International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

Scalable and Reliable Deep Learning Model to handle real-time Streaming Data

L. Amudha, R. Pushpalakshmi

Abstract: ***We have real-time data everywhere and every day. Most of the data comes from IoT sensors, data from GPS positions, web transactions and social media updates. Real time data is typically generated in a continuous fashion. Such real-time data are called Data streams. Data streams are transient and there is very little time to process each item in the stream. It is a great challenge to do analytics on rapidly flowing high velocity data. Another issue is the percentage of incoming data that is considered for analytics. Higher the percentage greater would be the accuracy. Considering these two issues, the proposed work is intended to find a better solution by gaining insight on real-time streaming data with minimum response time and greater accuracy. This paper combines the two technology giants TensorFlow and Apache Kafka. is used to handle the real-time streaming data since TensorFlow supports analytics support with deep learning algorithms. The Training and Testing is done on Uber connected vehicle public data set RideAustin. The experimental result of RideAustin shows the predicted failure under each type of vehicle parameter. The comparative analysis showed 16% improvement over the traditional Machine Learning algorithm.***

Keywords: Real-time stream data, deep learning, Kafka, TensorFlow, Analytics

I. INTRODUCTION

IoT devices and sensors produce a lot of data every second all over the world. There are many challenges in handling real-time streaming applications. Response time, accuracy, data loss, storage, scalability are a few to name. The data from the sensors or source devices are generated and transmitted at a very high speed. Hence if the arrival rate of the stream is faster than the processing speed, all items in stream cannot be considered for analytic processing. Some items shall be dropped out using sampling methods without affecting the performance and outcome. Hence a best method is required to handle the real-time streams.

With the advent of many tools and platforms by the major industry hulks, Apache, Amazon, Google, Microsoft like Apache Storm, Apache Hadoop, Apache Spark, Apache Flume, Apache Samza, Amazon AWS, Microsoft Azure, Google Cloud, and so many, analytics has become a feasible one for anyone who is in the earlier stages of data analytics[2,3,4]. All these platforms support traditional method of static analytics where large amount of data is collected from various sources, pre-processed and stored on large centralized data storage devices. But today all our applications are real-time or near real-time. Hence analytics should consider real-time data as and when generated and produce responses with low latency.

Revised Manuscript Received on February 05, 2020.

* Correspondence Author

L. Amudha*, Department of CSE, K.. Ramakrishnan College of Engineering. Tamilnadu, India. Email: la.cse09@gmail.com

R. Pushpalakshmi, Department of Information Technology, PSNA College of Engineering and Technology, Tamilnadu. India Email: push@psnacet.edu.in

II. APACHE KAFKA

Apache Kafka is one such platform that supports real-time stream handling with scalability, reliability and zero downtime. Apache Kafka accepts input insights from a variety of sensors and devices in real-time Fig.1. Apache Kafka processes event driven micro-services of many such devices. By now it is considered as the best Stream Processing platform. Data collected from different APIs and devices are fed into the Apache Kafka.

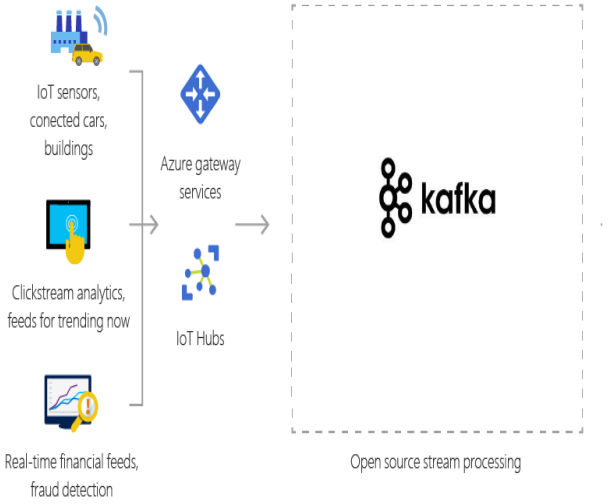

Fig.1 Kafka Input Sources

Apache Kafka is associated with a number of services. These services picks up data from Kafka and do data transformation, data processing and eventually at the end sends a data backup to Kafka itself. The current work is to apply deep learning on real-time streaming applications. Real-time streams are best handled by Kafka. At the same time Deep learning is best implemented with Tensor Flow. Tensor flow is an open source library used for large and complex mathematical computations that involves large amount of numeric arrays.

Tensor flow proves to be an apt choice to implement machine learning and deep learning algorithms. Tensor Flow also supports many built-in algorithms and data sets. One limitation is that these data sets are static. Recent data is not updated in these data sets. Hence in the proposed work it is decided to integrate Apache Kafka that deals with real-time data and Tensor Flow to apply deep learning algorithms on the streaming data.

III. GOOGLE TENSORFLOW

TensorFlow was developed by Google in the year 2015 and used for machine learning applications. This open source software library was basically written in Python, C++ and CUDA. The recent version being TensorFlow 2.0, its impact in real-time streaming pipeline helps to build models quickly. In the earlier versions TensorFlow had the concept of static graphs and the major trend is eager execution. We have to wait for the graph to completely run and wait for results. The major advantage of static graph is that re-declaration can be made reliably in the production environment itself. Even if we want to make any changes, we have to wait until the whole graph completes and get results. Since eager execution is enabled by default in TensorFlow, we will be able to run TensorFlow imperatively just like running a Python program. Another major trend in TensorFlow is the 'tf.data'. It is the data processing pipeline of TensorFlow. It allows us to efficiently process the data and do trimming and inference. The standard set of steps in a complete machine learning task are as follows – Data reading, Pre-processing, Model building, Training and Inference. The entire process is massively simplified in TensorFlow 2.0. The major focus of the data scientists is on Data Reading and Pre-processing.

IV. INTEGRATION OF APACHE KAFKA AND TENSORFLOW

While the earlier models works well with statically available in-built datasets like MNIST Dataset, today there is a huge need for real time stream processing with machine learning[8,9]. Integration of Apache Kafka with TensorFlow is a good solution for this. If there is a way to replace data reading and data pre-processing by some method like Kafka Connector, data from Apache Kafka can be taken as input, instead of taking it from static datasets. There are two major things that made data pipeline a mandatory requirement in Deep Learning – Performance and Flexibility.

A. Performance in Deep Learning:

It is vivid that learning requires lot of resources like IO, CPU and GPU. Since GPU is very expensive, if a machine learning task is waiting for IO to complete all the time, then likely we are going to waste our resources mainly the GPU processing capacity[1,11]. It is very much mandatory to make sure that the investments are getting back. Since GPU is very expensive it should be effectively utilized.

Consider the following scenario in Fig. 2. IO can be a disk input/output operation or from a network. We could wait till all the IO comes in. Most of the GPU is idle. The performance of the system comes down because of the fact th the power of GPU is not fully utilized.

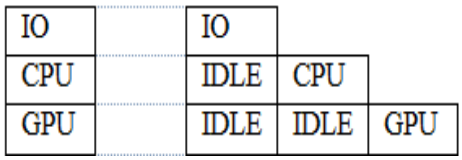

Fig. 2 Data pipeline scenario 1

Fig.3 portrays the scenario where IO is working continuously. At the same time GPU also started working as soon as the data is available for process in GPU. Therefore, there is an increase in utilization of GPU and correspondingly this will drastically improve the performance of the entire system.

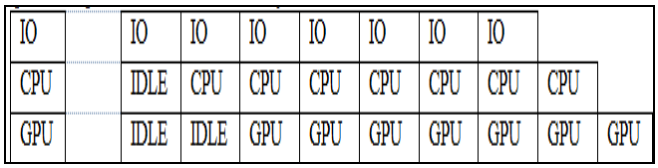

Fig. 3 Data pipeline scenario 2.

B. Flexibility in data formats

Real world has data is in diverse formats[12]. We should make sure that we could support all these format changes. The tf.data in TensorFlow has 3 attributes – TextLineDataset, FixedLengthRecordDataset, TFRecordDataset. While TextLineDataset has text of smaller length to read in bulletin, FixedLengthRecordDatase has text with limited number of lines. The third attribute TFRecordDataset is a sequence of binary strings. The default format of TFSensor is TFRecordDataset. The TFRecord data format is a sequence of binary strings or protocol buffer for serialization and deserialization. This Protocol buffer is also developed by Google and well known for its flexibility and simplicity. It can adapt and allow occupying data of any structure and size. TFRecord is used only by TensorFlow and there is no other ecosystem support for TFRecord out of TensorFlow. Different tools can be used to meet our requirements and make life easier. The schematic view of integrating Kafka with TensorFlow is shown in Fig. 4.

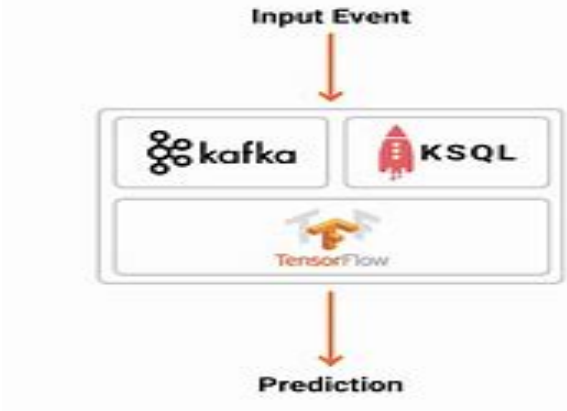

Fig. 4. Integration of Kafka with TensorFlow

TFRecord has no relation with Kafka. It by default works on static datasets from files and produces the output. But since we need to run machine learning on real time streaming applications, the TFRecord is integrated with Apache Kafka that has the real-time streaming input from various devices.
Now we can launch the TensorFlow tasks like pre-processing, training, prediction and then write the message back into Apache Kafka. One challenge here is that all the additional components must be always in "ON" state. Hence a simple way to address the need is required. Once data is imported to TensorFlow graph, TensorFlow allows to do all kinds of transformation and all kinds of graphs can be generated.

C. Steps in Stream analytics with integrated Kafka and TensorFlow

1.Read: Read data from Kafka to TensorFlow
2.Write: Write back TensorFlow to Kafka
3.Preprocessing: Data arrives at millions of messages per second. Depending on the input data format pre-processing is done to do type conversions. A data scientist spends most of the time in Data Pre-processing, filtering, transforming, anonimizing, extracting features. Kafka provides pre-processing functionalities like filtering and transformation features for generic streams. Therefore, smaller dataset will go into the analytic platform. Data Pre-processing solves the impedence mismatch between the data scientist who uses Python, Jupyter and JVM Engineers during both development and deployment.
4.Model building: Model building in TensorFlow creates objects of Tensors and then produces the graph of these objects. Every tensor is computed based on the value of other Tensor objects. Predictions are made by running the different parts of the graph separately and the experimental results are recorded.
5.Inference: Data from TensorFlow should be written back into Kafka after gaining the insights and these data should be written in a continuous fashion. So if a data is coming in sequence or batches it must be ensured that at the end of every batch it is updated into Kafka. We don't want to wait for all batches to complete to write the data back into Kafka. By default TensorFlow Kafka is a model predicting function that retains results.

D. Steps for model building and prediction

The following steps were used in generating an analytic working model,

```
1. import tensorflow_io.kafka as kafka_io
//Load dataset
2.dataset = kafka_io.KafkaDataset("")
#Preprocessing steps
3.dataset=dataset.map()
#Steps for model building
4.model = tf.keras.models
5.model.compile()
6.model.fit()
#Steps for keras callback
7.class OutputCallback(tf.keras.callbacks.Callback):
#initiate with server details and batch size
8.  def.__init__():
#define output sequence
9. self.sequence =
kafka_io.KafkaOutputSequence(topic=topic,
servers=servers)
#initialize batch size
10.  self._batch_size = batch_size
11. def on_predict_batch_end(self, batch, logs=None):
12. self._sequence.setitem(index,
class_names[np.argmax(output)])
#Initate model prediction and display output with
OutputCallback()
13.model.predict(test_dataset,
callbacks=[OutputCallback(32,'topic','localhost')])
```

At the end of every batch, update operation is done in Kafka. TensorFlow provides services with massive simplification. This entire set of process from read, write, pre-processing, model building and inference complete the stream data pipeline.

V. DIGITAL TRANSFORMATION OF A BUSINESS

The current work is built to test the Deep Learning model on real-time data set of connected vehicles Fig 6. Machine Learning framework provides support for predictive maintenance[13]. Whenever an anomaly is detected in any or some of the parameters, the model prediction step, generates the likelihood of occurrence of a specific problem like engine failure, brake failure, fuel about to drain, and so on. Action is taken immediately for any anomaly identified.
Every business provides these 3 types of services

1. Improve customer experience
2. Increase revenue
3. Reduce risk

The working condition of all the vehicles is always automatically monitored by the company's Machine Learning framework as in Fig 7. The training set has 911057 rows and 31 attributes related to a transport vehicle. The model building step identified the parameters of interest like number of surrounding vehicles, speed, latitude, longitude, source, and destination.

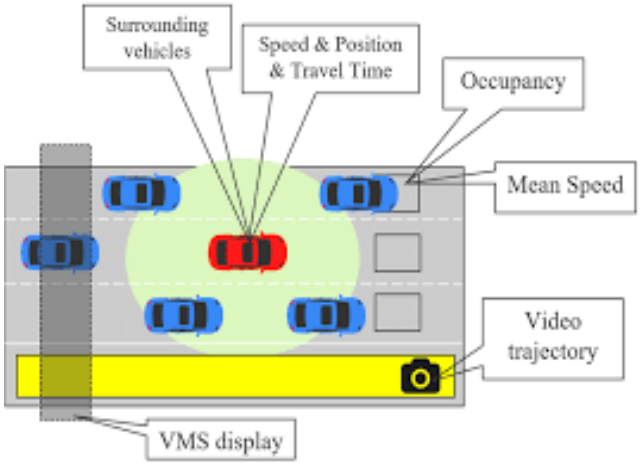

Fig.6 Prescriptive analytics of Connected Vehicles

Business solutions if integrated with the advanced analytic platform will surely provide multitude benefits for all the stakeholders[10].

This paper focuses on finding a problem well before it occurs. The issues would be given as alert to the customer or/and driver together with a set of actions to mitigate that situation. This would greatly reduce the repair work and effort.

VI. EXPERIMENTAL RESULTS OF ANALYTICS ON CONNECTED VEHICLES

Automated monitoring on connected vehicles with the advanced analytic technology proves to be a boon to the business people[7,8]. This architecture paves way for gaining best insights into the daily usage statistics of vehicles under this manufacturer or business provider. Best service ultimately leads to customer delight and in-turn hikes the turn-around of the company. The experimental results show the automatic detection and monitoring system of the transport industry.

The system predicts the problem earlier and prevents it from occurring. Prescriptive analytics can be closely related with the sentence, “Prevention is better than Cure”.

In the case of real-time UBER data set, millions of events get generated every second – 24/7. Analytics with kafka and Tensor is done at the server side. The evaluation resulted in identifying the major attributes that are a cause for failures.

Fig.7. Statistics of connected vehicles

Figure 8 shows that parameters like damaged core, faulty fan, loose fan bet, thermostat, damaged accessories plays a major role in causing machine failure.

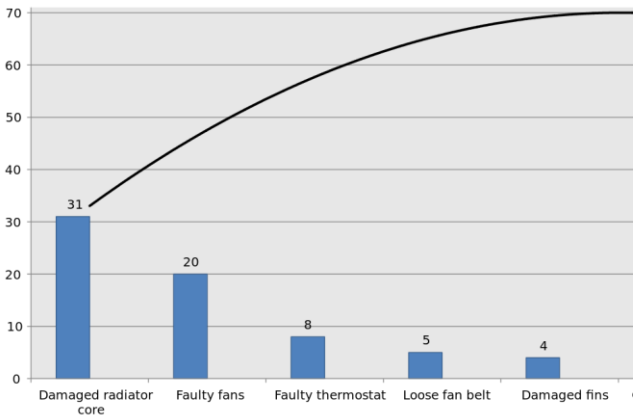

Fig.8. Parameters that cause brake failure

Business solutions if integrated with the advanced analytic platform will surely provide multitude benefits for all the stakeholders[10]. K. Simonyan and A. Krizhevsky et al discusses about the application of CNN in identifying movemetns in the video frames[14,15].

VII. CONCLUSION

This paper does an extensive study on the features and benefits of Apache Kafka with the real time streaming applications. Integration of Apache Kafka and Tensor flow makes possible to implement machine learning and deep learning algorithms in our real-time data. The performance analysis of implementing deep learning in CPU and GPU shows how the idle time can be reduced and the processing units can be effectively utilized. The implementations steps in Kafka and TensorFlow are also revealed. The mileage of vehicles is taken as an output metric. Experiment with real-time connected vehicles of Uber data set showed 16.2% improvement over traditional sets. Further, analytics can bring down the travel time, cost of travel, vehicle maintenance cost, failure frequency and improve customer delight if in future dedicated road structure for autonomous(smart) vehicles is designed and implemented.

REFERENCES

1. Shikhar Verma, Yuichi Kawamoto et al (2017) “A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues”, IEEE Communications Surveys & Tutorials, Vol 19, No. 3, Third Quarter : 1457-1477.
2. Son N. Tran (2018) and Artur S. d’Avila Garcez “Deep Logic Networks: Inserting and Extracting Knowledge From Deep Belief Networks” IEEE Transactions on Neural Networks and Learning Systems, Vol.29, No.2, 246-258.
3. STREAM Group (2003) “STREAM: The Stanford Stream Data Manager”, IEEE Data Engineering Bulletin, http://www.db.stanford.edu/Stream2, no.003
4. Yann LeCun (2015) “Deep Learning”, Yoshua Bengio et al NATURE, doi: 10.1038/nature14539 pp 436-444
5. Olubisi Runsewe (2018) Nancy Samaan “Cloud Resource Scaling for Time-Bounded and Unbounded Big Data Streaming Applications”, IEEE Transactions on Cloud Computing, 2017,pp(99):1-1, DOI: 10.1109/TCC.2018.2876242
6. H. El-Sayed(2018) S. Sankar et al "Edge of Things: The Big Picture on the Integration of Edge IoT and the Cloud", IEEE Access, vol. 6, pp. 1706-1717
7. Stephen Marshland(Second Edition) “Machine Learning – An Algorithmic Perspective”, Machine Learning and Pattern Recognition Series.
8. Charu C. Aggarwal, Data Streams: Models And Algorithms IBM T. J. Watson Research Center Hawthorne, NY 10532, Chapter 1
9. Hulten G. (2001), Spencer, L et al. Mining time-changing data streams. Proceedings of the 7th ACM SIGKDD International conference on Knowledge discovery and data mining San Francisco, California: ACM Press pp: 97–106.
10. Plamen P. Angelov (2012) Autonomous learning systems: from data streams to knowledge in real-time. John Wiley & Sons
11. Jiewan Zheng, Xianbin Cao , Senior Member, IEEE, Baochang Zhang, Xiantong Zhen , and Xiangbo Su , "Deep Ensemble Machine for Video Classification", IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 30, NO. 2, FEBRUARY 2019, pp 553 - pp 565
12. L. Niu, X. Xu, L. Chen, L. Duan, and D. Xu, “Action and event recognition in videos by learning from heterogeneous Web sources,” IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 6, pp. 1290–1304, Jun. 2017.

13. T. M. Nithya et al, "Scope Prediction Utilizing Support Vector Machine for Career Opportunities", I nternational Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958, Volume-8 Issue-5, June 2019, pp.2759-2762
14. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.
15. K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 568–576.

AUTHORS PROFILE

Mrs. .L. Amudha, is working as Assistant Professor of Computer Science and Engineering Department at K.RAMAKRISHNAN COLLEGE OF ENGINEERING, Trichy, Tamilnadu. She received her M.E degree in Computer Science and Engineering from Anna University in the year 2010 and a B.E degree in Computer Science and Engineering from Bharathidasan University in the year 2003. She has published a book titled "Problem Solving and Python Programming". She has also published more than 12 research articles in renowned journals. She is an active member of Institution of Engineers India (IEI). Her research area includes Data analytics, Edge analytics, Computational Intelligence, Parallel Programming.

Dr. R. PushpaLakshmi, is a Professor of Information Technology at PSNA College of Engineering & Technology (Anna University), TamilNadu, India. She received her PhD in Information and Communication Engineering from Anna University in 2014. She received the B.E. degree in Computer Science and Engineering from Madurai Kamaraj University in 2001, and the M.E. degree in Computer Science and Engineering from Anna University in 2004. She has published 3 books in area of object oriented programming. She has published more than 40 research articles in leading journals, and conference proceedings. She is a Life Member of the Indian Society for Technical Education (ISTE). Her main areas of research interest are Wireless Networks, Network Security, Soft Computing and Data Mining. She also holds keen interests in the area of object oriented programming.