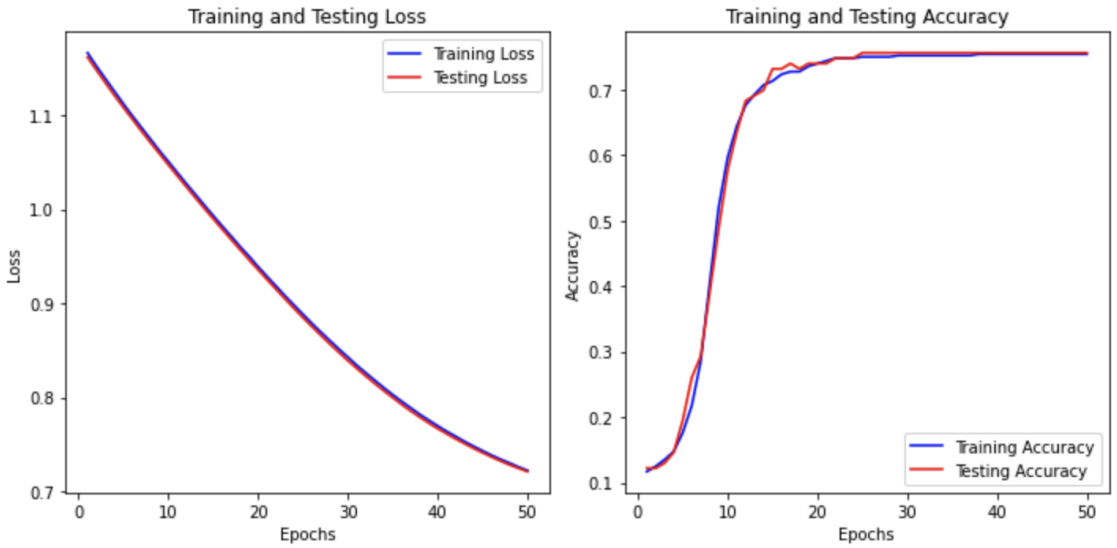
7. Final Results

Here are the final hyperparameter values we used for initial modelling:

- Activation Function: Tanh
- Pooling Operation: Global pooling
- Loss Function: Cross-entropy loss
- Number of Epochs: 30
- Optimizer: Adam
- Learning Rate: 0.00001

The Final Results:

Epoch [50/50], Train Loss: 0.7224, Train Acc: 0.7541, Test Loss: 0.7212, Test Acc: 0.7561



After turning, the model no longer overfitted, and the accuracy reached 75.61% from the initial of 65.86%.

8. Overcome Small Dataset Issue: K-fold Cross-validation

The dataset only has around 610 rows, which may lead to challenges such as limited model generalization, increased risk of overfitting, and difficulty in capturing complex patterns due to insufficient data samples.

To address this issue, we choose using K-Fold cross-validation. It systematically partitioned the data into multiple subsets for training and testing. In doing so, it maximizes the utilization of limited data by ensuring each data point contributes to both training and testing.

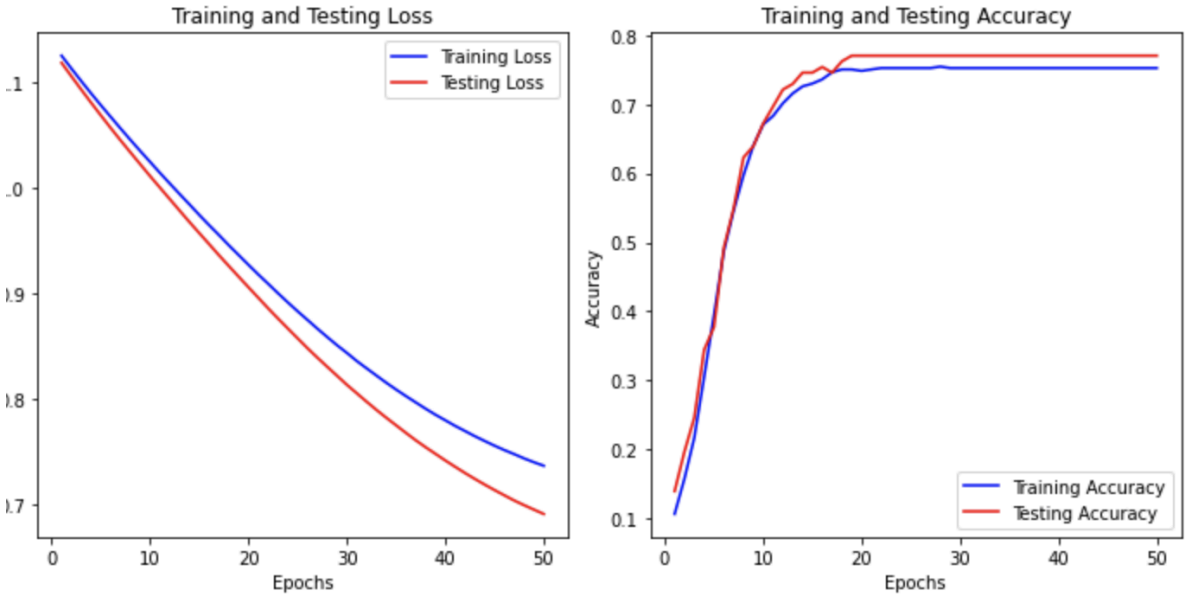
Implementation

To use, first we create an instance of the KFold class, set shuffle to True for random shuffling of data. Then, Iterate over each fold using enumerate(kf.split(X)), which yields the indices for the training and testing sets for each fold.

```
# Initialize KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Iterate over each fold
for fold, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Reshape X_train and X_test
    X_train_reshaped = X_train.reshape(-1, 32, 4, 4)
    X_test_reshaped = X_test.reshape(-1, 32, 4, 4)

    # Convert to PyTorch tensors
    X_train_tensor = torch.tensor(X_train_reshaped, dtype=torch.float32)
    y_train_tensor = torch.tensor(y_train, dtype=torch.long)
    X_test_tensor = torch.tensor(X_test_reshaped, dtype=torch.float32)
    y_test_tensor = torch.tensor(y_test, dtype=torch.long)

    # Create train and test datasets
    train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
    test_dataset = TensorDataset(X_test_tensor, y_test_tensor)

    # Create train and test data loaders
    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

The Results:

Epoch [50/50], Train Loss: 0.7366, Train Acc: 0.7526, Test Loss: 0.6908, Test Acc: 0.7705



After applying K-fold cross-validation, the accuracy increased from 75.61% to 77.05%!

9. Expand Dataset via CTGAN (Conditional Generative Adversarial Network)

Another approach to tackle the issue of small dataset size involves leveraging CTGAN to generate synthetic data based on the original dataset.

CTGAN is specifically tailored for generating synthetic tabular data, rendering it an apt choice for our survey data. It supports conditional generation, enabling the creation of synthetic data samples based on specific attributes or features, such as 'Primary streaming service' and 'Fav genre'.

Implement CTGAN

Please see file ”CTGAN and Data Preprocessing” for data synthesizing
Please see file ”cnn_model_CTGAN” for CNN model training and testing using expanded database

Set number of epoch to 200:

```
# Initialize the CTGAN model
ctgan = CTGAN(epochs=200)
```

State categorical data:

```
# List the names of the columns that are categorical in your dataset
categorical_columns = [
    'Primary streaming service',  # Options like Spotify, Apple Music, etc.
    'Hours per day',
    'While working',              # Yes/No or similar binary option
    'Instrumentalist',            # Yes/No or similar binary option
    'Composer',                   # Yes/No or similar binary option
    'Fav genre',                  # Genres like Classical, Jazz, etc.
    'Exploratory',                # Yes/No or similar binary option
    'Foreign languages',          # Yes/No or similar binary option
    'Frequency [Classical]',      # Likely categorical frequencies like Often, Rarely,
    'Frequency [Country]',        # -- do --
    'Frequency [EDM]',            # -- do --
    'Frequency [Folk]',           # -- do --
    'Frequency [Gospel]',         # -- do --
    'Frequency [Hip hop]',        # -- do --
    'Frequency [Jazz]',           # -- do --
    'Frequency [K pop]',          # -- do --
    'Frequency [Latin]',          # -- do --
    'Frequency [Lofi]',           # -- do --
    'Frequency [Metal]',          # -- do --
    'Frequency [Pop]',            # -- do --
    'Frequency [R&B]',            # -- do --
    'Frequency [Rap]',            # -- do --
    'Frequency [Rock]',           # -- do --
    'Frequency [Video game music]',# -- do --
    'Anxiety',
    'Depression',
    'Insomnia',
    'OCD',
    'Music effects'               # Effects like Improve, No effect, etc.
]
```

The scales for Anxiety, Depression, Insomnia, and OCD are categorical because the numbers range from 0 to 10.

Create 500 “fake” data:

```
# Create synthetic data
df_data_after = ctgan.sample(500)
```

Change the data type from float to integer:

```
df_data_after['Age'] = df_data_after['Age'].round().astype(int)
# df_data_after['Hours per day'] = df_data_after['Hours per day'].round().astype(int)
df_data_after['BPM'] = df_data_after['BPM'].round().astype(int)
```

Combine dataset:

```
# Combine original data with synthetic data
expanded_data = pd.concat([df_data_before, df_data_after], ignore_index=True)

# Optionally, save the expanded dataset to a CSV file
expanded_data.to_csv('expanded_dataset.csv', index=False)
```

Now the data has a size of 1111, up from 611:

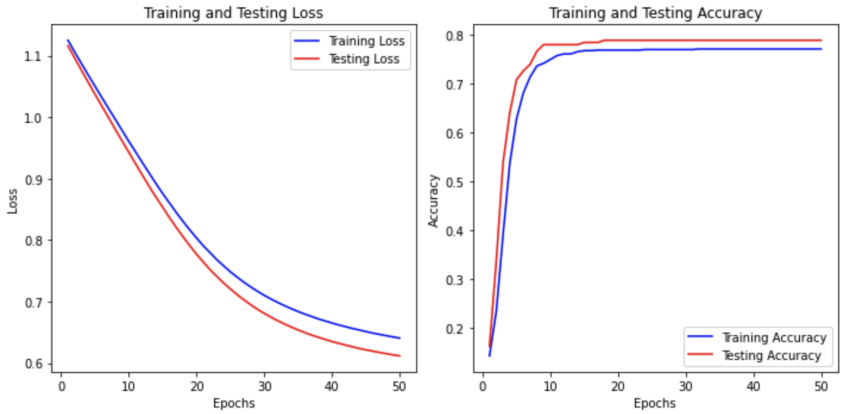
```
print("Number of rows:", df_encodedCTGAN.shape[0])
```

```
Number of rows: 1111
```

After comparing it to the original data, we concluded that the new dataset is suitable and ready for further analysis and application.

10. Training and Testing CNN Model Under Expanded Dataset

The Results:
Epoch [50/50], Train Loss: 0.6406, Train Acc: 0.7714, Test Loss: 0.6116, Test Acc: 0.7892



Although with the same hyperparameter setting, the accuracy increased from 77.05% to 78.92%.

11. Limitation and Challenge

Despite the theoretical appeal of using CNNs for this task, we encountered several challenges:

- **Data Nature:** Music and mental health data are inherently different from image data, lacking the spatial or temporal correlations that CNNs capitalise on.
- **Model Complexity:** CNNs might have been overly complex for this application, leading to difficulties in training and tuning the model effectively on our limited dataset.
- **Interpretability Issues:** Understanding how the CNN's detected features directly relate to music genre preferences and mental health impacts posed significant interpretability challenges.

E. Model Exploration 2: Multi-Layer Perceptron (MLP)

1. Overview of the Approach

Now that the feature vectors have been constructed, we explored into our second model, the Multi-Layer Perceptron (MLP). This model is well-suited for handling structured and tabular data, such as our dataset with different types of data. The layers of the MLP can effectively model interactions between all features, enabling it to identify complex patterns and relationships within the data through the input, hidden, and output layers.

2. Dataset

We will be using the expanded dataset for MLP Modeling.

Inputs

The inputs are the same as previous. The inputs include all the columns from the dataset in these 4 categories:

1. Demographic and General Information
2. Music Listening Habits
3. Specific Musical Preferences
4. Mental Health Metrics

The complete inputs are:

```
'Age', 'Hours per day', 'BPM', 'Frequency [Classical]',
'Frequency [Country]', 'Frequency [EDM]', 'Frequency [Folk]',
'Frequency [Gospel]', 'Frequency [Hip hop]', 'Frequency [Jazz]',
'Frequency [K pop]', 'Frequency [Latin]', 'Frequency [Lofi]',
'Frequency [Metal]', 'Frequency [Pop]', 'Frequency [R&B]',
'Frequency [Rap]', 'Frequency [Rock]', 'Frequency [Video game music]',
'Anxiety', 'Depression', 'Insomnia', 'OCD',
'Primary streaming service_Apple Music',
'Primary streaming service_I do not use a streaming service.',
'Primary streaming service_Other streaming service',
'Primary streaming service_Pandora',
'Primary streaming service_Spotify',
'Primary streaming service_YouTube Music', 'While working_No',
'While working_Yes', 'Instrumentalist_No', 'Instrumentalist_Yes',
'Composer_No', 'Composer_Yes', 'Fav genre_Classical',
'Fav genre_Country', 'Fav genre_EDM', 'Fav genre_Folk',
'Fav genre_Gospel', 'Fav genre_Hip hop', 'Fav genre_Jazz',
'Fav genre_K pop', 'Fav genre_Latin', 'Fav genre_Lofi',
'Fav genre_Metal', 'Fav genre_Pop', 'Fav genre_R&B', 'Fav genre_Rap',
'Fav genre_Rock', 'Fav genre_Video game music', 'Exploratory_No',
'Exploratory_Yes', 'Foreign languages_No', 'Foreign languages_Yes',
```

Outputs

How does a person's favourite genre affect their mental health: No effect, Improvement, or Worsening.

3. Implementing MLP

Define the MLP Model:

```
# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Split the dataset into an 80% training set and a 20% test set

Model:

```python
class MLP(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(MLP, self).__init__()
        self.layer1 = nn.Linear(input_dim, hidden_dim)
        self.layer2 = nn.Linear(hidden_dim, hidden_dim)
        self.output_layer = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = F.relu(self.layer1(x))
        x = F.relu(self.layer2(x))
        x = self.output_layer(x)
        return x
```

The network consists of three layers:

- 1st Layer: Takes the input data and processes it to a hidden form.
- 2nd Layer: Takes the output from the first layer, processes it further.
- 3rd Layer: Takes the output from the second layer and turns it into the final output format you need.

ReLU function which helps the model learn non-linear patterns.

```python
# Initialize Model, Optimizer, and Loss Function
input_dim = X.shape[1]
hidden_dim = 50
output_dim = y.shape[1]

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = MLP(input_dim, hidden_dim, output_dim).to(device)
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()
```

For the initial model, we set the size of the hidden layers as 50, the optimizer as Adam, and the loss function as Cross-entropy loss.

Here are the “Base” hyperparameter values we used for initial modelling:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu
- Loss Function: Cross-entropy loss

4. Initial Results:

Fold 5, Epoch 10, Train Loss: 0.1394, Train Acc: 0.9584, Test Loss: 0.2667, Test Acc: 0.8964

Both training and test loss generally decrease over epochs in each fold, which is a positive indicator that the model is learning and adapting to the features of the dataset.

In some cases, as the training loss decreases and training accuracy increases, where the model is performing well on the training data but not generalizing well to new data.

Similarly, accuracy, both on training and test datasets, generally increases with each epoch. This indicates the model's predictions are aligning better with the actual data over time.

5. Hyperparameters Tuning

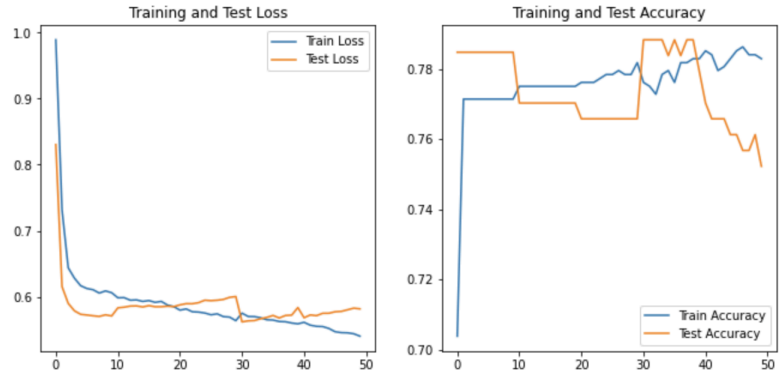
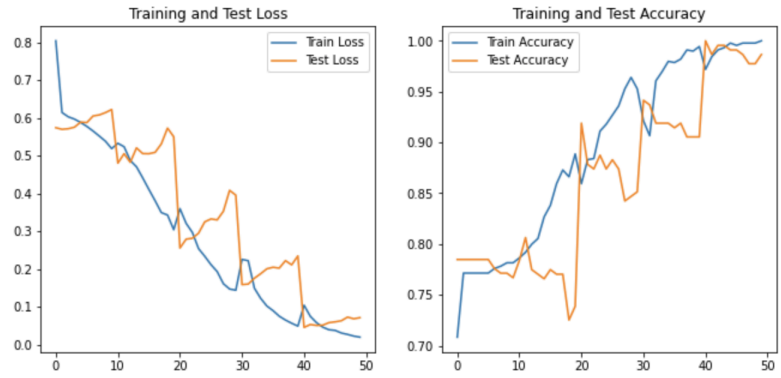
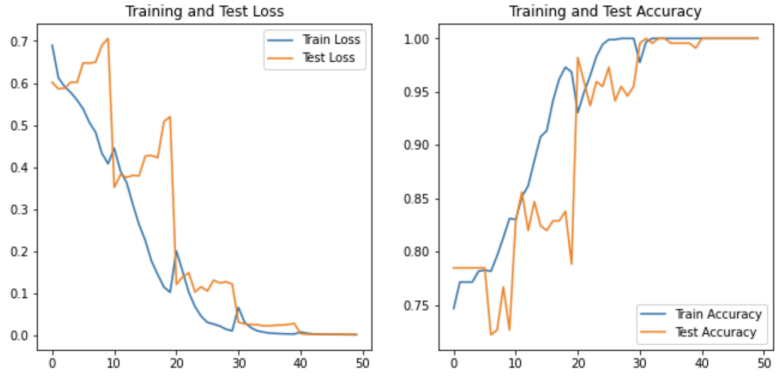
To address the above issue, firstly, we attempted to adjust the hyperparameters.

Here are the “Base” hyperparameter values we used for initial modelling:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu
- Loss Function: Cross-entropy loss

Number of Hidden Layers Tuning:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

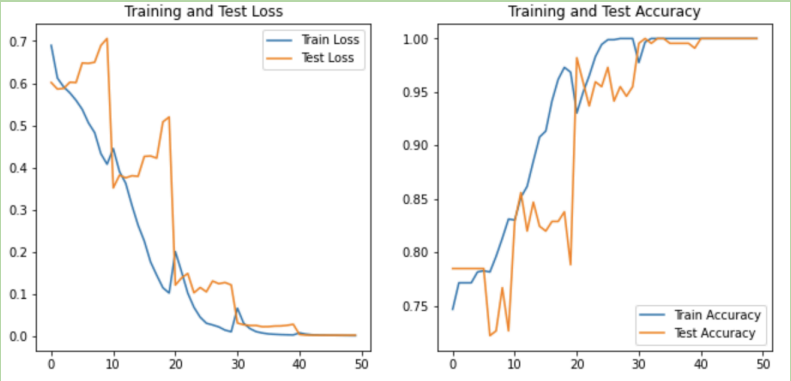
Number of Hidden Layers	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
50	0.9584	0.8964	
10	0.7829	0.7523	
100	1.0000	0.9865	
200	1.0000	1.0000	

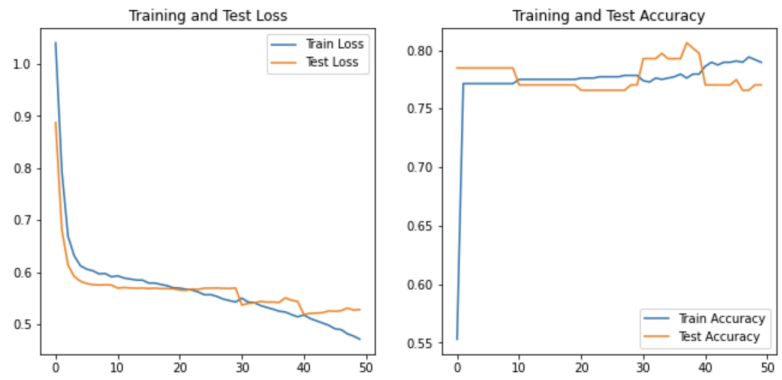
After the experiment, we chose the number of hidden layers =200 because of the highest test accuracy, 100%

.

Learning Rate Tuning:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

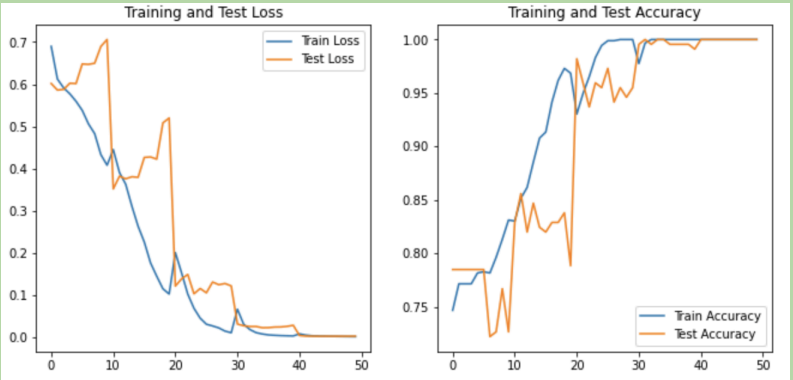
Learning Rate	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
0.001	1.0000	1.0000	
0.01	1.0000	0.9730	

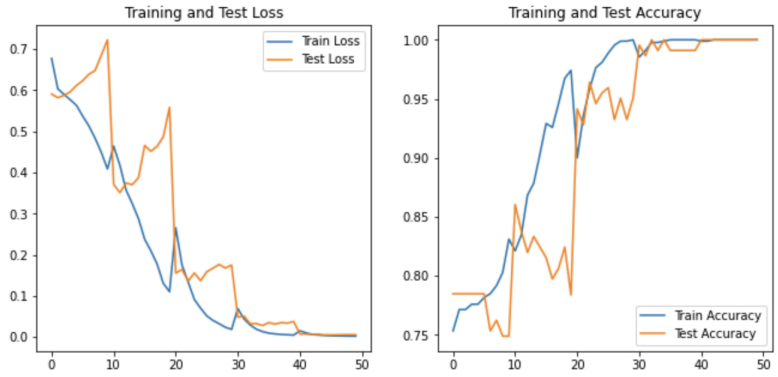
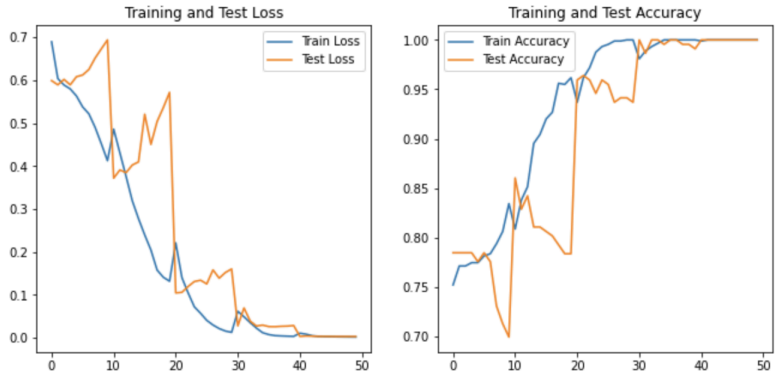
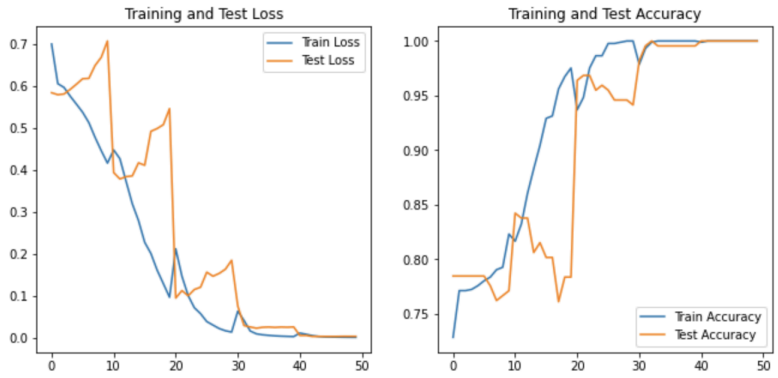
0.0001	0.7897	0.7703	

After the experiment, we still chose Learning Rate=0.001 because of the highest test accuracy.

Learning Rate Tuning:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

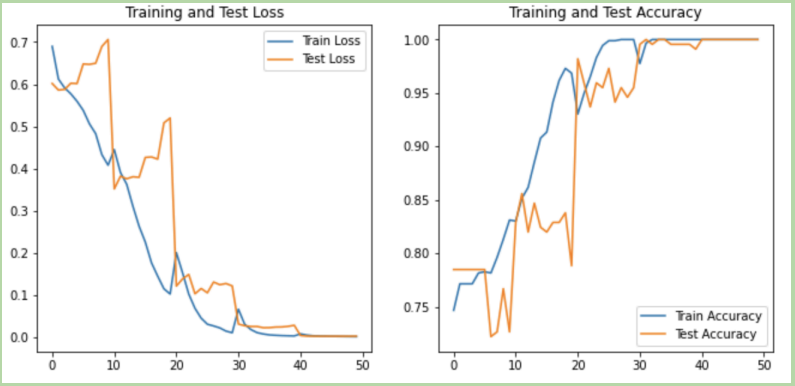
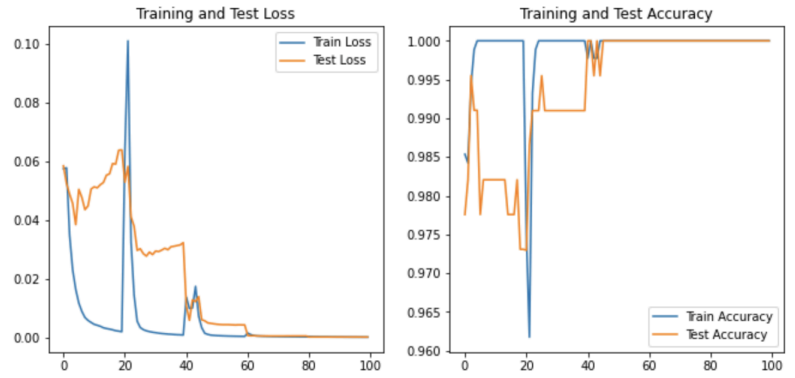
Batch Size	**Train Accuracy**	**Test Accuracy**	**Loss vs. Epochs & Accuracy vs. Epochs**
32	1.0000	1.0000	

8	1.0000	1.0000	
128	1.0000	1.0000	
1024	1.0000	1.0000	

After the experiment, we still chose Batch Size=32 because all have the same results.

Learning Rate Tuning:

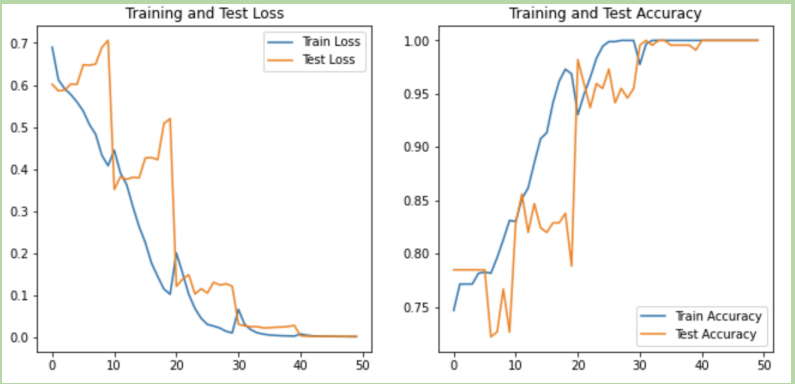
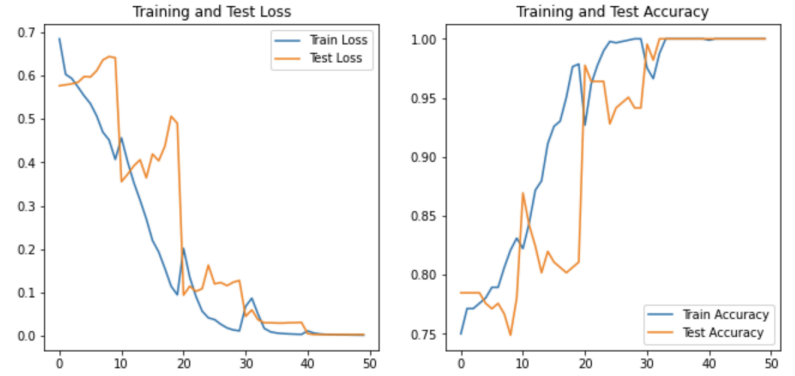
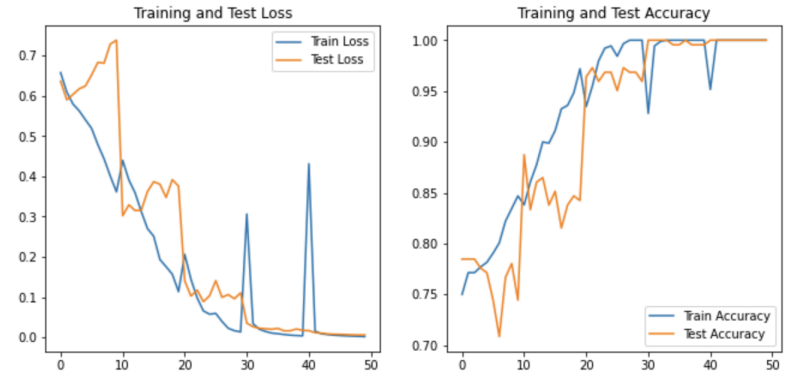
- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

Number of Epochs	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
10	1.0000	1.0000	
5	0.9944	0.9640	
20	1.0000	1.0000	

After the experiment, we still chose Number of Epoch=10 because of the highest text accuracy.

Learning Rate Tuning:

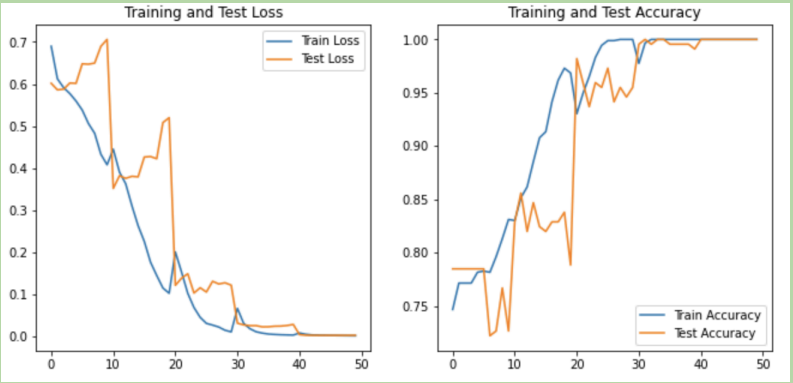
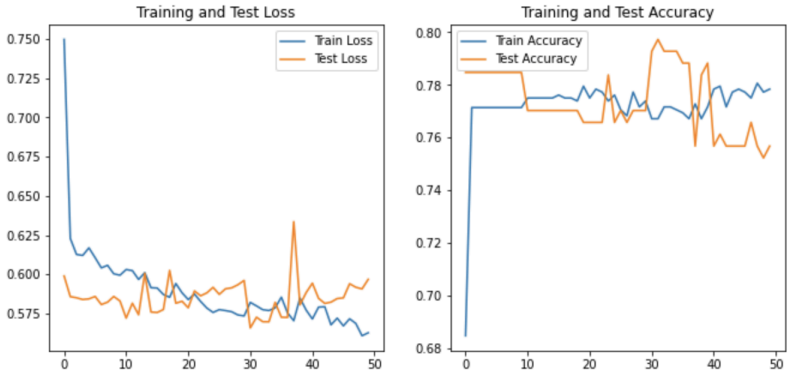
- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

Optimizer	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
Adam	1.0000	1.0000	
SGD	1.0000	1.0000	
RMSProp	1.0000	1.0000	

After the experiment, we still chose Adam as Optimizer because all have the similar results.

Learning Rate Tuning:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32
- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

Activation Function	Train Accuracy	Test Accuracy	Loss vs. Epochs & Accuracy vs. Epochs
ReLu	1.0000	1.0000	
Sigmoid	0.7784	0.7568	
Tanh	0.8931	0.8108	

After the experiment, we still chose ReLu as Activation Function because the highest test accuracy.

6. Final Results

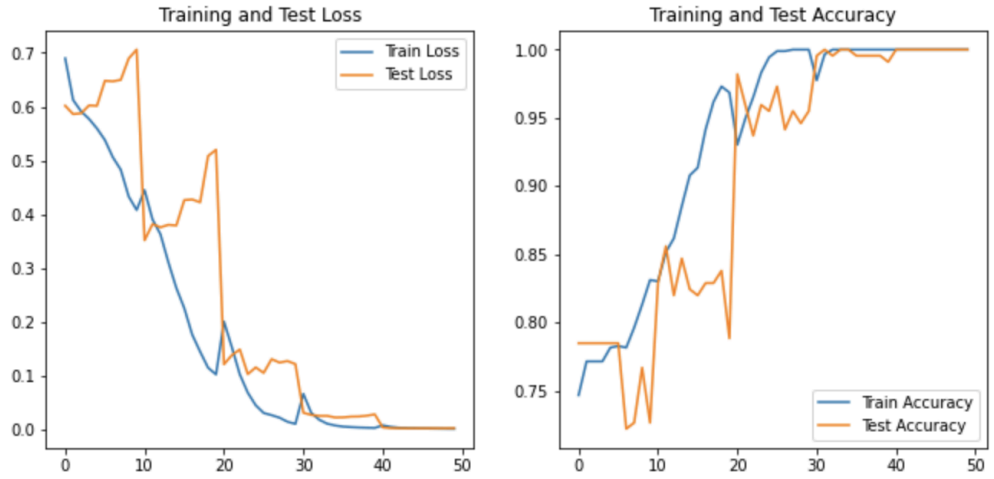
Here are the final hyperparameter values we used for initial modelling:

- Number of Hidden Layers: 50
- Learning Rate: 0.001
- Batch Size: 32

- Number of Epochs: 10
- Optimizer: Adam
- Activation Function: ReLu

Results:

Fold 5, Epoch 10, Train Loss: 0.0017, Train Acc: 1.0000, Test Loss: 0.0026, Test Acc: 1.0000



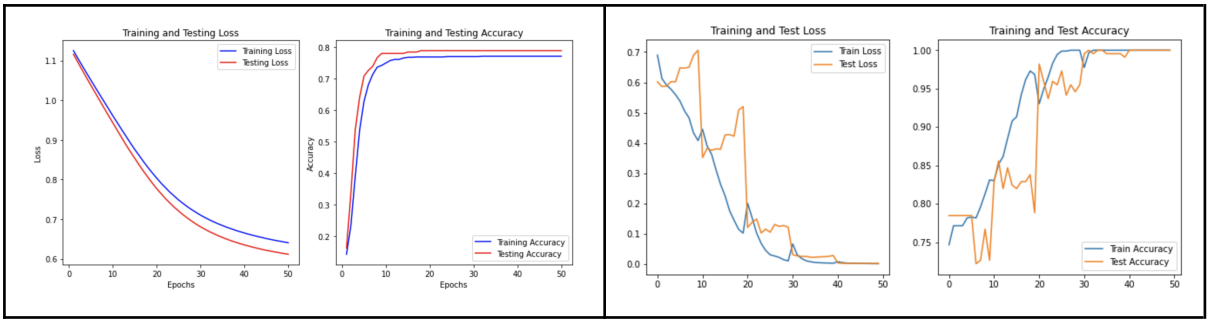
The 100% accuracy for both training and test data suggest that the model is performing very well and achieving perfect accuracy on both training and testing data.

However, it raises several concerns, including the possibility of overfitting, particularly if the test set is not large or representative enough, suggesting that the model may have learned the training data's noise and details rather than underlying patterns, affecting its performance on new data. Similarly, perfect accuracy might also be a sign of data leakage, where training and testing datasets inadvertently share information, or it might indicate that the dataset is too simple, with very distinct, easily separable classes, limiting the model's applicability to more complex, real-world tasks.

F. Conclusion

1.Model Comparison

CNN	MLP
Train Loss: 0.6406, Train Acc: 0.7714	Train Loss: 0.0017, Train Acc: 1.0000
Test Loss: 0.6116, Test Acc: 0.7892	Test Loss: 0.0026, Test Acc: 1.0000



The CNN shows a reasonable balance between training and testing accuracy, which typically suggests good generalization from training to unseen data. The training and testing losses are also relatively close, indicating that the model performs consistently across both sets.

The MLP, on the other hand, achieves perfect accuracy and extremely low loss on both training and testing datasets. While impressive, these results could suggest overfitting, despite the perfect test results. It's possible the MLP has memorized the training data, or the test set might not be challenging or diverse enough to test the model's generalization capability effectively.

2. Conclusion

Choosing a **MLP** for a simple dataset with straightforward relationships among features is a wise decision because MLPs are inherently simpler and more efficient to train on such data. The fully connected nature of MLPs makes them particularly adept at capturing linear relationships and interactions between features without the need for the complex architectures required for spatial or sequential data, such as those needed in CNNs or RNNs.

G. Final Thoughts

Lessons Learned

We understood that using CNN would be very ambitious as it is an unorthodox method. We wanted to try it out nonetheless. This initial attempt provided valuable insights into the limitations and potential of using advanced neural network architectures in unconventional domains. It highlighted the importance of matching the model architecture with the nature of the data and the specific requirements of the task.

Moving Forward

Building on these lessons, our future efforts will focus on exploring more suited models that align better with the dataset characteristics and the analytical goals of our project. This experience has underscored the need for adaptive model selection and continuous refinement in the field of AI-driven research in music therapy.

3. Implementation/Future work

A. Discussion

Challenges and Limitations of the Current Study:

Small Dataset Size
The study was conducted with a relatively small dataset comprising only 736 responses. This limited size restricts the robustness and generalizability of our deep learning model. With a small dataset, it is challenging to capture the full variability and complexity of music preferences and their psychological effects, which could lead to less reliable predictions and insights.

Reliance on Self-Reported Data
The mental health metrics (such as levels of anxiety, depression, insomnia, and OCD) were self-reported by respondents on a scale of 1 to 10, without external verification or clinical assessment. This method raises concerns regarding the accuracy and reliability of the data, as self-reported information can be subjective and influenced by the respondent's current mood or personal bias.

Potential for Confirmation Bias
Improvements in mental health conditions as a result of music listening were also self-reported, introducing a possibility of confirmation bias. Respondents may believe that music has a positive effect on their mental health, which could skew their responses to reflect expected outcomes rather than objective reality.

Individual Variations in Music Preference
Music preference is highly personal and varies widely among individuals. This variability means that even with a well-designed study, the results can differ significantly from one participant to another, making it difficult to draw broad conclusions about the effects of music genres on mental health.

Accuracy Concerns in Predictive Modeling
Given the subjective nature of the data and the inherent variations in individual music tastes and mental health states, creating a predictive model with high accuracy is challenging. The personal and subjective elements involved in music and mental health can lead to unpredictable model performance, especially when tested on new, unseen data.

Conclusion
The current study faces significant challenges that must be addressed to enhance the reliability and applicability of its findings. Future research would benefit from larger datasets, the inclusion of clinically validated mental health assessments, and more nuanced modelling techniques that can better account for individual differences in music preference and psychological impact. This approach will help minimise biases and improve the accuracy and relevance of the research in real-world settings.

B. Future work

Future Directions for Enhancing the Study on Music's Impact on Mental Health

To refine and expand our research on the effects of music on mental health, we have identified several strategic improvements and extensions that could significantly enhance the scope and accuracy of our findings:

Expanding the Dataset
A primary goal for future work would be to acquire a larger and more diverse dataset. A more extensive dataset would provide a more statistically robust foundation for our models, enabling us to capture a wider range of music preferences and mental health conditions. This expansion would also improve the generalisability of our results, making our conclusions more applicable to a broader population.

Enhanced Validation of Mental Health Metrics
Integrating clinically validated tools for assessing mental health conditions would be a crucial improvement. By moving beyond self-reported data to include expert assessments or standardised diagnostic tools, we can significantly reduce biases such as self-reporting errors and confirmation bias. This would provide a more reliable basis for evaluating the true impact of music on mental health.

Broadening the Range of Mental Health Issues
Exploring a wider array of mental health issues would allow us to address and potentially aid a larger segment of the population. By understanding how different types of music affect a variety of mental health conditions, we can tailor music-based interventions more effectively to meet individual needs.

Analysing Specific Songs and Their Components
Investigating which specific songs or song attributes (such as tempo, key, lyrics) most positively impact mental health could offer deeper insights into the mechanisms behind music therapy. This analysis would help identify the most therapeutic aspects of music, providing a targeted approach to treatment.

AI-Driven Personalized Song Generation
Looking ahead, we could explore the development of AI technologies to generate personalised music tracks tailored to an individual listener's mental health needs. By utilising AI in song generation, we could potentially create custom music therapies that are optimised for each user's specific emotional and psychological state, enhancing the therapeutic effectiveness of the music.

Conclusion
By implementing these enhancements, our future research could not only improve the accuracy and reliability of our findings but also broaden the impact of our work, offering new, innovative approaches to using music as a therapeutic tool for mental health. This expanded vision promises not just to advance our understanding of music's effects but also to translate these insights into practical, personalised interventions that can make a meaningful difference in people's lives.

4. References

1. *Ipsos. Singaporeans deem mental health as the biggest health problem. (2023, October 12).* https://www.ipsos.com/en-sg/singaporeans-deem-mental-health-biggest-health-problem

2. *Breaking down the most extreme BPMs in dance music - BOILER ROOM. (n.d.-b).* https://boilerroom.tv/article/extreme-bpms

3. *Wesseldijk, L.W., Ullén, F. & Mosing, M.A. The effects of playing music on mental health outcomes. Sci Rep 9, 12606 (2019).* https://doi.org/10.1038/s41598-019-49099-9

4. *Music & Mental Health Survey results. (2022, November 21). Kaggle.* https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results

5. *Wesseldijk, L.W., Ullén, F. & Mosing, M.A. The effects of playing music on mental health outcomes. Sci Rep 9, 12606 (2019).* https://doi.org/10.1038/s41598-019-49099-9