International Journal on Applications in Information and Communication Engineering
Volume 9 : Issue 2 : April 2023, pp 14 – 18 www.aetsjournal.com ISSN (Online) : 2394 - 6237

REAL-TIME VIDEO STREAM ANALYTICS USING EDGE ENHANCED CLOUDS

M.THANGAVEL , S. SURYA

***Abstract*—** Growing demand in internet usage, there is growing demand in improved technologies like Internet of Things (IoT), which is connected to devices such as sensors and video cameras, large amountsof streaming data is now being produced at high velocity. Applications which require low latency response such as video surveillance,augmented reality and autonomous vehicles demand a swift and efficient analysis of this data. Existing approaches employ cloudinfrastructure to store and perform machine learning-based analytics on this data. This centralized approach has limited ability tosupport real-time analysis of large-scale streaming data due to network bandwidth and latency constraints between data source andcloud. The proposed work is the RealEdgeStream (RES) an edge enhanced stream analytics system for large-scale, high performance dataanalytics. The proposed approach investigates the problem of video stream analytics by proposing (i) filtration and (ii) identificationphases. The filtration phase reduces the amount of data by filtering low-value stream objects using configurable rules. The identificationphase uses deep learning inference to perform analytics on the streams of interest. The phases consist of stages which are mappedonto available in-transit and cloud resources using a placement algorithm to satisfy the Quality of Service (QoS) constraints identifiedby a user.

***Keywords*—** Internet of Things, Quality of Service, RealEdgeStream, etc

I. INTRODUCTION

Video cameras are the most versatile form of IoT devices. The number of video cameras have grown at an unprecedented rate to increase public safety and security around the globe. France and UK have 1 million and 6 million CCTV cameras installed respectively. They are currently being monitored by human personnel who incessantly stare at a grid view screen to monitor actions or events of interest. Video surveillance has traditionally been performed by operators watching one or more video feeds, the limitations of which are well known. About 90 percent of incidents are missed after 20 minutes as the concentration of the CCTV operators drops.

Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals. The major concept of YOLO is to build a CNN network to predict a (7, 7, 30) tensor. It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location. YOLO performs a linear regression using two fully connected layers to make 7×7×2 boundary box predictions. To make a final prediction, we keep those with high box confidence scores (greater than 0.25) as our final reductions

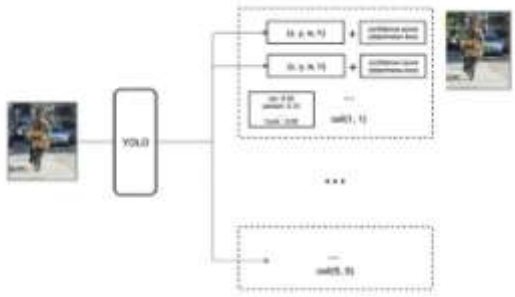


Figure1 : Overview- YOLO detection

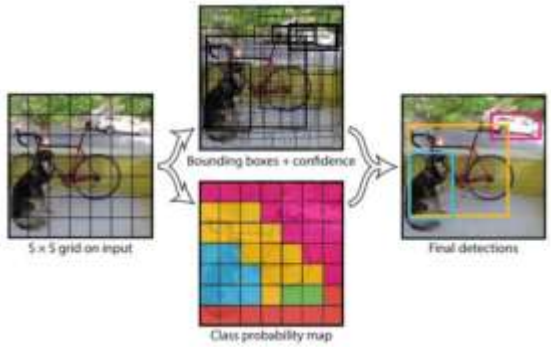


Figure2 : Objects with bounding boxes

Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes.

Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

M.Thangavel, Professor, Department of Computer Applications, Erode Sengunthar Engineering College (Autonomous), Perundurai, Erode. (Email id : thangavelpamu@gmail.com,)

S. Surya, PG Scholar, Department of Computer Applications, Erode Sengunthar Engineering College (Autonomous), Perundurai, Erode. (Email Id : suryamca99@gmail.com)

II. LITERATURE SURVEY

This chapter gives the overview of literature survey. This chapter represents some of the relevant work done by the researchers. Many existing techniques have been studied by the researchers on object detection, few of them are discussed below.

Low-altitude small-sized object detection using lightweight feature-enhanced convolutional neural network

Y. Tao, Z. Zongyang, Z. Jun, C. Xinghua and Z. Fuqiang, in Journal of Systems Engineering and Electronics, vol. 32, no. 4, pp. 841-853, Aug. 2021, doi: 10.23919/JSEE.2021.000073.

Unauthorized operations referred to as "black flights" of unmanned aerial vehicles (UAVs) pose a significant danger to public safety, and existing low-attitude object detection algorithms encounter difficulties in balancing detection precision and speed. Additionally, their accuracy is insufficient, particularly for small objects in complex environments. To solve these problems, we propose a lightweight feature-enhanced convolutional neural network able to perform detection with high precision detection for low-attitude flying objects in real time to provide guidance information to suppress black-flying UAVs. The proposed network consists of three modules. A lightweight and stable feature extraction module is used to reduce the computational load and stably extract more low-level feature, an enhanced feature processing module significantly improves the feature extraction ability of the model, and an accurate detection module integrates low-level and advanced features to improve the multiscale detection accuracy in complex environments, particularly for small objects. The proposed method achieves a detection speed of 147 frames per second (FPS) and a mean average precision (mAP) of 90.97% for a dataset composed of flying objects, indicating its potential for low-altitude object detection. Furthermore, evaluation results based on microsoft common objects in context (MS COCO) indicate that the proposed method is also applicable to object detection in general.

III. EXISTING SYSTEM

- Many edge-based stream analytics systems focus on deadline driven processing, bandwidth limited scheduling and real-time processing.
- Video analytics using edge and in-transit network nodes with a deadline-time for each job and priority based scheduling.
- Gigasight saves bandwidth by running computer vision algorithms on a cloudlet and sending the resulting reduced data (recognized objects) to the cloud. Vigil is an edge-based wireless surveillance system which saves bandwidth by scheduling techniques to support intelligent tracking and surveillance.
- In existing work, edge computing used to perform basic processing by rescaling the frame with increasing algorithm complexity to achieve high-performance.

A. Disadvantages

- Existing work focuses was on getting more jobs accepted and real-time performance, bandwidth conservation and analytics using deep learning models were not discussed.
- The priority of Vigil and Gigasight is to conserve bandwidth while our priority is to optimize performance while saving as much bandwidth as possible.
- Existing approach comes at the cost of accuracy

IV. PROPOSED SYSTEM

- The software running on the cloud performs basic processing such as video loading and motion detection in video frames.
- After basic processing, video frames are passed on to a machine learning model to support object classification and recognition.

A. Advantages

- Fast. Good for real-time processing.
- Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
- YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork.
- Region proposal methods limit the classifier to the specific region. YOLO accesses to the whole image in predicting boundaries. With the additional context, YOLO demonstrates fewer false positives in background areas.
- YOLO detects one object per grid cell. It enforces spatial diversity in making predictions.

V. IMPLEMENTATION AND EXECUTION

The implementation of this application is split into following modules.

- Object detection
- Find object location
- Find object movement
- Update to cloud server

This application detection moving objects using Yolo trained model, it uses recurrent neural networks for object detection. The objects are identified and its co-ordinates are arrived. Comparing it with previous frames we find the object movement. The object class identified is and detection time registered and updated to cloud server.

The project can be executed with web camera and it is the real time detection of objects.

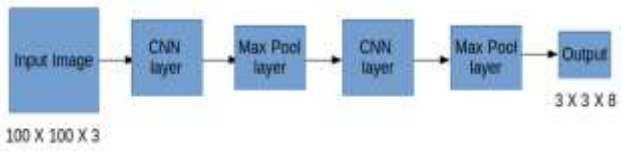
A. MODULES

1) Object detection

In object detection, module, the input of video stream is considered. The input video stream is identified for objects using R-CNN techniques and it primarily use regions to

localize the objects within the image. The network does not look at the entire image, only at the parts of the images which have a higher chance of containing an object.

YOLO first takes an input image from video stream. The framework then divides the input image into grids. Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for object.



During the testing phase, we pass an image to the model and run forward propagation until we get an output y.

2) Find object location

Object localization from the input video stream can be identified. Image classification from object detection, which goes through a ConvNet that results in a vector of features fed to a softmax to classify the object. Neural network have a few more output units that encompass a bounding box. In particular, we add four more numbers, which identify the x and y coordinates of the upper left corner and the height and width of the box (bx, by, bh, bw).

3) Find object movement

The Python's built in deque datatype to efficiently store the past N points the object has been detected and tracked at. Libraries imutils also used by collection of OpenCV and Python convenience functions. By checking the X,Y co-ordinates values and tracking them, the object movement can be detected.

4) Update to Cloud

Similarly, the machine learning stages for object detection were executed on two cloudlets followed by an object filtration stage. Finally, object recognition is performed on the cloud platform for the filtered frames. This configuration uses all the in-transit nodes from the data source to the destination and routes the data to the cloud platform.

B. DESIGN

This chapter gives overview of architecture design, dataset for implementation, algorithm used and UML designs.

C. DESIGN

This chapter gives overview of architecture design, dataset for implementation, algorithm used and UML designs.

VI. SYSTEM ARCHITECTURE

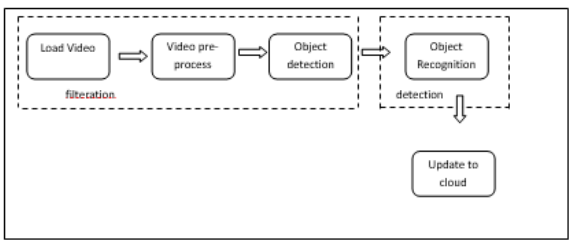


Figure 3: System Architecture

The above figure represents system architecture of proposed system, where we are showing the process of object detection and tracking the object.

A. UML DIAGRAMS

The design is a plan or drawing produced to show the look and function or workings of an object before it is made. Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved in object-oriented software development. It uses graphic notation to create visual models of software systems.

The different types of UML diagram are as follows.

- Use Case Diagram
- Class Diagram
- Activity Diagram
- Sequence Diagram
- Collaboration Diagram
- Component Diagram
- Deployment Diagram

B. USE CASE DIAGRAM

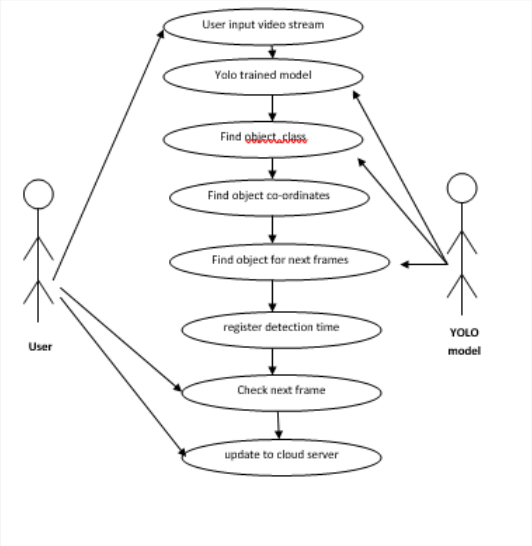


Figure 4 : Use case Diagram

The above figure represent use case diagram of proposed system, where user inputs video frame, the algorithm work to generate the identified output. The actor and use case is represented. An eclipse shape represents the use case namely input image, pre-process, recognition and output.

C. SEQUNCE DIAGRAM

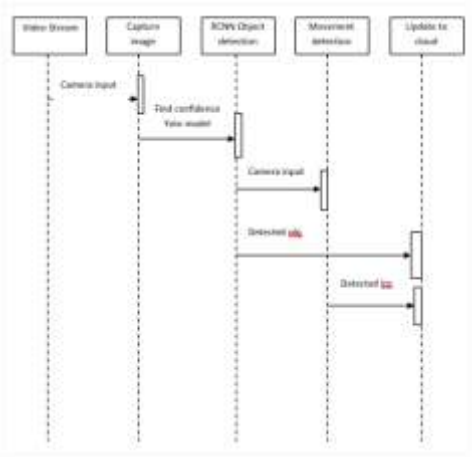


Figure 5: Sequence Diagram

A sequence diagram shows a parallel vertical lines, different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in order in which they occur. The above figure represents sequence diagram, the proposed system's sequence of data flow is represented.

D.ACTIVITY DIAGRAM

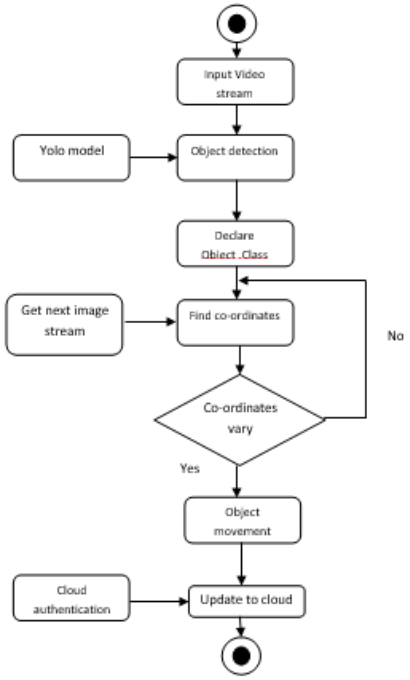


Figure 6: Activity Diagram

The above figure show the activity diagram of the proposed system, where we represented the identified activities and its functional flow.

E. DEPLOYMENT DIAGRAM

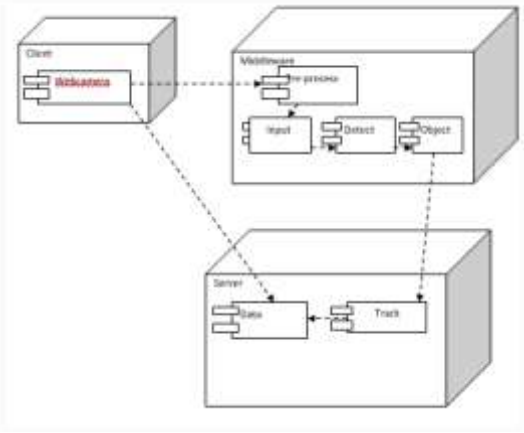


Figure 7 : Deployment Diagram

In the deployment diagram the UML models the physical deployment of artifacts on nodes. The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers

VII. FUTURE ENHANCEMENTS

Future enhancement can be with additional modules embedded to give more detection such as Single shot detection (SSD) module. As a future enhancement, we are also interested to extend YOLO model for more optimization and loss regularization parameters.

VIII.CONCLUSION

An architectural approach for supporting scalable real-time video stream processing using edge and in-transit computing. Current approaches to stream analytics are based on the use of centralized cloud platforms. With the increase in data volumes and velocity, a centralised analysis approach of this kind becomes infeasible for high-performance applications due to limited uplink bandwidth, variable latency and congestion between the data sources and the cloud platform. This approach to video analysis consists of a filtration phase followed by an identification phase. The filtration phase allows objects of low-value to be filtered (discarded) by the edge and in-transit nodes using configurable rules from a user. The identification phase performs deep learning inference on the objects of interest

IX. REFERENCES

[1] C. Shan, F. Porikli, T. Xiang, and S. Gong, Video Analytics for Business Intelligence. Berlin, Germany: Springer, 2012.

[2] M. U. Yaseen, A. Anjum, O. Rana, and R. Hill, "Cloud-based scalable object detection and classification in video streams," Future Gener. Comput. Syst., vol. 80, pp. 286–298, 2018.

[3] T. Abdullah, A. Anjum, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, "Traffic monitoring using video analytics in clouds," in Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput., 2014, pp. 39–48.

[4] Y.-L. Chen, T.-S. Chen, T.-W. Huang, L.-C. Yin, S.-Y. Wang, and T.-

C. Chiueh, "Intelligent urban video surveillance system for automatic vehicle detection and tracking in clouds," in Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl., 2013, pp. 814–821.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[6] W. Zhang, L. Xu, P. Duan, W. Gong, Q. Lu, and S. Yang, "A video cloud platform combing online and offline cloud computing technologies," Pers. Ubiquitous Comput., vol. 19, no. 7, pp. 1099–1110, 2015.

[7] R. Pereira, M. Azambuja, K. Breitman, and M. Endler, "An architecture for distributed high performance video processing in the cloud," in Proc. IEEE 3rd Int. Conf. Cloud Comput., 2010, pp. 482–489.

[8] K. Shvachko, H. Kuang, S. Radia, R. Chansler et al., "The hadoop distributed file system," in Proc. IEEE 26th Symp. Mass Storage Syst. Technol., 2010, vol. 10, pp. 1–10.

[9] C. Ryu, D. Lee, M. Jang, C. Kim, and E. Seo, "Extensible video processing framework in apache hadoop," in Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci., 2013, vol. 2, pp. 305–310.

[10] A. Kafka, "A high-throughput, distributed messaging system," 2014. [Online]. Available: URL: kafka.apache.o