MUSIC EMOTION RECOGNITION

CSCI-SHU 360 MACHINE LEARNING FINAL REPORT

Gracie, Zhou
NYU Shanghai
yz6212@nyu.edu

Qianyu, Zhu
NYU Shanghai
qz1086@nyu.edu

Sihan,Liu
NYU Shanghai
sl6964@nyu.edu

May 15, 2021

ABSTRACT

With the aim of classifying different songs based on the emotions they can inspire, this project trains different machine learning models with different sets of features. We stack traditional models including SVM and random forests, train it with the shallow features provided by Spotifys API, yielding an average precision of 0.76. recall 0.77, f1-score 0.76 for three-class classification task. This comparatively low accuracy can be attributed to the limited size of the dataset and the inherent errors in the labels. In comparison, the more complicated CLDNN model (CNN + LSTM + DNN), achieves an accuracy of 90.56% in binary classification.

Keywords CNN · MER

1 Introduction

Nothing evokes emotions in human beings quite the way music does. As a group of music lovers, we experience first-hand how our moods can fluctuate with the particular piece of music we are listening to. It would be very handy if an efficient music emotion identifier is incorporated in a music recommendation system, satisfying our emotional cravings. This is the inspiration for this project.

Music Emotion Recognition is usually regarded as a multiclass classification problem. The labels of this project include sad, calm, happy, and energetic. However, due to pronounced data imbalance, the team decides to treat it as a binary classification task, in which all songs are divided into negative and positive classes. Traditionally, models like Naive Bayes, SVM, and random forests are applied for this machine learning task. In recent years, emerging deep neural networks such as CNN have gained popularity due to their enhanced accuracy.(1)(2)

2 Dataset

2.1 Data Overview

The team created a dataset of about 7193 songs from Spotifys Web API. The process of selecting and retrieving data is illustrated in the figure 2.1.In order to suit the need for different machine learning models, two different datasets are subsequently created based on the original one. The first dataset uses Spotifys object metadata, and the second one consists of the audio features extracted from its 30-second song previews. Figure 2.1 also shows the distribution of the two datasets. The data is very imbalanced. This problem will be addressed in section 3.1.

2.1.1 Spotifys metadata

Spotifys AudioFeaturesObject includes 18 features, such as danceability and time signature. After discarding irrelevant columns like the URL of the song, we further include track_name and artist_name, which may increase model accuracy. The dataset that we get will be used for training traditional models such as SVM and random forests.

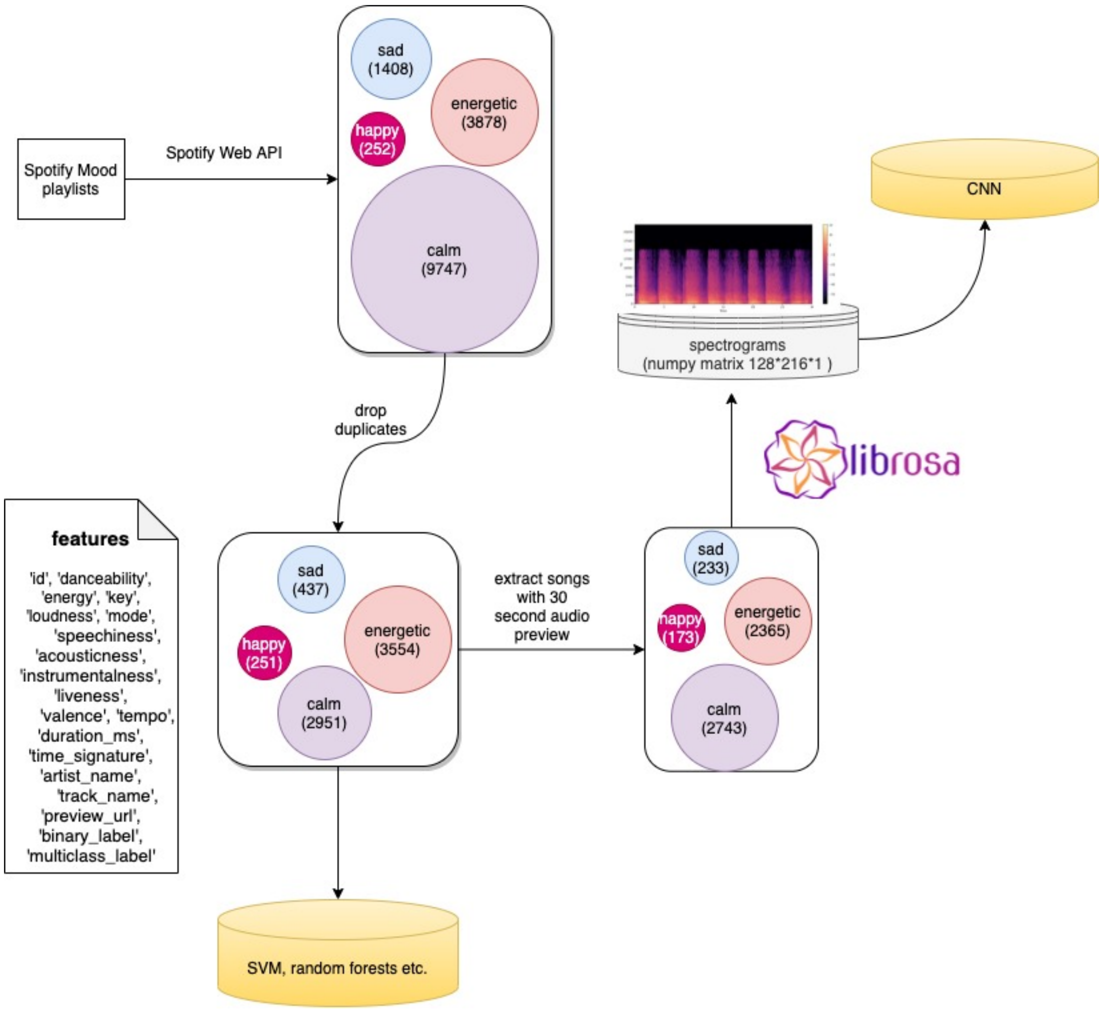

Spotify Mood playlists
Spotify Web API
sad (1408)
energetic (3878)
happy (252)
calm (9747)
drop duplicates
features
'id', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms', 'time_signature', 'artist_name', 'track_name', 'preview_url', 'binary_label', 'multiclass_label'
sad (437)
energetic (3554)
happy (251)
calm (2951)
extract songs with 30 second audio preview
sad (233)
energetic (2365)
happy (173)
calm (2743)
librosa
spectrograms (numpy matrix 128*216*1)
CNN
SVM, random forests etc.

Figure 1: Data Processing Pipeline

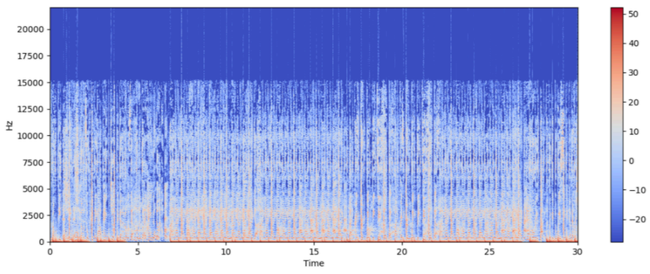


Figure 2: the spectrogram of a song labeled calm

2.1.2 Audio features

It is a common practice to convert audios into spectrograms, which can be fed to a Convolutional Neural Network (CNN). A spectrogram is the visual representation of the frequency spectrum of a signal, which encompasses all audio information. As demonstrated by figure 2.1.2 and figure 2.1.2, energetic songs and sad songs clearly differ in terms of their frequency spectrums. In other words, analyzing the spectrogram image is equivalent to analyzing the raw audio from which it is generated. Regarding the best model for processing images by researchers, CNN is a natural choice for this task.

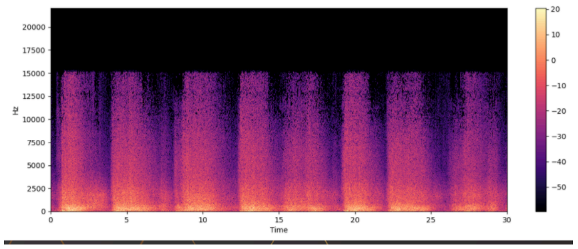


Figure 3: the spectrogram of a song labeled energetic

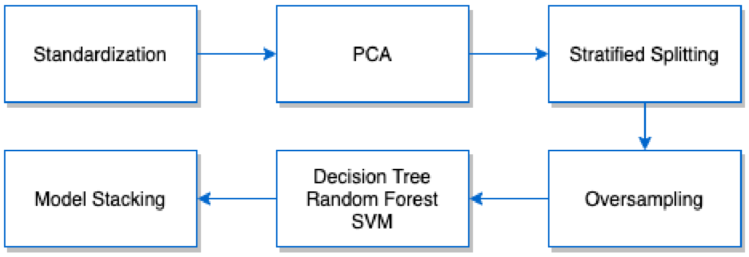


Figure 4: training pipeline

3 Traditional Machine Learning Models

3.1 Data Preprocessing

As mentioned in section 2.1, the data is very imbalanced, with many more energetic and calm songs than sad an happy ones. As a result, the first stacking model for the four-class classification task is very inaccurate. This is shown in the confusion matrix 3.1. It seems that the model can not tell happy songs from energetic ones. Therefore, the team decides to combine happy and energetic as one label, named joyful. In the preprocessing part, standardization, PCA and data resampling are applied. Among various standardizing techniques, we chose Z score standardization. Next, principal component analysis is used because we want the features to be uncorrelated.

3.2 Stratified Splitting

Using the train_test_split module from sklearn, we split the dataset into training set and testing set. The testing set is of one fifth of the whole dataset. We also specified the parameter 'stratify' to be Y, the labels of the dataset. This means that the class distribution is preserved in both training set and testing set.

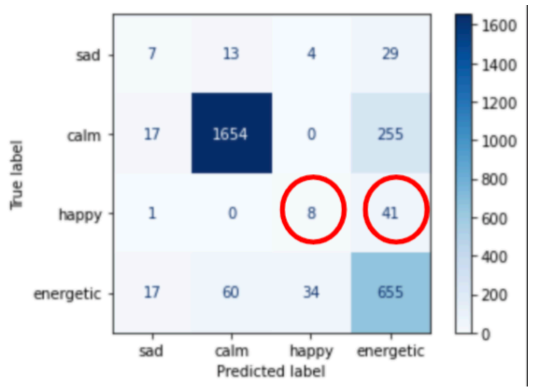


Figure 5: The confusion matrix for the first stacking model

Table 1: Classification Result of Default SVM

Classes	Precision	Recall	f1-score
Sad	0.75	0.06	0.11
Healing	0.93	1.00	0.96
Joy	0.95	0.84	0.90

Table 2: Classification Result of Decision Tree

Classes	Precision	Recall	f1-score
Sad	0.34	0.66	0.45
Healing	0.95	0.97	0.96
Joy	0.96	0.80	0.87

3.3 Oversampling

As mentioned previously, the data we have is severely imbalanced. To solve this problem, we oversampled the data of minority classes. The python library SMOTE is used. Each class is set to have equal weight in the training set while the testing set is left as it is. As a result , we get a training dataset with sad, happy and healing songs. 4752 songs included in each class.

3.4 Classifier

Since this is a classification task, several classical machine learning classifying models are tested, including SVM, Decision Tree and Random Forest. For each of the models, we used GridSearch for hyperparameter tuning. To do this, first we need a validation set. Therefore, the training data set which was oversampled was further split into two parts. This time, the validation set is again one fifth of the original training set. At the last step, we also tried to stack the models we have, including SMV, Decision Tree and Random Forest, each with the best hyperparameter. The parameter for stacking is not tuned.

3.5 Metrics

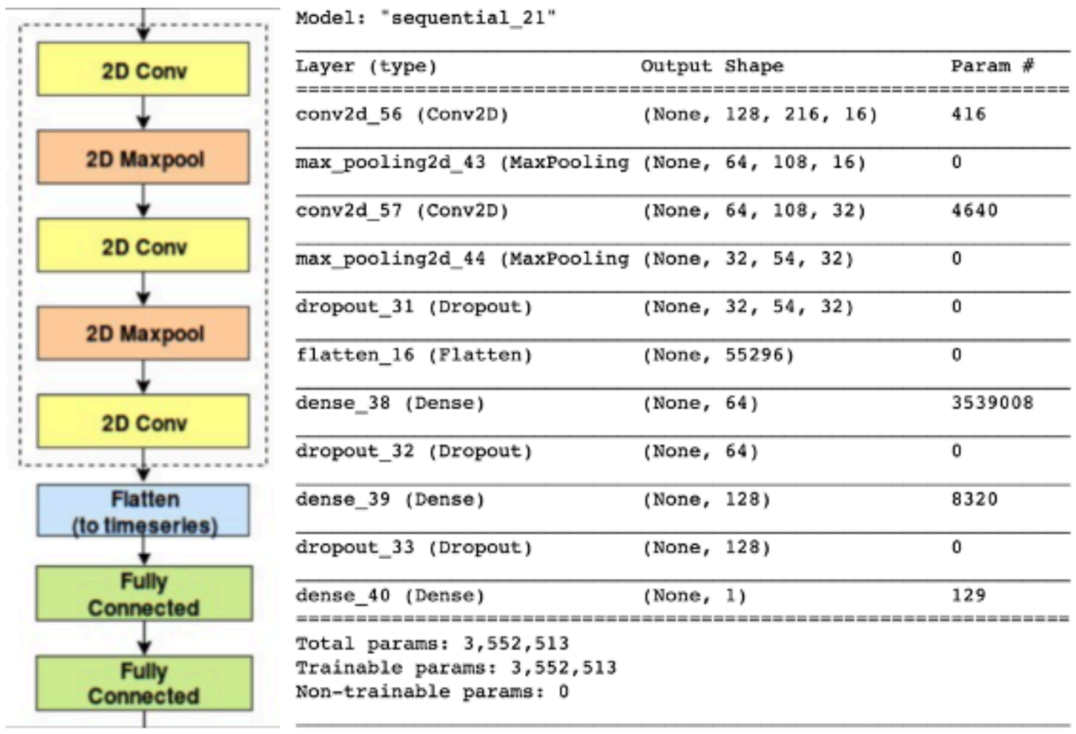
At first, we trained a SVM model with default parameters. The testing result is shown in Table 1. From this report, we can see that recall is extremely low for the sad music though its precision is relatively high. This makes us rethink whether it is a good choice to use precision as the only evaluation metric. AUROC and f1-score are taken into consideration. However, as this is a multi-class classification task, it is not easy to use AUROC as an evaluation metric. Thus, average f1 score is chosen for training and selecting the best model.

3.6 Result

Among all the models we tried, the Decision Tree with parameters "criterion: entropy, max_depth: 8, max_features: auto, n_estimators: 200 and the stacking model achieved the best performance. Model 1, Decision Tree on average achieved precision 0.75, recall 0.81, f1-score 0.75. Model 2, Stacking (SVM, Decision Tree and Random Forest) has achieved average precision 0.76. recall 0.77, f1-score 0.76.

Table 3: Classification Result of the stacking model

Classes	Precision	Recall	f1-score
Sad	0.43	0.48	0.45
Healing	0.95	0.97	0.96
Joy	0.90	0.86	0.88



```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_56 (Conv2D)           (None, 128, 216, 16)      416
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 64, 108, 16)       0
_________________________________________________________________
conv2d_57 (Conv2D)           (None, 64, 108, 32)       4640
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 32, 54, 32)        0
_________________________________________________________________
dropout_31 (Dropout)         (None, 32, 54, 32)        0
_________________________________________________________________
flatten_16 (Flatten)         (None, 55296)             0
_________________________________________________________________
dense_38 (Dense)             (None, 64)                3539008
_________________________________________________________________
dropout_32 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_39 (Dense)             (None, 128)               8320
_________________________________________________________________
dropout_33 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_40 (Dense)             (None, 1)                 129
=================================================================
Total params: 3,552,513
Trainable params: 3,552,513
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: CNN model pipeline

4 Neural Network with Spectrogram Input

4.1 Feature extraction

The input of our model is a matrix with size $128 \times 216 \times 1$ generated from a five-second audio segment in the middle part of a given song. We transform .wav audio files into numpy matrices which are computationally faster by librosa. Then the team proceeded with a train-test split of the dataset (80% training set, 10% validation set, 10% testing set).

4.2 Model selection & Optimization

4.2.1 baseline

To build a baseline model, we use artificial neural networks with one input layer (128×216 nodes), two hidden layers (128 nodes) with ReLu, one output layer with sigmoid activation to classify negative and positive emotions. This model yields an accuracy of 53.9%, behaving only slightly better than random guess. Pure FC layers turn out to be the wrong choice for our emotion recognition task.

4.3 Convolutional Convolutional Network

The first architecture the team explores is an extension of the baseline model (see figure 4.3). By introducing three convolutional layers and two Maxpooling layers, we achieve a training accuracy of 95%. However, the test accuracy is only 81% after 16 epochs.
This model has several severe drawbacks: first, it stops learning new patterns from the second epoch, and the loss curve shows a noticeable tendency of overfitting; second, the number of parameters is 500 times more than the size of our dataset, which leads to excessive training time.

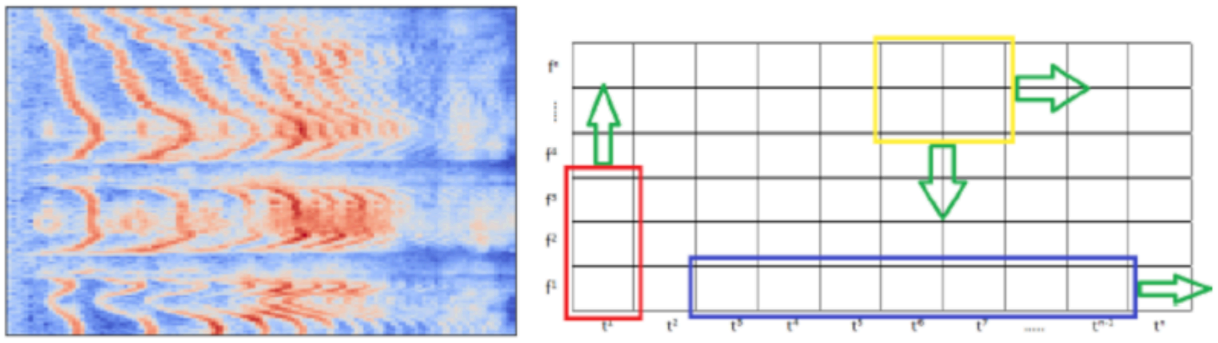


Figure 7: Spectrogram / Different filter sizes

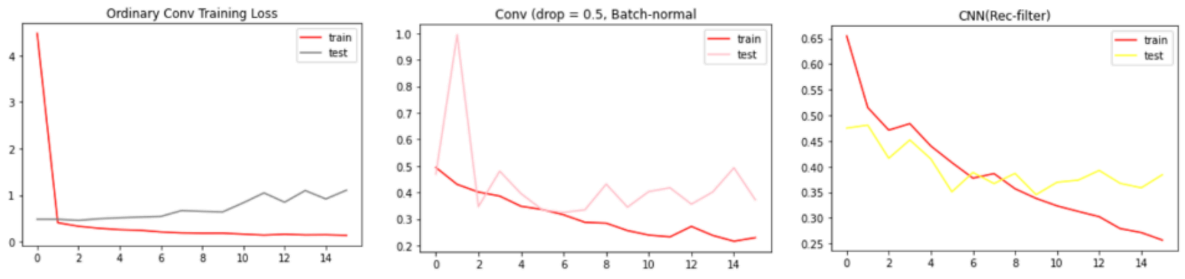


Figure 8: Loss Comparison (CNN, Batch-norm CNN, Rec-filter CNN)

4.4 CNN Optimization

4.4.1 filter size

Look back to our data, the x-axis denotes the time axis, and the y-axis denotes the frequency axis. They include different features. The updated convolutional layers have rectangular filters of size [2,1] or [1,3] such that they convolve only in frequency or time (figure 4.4.1). Also, the stride of the max pool layer is only along with the same dimension corresponding to the last convolution layer.(3)

4.4.2 Batch-normalization and batch-size

Our CNN model gets overfitted quickly in the first trial, so we set batch size = 32 to introduce random noise. This also shortens the training time, but leads to extra fluctuation in the epoch-loss plot.

4.4.3 Early Stopping

Later, we introduced the early stopping strategy by adding callback functions to our model. we increased the patient parameter from 0 to 5 to compensate for potential fluctuation.

4.4.4 Dropping Rate

We set the dropping rate to 0.5 during testing, but the difference is not obvious.

All the optimization process above gives us an increase in accuracy of 89.47% (figure 4.4.4). We believe a fair proportion of loss is due to the inherent random noise of manual labeling.

4.4.5 LSTM

The previous models fail to fully utilize the time-series information. It is for this reason that the team subsequently chooses to use an LSTM model to tackle this problem. However, in actual training, pure LSTM followed by FC layers is slower and has a lower accuracy than CNN.

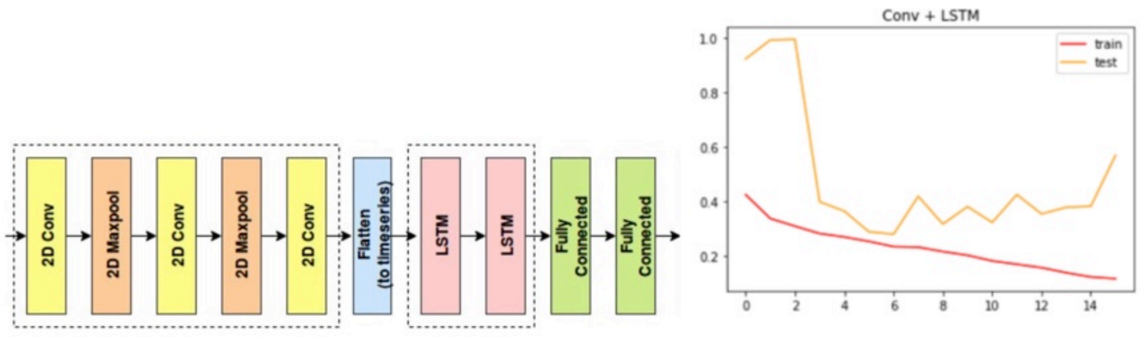


Figure 9: CNN + LSTM pipeline / Loss curves

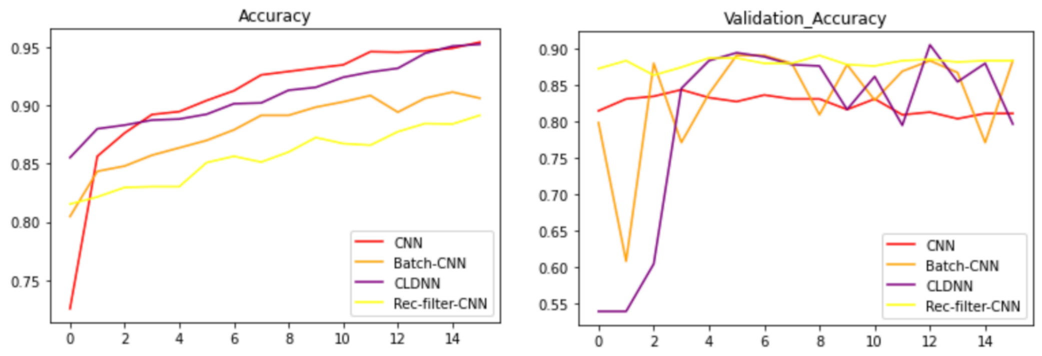


Figure 10: Training accuracy plot / Val-accuracy plot

The team finally applies the CLDNN model first raised by Rajat Hebbar in music embedding and genre classification(3). The model consists of one embedding layer, one two-layered, bidirectional LSTM layer, and one fully connected layer with dropout. A sigmoid function activates the final output, and we chose Adam to be the optimizer.
In the convolution layer, the filters are along the frequency axis, while in the LSTM layer, we only deal with the time axis. The result is a 90.56% testing accuracy.

4.5 Result

As shown in figure 4.5 and figure 4, the CLDNN model performs better than the others in testing and validation accuracy, with a training accuracy of 95.5% and a validation accuracy of 90.56%. After applying batch-normalization and rectangular filters, the optimized CNN shows a significant improvement in validation accuracy, with a historical maximum of approximately 90%. This is consistent with our assumption that separated processing along different axes of the spectrogram increases robustness of the model.

5 Conclusion

In conclusion, the team tries different methods to recognize the music emotions. Our project achieves high average precision score.

Table 4: Result Table

Accuracy/Model	CNN	Batch-CNN	Rec-Filter-CNN	CLDNN
Training Accuracy	95.48%	91.81%	88.79%	**95.50%**
Val-Accuracy	83.67%	89.11%	89.11%	**90.56%**

First, we create our own datasets containing both shallow features provided by Spotify's API and time-frequency matrices, which we extract from raw audios using librosa. Then, we stack SVM, Decision Tree, and Random Forest using the shallow features, which yields the following result: precision 0.76. recall 0.77, f1-score 0.76 for the three-class classification task. As for spectrogram analysis, CLDNN model (CNN + LSTM + DNN) outperforms other models with a validation accuracy 90.56%.

Both traditional models and NN models show huge potential in music emotion recognition. However, these two methods differ in input type and optimizing methodology, and fit into different business scenarios. For example, in cases of a well-known song, traditional machine models might be a sound choice, since input feature vectors are easy to obtain, and the training process requires less computation. But in situations where the user would like to analyze the raw audio from a new song, CNN/LSTM model is more preferable.

6 Limitations & Future Works

Taking one step forward, the models can be further improved by adding more data to each class, especially the minority classes. The team proposes to use audio data augmentation (4). We can double the dataset size by performing time stretching / pitch shifting / dynamic range compression on each music clip. Also, we will draw multiple clips from different positions in single audio, which will enrich our data without duplication.

Apart from this, another way to improve the accuracy of our model is to get data of higher quality. Since our dataset is obtained by crawling websites, the music labels are inclined to fall into the trap of subjectivity. We believe that the models can perform better if the songs are labeled by music professionals.

Lastly, the features used to train in traditional machine learning models are limited as there are only 16 features. More features might help in improving the model. One method worth trying is to write a program to extract as many features as we want directly from the audio itself.

References

[1] A. Khamparia, D. Gupta, N. G. Nguyen, A. Khanna, B. Pandey and P. Tiwari *Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network.*

[2] Liu, Xin and Chen, Qingcai and Wu, Xiangping and Liu, Yan and Liu, Yang. *CNN based music emotion classification.*

[3] Aytar, Yusuf and Vondrick, Carl and Torralba, Antonio *SoundNet: Learning Sound Representations from Unlabeled Video*

[4] Salamon, Justin and Bello, Juan Pablo *Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification*

[5] `https://github.com/Gracie-z/mer_ml_project`