Scalable Multimodal Machine Learning for Cervical Cancer Detection

T. Nhan[1], K. Poudel[1]

1. Middle Tennessee State University, Murfreesboro, Tennessee, USA

***Abstract*— Cervical cancer poses a significant global health concern, and machine learning techniques have shown great promise in early detection and diagnosis. However, the dominant focus on accuracy has overshadowed the crucial aspect of training time. This research paper investigates multimodal high-performance machine learning for cervical cancer detection, with a primary emphasis on scalability and training time optimization using Apache Spark. We conduct experiments on two distinct datasets – CSV and Images – analyzing the trade-offs between training time and accuracy. Moreover, we discuss the challenges encountered during the implementation of Spark. Our study aims to shed light on Spark's potential benefits and limitations in high-performance machine learning for cervical cancer detection. Our findings indicate that Spark ML showcases remarkable scalability potential compared to traditional platforms like SkLearn for text datasets. However, it faces challenges in tasks such as image classification, warranting further exploration and improvements in this domain.**

Keywords*— *Spark, Cervical Cancer detection, SparkML

I. Introduction

Cervical cancer is a type of cancer that predominantly affects the cervix in women. According to the World Health Organization (WHO) [1], it was the fourth most common cancer among women in 2020, with 90% of new cases and deaths occurring in low and middle-income countries. WHO recommends four different tests for detection, including HPV DNA testing, visual inspection of acetic acid, biopsy procedures, and liquid-based cytology [2]. However, these tests are expensive, time consuming, and unsuitable for large-scale testing. Consequently, there is a pressing need for a better and more efficient cervical cancer detection tool.

Thanks to advances in data processing, machine learning has emerged as a powerful tool in various industries. In the field of healthcare, machine learning has gained relevance and can be used to create advanced detection tools [3]. One of the key advantages of machine learning is its ability to rapidly deploy applications with a low barrier of entry for users, including healthcare professionals. Furthermore, machine learning can greatly enhance accuracy by analyzing vast amounts of data from multiple resources, allowing the identification of patterns and trends that may not be discernible to the human eye alone.

Using the potential of machine learning, a new generation of cervical cancer detection tools can be developed. These tools promise to provide a more efficient and accurate solution, significantly improving patient lives, reducing costs, decreasing mortality rates, and increasing the effectiveness of prevention programs. With seamless integration of machine learning, the limitations of current detection methods can be addressed, marking a significant step forward in the fight against cervical cancer.

Lots of works has been done on machine learning applications in the health industries. [4] tries to combine the usage of IoT devices and different machine learning models to reach results as high as 0.985 on average. However, most paper pay more attention to accuracy, which can be an oversight. In the era of Big Data, machine learning models must not only be accurate but also be able to handle large volumes of data in an efficient manner. This is a reasonable expectation as IoT devices are being used to improve accuracy and improve the life of users [4]. Since IoT devices can generate a large amount of data, other factors such as training time and scalability should also be considered.

In order to effectively handle data-rich data sets, parallelization is required. However, implementing parallelization can be tricky and system dependent. Additionally, code portability can also be a concern for parallelized implementation. Spark can be a solution to this problem. Spark is a big data platform that has a massive performance advantage compared to other big data platforms such as Hadoop MapReduce [5]. SparkML, the Spark's machine learning component, provides a scalable and parallelized machine learning library. In this project, we explore the usage of SparkML for Cervical Cancer prognosis and evaluate its accuracy and performance compared to nonparallelized platforms such as SkLearn. We also take a look at how portable the code is by taking a look at implementation differences between local and Google Cloud platform.

II. Litearture Review

Most of the research on cervical cancer prediction has focused predominantly on the Cervical Cancer Risk Classification dataset [6], a CSV text file. However, limited exploration has been made in harnessing images of cervical cells for prediction. Although some research groups have developed proprietary image datasets, the only publicly available labeled dataset is SipakMed [7], introduced in 2018. Remarkably, this particular dataset remains largely underexplored in the research domain.

Numerous papers have explored the application of machine learning in cervical cancer prediction, with a primary focus on the Cervical Cancer Risk Classification Dataset [6]. For example, in [8], the authors evaluated four machine learning models: Nasive Bayes, Decision Tree, Logistic Regression and Random Forest using the aforementioned dataset. Among them, the Decision Tree model exhibited an impressive accuracy of 96.8%.

Expanding the scope, another study, [9], delved into various machine learning models, including Logistic Regression, Bagging with Decision Tree as the base, Random Forest, and XG-Boost (Extreme Gradient Boost). Each proposed model demonstrated accuracies greater than 95%, with XG-Boost attaining the highest accuracy of 97.08%.

Furthermore, in [10], researchers explored different machine learning algorithms, culminating in an impressive accuracy of 99.71% using Random Forest. Beyond accuracy, this study also focused on the importance of the characteristics and the explainability of the model. Employing SHAP (SHapley Additive exPlanations), the authors identified key influential features affecting the model's performance, such as Schiller, Hinselmann, Age, and Hormonal Contraceptive.

Despite achieving remarkable accuracy levels, research related to improving the scalability of machine learning algorithms for the prediction of cervical cancer remains scarce. One notable exception is the work by [11], which used a multi-objective genetic algorithm to reduce the dimensions of the characteristics while maintaining a high precision of 0.965. In line with addressing this gap, our research aims to prioritize scalability as a critical performance metric in the context of cervical cancer prediction. By exploring both traditional CSV text files and cell images, we aim to advance the accuracy and applicability of cervical cancer prediction models.

III. Spark and SparkML

III-A. MapReduce

MapReduce is a programming framework originally introduced by Google to handle large datasets [12]. MapReduce allows data to be processed across cluster of commodity hardware. The MapReduce framework consists of 2 procedures, map and reduce. The map step is where the data is preprocessed. The reduce step is where the data are aggregated. In a typical MapReduce workflow, input is divided into smaller chunks. Then the small chunks of data are processed independently across different nodes in the cluster. The results from each node are then combined for the final output.

One of the most popular MapReduce implementations is Hadoop MapReduce. MapReduce is known for scalability and fault tolerance. MapReduce is used for batch processing and offline data processing jobs.

Features	Apache Spark	MapReduce
Data Processing Model	Batch processing, real time streaming, iterative task	Batch processing
In-Memory computing (RAM)	Yes	No
Processing Speed	Generally faster (In-Memory computing)	Slower (Disk based processing)
Ease of use	Higher level API and more expressive programming models	Lower-level API, more complex and verbose programming
Fault Tolerance	Yes	Yes
Scalability	Highly scalable	Highly scalable
Language Support	Java, Scala, Python, R	Java, support for other language is available but limited
Libraries	Multiple different built in libraries – (SparkMLLib, GraphX, ...)	Limited
Interactive Analysis of Jobs	Yes	No
Stream Processing	Yes	Yes, but with external tool (Apache Flink)

Figure 1. Spark vs MapReduce

III-B. Spark and SparkML

Apache Spark, introduced in 2014, was started at the University of California, Berkeley [13]. The goal of Apache Spark is to address the limitations of other Big Data processing frameworks such as MapReduces. Spark expands on the original MapReduce model by adding more feature sets and optimization. Figure 1 shows some of the differences between Spark and MapReduce. Sparks offers caching itermediated data in RAM, which significantly increases performance compared to disk-based processing in MapReduce. Spark also provides support and integration with other programming languages and platforms. Spark also provided higher-level API and more expressive programming models thus lower the barrier of entry compared to MapReduce.

SparkML is a Spark library that offers a scalable and distributed platform for machine learning. It supports multiple programming languages and streamlines the development of machine learning pipelines [14]. Through SparkML, users can easily deploy parallelized machine learning models on extensive datasets. By providing a high-level API for a wide range of machine learning tasks, SparkML significantly reduces the complexity of implementing distributed machine learning solutions. Furthermore, it incorporates implementations of several well-known machine learning algorithms, enhancing the efficiency of model deployment.

IV. Material and Methods

IV-A. Dataset

IV-A1. Cervical Cancer Risk Classification

This data set can be accessed on Kaggle[6]. The data set has 858 rows and 36 features. The features included are provided in figure 2 with "Biopsy" being the designated target class.

Age	STDs (number)	STDs: HPV
Number of sexual partners	STDs: condylomatosis	STDs: Number of diagnosis
First sexual intercourse	STDs:cervical condylomatosis	STDs: Time since first diagnosis
Num of pregnancies	STDs: vaginal condylomatosis	STDs: Time since last diagnosis
Smokes	STDs: vulvo-perineal condylomatosis	Dx:Cancer
Smokes (years)	STDs: syphilis	Dx:CIN
Smokes (packs/year)	STDs: pelvic inflammatory disease	Dx:HPV
Hormonal Contraceptives	STDs: genital herpes	Dx
Hormonal Contraceptives (years)	STDs: molluscum contagiosum	Hinselmann
IUD	STDs: AIDS	Schiller
IUD (years)	STDs: HIV	Citology
STDs	STDs: Hepatitis B	total_std
total_tests	Biopsy	

Figure 2. All features in Cervical Cancer Risk Classification dataset

IV-A2. Sipakmed

Sipakmed, introduced in 2018, is one of the first publicly available datasets for labeled Pap smear images. Its development aimed to standardize and increase the availability of Pap smear images. The data set encompasses five distinct classes, annotated based on the cytomorphological characteristics of the cells as follows:

- Normal Cells
  - Superficial-Intermediate Cells
  - Parabasal Cells
- Abnormal Cells
  - Koilocytotic Cells
  - Dyskeratotic Cells
- Begnign Cells

Category	Num of Images	Num of Cells
Superficial/Intermediate	126	813
Parabasal	108	787
Koilocytotic	238	825
Metaplastic	271	793
Dyskeratotic	223	813
Total	966	4049

Figure 3. Sipakmed's Distribution

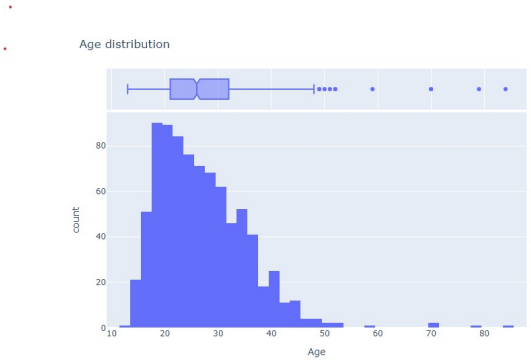

Figure 4. Age distribution

  - Metaplastic Cells

Sipakmed has 966 images, including 4049 cells in total. The specific distribution is shown in Figure 3. For the purpose of this project, only the photo of each individual cell is considered for training and testing purposes.

IV-B. PreProcessing

IV-B1. Cervical Cancer Risk Classfication

Before the training, several basic steps were taken to make sure the data were ready for machine learning models. First, all "?" values are replaced with the median of their respective columns. Additionally, all duplicated data was removed, resulting in a reduction in the number of rows from 858 to 835 rows.

The data set was heavily skewed. Specifically, most of the data set was within the age group of 21 to 32 (Figure 4). Other features are also skewed. However, no modification was done to ensure balance on the data set apart from using ADASYN (a super sampling technique) on the "Biopsy" target class. This was necessary due to the imbalanced ratio of 781 to 54 between the values of 0 and 1 for the class of 'biopsy', as demonstrated in Figure 5.

IV-B2. Sipakmed

For the Sipakmed dataset, the initial step involves compressing the images to alleviate the computational load. In this process, the images are resized to a squared dimension of 77x77 pixels. Although it is common to convert images to grayscale to further mitigate computational demands, it was found that grayscale conversion resulted in a 20% decrease in accuracy during testing.

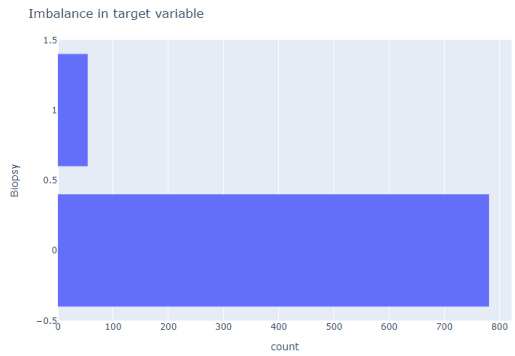

Figure 5. Imbalance of target variable

As a result, the decision was made to retain the images' original RGB values without converting to grayscale. Subsequently, the compressed images are transformed into a dataframe structure, where each feature corresponds to the color value of a specific pixel.

Given the considerable number of pixels present in the images, Principal Component Analysis (PCA) is conducted to project the dataset into a lower-dimensional space. SkLearn is employed for these pre-processing stages, a choice that will be rationalized in a subsequent explanation. Following PCA, the resulting dataset is saved as a CSV file and input into Apache Spark for subsequent training and testing phases.

IV-C. PCA

Principal Component Analysis (PCA) is a fundamental technique used to reduce the dimensionality of a dataset while retaining crucial information. In the context of this paper, PCA serves as a vital tool for dimensionality reduction in pap smear cell images. Given an image dataset X comprising n samples, where each sample x_i represents a vector in a high-dimensional space, PCA aims to identify a set of orthogonal vectors (principal components) that capture the maximum variance present in the data.

Mathematically, the PCA process involves calculating the covariance matrix $\Sigma = (1/n) * X^T * X$, followed by determining its eigenvectors and eigenvalues. Subsequently, the eigenvectors are ordered according to their associated eigenvalues, signifying the directions of maximum variance within the dataset. Through projection onto these principal components, an alternative, lower-dimensional representation of the original data is generated, which effectively retains the predominant variability encapsulated within the dataset. By integrating PCA into our methodology, we can effectively transform patterns inherent in pap smear cell images into a more manageable and informative feature space, thereby facilitating subsequent machine learning tasks.

PC	explained_ratio	explained_ratio_sum
0	5.594232e-01	0.559423
1	1.328089e-01	0.692232
2	4.171268e-02	0.733945
3	3.804913e-02	0.771994
4	3.212373e-02	0.804118
...	...	...
995	9.494518e-07	0.999483
996	9.395253e-07	0.999484
997	9.382255e-07	0.999485
998	9.353575e-07	0.999486
999	9.305075e-07	0.999487

Figure 6. PCA explained variance

In this case, the images dataframe after compression has dimensions of 4049 by 17787. This indicates that the dataframe represents 4049 photos, each with 17787 features. The result from PCA analysis demonstrates that 1000 features are sufficient to explain more than 99% of the original dataframe variance (see Figure 6). This substantial reduction in the number of features significantly alleviates the computational overhead required for the classification task.

IV-D. Model chosen

IV-D1. Logistic Regression

Logistic regression is a linear classification algorithm that models the relationship between input variables and a binary output using the logistic function. It is widely used due to its simplicity and interpretability. Logistic Regression performs well when the relationship between features and the target class is linear. It is computationally efficient and works well with large datasets. However, Logistic Regression may struggle with complex relationships and non-linear data. It assumes linearity and may not capture intricate interactions between variables.

IV-D2. Decision Tree

Decision Tree Classifier is a non-parametric algorithm that builds a tree-like model of decisions and their possible consequences. It is intuitive and easy to understand. Decision trees can handle both numerical and categorical data, require minimal data preprocessing, and can capture complex interactions between variables. However, decision trees tend to be prone to overfitting, especially with deep trees. They can be sensitive to

small changes in the data, and their interpretability decreases as the tree grows larger.

IV-D3. Random Forest

Random Forest Classifier is an ensemble method that combines multiple decision trees to reduce overfitting and improve generalization. It constructs an ensemble of decision trees, where each tree is trained on a random subset of the data and a random subset of the features. Random Forest can handle large datasets, high-dimensional data, and capture complex relationships between variables. However, Random Forest models can be challenging to interpret compared to single decision trees. Training time and memory requirements may increase with a large number of trees in the forest.

IV-D4. Gradient Boosted Classifier

Gradient Boosted Classifier is an ensemble method that sequentially builds models, each improving upon the mistakes of the previous ones. It combines weak learners, typically decision trees, in a stage-wise manner. Gradient Boosting can handle complex relationships, works well with heterogeneous data, and can capture feature interactions. It is especially effective in dealing with imbalanced datasets. However, Gradient Boosting can be computationally expensive and may require careful tuning of hyperparameters. It may also be prone to overfitting if the number of boosting iterations is too high.

IV-D5. Linear Support Vector Machine

Linear Support Vector Machine (SVM) is a popular classification algorithm that finds the hyperplane that maximally separates the classes in the feature space. SVMs are effective in high-dimensional spaces and can handle both linear and non-linear data by using different kernel functions. They provide robustness against outliers and perform well with small to medium-sized datasets. However, SVMs can be computationally expensive for large datasets and require proper scaling of features. The choice of the kernel function and regularization parameter can significantly impact the performance of SVMs.

IV-D6. Naives Bayes

Naive Bayes is a probabilistic classification algorithm that applies Bayes' theorem with the "naive" assumption of feature independence. It is simple, fast, and performs well with high-dimensional datasets. Naive Bayes can handle categorical features and works well with small training sets. It is robust to irrelevant features. However, Naive Bayes assumes feature independence, which may not hold true in real-world scenarios. It may struggle with capturing complex relationships and interactions between variables.

IV-D7. Factorization Machine Classifier

Factorization Machine Classifier is a versatile algorithm that models feature interactions in a flexible and scalable manner. It captures complex interactions between features and can handle high-dimensional and sparse data. Factorization Machines are robust to overfitting and require less memory compared to other models. They work well with large-scale datasets and have been successful in various applications, including recommendation systems, click-through rate prediction, and text classification.

IV-E. Methodology

The project has two specific goals:

- **Demonstrating Spark's Viability as an Alternative Platform**: One of the key aims is to establish Spark as a compelling alternative platform in comparison to traditional machine learning frameworks such as SKLearn or CNN.
- **Comparing Spark's Scalability**: Additionally, the project strives to conduct a comparison of Spark's scalability in contrast to other non-parallelized platforms, with a particular focus on SKLearn.

To address the first objective, the preprocessed data undergoes shuffling and is then divided into a 70-30 training-testing split. Subsequently, the training data is fed into each of the aforementioned machine learning algorithms. This process is repeated ten times, and the resulting outcomes are averaged over these iterations. The default implementation of each algorithm provided by Spark is employed. The methodology is also replicated for the Sipakmed dataset. To evaluate the model performance, a range of performance metrics including accuracy, precision, F1 score, and recall are computed. These results are subsequently compared with those from SkLearn (for CSV files) and CNN (for images) to provide an overview that supports the claim of Spark's viability as a platform.

In relation to the second objective, given the challenges encountered during image classification in Spark, the assessment of scalability focuses exclusively on the Cervical Cancer Risk Classification dataset. The initial step involves recording the training time for each machine learning algorithm. Subsequently, the algorithm with the longest training time is selected for further analysis. This chosen algorithm is implemented both in SkLearn and Spark. The chosen algorithm is then applied to a CSV file, which is subsequently duplicated in size with each iteration of the loop. Starting at an initial file size of 268KB, the file grows to a substantial 217MB following ten iterations. Within this context, the singular evaluation metric considered is training time, given that the duplicated data points introduce an element of artificiality.

Classifier Name	Accuracy Score	Precision Score	Recall Score	F1 Score
LogisticRegression	0.932432	0.932756	0.932432	0.932424
RandomForestClassifier	0.851351	0.855269	0.851351	0.850977
KNeighborsClassifier	0.932432	0.935275	0.932432	0.932311
SupportVectorClassifier	0.967181	0.969196	0.967181	0.967142

Figure 7. SkLearn's Accuracy on Cervical Cancer Risk Classification

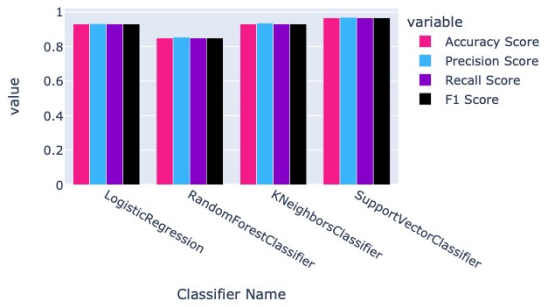

Figure 8. SkLearn's Accuracy on Cervical Cancer Risk Classification

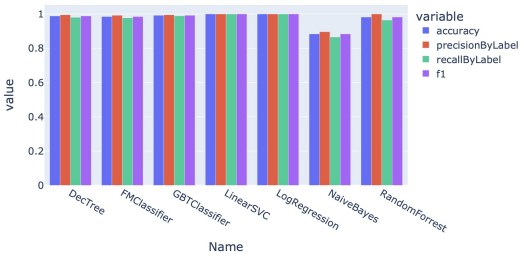

Figure 9. Spark's Accuracy on Cervical Cancer Risk Classification

Name	accuracy	precisionByLabel	recallByLabel	f1
DecTree	0.988297	0.995238	0.980288	0.988303
FMClassifier	0.984677	0.992093	0.977215	0.984675
GBTClassifier	0.992204	0.994841	0.988871	0.992212
LinearSVC	1.000000	1.000000	1.000000	1.000000
LogRegression	1.000000	1.000000	1.000000	1.000000
NaiveBayes	0.883407	0.896042	0.865482	0.883394
RandomForrest	0.981989	1.000000	0.964434	0.981988

Figure 10. Spark's Accuracy on Cervical Cancer Risk Classification

These methodological steps collectively contribute to an exhaustive assessment of Spark's potential as a scalable and viable alternative platform for the context of cervical cancer prediction.

V. Result and Discussion

V-A. Accuracy

V-A1. Cervical Cancer Risk Classification

Figure 9 and Figure 10 showcase the robust performance of all Spark-implemented algorithms. Notably, except for Naive Bayes, all algorithms exhibit excellent performance, maintaining accuracy rates consistently above 98% for all metrics. The Naive Bayes algorithm performs comparatively worse among the selected algorithms; however, it still achieves a respectable accuracy of 88%, along with consistent precision, recall, and F1 scores hovering around 88%.

Figure 7 and Figure 8 illustrate the accuracy results when implemented using SkLearn. While the accuracy rates are somewhat lower than those of Spark's implementation, SkLearn's performance remains commendable. The best result comes from the Support Vector Classifier, achieving accuracy scores exceeding 96% for all metrics. On the other hand, Random Forest performs comparatively less favorably, with accuracy scores hovering around 85% across all metrics.

The findings demonstrate that Spark's accuracy is consistent and comparable to that of SkLearn. However, the exceptional performance of the majority of Spark's algorithms raises valid concerns about potential overfitting. Rigorous exploration of this possibility is vital, necessitating an in-depth analysis of preprocessing choices and the impact of ADASYN for data balancing. A comprehensive investigation is essential to ascertain the reliability and generalizability of the developed models.

Assuming the absence of major implementation issues, the Decision Tree algorithm emerges as a promising solution for the given problem. As depicted in Figure 15, this algorithm boasts an average training time of 0.997 seconds, ranking as the second most efficient. Remarkably, despite its efficient training, the Decision Tree algorithm maintains consistently high performance levels (above 98%) across all other metrics, further affirming its suitability for the task at hand.

V-A2. Sipakmed Dataset

Figure 12, 13, and 14 present the results for the SipakMed dataset. Overall, the outcomes are suboptimal, with no model surpassing the 80% mark for any metrics. The most favorable performance stems from the Decision Tree classifier, achieving scores above 72% for all metrics. On the other end of the spectrum, Random Forest performs the least effectively, with a recall score of 57%, although other metrics achieve scores above 65%. While this performance surpasses random guessing, considering the classification task's complexity across five classes, the results remain far from desirable.

Figure 11 showcases the significant superiority of CNN performance compared to this paper's Spark implementation. Despite being designed with 32 epochs in

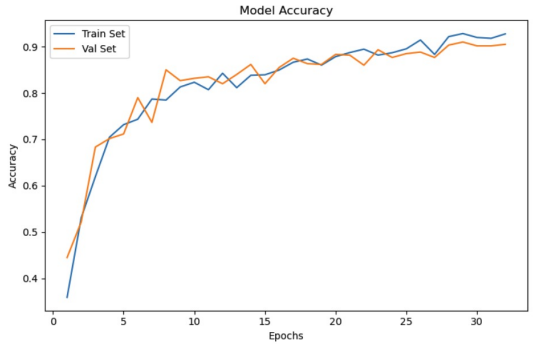

Figure 11. CNN's Accuracy (SipakMed)

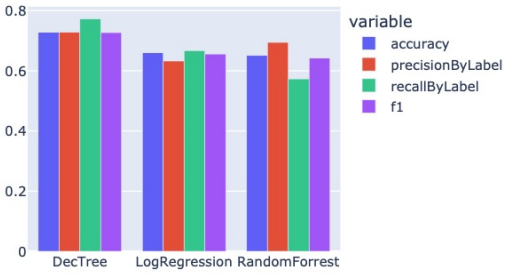

Figure 12. Spark's Accuracy (SipakMed)

mind, CNN outperforms all of Spark's implemented algorithms as early as epoch 10. By the 32nd epoch, CNN attains an accuracy of 92.75% on the same image dataset.

These results show that Spark or this paper's specific implementation of Spark may not be as suitable for image classification tasks. This observation aligns with the expected behavior, as CNN has demonstrated excellent performance in the realm of image classification tasks [15].

V-B. Scalability

Figure 15 illustrates the training times of various algorithms, guiding the selection of the Gradient Boosted Tree algorithm for scalability testing due to its longer training duration.

As shown in Figure 17, SkLearn's fitting time doubles with each iteration, reflecting its lack of parallelization. This behavior aligns with expectations, considering the proportional increase in training time with growing file size. In contrast, Spark exhibits a distinct pattern. While the initial iteration involves additional setup time for parallelization, subsequent iterations demonstrate a significant performance advantage over SkLearn. Notably, Spark maintains consistent and comparatively faster training times from the second to the fourth iteration, hovering around 1.6 seconds.

Figure 13. Spark's Training time (SipakMed)

Name	accuracy	precisionByLabel	recallByLabel	f1
DecTree	0.728089	0.727769	0.772028	0.726851
LogRegression	0.660797	0.632435	0.667035	0.655694
RandomForrest	0.650785	0.694339	0.572962	0.642461

Figure 14. Spark's Accuracy (SipakMed)

Spark's training time, after the initial setup in the first iteration, remains relatively steady across iterations 2, 3, and 4. Remarkably, starting from the seventh iteration, Spark exhibits a substantial performance improvement, achieving considerably faster training times.

While SkLearn showcases a relatively faster training time in percentage terms before the sixth iteration, the practical difference between instant completion and a 2-second interval remains minimal. However, as the iterations progress, Spark's lead becomes evident, resulting in a 16-second difference in training time by the tenth iteration. This emphasizes Spark's superiority in terms of scalability.

These findings show that Spark has significant advantage over SkLearn when managing larger datasets and achieving efficient parallel processing. Spark's capability to deliver consistent and fast performance, even as data size increases, solidifies its position as the preferred platform for scalable text based machine learning tasks.

VI. Spark's Implementation Issues

Despite Spark's demonstrated scalability with larger text data and satisfactory performance with smaller datasets, certain issues arise in its implementation:

- **Performance Discrepancies Between Platforms**: In the scope of this research, notable performance disparities were observed between Spark running on distinct hardware setups. For instance, Spark exhibited substantially superior execution speed on an M1 chip in comparison to a Windows machine equipped with an i7-12700 processor. While the smallest model's training on the Windows machine consumed a

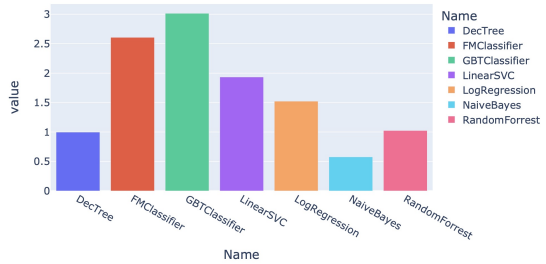

Figure 15. Spark's Training Time (Cerivical Cancer Classification)

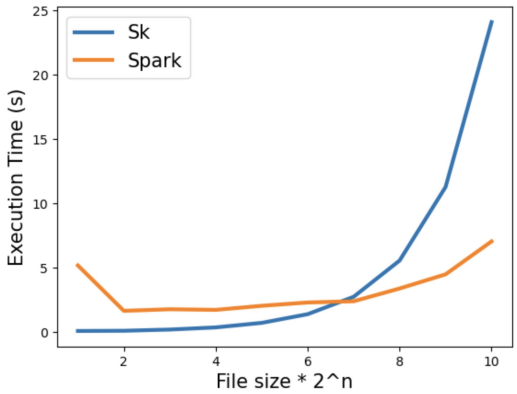

Figure 16. Scalability (Chart)

Size	SkLearn	Spark
1	0.110263	5.197456
2	0.126359	1.664117
3	0.217357	1.639954
4	0.378548	1.661266
5	0.700354	1.991977
6	1.388288	2.140520
7	2.694753	2.308474
8	5.468835	3.224648
9	11.219578	5.624002
10	23.706132	7.079738

Figure 17. Scalability (Table)

minimum of 50 seconds, the M1 chip accomplished the same task, inclusive of setup time, in just 5.3 seconds.

- **Popularity and Algorithm Availability**: Unlike SkLearn, Spark enjoys less widespread adoption. Although the SparkML library boasts a diverse array of algorithms, it might not encompass an equally extensive selection of pre-implemented algorithms as SkLearn. Users might discover a broader repertoire of readily accessible algorithms within SkLearn's ecosystem.

Moreover, while this might not pertain directly to Spark as a library, crafting custom algorithms within the Spark framework can be notably more intricate compared to SkLearn. To harness Spark's full potential, users must grasp supplementary concepts like MapReduce and RDDs (Resilient Distributed Datasets). This steeper learning curve might demand users to invest heightened time and effort for extracting the utmost benefits from Spark.

The challenges in utilizing Spark for image classification within traditional workflows became evident during my personal experience. Initially, as I attempted to follow the conventional procedure of image compression, grayscale transformation, and subsequent PCA, I encountered memory limitations. Despite my system's 32GB RAM, loading the images and applying PCA led to memory overflow. Upon upgrading to a system with 128GB RAM, the PCA process began. However, after approximately 15 minutes, Spark's PCA implementation prompted me to reduce the dataset's dimensionality—a counterintuitive directive given PCA's intended function. This dilemma could stem from both my code and potential limitations in Spark's implementation of PCA. While I managed to circumvent this issue by employing PCA from SkLearn, saving the outcome as a CSV, and then feeding it into Spark, the inability to perform all tasks solely within the Spark ecosystem remains a constraint.

Overall, Spark offers remarkable scalability benefits and an expansive machine learning library. Nevertheless, it is imperative to acknowledge potential performance discrepancies based on hardware configurations, algorithm availability with respect to SkLearn, and the additional expertise required to effectively wield Spark.

VII. Methodological Weakness

While the methodology used in this paper facilitates exploration of Spark's potential in multimodal big data processing in general, and cervical cancer prediction in particular, the proposed methodology harbors several significant weaknesses that we aim to address in future work.

To begin with, concerning the Cervical Risk Classification dataset, four features can be utilized as target variables: Hinselmann, Schiller, Citology, and Biopsy. These diagnostic tests are central to cervical cancer assessment, but two issues arise:

- The absence of a definitive target class that unequivocally indicates a patient's actual cervical cancer status, given that all these variables pertain to diagnostic tests.
- This paper solely employs Biopsy as the target variable, a choice influenced by its widespread use as the target variable in similar research using the same dataset. However, since the results of the other three diagnostic tests are included in the training features, there's potential for inflated accuracy figures, making the outcomes appear more impressive than they genuinely are.

To mitigate these concerns, it would be beneficial to either work with a dataset containing a designated column for true cervical cancer conditions or, if utilizing the current dataset, to exclude other diagnostic test results from the training process.

Turning to the Sipakmed dataset, our methodology encounters some challenges. Primarily, our attempt to compare traditional CNN accuracy with SparkML accuracy—without employing CNN—poses a methodological issue as it entails comparing accuracy between two distinct methodologies while striving to establish Spark as a viable alternative. To rectify this, future research should apply the same traditional machine learning algorithm to both Spark and Sklearn.

A further methodological limitation arises from the prominent role GPUs and deep learning play in image classification tasks. Unfortunately, we did not incorporate GPU acceleration due to time and knowledge constraints. Although Spark provides GPU support via Rapids, promising performance enhancements of over 4 times and up to 50% infrastructure cost savings [16], implementing this approach is complex due to its relatively recent introduction and limited adoption, resulting in scarce documentation. Future research will encompass investigating CNN with GPU acceleration to fully leverage its capabilities in the realm of image classification tasks.

VIII. Conclusion

Our research has revealed that Spark is a viable option for the medical sector, particularly in the area of cervical cancer prediction. All algorithms tested for CSV file showed excellent results, with most achieving an accuracy of 98% or higher. Furthermore, Spark was able to efficiently manage large datasets, which is essential given the growing availability of data from IoT devices that can be used to improve machine learning model performance. It is important to note that Spark is not a universal solution for all machine learning problems. There are many difficulties encounter when trying to make image recoginition works on Spark. But in the context of big data, the field of machine learning should be moving towards increased parallelization. Spark's capabilities are well-suited to this trend, making it a valuable tool for taking advantage of the potential of big data in medical applications and beyond.

REFERENCES

[1] W. H. Organization. (available at: *https://www.who.int/health-topics/cervical-cancer#tab=tab_1*).

[2] M. Jha, R. Gupta, and R. Saxena, "Cervical cancer risk prediction using xgboost classifier," *2021 7th International Conference on Signal Processing and Communication (ICSC)*, 2021, pp. 133–136.

[3] K. Allawadi, M. K. Singh, and C. Vij, "Using machine learning to improve healthcare: A disease prediction and management system," *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, 2023, pp. 281–285.

[4] S. J P, T. N. Deepak, and M. M, "Prediction of health problems and recommendation system using machine learning and iot." *2021 Innovations in Power and Advanced Computing Technologies (i-PACT), Innovations in Power and Advanced Computing Technologies (i-PACT), 2021*, pp. 1 – 8, 2021. (available at: *https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9696622&site=eds-live&scope=site*).

[5] S. Gopalani and R. Arora, "Comparing apache spark and map reduce with performance analysis using k-means," *International journal of computer applications*, vol. 113, no. 1, 2015.

[6] Gokagglers, "Cervical cancer risk classification," Aug 2017. (available at: *https://www.kaggle.com/datasets/loveall/cervical-cancer-risk-classification*).

[7] M. E. Plissiti, P. Dimitrakopoulos, G. Sfikas, C. Nikou, O. Krikoni, and A. Charchanti, "Sipakmed: A new dataset for feature and image based classification of normal and pathological cervical cells in pap smear images," *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3144–3148.

[8] S. Srivastav, K. Guleria, and S. Sharma, "Predictive machine learning approaches for cervical cancer detection: An analytical comparison," *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2023, pp. 951–956.

[9] P. Rawat, M. Bajaj, S. Mehta, V. Sharma, and S. Vats, "A study on cervical cancer prediction using various machine learning approaches," *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, 2023, pp. 1101–1107.

[10] M. Hasan, P. Roy, and A. M. Nitu, "Cervical cancer classification using machine learning with feature importance and model explainability," *2022 4th International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, 2022, pp. 1–4.

[11] M. Kuanr, P. Mohapatra, and J. Piri, "Health recommender system for cervical cancer prognosis in women." *2021 6th International Conference on Inventive Computation Technologies (ICICT), Inventive Computation Technologies (ICICT), 2021 6th International Conference on*, pp. 673 – 679, 2021. (available at: *https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9358540&site=eds-live&scope=site*).

[12] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters." *Communications of the ACM*, vol. 51, no. 1, pp. 107–113 – 113, 2008. (available at: *https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-37549003336&site=eds-live&scope=site*).

[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010, pp. 10–10.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark." 2015. (available at: *https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=edsarx.1505.06807&site=eds-live&scope=site*).

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks." 2013. (available at: *https://ezproxy.mtsu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=edsarx.1311.2901&site=eds-live&scope=site*).

[16] (available at: *https://nvidia.github.io/spark-rapids/*).