An industrial IoT edge node for buffer level detection in a cardboard production line

G Avramidis¹ and D Karampatzakis¹

¹IEES Laboratory, Dept. of Computer Science, International Hellenic University, Kavala, 65403, Greece

E-mail: dkara@cs.ihu.gr

Abstract. Computer Science and Internet have evolved rapidly the last decades and equally impressive is the evolution of the Industrial Internet of Things technology into factories' shop floors. Among other technologies: modern CPU Architectures, Edge Computing, Deep Learning, Computer Vision, and Low Power Wide Area Networks, are playing a key role in this new competitive environment. In this paper, we present an Industrial IoT Edge Node for level detection on an overhead bridge conveyor (buffer) which is part of a 5-ply corrugated cardboard production line. We focused on the Edge Node and the development of the system was accomplished by using state of the art technologies from disciplines of computer vision and deep learning. We present two implementations using contour detection and CNN techniques. Finally, we implemented a LoRaWAN solution in the IIoT node to send alert messages to the control room. Experimental results are presented for the proposed system implementations.

1. Introduction

A new disruptive technology, the Industrial Internet of Things (IIoT), grows rapidly and its adoption is a challenge for industries worldwide. Based on claims of the research firm Meticulous (2020), the industrial IoT market is expected to grow at a CAGR of 16.7% from 2019 to 2027 to reach $263.4 billion by 2027. The IIoT applications requires to address new concerns about scalability, edge computing, communications, cyber security, and artificial intelligence.

Edge computing is an upcoming set of enabling technologies offering computational and storage resources at the edge of the network [1]. The increased number of IoT devices and applications produces huge amounts of data that needs processing closer to the data sources than to cloud infrastructures. In the IIoT environment the systems are time-sensitive and industrial applications generates enormous raw data which increases the network traffic causing high network latency and packet losses. So, an IIoT system is important to process data closer to the edge of the network performing complex computational tasks in limited hardware resources. Offloading a part of the workload to the edge in IIoT applications is an open research issue and several technological solutions are in progress. In the Industrial IoT because of the complex system characteristics the edge node is not only the producer of the data, but it is critical to be also the consumer of this data. In this context, Deep Learning [2] is a promising approach for data processing in proximity to the data source and succeeding computational offloading to the edge. Deep Learning uses big data to study complex data and in an IoT application could be more efficient compared to machine learning because it is able to exploit large data produced by IoT devices. Another important feature of Deep Learning is that intermediate data extracted from inner neural network layers is reduced compared to input data size which reduces network traffic to cloud servers and has different semantics which ensures privacy.
preservation. So, implementing Deep Learning techniques to the IoT edge layer (End Node or Gateways) is critical for industrial IoT applications and various solutions are demonstrated in recent literature. In the field of IoT communications one on the most widespread Low Power Wide Area Networks (LPWAN) is LoRaWAN [3]. LoRaWAN is an open MAC specification which uses the LoRa PHY. LoRa PHY is a proprietary radio technology of Semtech implementing a Chirp Spread Spectrum (CSS) modulation. In a typical start topology, a LoRaWAN consists of one or more LoRa End Node(s), one or more LoRaWAN Concentrator/Gateway(s), a LoRaWAN Network Server (NS) and Application Server(s) (AS). Depending on the application End Nodes and Gateways are at the edge of the network and NS and AS at the cloud. Also, LoRaWAN is a low power radio technology which extends End Nodes’ battery life and provides two layers of security (network and application) using AES encryption.

In the industrial domain except for the big and multinational companies, in developing countries, many small to medium factories will implement IIoT solutions to improve their productivity and competence. In this paper, we present a low-cost IoT solution at the edge of the network which implements deep learning techniques to solve a specific industrial case. In more detail, we observe the level of flowing paper on the overhead bridge conveyor (buffer) which is part of a 5-ply corrugated cardboard production line. Our approach implements an edge device equipped with a camera that performs video sensing of a standard frame of buffer, implements a deep learning CNN to detect alerting states, and uses LoRaWAN to transmit alerts to the control room.

The remainder of this paper is organized as follows. In the next section we present related work. In Section 3 we present the architecture of our system and in Section 4 we provide more details about the two different implemented methods. In Section 5 we present experimental results and finally in the last section we conclude this article.

2. Related Work
Edge-based Deep Learning in Industrial Internet of Things is a hot issue in research and many approaches for edge offloading are presented in literature. The classical IoT applications with cloud offloading are not ideal for critical and time-sensitive industrial applications because data produced from end nodes needs massive data exchange between servers and IoT devices [4]. Also, because of the resource-constrained end nodes novel frameworks are presented for cooperative edge computing [5]. In the proposed approaches running deep learning models at the edge is challenging because of the limited computing capabilities of edge nodes. Nguyen Gia et al. [6] proposed a 5-layer network framework for smart farming applications with the following layers: Sensor, Edge, Fog, Cloud, and Terminal. In this paper, the authors are using CNN techniques at the edge gateway (Raspberry Pi3) to compress data produced at the end nodes. This approach reduces data transfer to cloud servers and make a LoRaWAN implementation feasible. A. Seferagic et al. in their review paper [7] use International Society of Automation (ISA) industrial systems classification and after their research conclude that LoRa networks, as a single-hop long range network, could be applied to non-critical IIoT systems of 4-6 classes. Based on ISA classification such kind of systems are: Open loop systems, alerting systems and information gathering systems. Additionally, Filho et al. [8] present the performance of LoRaWAN for co-existing telemetry and alarm messages in industrial applications. As LoRaWAN is a pure ALOHA network and packet collisions is one of the biggest issues, the authors presented three configurations of alarm topology using their Spreading Factor (SF) allocation schemes and test them using system level simulation (ns3). Simulation results show that it is possible to achieve 100% delivery of alarm messages in industrial environments but need to design a LoRa network with custom made configurations to have increased network performance. In applied domain, Bhattacherjee et al. [9] implemented an emotion detection IoT enabled edge-node for citizen security in a smart city environment. In this work, the authors present an edge-node running a trained CNN algorithm for face recognition on a Raspberry Pi3, detects facial emotions, and in case of distress situations sends an alarm signal via LoRaWAN. Finally, D. Ziouzios and M. Dasygenis [10], proposed a smart recycling bin which detects recycled materials captured by a camera module attached to the entry area of the bin and communicates with LoRaWAN. In this work the authors implemented the Convolutional Neural Network (resnet34 CNN) for image classification using a Single Board Computer (SBC) with an extra
FPGA SoC hardware (Xilinx Pynq-Z1). FPGAs is a promising hardware technology for executing computing tasks closer to the edge.

3. System Architecture

Our system architecture is presented in figure 1. The proposed IIoT system is attached to the frame of overhead bridge conveyor (buffer). The Edge node is composed of a Single Board Computer (Raspberry Pi3 Model B - ARM Cortex A53 – 1GB SDRAM) to run the trained CNN model (LeNet-5), a USB web camera (Microsoft LifeCam HD-3000) for video sensing, and also the SBC has an attached module of a LoRa Radio (Dragino LoRa/GPS for Raspberry) for wireless connectivity. In our experiments we used an outdoor RAK Wireless 7258 – EU868 LoRaWAN Gateway which communicates via internet with The Things Network (TTN) LoRaWAN Server. Finally, the Edge node system has a wall mount power supply.

![Image of system architecture](image)

**Figure 1.** The overall architecture of the proposed Industrial IoT system.

4. Implementation

In our IIoT system, warning state detection was implemented with two different techniques: a) Contour detection using OpenCV and b) Deep learning using LeNet-5 CNN. In more detail, the two implementations are described in the following subsections.

4.1. Method A – Contour Detection

The Method A is based on contour detection using OpenCV library. The basic concept is that, when there is enough paper on the overhead bridge conveyor (buffer), the level of paper mass is upper the pre-defined limit and when paper mass is lower from the limit, we have a warning state. This can be verified because as more paper is accumulated on the buffer, it will produce bigger layers of paper mass, that will sit one on top of the other. Responsible for the height detection is a USB camera that is connected to a Raspberry Pi3 computer. The USB camera is mounted in way that it captures the flow of paper.

Our system, in order to detect if there is paper in a specific area, tries to detect the colour of the paper in the specific area. So, at the start, the system asks the user to click on a pixel of the camera frame that is presented to the user using the RGB colour space. The next step is to increase the effectiveness of the system, regarding the colour detection, against the variations of the light during video capture from the camera. So, the system “moves” the frame from RGB colour space to HSV colour space, and as the user clicks the pixel with the wanted RGB values, it creates a range of HSV values that correspond to the wanted colour of the pixel. In other words, the system does not try to detect one specific only colour but a family of different shades of the first colour. At this point, the system can detect the wanted colour more easily, even with variations of light during capture. This colour detection is based on the use of a mask on the origin frame to produce a binary image. In order to create this mask, the system needs the upper and the lower HSV values from the colour range that was created in the previous step. Also, it must be noticed, that before the use of the mask the system
performs a smoothing operation on the frame to eliminate some strong edges of the frame. As the mask has been used, and the binary image is created, the system performs morphological operations of erosions and dilations on the binary image of the frame. As a result, a modified binary image is created. In this modified binary image, the system will try to detect contours, that eventually correspond to the wanted colour.

4.2. Method B – CNN Deep Learning
The Method B is based on frame classification using the Lenet-5 Convolutional Neural Network. More specifically, this approach consists of the following parts.

**CNN model:** The Lenet-5 model is considered as one of the very first convolutional neural networks for deep learning as it has six layers. Among these layers there are 3 convolution layers, 2 sub-sampling layers and 1 fully connected layer. Finally, there is one output layer. The model was first used for handwritten digits classification. It takes a picture of small dimensions (e.g. 32 x 32 or 28 x 28) and it provides a classification label for this input image as an output. Deep learning models usually need large amount of data for better training.

![Figure 2. Training Loss and Accuracy on Warning State / Normal State.](image)

**Supervised training:** As was mentioned above, the frames that fed our model were taken using a USB camera that was connected to a Raspberry Pi computer. The position of the camera was fixed but there was no need for a specific view angle. For the training process there were 4,840 frames captured, from actual paper flow on the moving belt. The operator of the machine labeled each frame as warning frame or as normal frame according to his specialized decision. The training process took approximately 15 minutes using an i7 CPU with 16GB of RAM. For comparison purposes the training was also executed on a Raspberry Pi 3 Model B and was completed after 105 minutes.

![Figure 3. Warning and Normal states with high confidence.](image)

**Learning Rate / Epochs / Batch size:** Regarding the training procedure, we should refer that we used 25 epochs to train our model, meaning that the whole training data set was loaded into our model
25 times. The batch size, which is the segment of the training set that is loaded into the model at each step, was set to 32. The learning rate was set to 0.001, which means that the weights of the model are being updated by 0.001 during the training procedure.

For network model implementation, training, and usage, the Keras and Tensorflow libraries are needed to be used in Python environment. In both methods, when the system detects 500 consequent warning frames the machine status is characterized as warning status (figure 3) and the system informs the operator to check the paper flow to prevent pausing of production machinery. In general, both methods succeed in detecting warning states.

5. Results

As it is shown in table 1, Method A seems to have difficulties in detecting frames of normal state, in comparison with Method B. This can be justified considering the difficulty of the detection of the contour above the line limit. Even if the HSV values of the colour that the user has selected, are expanded, it is still insufficient for the perfect detection of the contours inside the wanted area of the frame. On the other hand, Method B seems to analyse patterns from the frames and jumps to semantic conclusions during the training of the model. The process of the training procedure is shown in figure 2. It should be noticed that during the early epochs of the training the train loss is high and the train accuracy is low while during the last epochs of the training the model has achieved higher train accuracy and lower train loss, meaning that its training procedure is normal and efficient. This pattern analysis has as a result, Method B, to be independent of lighting conditions, color selection and the use of a limit line. Regarding the process time, it is shown that there is no significant difference for small duration videos (Video 1 and 2 have 60sec length) between the two Methods. However, for bigger videos (e.g. 240 sec) there is a noticeable difference between 10% and 17%. This can be justified because Method A uses procedures (Gauss, binary conversion, color space change etc.).

| Method | Total Frames | Normal State Frames | Warning State Frames | Time (sec) | Warnings |
|--------|--------------|---------------------|----------------------|------------|----------|
| Method A | 1.800 | 420 | 1.380 | 64 | 0 |
| Method B | 1.800 | 933 | 867 | 60 | 1 |

In table 2 it is shown the noticeable difference regarding the execution time of the video on the Raspberry Pi3 SBC, with and without, the display of the frames, after they were classified by the model of Method B. The display of the frames causes an increment to the execution time of 50 to 70%. This shows that a CNN model can be executed on SBC and classify its input successfully, but when it comes to images, it is better not to force their display as this procedure has a high computational cost.

| Displaying Output Frames | Not Displaying Output Frames |
|--------------------------|-------------------------------|
| Total Frames | Normal State Frames | Warning State Frames | Time (sec) | Warnings | Normal State Frames | Warning State Frames | Time (sec) | Warnings |
| Video 1 | 1.800 | 936 | 864 | 163 | 1 | 936 | 864 | 96 | 1 |
| Video 2 | 1.800 | 1.710 | 90 | 162 | 0 | 1.710 | 90 | 95 | 0 |
| Video 3 | 7.200 | 4.190 | 3.010 | 609 | 5 | 4.190 | 3.010 | 385 | 5 |
| Video 4 | 7.200 | 4.838 | 2.362 | 621 | 4 | 4.838 | 2.362 | 418 | 4 |

Table 1. Comparison of proposed status detection methods on i7 / 16GB.

Table 2. Results of running trained CNN model on SBC.
6. Conclusion and future work

This work presents an Industrial IoT system for buffer level detection in a cardboard production line. The system combines a low-cost SBC equipped with a USB camera, using Computer Vision methods and a LeNet-5 CNN model for warning state detection and communicates via a LoRaWAN network. Our experience from the implementation and the experimental results approves that both Methods A and B can solve the detection problem by analysing the frames from the USB camera. Method A requires the use of several procedures that have to do with image processing (filters, color space, etc.) and OpenCV library makes this task easier, while Method B uses a trained CNN model. Its supervised training needs time and processing power. In comparison with Method A, Method B gives better results, as during the training of the model, the pattern of the frame is analyzed, and semantic conclusions are made. It should be noticed that Method B is more robust as there are no tight restrictions regarding the camera capture position. The main requirement is the training procedure of the model with a training data set taken from real time data. Because of Covid-19 restrictions it was not feasible to test the system in real conditions, so for further work we will plan to test our system in the industrial shop floor and also to test more CNN models and SBCs.

References

[1] Shi W, Cao J, Zhang Q, Li Y and Xu L 2016 Edge computing: vision and challenges. *IEEE Internet of Things Journal*, 3 (5), pp 637-646, doi: 10.1109/JIOT.2016.2579198
[2] Li H, Ota K and Dong M 2018 Learning IoT in edge: deep learning for the Internet of Things with edge computing. *IEEE Network*, 32 (1), pp 96-101, doi: 10.1109/MNET.2018.1700202
[3] Haxhibeqiri J, De Poorter E, Moerman I and Hoebeke J 2018 A survey of LoRaWAN for IoT: from technology to application, *Sensors*, 18 (11), 3995, https://doi.org/10.3390/s18113995
[4] Liang F, Yu W, Liu X, Griffith D and Golmie N 2020 Toward edge-based deep learning in industrial Internet of Things. *IEEE Internet of Things Journal*, 7 (5), pp 4329-4341, doi: 10.1109/JIOT.2019.2963635
[5] Galanopoulos A, Iosifidis G and Salonidis T 2019 Cooperative analytics for the Internet of Things Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc’19), New York, NY, USA, pp 395–396. https://doi.org/10.1145/3323679.3326631
[6] Gia T N, Qingqing L, Queralta J P, Zou Z, Tenhunen H and Westerlund T 2019 Edge AI in smart farming IoT: CNNs at the edge and fog computing with LoRa. *IEEE AFRICON*, Accra, Ghana, pp 1-6, doi: 10.1109/AFRICON46755.2019.9134049
[7] Seferagić A, Famaey J, De Poorter E and Hoebeke J 2020 Survey on wireless technology trade-offs for the industrial internet of things, *Sensors* 20 (2) 488
[8] Santos Filho F H C, Dester P S, Stancanelli E M G, Cardieri P, Nardelli P H J, Carrillo D, Alves H 2020 Performance of LoRaWAN for handling telemetry and alarm messages in industrial applications, *Sensors*, 20 (11) 3061
[9] Bhattacharjee S S, Kumar N T S and Rajalakshmi P 2019 Emotion detection IoT enabled edge-node for citizen security, *IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, pp 925-930, doi: 10.1109/WF-IoT.2019.8767173
[10] Ziouzios D and Dasygenis M 2019 A smart recycling bin for waste classification, *Panhellenic Conference on Electronics & Telecommunications (PACET)*, Volos, Greece, pp 1-4, doi: 10.1109/PACET48583.2019.8956270

Acknowledgments

This work was supported by the MPhil program “Advanced Technologies in Informatics and Computers”, hosted by the Department of Computer Science, International Hellenic University, Greece.