Performance Evaluation of Machine Learning Algorithms using Multithreading

Sophia Kennedy
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
sophiak1@umbc.edu

Nitu Choudhary
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
nituc1@umbc.edu

Ankita Rathod
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
arathod1@umbc.edu

***Abstract*—Our idea is to evaluate the performance of machine learning algorithms such as Support Vector Machines(SVM) and Convolutional Neural Network(CNN) using both CPU and GPU. Implementation is done using multithreading and the number of threads are limited to analyze performance for different cases having different number of threads. Performance is evaluated by calculating the amount of time an algorithm takes to training its model in both CPU and GPU. Comparison of the results is shown by plotting graphs.**

***Index Terms*—SVM, CNN, multithreading, GPU**

I. Introduction

Machine learning algorithms are widely used in computer science such as Data security, Marketing Personalization, Fraud detection, online search, recommendations, NLP and smart cars. Machine learning engineers are in huge need by companies for implementing machine learning algorithms. The concern every ML engineer have is the performance some Algorithm takes just very seconds to complete their execution whereas certain algorithms may take a whole night or even a day or two to completely finish its execution. Our idea is to evaluate this performance of ML algorithms for better results using various test cases.

II. Machine Learning Algorithms

The Machine Learning algorithms we used for this project are as follows:

- A) Support Vector Machine
- B) Convolutional Neural Network

A. Support Vector Machine

Support Vector Machine is a Supervised Learning Algorithm that classifies data, it takes training data set as input and predicts the class of input data based on the output labels provided in the training data. In our project, we have utilized SVC library from scikit-learn with ensemble techniques for implementation of SVM. Ensemble methods use Support vector classifier as base classifier. These ensemble techniques considers the euclidean distance between the new data and train data in order to relieve from the conventional drawbacks.

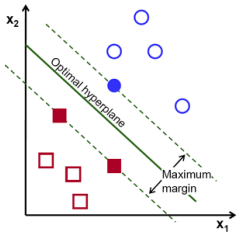

Fig. 1. Support Vector Machine(SVM)

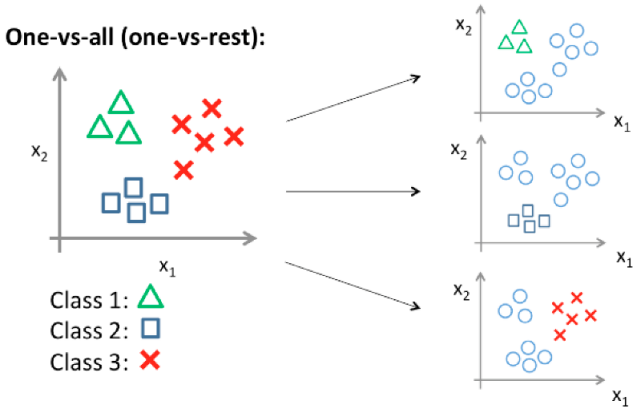

Fig. 2. One vs All Classifier

The Ensemble technique that we use is One-vs-All Classifier and Bagging Classifier.

1) One vs Rest Classifier: This is the first techinque used. The fitting is done as a single classifier per class. The fitting of class is against all other classes for the each case of classifier. We have used this to show the training of the model without any distribution of the training data in multiple threads unlike Bagging Classifier. The knowledge gained about the required class by inspecting it against its corresponding classifiers makes it more interpretable.

2) Bagging Classifier: This is the second technique used for improving the training time in SVM. This improves the training time 10 times leading to better performance of SVM. A Bagging classifier is an ensemble meta-estimator. Random subsets of the data are generated and then fitting is done and multiple models are generated. It fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions (either by voting or by averaging) to form a final prediction. The prediction accuracy is similar to the case when a single model is trained. The execution time is enhanced 10 times more when using Bagging classifier than training only one model using OneVsRest classification method. We have also implemented multithreading in this which improves the time in training for OneVsRest classifier but there is not much improvement in the training time.

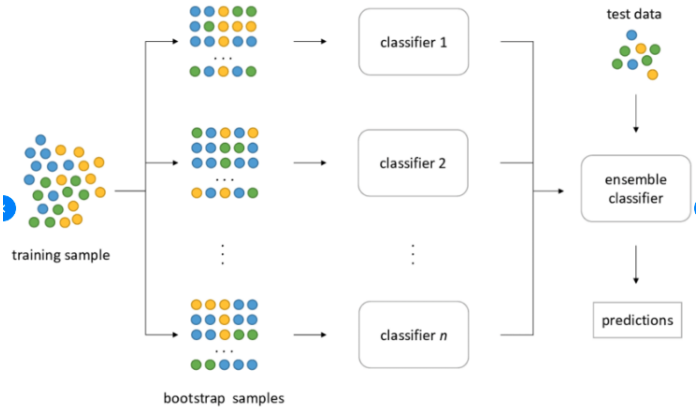

Fig. 3. Bagging Classifier

B. Convolutional Neural Network

Neural Networks learns the relationship from the data features and response observed. From the input, it is able to make predictions (e.g. the best time to go visit mall to avoid long queues in billing). In Convolutional Neural Network (CNN) every network layer acts as a detection filter for the presence of specific features or patterns present in the original data. The first layers in a CNN detect (large) features that can be recognized and interpreted relatively easy. Further, layers detect more abstract features. The features recovered in this layer is smaller in compared to the first layer (and are usually present in the larger features detected by earlier layers). The last layer of the CNN is able to make an ultra-specific classification by combining all the specific features detected by the previous layers in the input data.

III. Implementation

We implemented Machine learning algorithms in both CPU and GPU. The dataset we used for SVM is iris dataset and for Convolutional Neural Network is MNIST dataset. The machine learning algorithms were coded in Python. First, We implemented SVM in CPU by importing sci-kit learn library in python. As we mentioned earlier, we have used

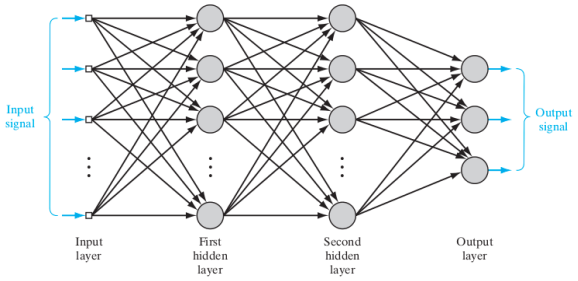

Fig. 4. Convolutional Neural Network (CNN)

OneVsRest classifier and Bagging classifier for SVM in order to improve its accuracy of prediction. We multithreaded the training part of the algorithm in order to know how it affects the performance of the Algorithm. Then, We implemented the same SVM code to train same dataset in GPU with and without multithreading. Secondly, we implemented the Convolutional Neural Network in CPU by importing tensorflow. The same CNN implementation was done in GPU. Then, we multithreaded the training part of the code in order to know the amount of time it takes for training with and without multithreading in CNN. Different number of threads were used for training to get the execution time for these case for better evaluation. Based on the time taken for training the model by given machine learning algorithms with and without multithreading in both CPU and GPU is noted for evaluation. We plotted graphs for that data to visualize the obtained results.

A. Hardware specifications for CPU and GPU

Since we used Google Colaboratory for running our code in CPU and GPU. The specifications of CPU and GPU are given below:

- **GPU:** 1xTesla K80 , having 2496 CUDA cores, compute 3.7, 12GB(11.439GB Usable) GDDR5 VRAM
- **CPU:** 1xsingle core hyper threaded i.e(1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost), 45MB Cache
- **RAM:** 12.6 GB Available
- **Disk:** 33 GB Available

For every 12hrs or so Disk, RAM, VRAM, CPU cache etc data that is on our allotted virtual machine will get erased.

IV. Datasets Used

We used two datasets IRIS and MNIST for training SVM and CNN respectively.

- MNIST (Modified National Institute of Standards and Technology database) is a database of handwritten digits which is commonly used for training various image processing systems and also for training and testing in the field of machine learning. There are 60,000 training images and 10,000 testing images in MNIST database.
- Iris flower data set measure features from the length and the width of the sepals and petals, in centimeters. Iris

setosa, Iris virginica, and Iris versicolor are the three species of Iris each having 50 samples.

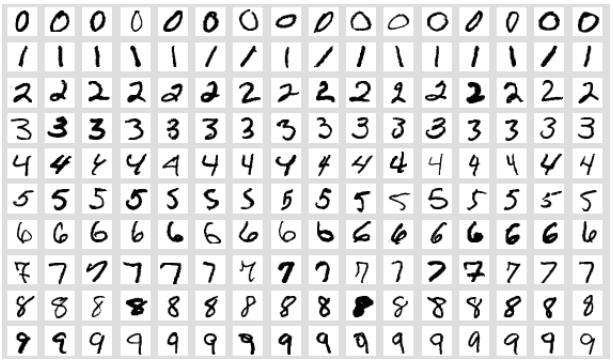

Fig. 5. Sample images from MNIST dataset

V. Results

The Results of our project are shown below for both SVM and CNN in CPU and GPU, with and without threads. Fig.6 and fig.7 shows the results for implementation of CNN in both CPU and GPU. Fig. 8 and Fig. 9 shows the results for implementation of SVM in both CPU and GPU. It is visible that performance is neither in the scale of complete increase nor in the scale of complete decrease. It depends on the size of the dataset used for training the model and also the number of threads used. Sometimes the overhead due to multithreading adds more to the execution time, affecting the performance of the algorithm. Depending on the size of the dataset and the number of threads used, the performance varies. There comes a point when the execution time increases on increasing the number of threads due to the overhead of context switching in threads.

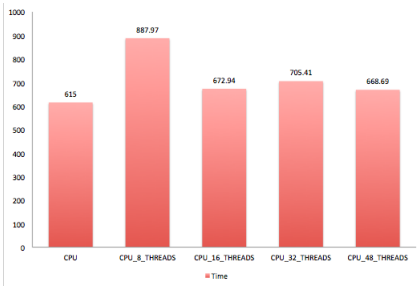

Fig. 6. Implementation of CNN in CPU

VI. Future Work

As of now, Dataset we have used are limited in size and variation. We can test these performance evaluation techniques for more diversified datasets and improvements can be done accordingly. This would give a much better perspective of the variation in performance when training models using machine learning algorithm. We have just considered SVM and CNN, this can be done using other algorithms too using the same techniques.

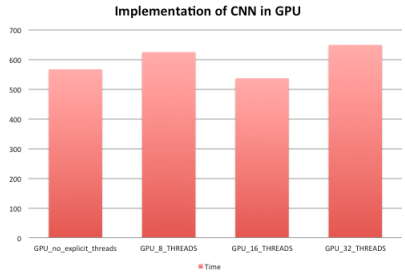

Fig. 7. Implementation of CNN in GPU

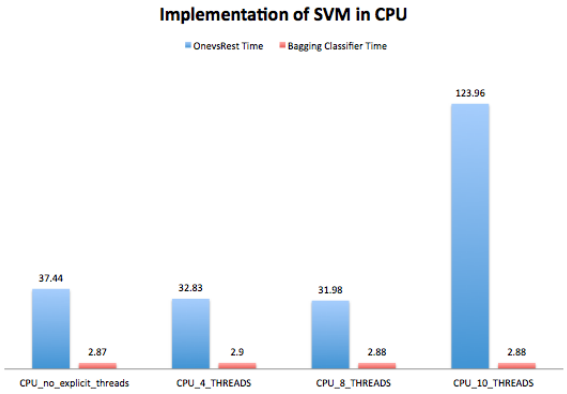

Fig. 8. Implementation of SVM in CPU

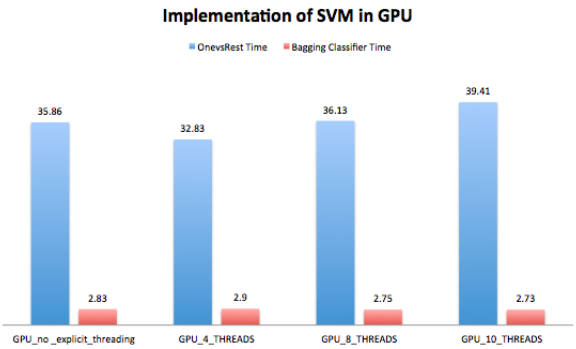

Fig. 9. Implementation of SVM in GPU

VII. Conclusion

We conclude that our project paper solemnly focuses on analyzing the variations that occur in the performance of two machine learning algorithms that are widely used in industries and research field. We evaluated training time in both CPU and GPU using multithreading. Therefore, these evaluations of performance using various cases help ML engineers in adapting the appropriate methodology in order to produce the effect in both prediction accuracy and performance.

Acknowledgment

We would like to express our gratitude towards Professor Dr. Ting Zhu and teaching assistant Yao Yao for their time,

guidance and constant supervision as well as for providing necessary information whenever needed.

REFERENCES

[1] Yuriy Kochura, Sergii Stirenko, AnisRojbi, Oleg Alienin, Michail Novotarskiy, Yuri Gordienko (2017), Comparative Analysis of Open Source Frameworks for Machine Learning with Use Case in Single-Threaded and Multi-Threaded Modes, IEEE XII International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT 2017), Lviv, Ukraine.

[2] Athanasopoulos, A. Dimou, V. Mezaris, I. Kompatsiaris, "GPU Acceleration for Support Vector Machines", Proc. 12th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2011), Delft, The Netherlands, April 2011.

[3] X. Zhang, Y. Zhang, C. Gu.,GPU Implementation of Parallel Support Vector Machine Algorithm with Applications to Intruder Detection, JOURNAL OF COMPUTERS, VOL. 9, NO. 5, MAY 2014.

[4] Q. Li, R. Salman, V. Kecman, An Intelligent System for Accelerating Parallel SVM Classification Problems on Large Datasets Using GPU.

[5] V. Fursov, S. Bibkov, P. Yakimov, Localization of objects contours with different scales in images using Hough transform [in Russian], Computer Optics. 37, 4 (2013) 502-508.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint, 1603.04467, 2016. arxiv.org/abs/1603.04467. Software available from tensorflow.org.

[7] Abdulrahman Alalshekmubarak and Leslie S Smith. 2013. A novel approach combining recurrent neural network and support vector machines for time series classification. In Innovations in Information Technology (IIT), 2013 9th International Conference on. IEEE, 4247.

[8] C. Cortes and V. Vapnik. 1995. Support-vector Networks. Machine Learning 20.3 (1995), 273297. https://doi.org/10.1007/BF00994018