Real-time Object Detection & Tracking and Scene Categorization on Traffic Data in Real-time video.

Pooja Shekhar-University of Missouri, Kansas City

Abstract— **Intelligent and automated security surveillance systems have become an active research area in recent time due to an increasing demand for such systems in public areas such as airports, underground stations and mass events. In this context, tracking of stationary foreground regions is one of the most critical requirements for surveillance systems based on the tracking of abandoned or stolen objects or parked vehicles. Shift to lambda architecture in this field of surveillance systems, is moving towards a real-time requirement for faster analysis of data. In this research, we used the application of a lambda architecture using Apache Spark, Apache Storm and Kafka with streaming traffic video data to achieve a real-time classification of objects in the video, which in our case primarily consists of three objects – home, man and car.**

Keywords—**SIFT,feature-extraction,lambda architecture, Storm, Kafka, Spark**

I. Introduction

Videos are actually sequences of images, each of which called a frame, displayed in fast enough frequency so that human eyes can percept the continuity of its content. It is obvious that all image processing techniques can be applied to individual frames. Besides, the contents of two consecutive frames are usually closely related. Object detection and tracking are important and challenging tasks in many computer vision applications such as surveillance, vehicle navigation, and autonomous robot navigation. Video surveillance in a dynamic environment, especially for humans and vehicles, is one of the current challenging research topics in computer vision.. The use of big data analytics has been significant in this area. We attempted to harness these new advances in the field to solve real time data processing problems. Surveillance networks are typically monitored by a few people, viewing several monitors displaying the camera feeds. It is very difficult for a human operator to effectively detect events as they happen. We have focused mainly on automating the surveillance systems. Recently computer vision research has to address ways to automatically some of this data, to assist human operators are readily becoming available. The framework is in place to implement the system, however, we have yet to show meaningful results in regards to classification. We suspect this could be achieved by refining the machine learning techniques used. Due to non-availability of ample hardware requirements to handle big data, our classification results are not very impressive as they have been done on low training data, though they are meaningful.

II. Related Works

A. *Object Detection & Tracking Using Real time Video Analysis & Scene Categorization*

Visual content can be modeled as a hierarchy of abstractions. At the first level are the raw pixels with color or brightness information. Further processing yields features such as edges, corners, lines, curves, and color regions. A higher abstraction layer may combine and interpret these features as objects and their attributes. At the highest level are the human level concepts involving one or more objects and relationships among them. To monitor an object′s spatial and temporal changes during a video sequence, including its presence, position, size, shape, etc. has already been done in the past. This is done by solving the temporal correspondence problem, the problem of matching the target region in successive frames of a sequence of images taken at closely-spaced time intervals.

Many object detection algorithms are based on the construction of a statistical model for the values taken by background pixels. Any pixels with values incompatible with the statistical model are considered to be in the foreground. The simple methods for modeling the values of background pixels can fail because different objects may project to the same pixel at different times (if the objects move) and the lighting can change. CENTRIST (CENsus TRansform hISTogram), is a visual descriptor for recognizing topological places or scene categories ,has been developed in the past. We show that place and scene recognition, especially for indoor environments, require its visual descriptor to possess properties that are different from other vision domains (e.g. object recognition).Experiments demonstrate that CENTRIST outperforms the current state-of-the-art in several place and scene recognition datasets, compared with other descriptors such as SIFT and Gist. Besides, it is easy to implement and evaluates extremely fast. However, none of the above discussed experiments work on big data frameworks, but our experiments mainly works on coming up with a online/offline streaming and classification of the video.

B. Lambda Architecture

With the MapReduce framework, the data is split up into partitions and undergoes two phases, Map and Reduce. These two phases are the building blocks to many tasks and help to achieve a fairly efficient and fast performance that is unparalleled near the time of its creation. But it has some serious pitfalls. The growing complexity of Big Data problems of today is too great for the MapReduce with severe overhead and CPU issues. This led to the development of frameworks to illustrate the scalability, adaptability and efficiency of the lambda architecture.

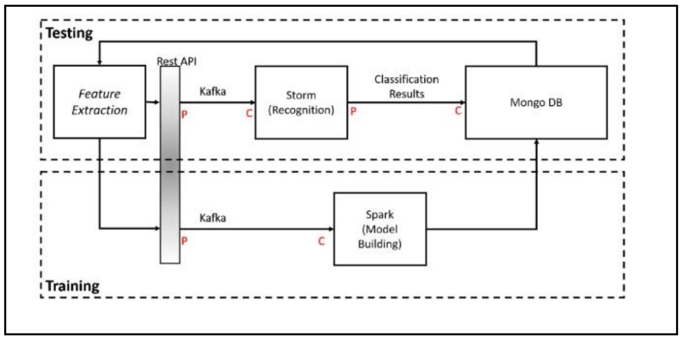
Apache Spark is a framework that is similar to MapReduce while still supporting a lambda architecture. By itself, it is a micro-batch processing framework that can outperform MapReduce’s efficiency. It has the Resilient Distributed Dataset (RDD), that is immutable and helps to allow fast access to data. It is also fault-tolerant and helps to reduce overhead while running MapReduce operations. It works well in a lambda architecture with Storm or Heron. Storm and Heron are both primarily stream processing engines that take data continuously (in the form of tuples) and distribute them to bolts to carry out tasks. Heron is a much more complex, but also more efficient and fault-tolerant than Storm. These streaming engines have the capability of incorporating their input and output with Spark to create simultaneously a stream and batch framework, not unlike the lambda architecture. It provides the capability to create a model that can both learn from stored datasets as well as new data coming in, especially with Kafka. Kafka is a publisher-subscriber system to which data can be streamed and from which data can be consumed. This allows for a user to publish data, and have their Storm-Heron-Spark lambda architecture consume that data in real-time. It can also be the broker between Spark and Storm, and serve as a reliable, scalable, efficient piece of the lambda architecture, without which the key elements of streaming and batch processing would not be able to efficiently combine to tackle the scale of Big Data.

While this model does not entirely implement a streaming engine of any sort, the fact that it can process both visual, audio, numerical, etc., data in large quantities shows that it has the adaptability and scalability of a lambda-like architecture, for these different data types cannot generate a similar model with a MapReduce-like framework. Indeed, the power of flexibility, scalability, and efficiency of these Big Data frameworks can only be achieved with a lambda architecture.

III. Implementation

Our lambda architecture consists of Spark, Storm, and Kafka [Fig.a] which receives data live from the client and uses a MongoDB database as an intermediary and final storage for model and results. Spark is used for our offline learning and model building, where we used Sparks MLLib to create a decision tree. Storm is used for online classification where different bolts provide true and false answers for classifying objects. Kafka and to an extent MongoDB acts as the intermediary communication channel between Spark and Storm and the client.

Fig.a. Lambda Architecture Design.

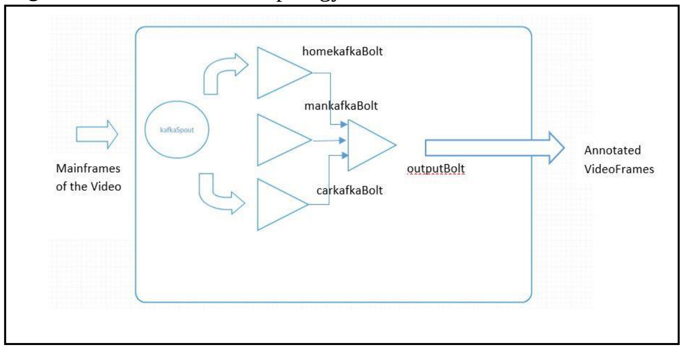


Architecture of our experiment.

Our real time classification workflow consists of the client extracting key frames and sift points from the traffic video and sending them to Kafka. Kafka queues this data and then acts as a spout for Storm to pull this data from and perform classification. The tuple of the form {frame metadata, SIFT point} are streamed into each bolt in our topology [Fig. 2] and are emit true or false based on the classification to the MongoDB with the frame metadata which allows for validation.

Our model was trained in Spark, where we extracted SIFT points from 254 mainframes. The images were converted to grayscale before SIFT extraction and principal component analysis was not applied. We then used Spark's MLLib to create a decision tree to decide on object classification. The model is then sent to the MongoDB and parsed to create bolts for Storm.

Fig. 1. Generated Storm Topology.



Tuples of frame metadata and SIFT points are streamed in from the Kafka spout and flow into each classification bolt. The results of the classification are sent the mongodb.

IV. Results and Evaluation

TABLE I. Confusion Matrix

Predicted **Actual**			
	home	Man	Car
home	166.0	63.0	47.0
car	281.0	1427.0	405.0
man	84.0	140.0	364.0

- F-MEASURE: 0.6817523304581611
- RECALL: 0.6573731944910984
- PRECISION: 0.7384695474593782

In our evaluation, we used two videos of traffic video. Table 1 shows the confusion matrix of this traffic video data. Our model achieved a 65% precision rate using decision tree model with training data.

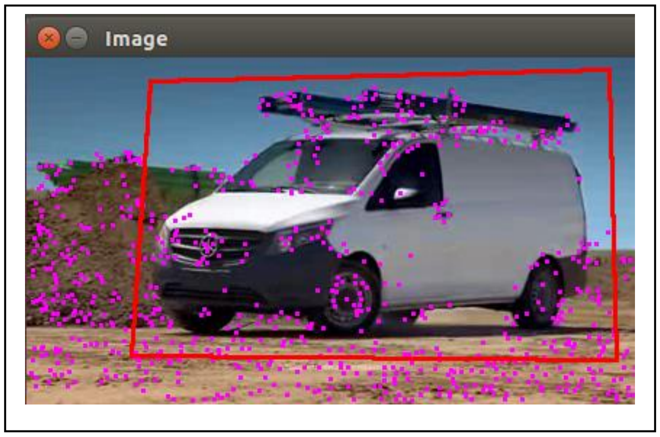
V. Conclusion

Spark uses the MLLib library, use Storm for online classification, and Kafka as the communicator bridge between them. We observe parsing of the model from spark is dubious as the confusion matrix of the test data contains many unclassified points between the bolts which could never happen in a single decision tree. The time taken process the points is a little too long; however, if aggregation schemes are utilized, we would see this reduced as our feature size per frame would also reduce.

VI. Future Work

Our project can be extended from and improved upon the parsing error of the model. Since the current research work is based on less amount of training data ,therefore Mean Square Error is high since the operations are computationally expensive. We can perform on a better on a high specification machine. Furthermore, applying dimension reduction to our SIFT points for model building would be beneficial.

Fig. 2. Annotated image of car.



References

[1] https://people.eecs.berkeley.edu/~pulkitag/scene_report.pdf Goutam & Sailaja. “Classification of acute myelogenous leukemia in blood microscopic images using supervised classifier.” 2015 IEEE International Conference on Engineering and Technology (ICETECH), March 2015. http://ieeexplore.ieee.org/document/7275021/

[2] https://smartech.gatech.edu/bitstream/handle/1853/44929/PAMI_CENTRIST.pdf Tran, Pham & Zhou. “Cell phase identification using fuzzy Gaussian mixture models.” Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, 2005. http://ieeexplore.ieee.org/document/1595447/

[3] "What is big data?" IBM - Bringing Big Data to the Enterprise. URL: https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[4] Dean J, Ghemawat S. "MapReduce: simplified data processing on large clusters". Communications of the ACM Magazine - 50th anniversary issue: 1958-2008, Vol 51: Issue 1, Jan 2008.

[5] Zaharia, Matei, et al. "Spark: cluster computing with working sets." HotCloud 10 (2010): 10-10.

[6] Toshniwa A, Taneja S, Shukla A, Ramasamy K, Patel JM, Kulkarni S, Jackson J, Gade K, Fu M, Donham J, Bhagat N, Mittal S, Ryaboy D. "Storm @Twitter". Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp 147-156, 2014.

[7] Kulkarni, Sanjeev, et al. "Twitter heron: Stream processing at scale." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

[8] Bijnens N. "A real-time architecture using Hadoop and Storm". DataCrunchers, Dec 2013. URL: http://lambda-architecture.net/architecture/2013-12-11-a-real-time-architecture-using-hadoop-and-storm-devoxx.

[9] Kreps J, Narkhede N, Rao J. "Kafka: a Distributed Messaging System for Log Processing". NetDB Workshop, 2011.

[10] Wu, Dongyao, et al. "Building Pipelines for Heterogeneous Execution Environments for Big Data Processing." IEEE Software 33.2 (2016): 60-67.

[11] Neumeyer, Leonardo, et al. "S4: Distributed stream computing platform." 2010 IEEE International Conference on Data Mining Workshops. IEEE, 2010.

VIDEO REFERENCES

A. Training Videos

Commercials of Mercedes Benz collected from YouTube.

B. GitHub Repository

[12] Source: https://github.com/PoojaShekhar/CS5543-Real-Time-Big-Data-Analytics--Lab-assignments/tree/master/Project

[13] Demo:https://github.com/PoojaShekhar/CS5543-Real-Time-Big-Data-Analytics--Lab-assignments/tree/master/Project/demo

The source code can be viewed in segments to indicate milestones and progression by feature. The demo is divided into two videos and shows the full implementation.