Image Identification using Natural Images Dataset

Srikrishna S
Masters, Data Science
Indiana University
Bloomington, Indiana
+1(812)955-8935
srsridh@iu.edu

ABSTRACT

One of the most pressing issues in the field of analytics is the increase in time involved during solving problems with huge amounts of data. Machine learning is the process of building models to solve classification/ Regression problems. Big- data systems are capable of handling huge amounts of data with ease. We can store, process and retrieve information very efficiently from big data systems. In this paper we aim to combine Machine learning with Big data systems and solve an Image classification problem. The dataset "Natural Images Dataset" [1] was obtained from a competition in Kaggle. Our aim is to implement the image classification in parallel, on SPARK and compare it's running time to the running time of the same machine learning model, implemented on a local system using Python.

Keywords

Analytics, Machine learning, Big data, Kaggle, parallel, Spark

1. INTRODUCTION

Image classification is the need of the hour. The most common obstacles faced by the Data scientists are the time involved in the process of image classification. The pixel matrix for each image is very huge, which in turn makes our data set extremely dense. In our paper, we have taken a data set having 6899 images belonging to the classes of Airplane, Car, Cat, Dog, Flower, Fruit, Motorbike, Person and built machine learning models to accurately classify the images belonging to different classes.

This dataset demanded extensive feature engineering. At first, we tried a naïve approach of using just the pixels as features, which as expected did not scale well. So, we had to extract and combine multiple features like HuMoments, Mahotas, Histogram and pixels.

Thus, this project not only involves machine learning model building, but also scaling it up in a parallel environment using SPARK, to compare running times.

1.1 Related Work

There is considerable amount of related work like building different classification machine learning algorithms to identify images. However, our paper stands apart from all the existing work in many ways. There is no known implementation of algorithms to solve this problem in SPARK. Also, no one has attempted to compare the running time in distributed environment like Spark vs the running time in the local system. We would like to acknowledge the code samples for the machine learning algorithms available in spark.aparche.org [2]

2. DATASET

2.1 Dataset Information

The dataset used for this project is very simple. It has a collection of 6899 color images belonging to the classes of Airplane, Car, Cat, Dog, Flower, Fruit, Motorbike, Person. Each image has a different pixel range and none of the images have uniform dimensions. Figure [1] explains the distribution of data across different classes.

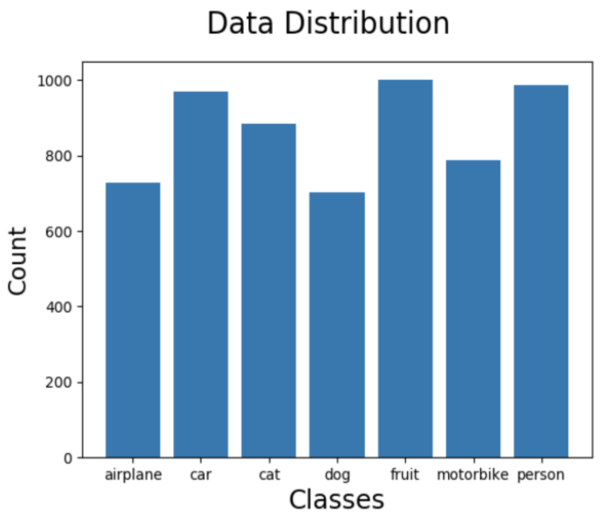

Figure 1. Data Distribution

2.2 Feature Engineering

Since our project is an image classification problem, as mentioned earlier the only source of data available is images. So, we are responsible for generating our own features. We first re-scaled our image into a uniform 100 * 100 pixels. This value of 100 * 100 was chosen carefully after lot of trial and error of pixels. Choosing the right dimension of images is highly challenging as different images have important features on various parts of the file. So, this combination will work fine. We created a numerical feature vector having all the below combination of features using 'OpenCV' library in Python.

2.2.1 Pixels as features

The color images are resized and converted to a Black and white image. These pixels are flattened and are converted to a list of features for each image

2.2.2 Features using Mahotas

Mahotas is a library available in python and coded using C++. It is based on NumPy array and its execution is very fast. It combines different elements of an image such as convex points, hit/miss, convolution, edge detection etc.

2.2.3 Features using Hu Moments

Hu Moments are used to describe the shape of the object. It is extracted from the outline of the object and it gives us the feature vector of numbers to describe the shape of the object. This feature is very useful as animals and objects tend to different shapes and outlines. This feature when used, will help to resolve serious misclassification errors like airplanes being predicted as dogs or cats and vice versa.

2.2.4 Histogram
It is a way of understanding the image. It gives us an idea about the contrast, brightness, intensity distribution of the images.

3. Architecture and Implementation
The usage of Machine learning algorithms for performing image classification are divided into two phases. We implemented algorithms in both Python and Spark environment. Though the basic working of the algorithms is the same, the implementation varies with the environment. For Python the dataset was converted into a Pandas data frame and an 80-20 train- test split was used uniformly across all the algorithms. For Spark, numerous combinations of train-test split were tried and 70-30 was chosen as an apt ratio. Also, instead off Pandas the features and the classes for training were converted as vectors and were given as input to our machine learning model.

3.1 Random Forest
Random Forest is a machine learning algorithm capable of performing both regression and classification tasks. We are using it for Classification. It also undertakes dimension reduction, treats missing values, outliers' values and other essential steps of data exploration. It is a type of ensemble learning, where a group of weak models (trees) combine to form a powerful model(forest). It uses bagging approach for solving the instability problem. The desired Regression Tree is estimated on many bootstrap samples and then the final prediction is made as the average of predictions by all the trees. Here, Random Forest adds a new source of instability to the individual trees and new variable having optimal observation is noted. This point is used to split the tree. In all this process, bagging gets benefits from instability.

3.2 Gradient Boosting
Gradient boosting models are usually preferred when an improvement in accuracy is needed. It usually operates on combining different decision trees together in order to achieve higher accuracy. Thus, Gradient boosting comes under the category of ensemble models. They combine multiple weak learners, typically decision trees in order to build a strong classifier. It can be classified into two types. Extreme gradient boosting and light gradient boosting.

3.3 Support Vector Machines
SVM'S are supervised learning algorithms that can be used for classification and regression. Originally, they were designed to solve binary classification problems. But, with the development of one vs the rest classification techniques, they can be used to solve multiple classification as well. The goal in SVM is to maximize the distance of the points from the hyper plane and also to maximize the margin of separation. Thus, it identifies the right hyper plane to separate the classes. These hyper planes in SVM's can also be built in 3-D instead of 2-D. It uses an inbuilt trick called kernel trick, to do better predictions

3.4 ADA Boost
ADA boost is the short form of adaptive boosting. It is a machine learning meta algorithm that combines the weak learners using a weighted sum. When one cycle of implementation of an algorithm completes, it pre-calculates and assigns weights to data points. So, for the next iteration, using these weights, we can assign more importance to certain data points.

3.5 Logistic Regression
Logistic regression is a probability based binary classification algorithm. It can only predict classes like yes/no, true/false, 0/1 etc. It converts the probability values into logit and predicts by performing the Maximum likelihood. Generally, the logistic regression uses sigmoid function for prediction. There are three types of Logistic regression- Binary, multinomial and Ordinal logistic regression. The probability values get classified into different classes based on the decision boundary values that are pre-set.

3.6 SPARK Architecture
Spark is an open source frame work like HADOOP. It is extremely fast, and it is capable of handling Petabytes of information at a time. It is well integrated with all popular Machine learning algorithms. It is based on "Cluster computing" which means it can combine the power of multiple clusters to do processing. It is extremely scalable, and it is easily interfaceable with major programming languages like python. be obtained by any reader. It distributes the data in parallel and runs algorithms, so the program gets executed in different clusters quickly. The various components of Spark are listed below [2].

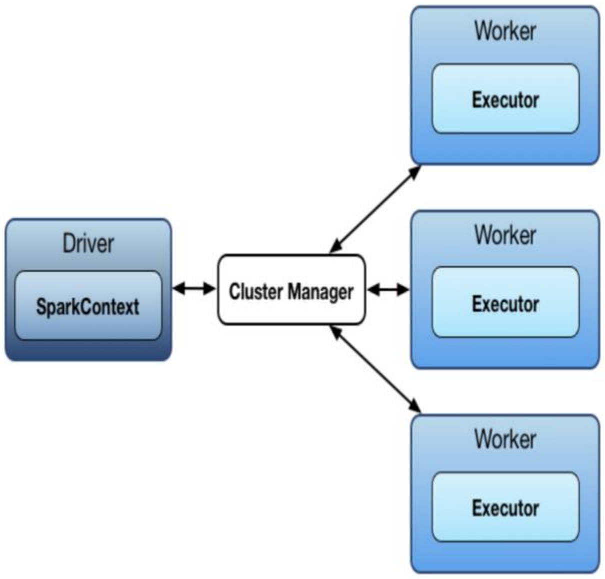

Figure 2. SPARK Layout

3.6.1 Spark Core
SPARK cores have built in memory and are capable of referencing an external dataset to solve problems. It operates on a general execution engine [3]

3.6.2 Spark SQL
It is responsible for introducing RDD, which supports both structured and semi-structured data

3.6.3 MLLIB
It is the machine learning library that is available in SPARK which helps in running machine learning algorithms.

3.6.4 SPARK Streaming
It has an API service that can be used to modify and manipulate data to match with the RDD API [3].

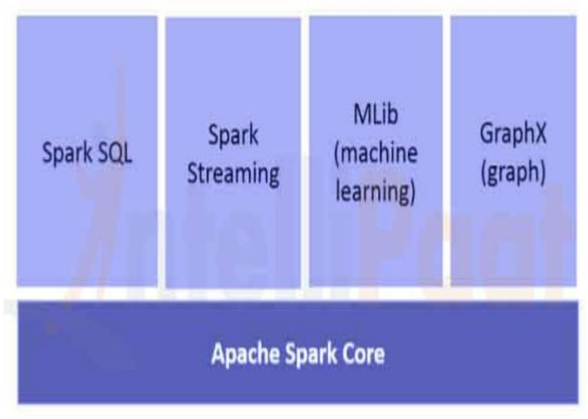

Figure 3. SPARK Components

3.7 HDFS

The dataset was first uploaded into HDFS (Hadoop distributed file system). HDFS [4] is a key component for parallel processing. It is closely related to MapReduce. HDFS is responsible for breaking data and distributing them across clusters for efficient parallel processing. Since it makes multiple copies of data and stores it in a server rack, crashes at nodes can be identified and the processing can still continue with the copy of data [4].

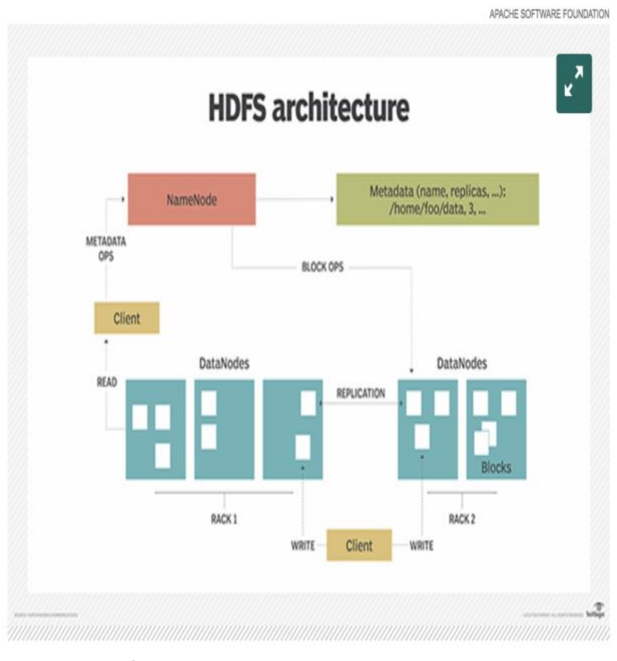

Figure 4. HDFS

HDFS is based on the Master/ Slave architecture. The Master device or process controls the slaves. The Master is responsible for boosting the I/O operation by dividing blocks of data over multiple disks. HDFS is very powerful and was first used by Yahoo. All big companies like Facebook, eBay, LinkedIn, Twitter, use HDFS. HDFS is also the core for data source alternatives and is called as Data lakes. HDFS is a very powerful architecture as it is capable of handling 1000's of petabytes of data across multiple nodes

4. EXPERIMENTS

The Image classification phase was carried out in two phases. First, the algorithms were implemented in our local system using Python with the help of PyCharm IDE. An entirely different code was written to scale the algorithm to work in a distributed environment (Spark). As expected, on the execution on Local system took a lot of time. In SPARK, the number of executors used were 2, and 4G memory was allocated for execution.

4.1 Random Forest

Random Forest [5] is a tree-based classifier which was used to classify the type of our images. The number of Trees were limited to 100 to avoid over-fitting, "GINI" was used to measure impurity, depth of the trees was limited to 12 and 40 bins were used. In Python, the execution time was 2260.85 seconds with 77% accuracy and in SPARK it was 599.74 seconds with 73% accuracy

4.2 SVM

SVM's [6] are binary classifiers and are usually used to solve problems that have binary class values. For instance, 0/1, Yes/No, True/False etc. In python, when the scikit learn [5] library is used, it automatically scales it to work for multiple class using the one vs the rest technology. An accuracy of 84.05% was achieved. However, in PySpark, SVM's can only be used to solve binary classification problem.

4.3 ADA Boost- Decision Trees

Boosting algorithm was used to use an ensemble machine learning approach to reduce Bias and improve the accuracy of the model. The ADA BOOST (Adaptive boosting) classifier coupled with decision trees performed the best in Python. The parameters used were max_depth = 2, n_estimators = 200 and learning rate = 0.8 In Pyspark, since a library to perform ADA Boost was not available, Gradient Boosting was performed. The execution took 1353.25 seconds to complete and the RMSE was 1.03. Though boosting is a efficient technique to improve accuracy, for our classification problem it did not perform that efficiently.

4.4 Gradient Boosting

Gradient Boosting technique was used to scale the execution of our machine learning classifier. In-order to achieve a better accuracy in SPARK than any one of the previous classifiers implemented, Gradient boosting regressor was implemented. The maximum Iteration of the booster was limited to 10 and RMSE was used as an evaluation metric. An RMSE value of 0.87 with a run time of 628.77 seconds was achieved.

4.5 Logistic Regression

Logistic regression is a probability-based classifier which are used to train a model for binary classification problem. For our Image classification goal, we implemented the one vs the rest Logistic regression algorithm. In this case, one classifier is trained for each class and it generates a numeric value. All the classes apart from the classes that are considered are ignored.

The run-time on the local was around 2574.33 seconds, RMSE 2.12.

The run-time in Spark was 420.12 seconds, RMSE 0.64

4.6 Confusion Matrix

The below image is the confusion matrix obtained for image classification using SVM. It depicts the number of images correctly classified according to the class of images in the test data have been correctly classified according to the class they belong to. Also, it can be observed which images are being misclassified and also to what classes they are being misclassified to [5].

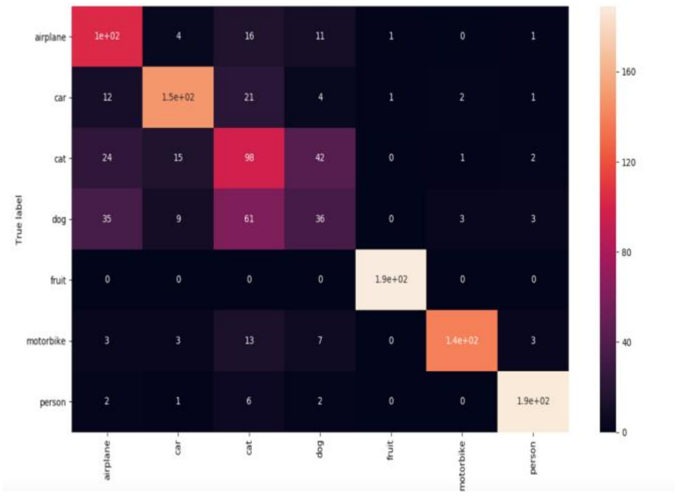

Figure 5. Confusion Matrix

From Figure [6] we can see that the Random Forest implementation was 1660 seconds faster, Logistic regression was 2100 seconds faster and Boosting was 750 seconds faster in SPARK as compared to Python. This is a great improvement in the running time of the algorithms on SPARK, when compared to the local system. Thus, the algorithms when implemented on a distributed network in parallel, scales the operation to a great extent and produces considerable savings in the running time.

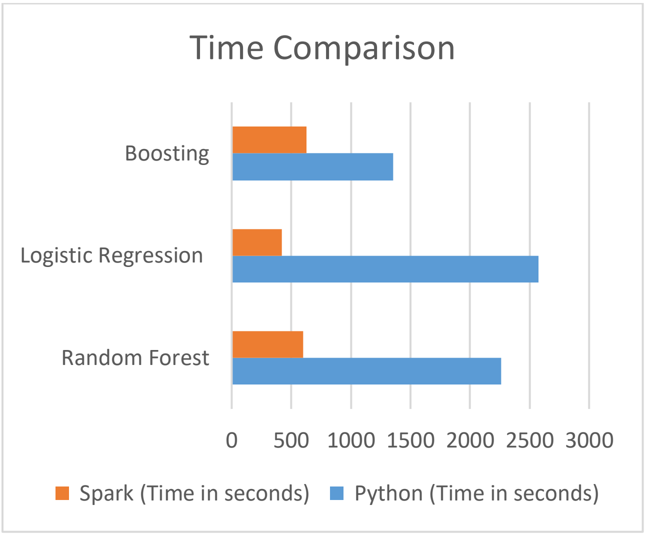

Figure 6. Time comparison

Table 1. Accuracy and Time consumption in Python

Algorithm	Python (Seconds)	Accuracy
Random Forest	2260.85	77.17%
Logistic Regression	2574.33	2.12(RMSE)
Boosting	1353.25	1.03(RMSE)
SVM	890.33	84.05%

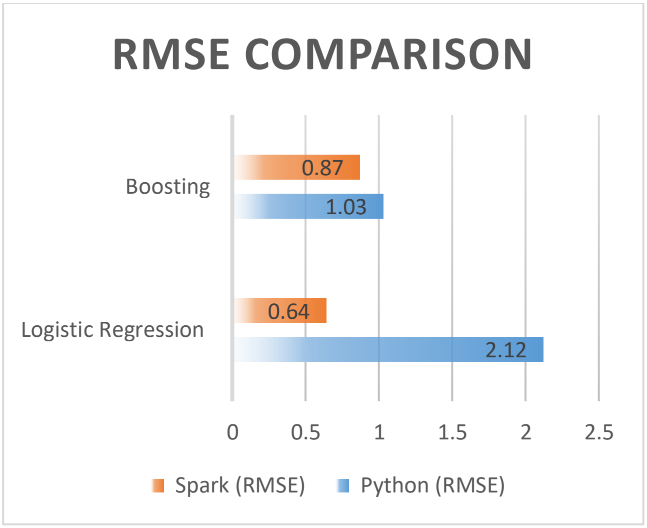

Figure 7. Time comparison

Table 2. Accuracy and Time consumption in SPARK

Algorithm	SPARK (Seconds)	Accuracy
Random Forest	599.74	73%
Logistic Regression	420.14	0.65(RMSE)
Boosting	628.77	0.8(RMSE)

From Figure [7] we can observe that the RMSE values for Boosting and regression are comparatively better when the algorithm is implemented in SPARK. We can observe that the improvement in accuracy is not that high as compared to the reduction in time for implementing the algorithm in SPARK. We aim to overcome this by exploring additional options (More of it in Future work). Also, the Table 1 gives us more details about the running time and accuracies of the various models that were built in Python. We can see that the accuracy of the SVM is the highest. This high accuracy was achieved by scaling the inputs that were given to the SVM model. Table 2 depicts the running time and accuracy of the algorithms in SPARK

5. CONCLUSION

SPARK surpasses python implementation in local system owing to parallel processing and implementation using a distributed frame work. Since the input data to the spark was given via Hadoop distributed File System (HDFS), it broke down the data and distributed it across the clusters for parallel processing. Since SPARK has a multi-stage in memory and a map-based computing engine, comparing Table 1 and Table 2, we can see that there is reduction in times of more than 30% was achieved as compared to Python implementation. Also, there was also great improvement of accuracy in Logistic Regression and Boosting. Surprisingly, Random forest accuracy dropped in SPARK, which may be attributed to the change in parameters and the number of trees constructed. The challenges faced were, initially it was very difficult to scale the program to work in SPARK. There were lot of issues with the yarn and cluster and it even ran out of memory after multiple trials. Synchronization issues happened due to low memory and at times there were no enough resources in Yarn. Thus, a distributed platform like SPARK which uses the RDD – Resilient Distributed Dataset surpasses implementation on a Non-distributed frame work. Hence, for computational expensive tasks like Image

Classification involving a dataset with numerous features, a distributed clustering frame work like SPARK is an ideal choice.

6. FUTURE WORK

In future, we aim to deploy the algorithms in SPARK to achieve more accuracy. Currently, our models are performing well but still we aim to explore more parameter tuning techniques for the machine learning algorithms in-order to improve the accuracy of the model. There are lots of variations in parameters which can be tried and tested. Also, we believe that the right feature selection is important to improve model's accuracy for image classification problems. Each image has its own unique features and it is of a different dimension. Currently we are exploring only specific features like pixels of the image, Hu Moments, Histogram and Mahotas, where all these features are extracted using a library called OpenCV in Python. Henceforth, we are planning to explore other feature learning libraries such as Over Feat [7] to extract more relevant and improved features. Over feat is a library which deploys Integrated recognition, Localization and Detection using Convoluted Neural Networks. It belongs to the class of techniques which enable us to automatically learn the right feature from the images for accurate classification.

7. ACKNOWLEDGMENTS

I would like to thank Professor Ms. Judy Qiu, Associate Professor of Intelligent Systems Engineering Professor Mr. Bo Peng for the constant support and encouragement. It would have been very challenging to work on this project without their guidance and support

I also would like to thank the Associate Instructors, Mr. Selahattin Akkas and Mr. Langshi Chen for guiding me in implementing the Image Classification problem in SPARK.

8. REFERENCES

[1] https://www.kaggle.com/prasunroy/natural-images

[2] https://spark.apache.org/

[3] https://intellipaat.com/tutorial/spark-tutorial/apache-spark-components/.

[4] https://searchdatamanagement.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS

[5] B Xu, Y Ye, L. Nie, "An improved random forest classifier for image classification[C]", International Conference on Information and Automation, pp. 795-800. DOI= https://ieeexplore.ieee.org/document/6246927/authors#authors

[6] B. Demir, S. Ertürk, "Hyperspectral image classification using relevance vector machines", *IEEE Geoscience* and Remote Sensing Letters, vol. 4, no. 4, pp. 586-590, Oct. 2007. DOI = https://ieeexplore.ieee.org/document/4317528/authors#authors

[7] https://arxiv.org/abs/1312.6229