# CUDA-Driven Support Vector Machines: Parallel Computing Approaches for Music Recognition

Wei Xi
1023040915
*school of Computer Science*
*NJUPT*
Nanjing, China

***Abstract*—This paper begins by exploring the audio characteristics of music, extracting multiple audio features from the musical content. Various features such as timbre, pitch, and loudness are extracted while retaining the temporal sequences of these features. To reduce computational complexity, dimensionality reduction techniques are applied to these features. The paper employs a support vector machine (SVM) with a radial basis function (RBF) kernel for music classification. Furthermore, CUDA is utilized to accelerate the training of the SVM, and corresponding experiments are conducted. The paper evaluates the accuracy achieved by different feature selection and processing methods in the context of SVM-based music classification.**

***Index Terms*—music, support vector machine, classification**

## I. Introduction

The concept of music genre is formed by unique elements such as beats, timbre, and melody in musical compositions, giving rise to various styles such as pop, classical, and jazz. Currently, one of the commonly used classification and retrieval criteria on most music websites is the music genre. Additionally, music genre serves as a crucial classification attribute in the management and storage of digital music databases. Therefore, automatic classification of music genres holds significant research significance and practical value.

The Support Vector Machine (SVM) is a powerful supervised learning algorithm primarily used for classification and regression tasks. Originally designed for linearly separable data, real-world datasets often exhibit non-linear separability. To address this challenge, researchers introduced the concept of kernel functions, enabling SVM to perform non-linear mapping in high-dimensional spaces. Despite the significant progress made by deep learning in various fields in recent years, SVM continues to hold a prominent position in specific problems and scenarios.SVM performs well in high-dimensional data and is a good choice for music classification. Music data typically has various features like timbre, pitch, loudness, etc. SVM excels in handling datasets with numerous features. By choosing a suitable kernel function, SVM can effectively deal with non-linear relationships, which is beneficial for the complex and diverse features of music.

For large-scale datasets or scenarios demanding higher computational efficiency, one can utilize optimized implementations of SVM or integrate additional methods such as deep learning to further enhance performance. With the evolution of GPUs, the emergence of CUDA provides another option for accelerating training methods.

## II. Related Works

### A. The Support Vector Machines

In the 1990s, Vapnik and his colleagues introduced the Support Vector Machine [1] (SVM) method to address the issue of small-sample classification. SVM is a learning method based on the principle of structural risk minimization, and its generalization ability surpasses several traditional learning methods. Since its inception, SVM has garnered significant attention and recognition from researchers, leading to a series of important research advancements. It has developed a comprehensive theoretical foundation and a fundamental framework, gradually becoming an effective tool, aided by optimization methods, to address various issues in data mining. SVM has proven to be capable of addressing traditional problems such as high dimensionality and overfitting to some extent.Presently, SVM finds widespread applications in various fields [2], including but not limited to speech recognition, remote sensing image analysis, fault detection and prediction, information security, and time series forecasting.

### B. CUDA

CUDA [3]–[5] is a universal parallel computing architecture introduced by NVIDIA. The CUDA architecture revolutionizes traditional GPU programming by eliminating the need to convert program tasks into GPU graphics processing tasks, thereby reducing development complexity. The main components of the CUDA architecture include computing cores and a memory hierarchy. Each GPU computing core is composed of multiple stream multiprocessors (SM), and each SM consists of multiple stream processors (SP). Simultaneously, the GPU provides a multi-level memory system accessible to threads. Each thread has private local memory, and each thread block shares a shared memory that facilitates data sharing among all threads within the block, with a lifespan identical to the block and high-speed read/write capabilities. Additionally, constant memory and texture memory serve as read-only storage optimized for different purposes, while global memory is used to receive data transmitted from the CPU.The structural diagram of CUDA is illustrated in Figure 1. In this diagram, a Thread represents a thread, with multiple threads forming a Block, and multiple Blocks constituting a Grid.

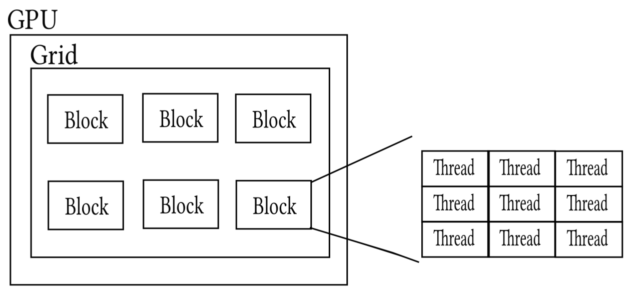


Fig. 1. CUDA architecture

## III. Music Classification

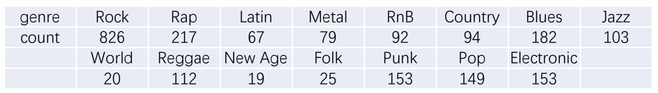
Music classification is fundamentally a pattern recognition process. The processing of music classification should adhere to the general processing flow of pattern recognition applications, thus allowing the design of the music classification technical process with the principles of pattern recognition. Data used for training and testing in music, first and foremost, must be collected. The selection of features and models is determined based on the characteristics of the collected data.

### A. Dataset

One of the main challenges in supervised learning is the computational complexity required for effective learning and the quantity of labeled data. The former has been addressed to some extent through advancements in hardware and the utilization of general-purpose GPU computing. The latter depends on the availability and size of labeled dataset.

In this paper, we opted for the Million Song Dataset [6] (MSD) as our chosen publicly available dataset. Despite the absence of audio files in this dataset, it encompasses numerous features that are deemed sufficient for the purposes of this paper. The Million Song Dataset includes dozens of features, including timbre and loudness, and its sheer size is impressive. Unfortunately, this dataset lacks annotated music genres. To address this issue, we concurrently selected another dataset: tagtraum [7]. This dataset serves as an enhancement to the MSD and merging it with the MSD provides a dataset that meets our criteria.

We selected a subset of the MSD, comprising 2250 music tracks. The distribution of various music genres within this subset is illustrated in Figure 2. It is evident that there are a few genres with a disproportionately small number of instances. To mitigate the impact of data skewness, we decided to exclude these genres from the training set.



| genre | Rock | Rap | Latin | Metal | RnB | Country | Blues | Jazz |
|---|---|---|---|---|---|---|---|---|
| count | 826 | 217 | 67 | 79 | 92 | 94 | 182 | 103 |
|  | World | Reggae | New Age | Folk | Punk | Pop | Electronic |  |
|  | 20 | 112 | 19 | 25 | 153 | 149 | 153 |  |

Fig. 2. music genre count

### B. Feature Selection

When performing feature selection, we focus on the audio features of the music itself and do not consider music metadata such as the year, artist, etc. The final selected features are as follows:

- Timbre: The timbre data in the Million Song Dataset (MSD) consists of twelve dimensions, each corresponding to different aspects of timbre. By combining these twelve dimensions, a reasonably representative feature can be obtained. We choose to extract the timbre feature using the following formula, where cx is the value of the x-th dimension, and dx is the weight for the x-th dimension:

$$timbre = c1 \cdot b1 + c2 \cdot b2 \ldots c12 \cdot b12 \qquad (1)$$

- Pitches: Similar to timbre, pitches also have twelve dimensions, each corresponding to different aspects of pitch. We extract the pitches feature using the same approach as timbre.
- In addition to these two crucial features, we also selected features such as loudness and maximum note duration. When extracting these features, we employed the method of taking the mean
var to process the data.

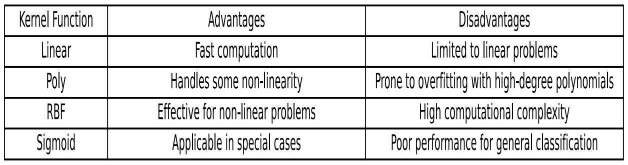
### C. Data preprocess

Data preprocessing is a crucial step in music classification tasks, ensuring the effectiveness and performance of model training. In this paper, we take some common methods for preprocessing.

- Data Cleaning:Data cleaning involves addressing missing values and handling outliers. As mentioned earlier, there is a significant disparity in the number of music genres, and to avoid data skewness, we chose to perform data cleaning by discarding music genres with a smaller number of instances.
- Normalization:Normalize the data to ensure all features have similar scales and prevent certain features from having a disproportionate impact on the model. As we extract features using variance, there is a significant difference between features in the data. To address this issue, normalization is necessary to improve accuracy and reduce computational complexity.
- Data splitting:80% of the data is used as the training set, while 20% is allocated to the testing set.

### D. Design of SVM

*1) Choice of kernel function:* The kernel functions for Support Vector Machines (SVM) can be broadly categorized into the following types: Linear, Poly, RBF (Radial Basis Function), and Sigmoid. The advantages and disadvantages of each function are illustrated in Figure 3. The Linear function is suitable for linearly separable cases, and when the number of features is comparable to the number of samples, using Linear can often achieve good results while significantly reducing the computational load. However, in this case, with a dataset size of 2250 and a feature dimension of approximately 200, we did not choose to use the Linear function. RBF performs well in classification tasks with high-dimensional data. Although it requires substantial computational resources, the dataset size is not large in this instance, and techniques such as CUDA can be employed for acceleration. Considering these factors, we choose to use the RBF as the kernel function for SVM.



| Kernel Function | Advantages | Disadvantages |
|---|---|---|
| Linear | Fast computation | Limited to linear problems |
| Poly | Handles some non-linearity | Prone to overfitting with high-degree polynomials |
| RBF | Effective for non-linear problems | High computational complexity |
| Sigmoid | Applicable in special cases | Poor performance for general classification |

Fig. 3. comparison for functions

*2) Gridsearch for best parameters:* In SVM training, numerous parameters can influence the training results. To achieve the best training performance, researchers often invest considerable time in finding optimal parameters. To mitigate the time spent on parameter tuning, automated parameter tuning has emerged. In this paper, we opt for the grid search method to perform automated parameter tuning for finding the optimal parameters. Grid search systematically explores all possible hyperparameter combinations, determining the combination that yields the best performance on a given dataset. It conducts a search over all possible hyperparameter combinations under cross-validation and outputs the combination that results in the best performance.Cross-validation (CV) is a statistical method used to validate the performance of a classifier. The basic idea is to partition the original data into groups, with one portion serving as the training set and another as the validation set. Initially, the classifier is trained using the training set, and subsequently, the model's performance is assessed using the validation set. This process is repeated, allowing for a comprehensive evaluation of the classifier's performance.

### *E. Cuda for SVM*

To meet the substantial computational demands of machine learning, particularly in the current scenario where processor speed improvements are slowing down, NVIDIA employs CUDA technology on GPUs. This enables machine learning to leverage the parallel processing capability of GPUs, thereby enhancing training speeds. Due to GPUs having a large number of processing units and high-speed VRAM, they are easily parallelizable.

CUDA is a universal parallel computing model that provides a framework for general-purpose computing on GPUs. Through CUDA, programmers can write code using familiar programming languages without delving into intricate parallel programming techniques [8]. This structure allows programmers to effortlessly harness the parallelism of GPUs and attain higher acceleration ratios without grappling with complex parallel programming issues. This proves beneficial for accelerating machine learning training processes and handling large-scale datasets.

*1) analysis:* In order to parallelize SVM, it is necessary to analyze the SVM algorithm [9]–[11], identify components that are amenable to parallelization, and parallelize them while minimizing the non-parallelizable aspects as much as possible.

We know the objective function of SVM is :

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to} \quad 0 \le \alpha_i \le C \tag{2}$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Among of (2), $\alpha$ represents the Lagrange multiplier vector, C is the regularization parameter, n is the number of samples, and K is the kernel function.

$$Q = \sum_{i,j} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

We define Q like (3). Obviously, Q occupies a significant amount of computational time. Moreover, matrix operations are highly amenable to parallelization.In conclusion, we have decided to parallelize the computation of the Q matrix.

## IV. Evaluation

We have chosen two aspects to evaluate our designed SVM: accuracy and training speed. The next three experiments will assess our designed SVM from these two perspectives and provide corresponding analyses.

The hardware, software, and operating environment for this experiment are as follows:

Processor: Intel Core i5 9300H
GPU : NVIDIA GeForce RTX 1660Ti
Memory ; 8GB*2
CUDA version : 11.6
OS : Windows 11
Python : 3.8

### *A. compare with kernel*

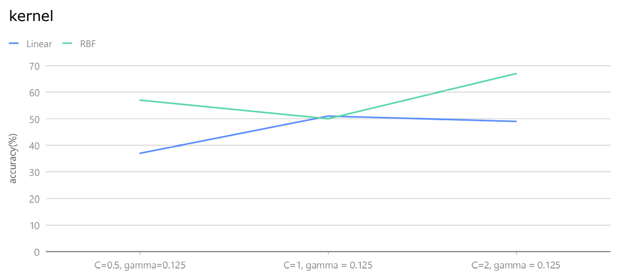


Fig. 4. compare with kernel

As shown in the figure 4, we compared the accuracy of various kernel functions after training with different C and gamma parameters.

It is evident that the RBF kernel outperforms the linear kernel in this music classification task. The linear kernel performs suboptimally across different parameters. Additionally, focusing on the RBF curve, the accuracy of SVM varies with different C and gamma parameters. With a higher C value, SVM fits the training set by penalizing terms, while the gamma parameter maps the SVM space to a higher dimension for better fitting of the training set. Therefore, even with parameter adjustments to enhance fitting, there is no guarantee of improved accuracy, and the specific values need to be analyzed case by case. Increasing the fitting may come at the cost of reducing the model's generalization ability.

### *B. normalization and standardization*

Normalization and standardization [12]–[14] play a crucial and indispensable role when dealing with skewed data during training. Skewed data can lead to inaccurate partitioning of the hyperplane during training, favoring data with larger weights and affecting the overall training effectiveness. In SVM, normalization has a significant impact. Therefore, in this experiment, we focused on exploring the performance differences of SVM before and after normalization.

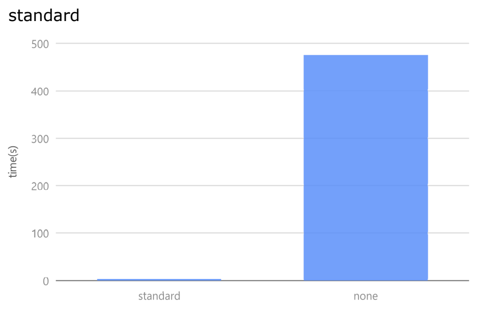


Fig. 5. training time

First, we tested the impact of normalization on training speed. As shown in the figure 5, before normalization, the training time was seconds. After normalization, the training time reduced to only about three seconds. In contrast, the training time without normalization was over four hundred seconds.

This clearly illustrates the importance of normalization for SVM. Beyond its impact on training time, normalization also significantly influences the accuracy of training. Skewed data, if not normalized, can greatly affect the parameters during training, leading to a decision function that deviates significantly from the expected one. Even though fitting to the training set may still be good in such cases, it is not conducive to data prediction. For the test set, accuracy would be very low, indicating poor generalization performance and practicality.

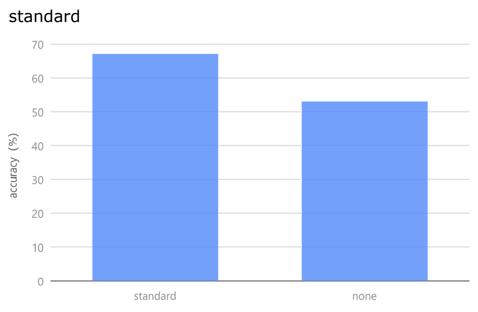


Fig. 6. accuracy

From Figure 6, it is evident that SVM without normalization has a 10% decrease in accuracy compared to SVM with normalization. Therefore, the impact of skewed data on the training effectiveness of SVM is evident.

In conclusion, normalization in the SVM training process is indispensable. It can simultaneously improve training speed and accuracy without incurring additional costs.

### *C. parallel*

Although parallel algorithms can effectively utilize multi-core processors or other parallel execution entities, the additional communication overhead and increased complexity of the algorithm make it not suitable for all tasks, especially small-scale tasks.

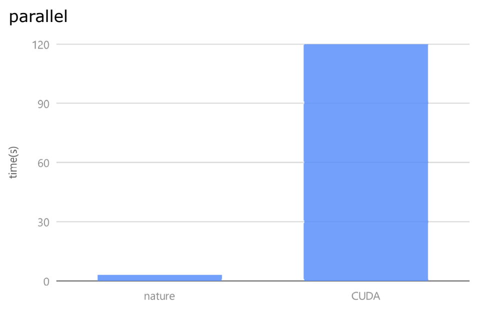


Fig. 7. parallel

The figure 7 compares the training speeds of single-threaded SVM and CUDA parallelized SVM. Unfortunately, due to the small size of the training set in this SVM experiment, using CUDA ended up increasing the time consumption rather than reducing it. After adopting CUDA for computation, due to the limitations of memory and GPU bandwidth, as well as communication overhead, transporting data to the GPU for computation turned out to be more time-consuming than performing the computation on the CPU. The primary time overhead of parallel algorithms is concentrated on data transportation rather than computation. This cost is unacceptable. Therefore, parallel computation is slower than single-threaded computation within the CPU. Although we believe that employing CUDA on a large dataset would significantly enhance training speed, due to time and hardware constraints, we ultimately did not conduct this experiment.

## V. Summary

In this project, we designed a classification algorithm based on SVM for music recognition tasks and attempted to accelerate it using CUDA. Although the attempt did not effectively accelerate the small dataset, we still explored it, gaining a deeper understanding of the SVM algorithm. Meanwhile, we conducted a series of experiments to assess the effectiveness and performance of the algorithm we designed.

## Acknowledgments

This semester's course covered knowledge related to big data analysis, and after a semester of learning, I have gained a lot. During the course, I acquired knowledge about Support Vector Machines (SVM). Therefore, for this experiment, I decided to use SVM for design and apply it to music recognition tasks. During the experiment, due to the significant time required for SVM parameter tuning, I attempted to parallelize the SVM algorithm, opting to use CUDA to accelerate the SVM algorithm. Unfortunately, this experiment did not delve into extensive research. Finally, I would like to express my gratitude to Professor Zou Zhiqiang, who taught the course this semester.

## References

[1] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[2] Q. Wu, L. Zhang, and W. Wang, "New family of piecewise smooth support vector machine," *Journal of Systems Engineering and Electronics*, vol. 26, no. 3, pp. 618–625, 2015.

[3] S. Cook, *CUDA programming: a developer's guide to parallel computing with GPUs*. Newnes, 2012.

[4] J. Cheng, M. Grossman, and T. McKercher, *Professional CUDA c programming*. John Wiley & Sons, 2014.

[5] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[6] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[7] H. Schreiber, "Improving genre annotations for the million song dataset." in *ISMIR*, 2015, pp. 241–247.

[8] L. Oden, "Lessons learned from comparing c-cuda and python-numba for gpu-computing," in *2020 28th Euromicro international conference on parallel, distributed and network-based processing (PDP)*. IEEE, 2020, pp. 216–223.

[9] K. Sopyła, P. Drozda, and P. Górecki, "Svm with cuda accelerated kernels for big sparse problems," in *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012, Zakopane, Poland, April 29-May 3, 2012, Proceedings, Part I 11*. Springer, 2012, pp. 439–447.

[10] A. Van Craen, M. Breyer, and D. Pflüger, "Plssvm—parallel least squares support vector machine," *Software Impacts*, vol. 14, p. 100343, 2022.

[11] M. E. Paoletti, J. M. Haut, X. Tao, J. P. Miguel, and A. Plaza, "A new gpu implementation of support vector machines for fast hyperspectral image classification," *Remote Sensing*, vol. 12, no. 8, p. 1257, 2020.

[12] K. Bach, "Standardization vs. conventionalization," *Linguistics and Philosophy*, pp. 677–686, 1995.

[13] M. S. Gal and D. L. Rubinfeld, "Data standardization," *NYUL Rev.*, vol. 94, p. 737, 2019.

[14] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.