DJ BOX

Automatic playlist generation

資工四 108AC1026 林寧

Table of contents

01

Research Motivation and Goals

01 Research Motivation

Background music is crucial for setting the right atmosphere in various situations

Manually creating playlists

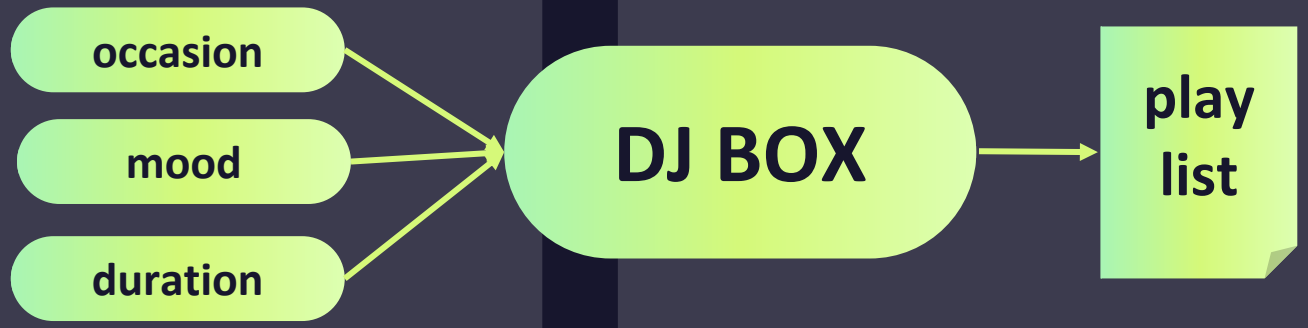
time-consuming, and may not fit the desired ambiance

Pre-existing playlist

standardized and may not cater to individual needs

01 Goals

Personalization



Input specific occasion, mood, duration

Generation

Customized playlist suited to conditions

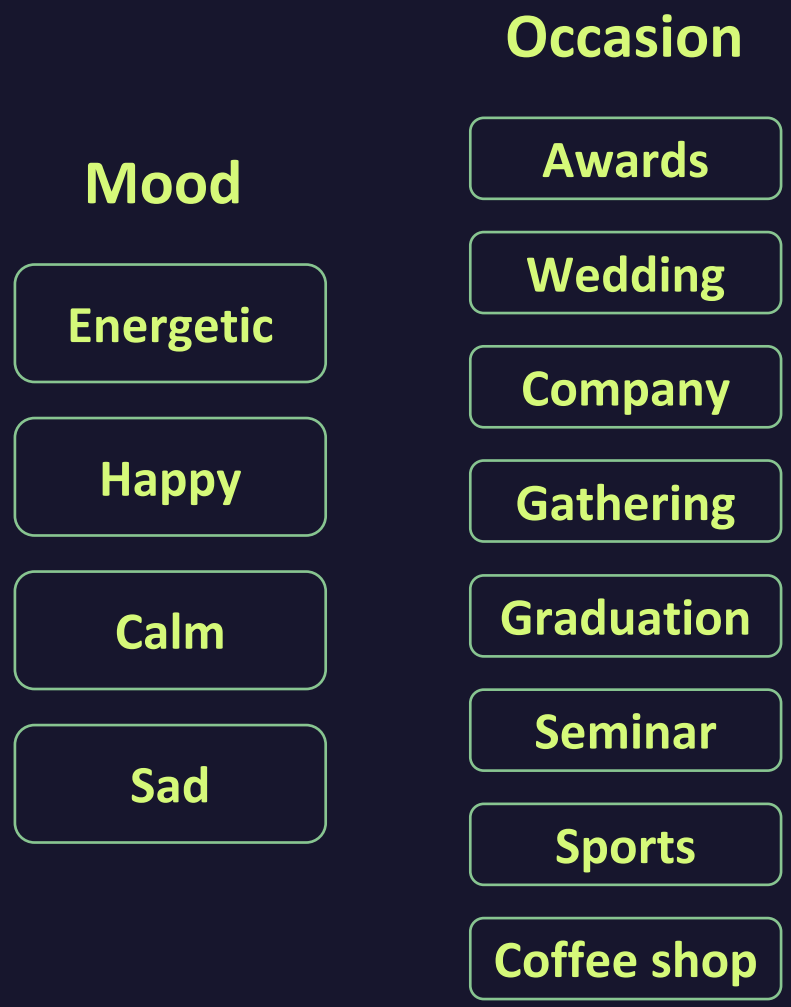
02

Methods and Steps

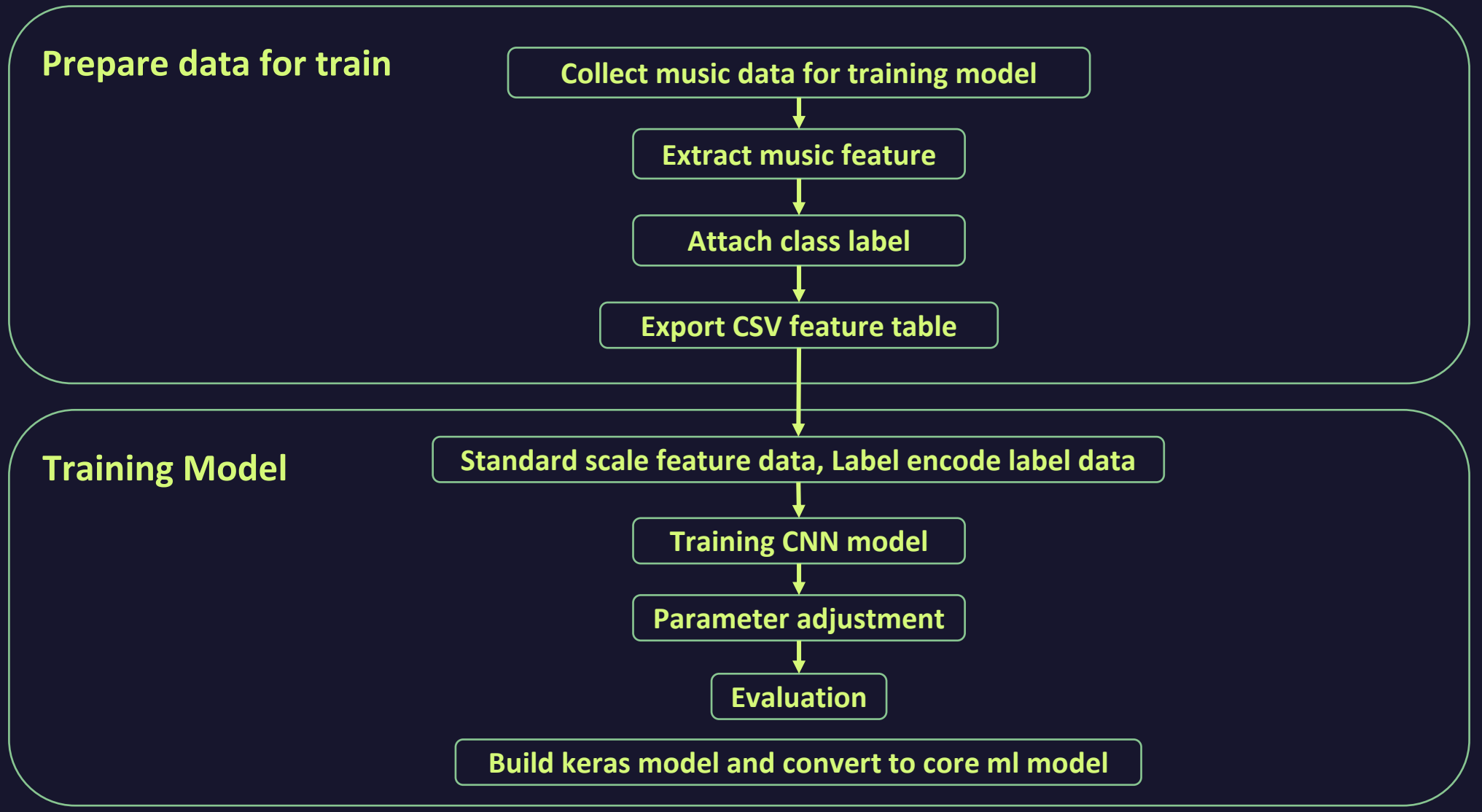
02 Methods and Steps

Classify music by occasion and mood

Classifying music based on occasion and mood can help people better choose appropriate music for different contexts.



02 Methods and Steps



02 Methods and Steps

Prepare data for app

- Collect music data for firebase
- Extract music feature
- Attach class label
- Import into firebase

Implement app
(predict song, generate list)

- Read data from firebase
- Convert data type to mlmutiarray type
- Predict song occasion, mood and fetch duration time
- Generate song list

02 Methods and Steps

Collecting a large amount of music data

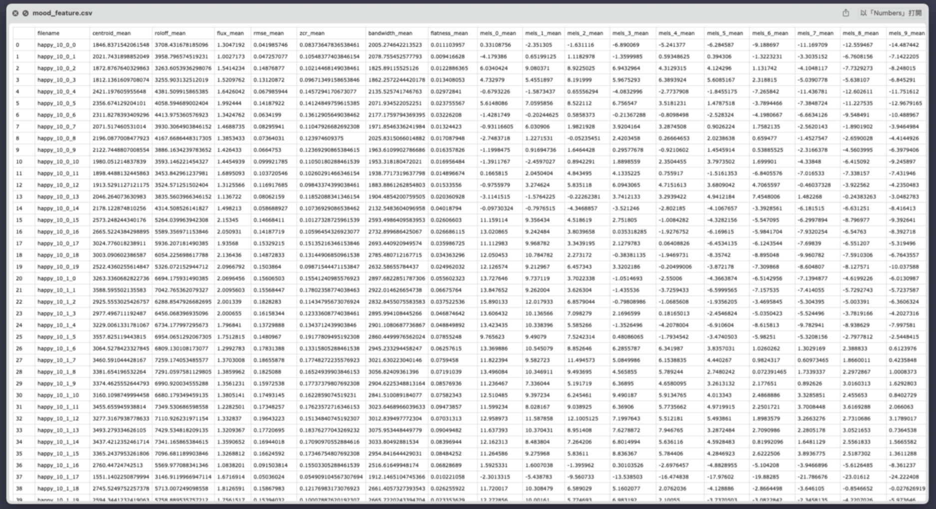
Manually collect music data according to different scenes and moods on YouTube.

02 Methods and Steps

Extract music feature and attach label

Extract feature using Python and the Librosa library, such as Mel Spectrogram, spectral centroid, and many others, also attach label.

songs_feature.csv



02 Methods and Steps

Training data

by CNN Model

Training mood dataset and occasion dataset into cnn model by tensorflow keras.

```
# 建立神經網絡架構
model=keras.models.Sequential([
    keras.layers.Flatten(input_shape=(X.shape[1],)),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(8, activation='softmax'),
])
```

02 Methods and Steps

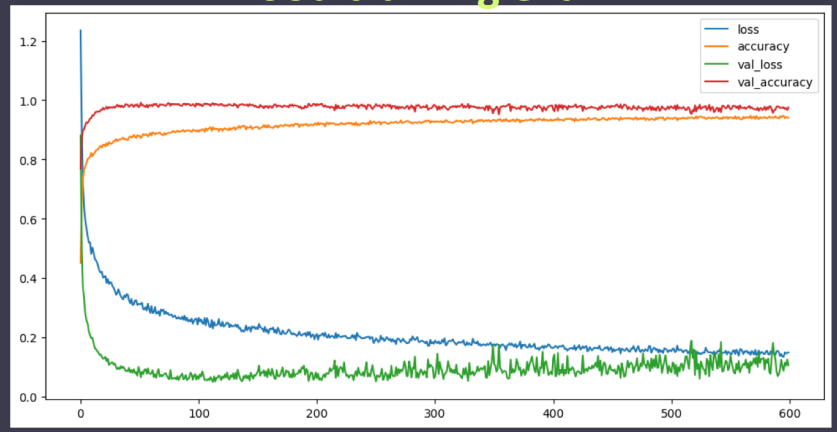
Mood CNN Model

The best accuracy is: 97%

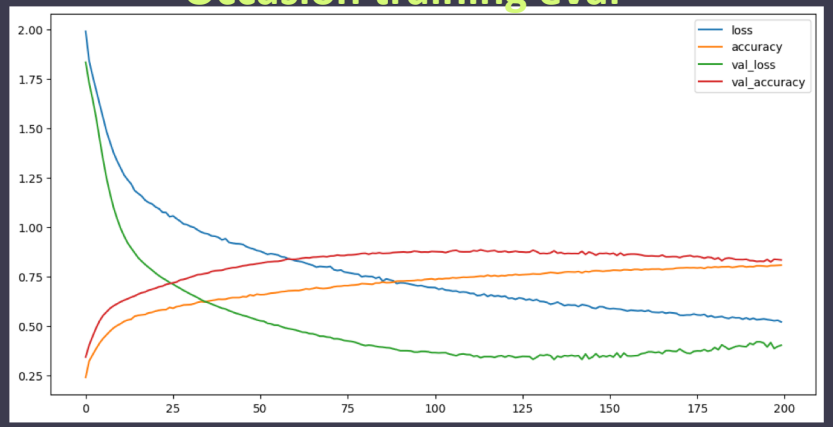
Occasion CNN Model

The best accuracy is: 80%

Mood training eval



Occasion training eval



02 Methods and Steps

Feature Extraction adjust

```python
# 提取質心特徵 (spectral centroid)
centroid = librosa.feature.spectral_centroid(y=y, sr=SR, n_fft=N_FFT, hop_length=HOP_LEN).ravel()
centroid_mean = centroid.mean()

# 提取衰減特徵 (spectral rolloff)
roloff = librosa.feature.spectral_rolloff(y=y, sr=SR, n_fft=N_FFT, hop_length=HOP_LEN).ravel()
roloff_mean = roloff.mean()

# 提取過零率特徵 (zero-crossing rate)
zcr = librosa.feature.zero_crossing_rate(y=y, frame_length=N_FFT, hop_length=HOP_LEN).ravel()
zcr_mean = zcr.mean()

# 提取均方根能量特徵 (root mean square, RMS)
rmse = librosa.feature.rms(y=y, frame_length=N_FFT, hop_length=HOP_LEN).ravel()
rmse_mean = rmse.mean(

# 提取起點強度特徵 (onset strength)
flux = librosa.onset.onset_strength(y=y, sr=SR).ravel().mean()
flux_mean = flux.mean()

# 提取頻譜帶寬特徵 (spectral bandwidth)
bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=SR, n_fft=N_FFT, hop_length=HOP_LEN).ravel()
bandwidth_mean = bandwidth.mean()

# 提取頻譜平坦度特徵 (spectral flatness)
flatness = librosa.feature.spectral_flatness(y=y, n_fft=N_FFT, hop_length=HOP_LEN).ravel()
flatness_mean = flatness.mean()

# 提取色度特徵 (Chromagram)
chroma = librosa.feature.chroma_stft(

# 提取梅爾頻譜特徵 (Mel spectrogram)
mels = librosa.feature.melspectrogram
mels = librosa.power_to_db(mels)
```

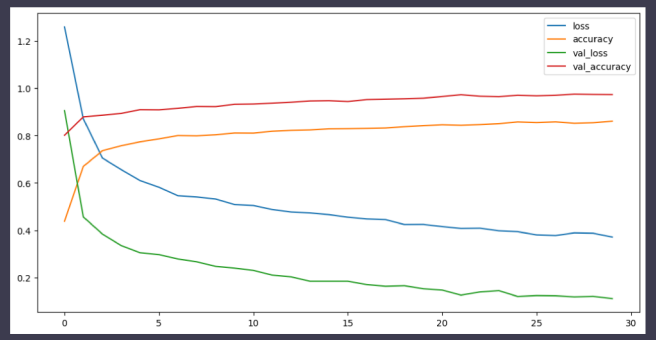
```python
# 將提取到的特徵儲存
if len(mels.T) == EXP_LEN:

    features['filename'] = f'{filename}'
    features['centroid_mean'] = centroid_mean
    features['roloff_mean'] = roloff_mean
    features['zcr_mean'] = zcr_mean
    features['rmse_mean'] = rmse_mean
    features['flux_mean'] = flux_mean
    features['bandwidth_mean'] = bandwidth_mean
    features['flatness_mean'] = flatness_mean
    for idx, v_mels in enumerate(mels):
        features[f"mels_{idx}_mean"] = v_mels.ravel().mean()

    for idx, v_chroma in enumerate(chroma):
        features[f"chroma_{idx}_mean"] = v_chroma.ravel().mean()
    features['label'] = label
```

Model adjust

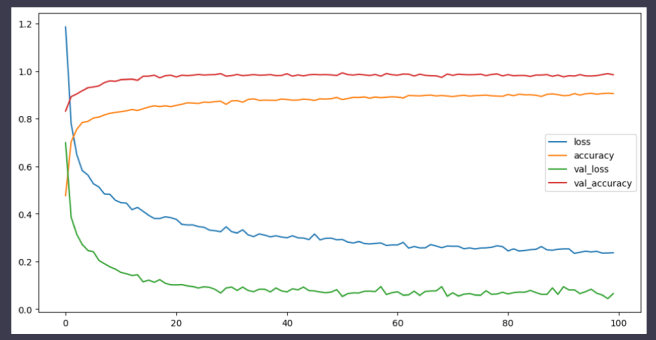
```python
# 建立神經網絡架構
model = keras.Sequential([

    # 輸入層
    keras.layers.Flatten(input_shape=(X_train.shape[1], X_train.shape[2])),

    # 第一層隱藏層
    keras.layers.Dense(512, activation="relu"),

    # 第二層隱藏層
    keras.layers.Dense(256, activation="relu"),

    # 第三層隱藏層
    keras.layers.Dense(64, activation="relu"),

    # 輸出層
    keras.layers.Dense(classes, activation="softmax")
])
```

02 Methods and Steps

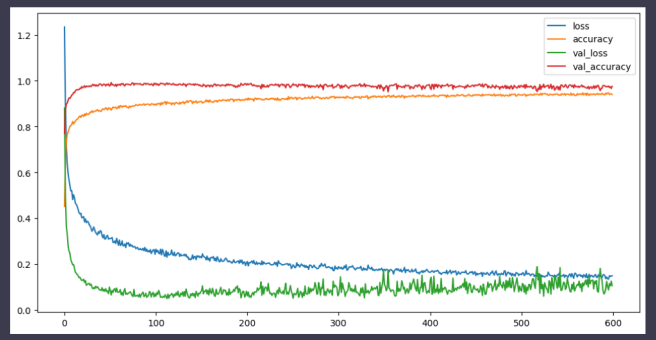
Mood training eval version1



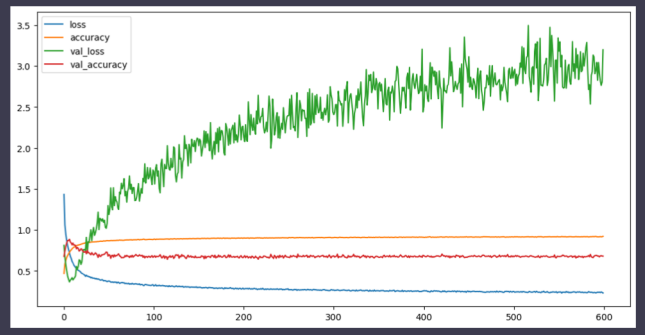
Mood training eval version2



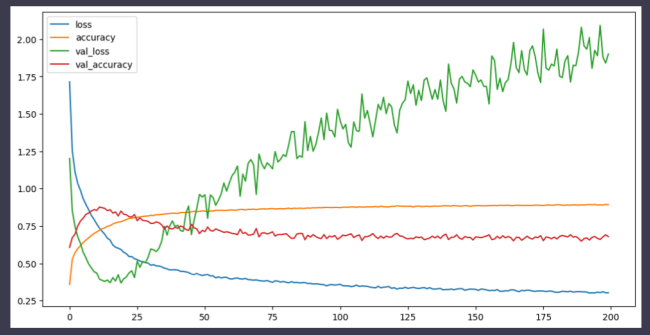
Mood training eval version3



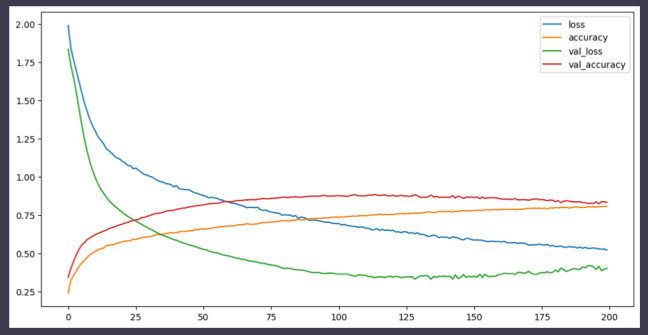
Occasion training eval version1



Occasion training eval version2



Occasion training eval version3

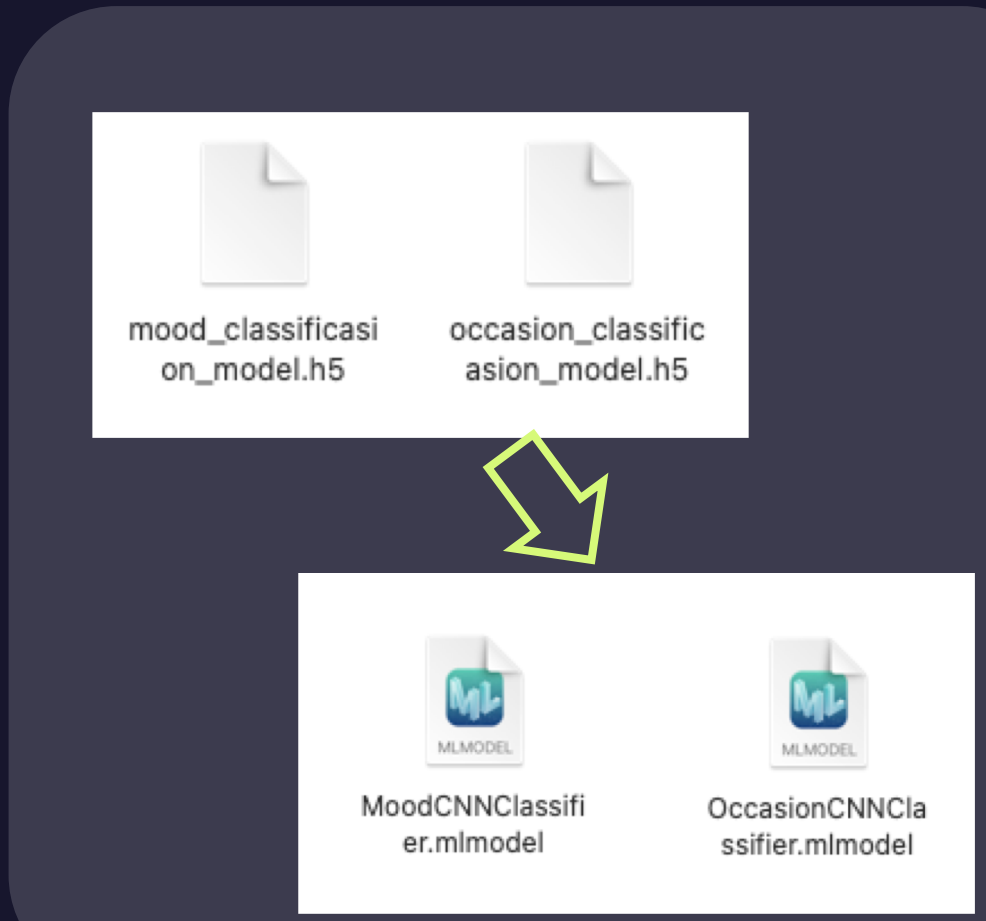


02 Methods and Steps

Convert Model

to CoreML Model

Use coremltoos to convert model.



02 Methods and Steps

Firebase Data ⇨ Convert into MLMultiArray ⇨ OccasionCNN Classifier ⇨ MoodCNN Classifier ⇨ Fetch Duration

Use CoreML on IOS app

Implementing a user interface on the iOS platform.

```
// 篩選出符合 occasion condition 的 song
if let occasionPrediction = ClassifyOccasion(tensorInput) {
    if occasionPrediction.Identity[occasion.identity()].floatValue > 0.7 {
        print("occasion > \(song.title):\(occasionPrediction.Identity[occasion.identity()].floatValue)")

        // 篩選出符合 mood condition 的 song
        if let moodPrediction = ClassifyMood(tensorInput) {
            if moodPrediction.Identity[mood.identity()].floatValue > 0.8 {
                print("mood > \(song.title):\(moodPrediction.Identity[mood.identity()].floatValue)")

                // 找出最貼近 duration 的 song
                if let songDuration = Int(song.duration) {

                    // 目前 recommend song list 中的總長度
                    let curDuration = self.recommendSongs.map { Int($0.duration) }.reduce(0, +)

                    if abs(duration - (curDuration + songDuration)) <= abs(duration - curDuration) {
                        if let song_id = song.id {
                            self.recommendSongs.append(Song(id: song_id, title: song.title, duration: songDuration, url: song.url))
                            print("Current Duration: \(curDuration + songDuration)")
                        }
                    }
                }

            }
        }
    }
}
```

02 Methods and Steps

03

Demo

03

Demo


12:23
搜尋
場合
婚禮
聚會
咖啡廳
頒獎典禮
畢業典禮
講座
運動會
公司活動
情緒
活力
開心
平靜
悲傷
時長
0 h
1 h
2 h
0 m
1 m
2 m
0 s
1 s
2 s
重置
儲存

Thanks You for Listening