Deep-Learning-Based Insulator Detector for Edge Computing Platforms

Batbayar Battseren
Department of Computer Engineering
Chemnitz University of Technology
Chemnitz, Germany
batbayar.battseren@informatik.tu-chemnitz.de

Mohamed Salim Harras
Department of Computer Engineering
Chemnitz University of Technology
Chemnitz, Germany
mohamed-salim.harras@informatik.tu-chemnitz.de

Soaibuzzman
Department of Computer Engineering
Chemnitz University of Technology
Chemnitz, Germany
soaibuzzman@s2019.tu-chemnitz.de

***Abstract*—In the past years, the object detection applications have witnessed a rapid increase in usage of deep-learning (DL) based solutions, due to their accurate object detection, and robustness to illumination, scale, clutter, rotation changes, etc. Therefore, DL-based approaches started to be used in real-time applications. In an autonomous aerial inspection system, the robust detection and time requirement are critical aspects for the real-time perception and decision making. However, most of the DL models are not suitable for the edge computing platforms, due to their heavy sizes and poor reliability. This paper presents the experimental results obtained from a YOLOv4-Tiny model with a CSPDarknet-53 backbone on different single board computers. The study demonstrates that the performance of the adopted approach is highly dependent on the target platform; and the real-time object detection is reachable in specific cases.**

Keywords—deep learning, object detection, power line inspection, insulator, embedded device, edge computing, autonomous system, unmanned aerial vehicle

I. INTRODUCTION

Unmanned aerial vehicles (UAV) have been used in the past years for different kinds of critical infrastructure inspection, due to the benefits that they offer. The UAV-based inspection provides a number of advancements in the high voltage power line inspection process compared to the conventional methods. For instance, no human life is at risk, no voltage cut-offs, high-quality data collection, high accessibility, less time consumption, and lower cost. However, there are still some weaknesses in the UAV-based inspection which are mainly caused by external factors like the weather condition, the environmental difficulty, or by the pilot experience, response time, or fault.

The power line inspection is a highly automatable sequential process, and most of the object inspections can be performed using vision cameras. The Autonomous Power Line Inspection (APOLI) project addresses the abovementioned problem by offering an autonomous solution for these processes [1]. The advantages are that the autonomous system can perform repeated processes without any problem, have a better control and response time in strong environmental conditions, and provide a better quality inspection.

The unmanned aerial system (UAS) that was developed consists of three main components, which are the copter, the camera setup, and the companion computer. Technically, the copter could be any type of multi-copter that has enough payload to carry the autonomous system's additional components. The flight controller should be open source or controllable by external commands. The camera setup is equipped with navigation and inspection cameras which are mounted on a 3-axis gimbal. The dual-camera configuration delivers simultaneously a wide-angle navigation view and a narrow-angle high-resolution inspection view. The companion computer is directly mounted on the copter in order to run the image processing and autonomous decision-making algorithms in real-time.

In order to develop this type of autonomous system, we proposed an Adaptive Research Multicopter Platform (AREIOM) destined for being applied within the APOLI project [2]. Fig. 1 illustrates the AREIOM software and hardware architecture, where the software components are mapped on a dedicated hardware component. There are several software components developed to execute the inspection mission automatically, namely Expert System (EXS), MAVLink Abstraction Layer (MAL), Camera Gimbal Control (CGC), Acquisition (ACQ), Navigation Image Processing (NIP), and Inspection Mission Recorder (IMR).

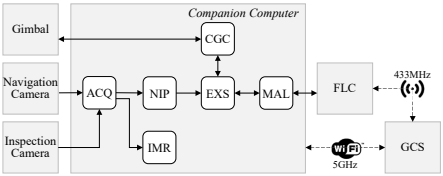

Fig. 1. AREIOM software and hardware architecture

The EXS plays a leading role in running the inspection mission and making real-time decisions [3], while the MAL sends and receives messages from the Flight Controller (FLC) [4]. Together, they work as a flight subsystem. At the same time, the ACQ, the NIP, the IMR, and the CGC are acting as a vision subsystem. The ACQ acquires the live video feed from the cameras and provides it to other processes like the NIP and the IMR. The CGC points the cameras into the desired direction with the help of the NIP output. The NIP detects the inspection object, and the IMR records high-resolution inspection data.

In this paper, we discuss the navigation image processing part, specifically the deep-learning (DL) based insulator detection, including the model training and the experiments performed on different embedded devices.

II. RELATED WORK

Over the past decade, the DL approaches for object detection have seen a consistent increase. This increase was met with a high demand for inspection routines using UAVs equipped with cameras. With the help of edge computing devices and high-performing cameras, inspection routines based on DL became a topic of active research lately. Throughout the DL training and learning of image features, structures are detected accurately and at faster speeds. UAV-based autonomous inspection system requires a continuous stable detection of the insulator string in order to control the flight and inspect the insulators.

A. Feature-Based Insulator Detection

We developed a traditional image processing algorithm for insulator detection, which detects the target object without any machine learning method and parallelism [5]. This approach detects the insulator string based on its unique features, which are symmetrical shape and repeated color pattern. Initial localization of the insulator takes place based on symmetry analysis. The aforementioned analysis extracts image features (pixel intensities, edges) and analyses the edges for symmetry in the Hough space. In the study, we tested the approach on lab images with a solid white background which led to a robust detection of insulators. The same approach was developed further for real-life images in [6]. To achieve the same result in a real-life scenario without using the DL, a color feature-based signal representation analysis method was introduced. In addition to the previous method, this approach detects the repeated color pattern of the insulators on the symmetry axes, which completes the region of interest (ROI) of the insulator. Moreover, an insulator burn-mark damage detection method in [7] was improved and used in the real-life scenario in this study.

The measured average detection frame rate was 15 fps on the machine with an i5 CPU and 8GB RAM configuration. Though this method was computationally expensive and very slow in terms of performance, it has been tested on the Odroid XU4 single board computer (SBC), and the average detection rate was only 5.48 fps. It also led to many false detections since the color feature could be affected by any external factors like illuminance, view angle, reflection, camera configuration, etc.

Jabid et al. [8] defined an approach based on spatial features. The method makes use of morphological operations and defines a spatial model for the detected object. In their method, the concept of a sliding window based local directional pattern (LDP) feature is extracted and support vector machines are used for the classification of the extracted windows.

B. Deep-Learning-Based Insulator Detection

In the past year our team have already conducted several DL based solutions in related studies and projects [9, 10]. In these studies, DL were used in collision warning, outdoor navigation, and depth estimation application. In [11], DL-based single insulator detector approach presented. In the Master Thesis report published at Chemnitz University of Technology [12], a student proposed a DL-based approach for burn-mark detection. A YOLOv5 network has been chosen, and the learning stage is performed with both passive and active approaches. The active learning approach allowed to have a lightweight model with similar or higher accuracy.

Naeem et al. [13] presented a DL-based autonomous vision system to detect faults on insulators based on a YOLOv4-tiny network architecture [14]. To evaluate the performance of their model, the YOLOv4-tiny results were compared to YOLOv3-tiny network architecture on different edge-computing devices such as the Raspberry Pi 4, Nvidia Jetson Nano, Nvidia Jetson TX2, and Nvidia Jetson AGX Xavier. They concluded, using YOLOv4-tiny, real-time detection can be realized on-board of Nvidia Jetson AGX Xavier without a trade-off of accuracy.

Adou et al. [15] proposed a method for insulator bunch-drop detection based on deep learning. The authors opted for the YOLOv3 model to train their network. In fact, two labels were defined for the insulator and the bunch-drop. The model classifies the detected objects and allows to localize the insulator. Logistic regression is the tool used for performing classes probabilities and labels predictions within the YOLO model. The proposed method was able to perform well on a desktop, processing frames at 45 frames per second with a mAP of 83,52%.

In [16], the authors defined a cascading approach composed of the detection network and the classification network. The first network allows to detect the insulators based on RPN, which are then fed to the classification network for fault detection. This allowed for the reduction of the computational cost, while the accuracy and the robustness of the model improved considerably.

A different study targeted the detection of the insulator using the YOLOv2 model [17]. The proposed method used data augmentation to increase the size of their dataset and avoid over-fitting. Hence, they recorded a mAP of 88% with an average prediction time of 0,04 seconds, allowing them to run at real-time.

A deep CNN cascading architecture is introduced in [18] based on the concatenation of VGG and ResNet in a Region-Based Convolutional Neural Network (R-CNN) for the localization of defects on insulators. The cascading architecture allows for two-stage object detection. The localization of the insulator is based on a region-based proposal (RPN) approach and is then followed by the localization of defects in the proposed regions. Another research [19] based the localization of the insulator on a Faster R-CNN, an improved version of R-CNN that cuts the running time of the RPN [20].

In a benchmark published by Nvidia [21], several models with their image resolutions have been tested for inference on most edge computing devices produced by Nvidia, and the ResNet network along with YOLOv4-tiny and SSD (Single-Shot Detection) networks perform best on embedded devices.

In contrast to two-stage detection methods, Lui et al. [22] propose a single-shot detector (SSD) based on the VGG16 model for feature extraction. The latter bases its prediction on 21 class scores for every detected instance. By excluding the region proposal network (RPN), the network is able to meet

the real-time requirements while allowing for fair detection accuracy.

III. Deep Learning Model Training

Fig. 2 illustrates the main steps of the DL-based insulator detector implementation. The data preparation and model training points are discussed in this section. The model deployment is discussed in detail in the next chapter.

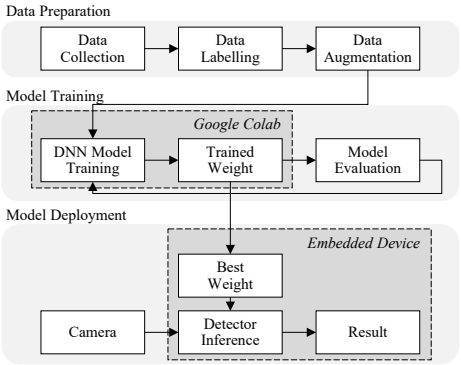

Fig. 2. Workflow of the DL-based insulator detector implementation

A. Dataset Preparation

In the APOLI project, the main inspection object is the high voltage power line insulator, which is an essential part in the power pole. It holds the wire without making any electrical connection with the power pole. The insulators are generally linked together to form what is called an insulator string, and the number of the insulator is defined by the voltage level. Though there are different types of insulators available on the market, the specific insulator targeted in this study is the PS-70 glass insulator, which is commonly used in many countries (Fig. 3).

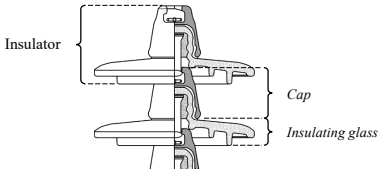

Fig. 3. High Voltage Power Line Insulator

From the computer vision perspective, insulators have their unique features. As they are rotating objects, they have symmetric representation on the image. Furthermore, as they are partially made of glass, they are reflective, and their color changes depending on the environment and the view angle.

1) Data collection

According to empirical bounds for training datasets on computer vision, a rule of thumb is 1,000 images per class [23], though this number can be reduced if pre-trained weights are used. The image resolution is 640×640 and the dataset is collected from real high voltage power line inspection videos, which are recorded under the APOLI project from 2016 to 2019. The whole dataset is categorized into four distinct categories based on their quality and captured environment: good, average, bad, and indoor. The good dataset contains a huge diversification on images with different angles and backgrounds in good quality, whereas the average and bad datasets hold less quality and blurry images. The indoor images are captured in the lab in different lighting conditions. This variation of the dataset helps to make the model more robust, hence, to deal with the over-fitting problem.

After that, images are selected randomly from the training set based on a constant selection ratio. This ratio is selected heuristically: 35% of the image from the good category, 25% of the image from the average category, 20% of the image from the bad category, and 20% of the image from the indoor category.

2) Data augmentation

In order to provide more robustness to the network, data augmentation techniques are applied to the selected images. Detecting the insulator at different angles is a significant challenge when these kinds of images are not included in the dataset. This challenge is overcome by rotating and flipping the training images at a certain angle. Moreover, a salt and pepper filter was applied to make the model robust against the high-frequency noise. The following list depicts the three chosen augmentation techniques:

- Rotation: Both 90° clockwise and anti-clockwise rotations are applied to create scenes with horizontal insulator strings. The dimension is preserved after rotation.
- Flip: Only horizontal flip is applied. This is mainly to create scenes with inversed insulators.
- Noise: A high pass filter is used. The purpose of this filter is to create scenes where the UAV's camera feed is blurred by sudden UAV movements. A 2% salt-and-pepper filter is applied to each image to distort the high-frequency features.

A canvas of the original image and the augmented images are illustrated in Fig. 4.

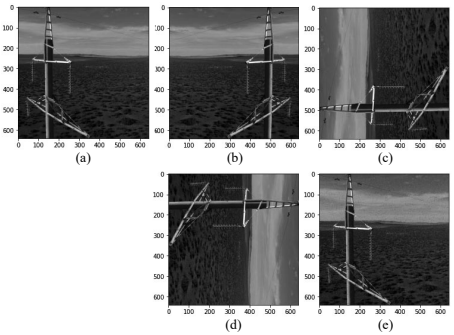

Fig. 4. Illustration of image samples with augmentation, (a) original image, (b) horizontal flipped image, (c) 90° anti-clockwise and (d) 90°clockwise rotation, and (e) 2% salt & pepper noise is added in the original image.

B. Model Training

Being computationally extensive, region proposals-based two-step object detection algorithms are not feasible for resourced constrained embedded systems. By compromising some accuracy, it is possible to gain a significantly higher inference speed with a regression-based one-stage approach. There are a few prominent one-stage object detection algorithms are available out there such as SSD and YOLO. To meet the edge computing requirements, a trade-off in accuracy is unavoidable.

1) Model Selection

Nvidia Jetson Nano is considered as the main target platform for the inference phase in our application. An inference benchmark for DL-based object detection methods on Jetson Nano is developed by the Nvidia developers' community [24]. In this benchmark, SSD with a mobilenet-v2 backbone performed the inference at a faster speed compared to Tiny YOLOv3. Though the inference speed is a bit higher for SSD, the Tiny YOLOv3 showed a much better performance in terms of accuracy with a mean average precision (mAP) of 0.70 at an intersection over union (IoU) threshold of 0.5 on the MS COCO dataset [25]. The further success of the YOLO family, YOLOv4-tiny [26], allowed for an increase in the usability of object detection applications in edge computing systems.

The size and required computational power for YOLOv4-tiny are much smaller compared to YOLOv4. It uses the CSPDarknet53-tiny network as the backbone which uses the CSPBlock module in cross SPN instead of the ResBlock module in the residual network of CSPDarknet53 (Fig. 5). Hence the numbers of convolutional layers are compressed. However, considerable precision is still maintained. To make the computation procedure more efficient, the LeakyReLU activation function is used in the YOLOv4-tiny instead of the Mish activation function. To increase the object detection speed, rather than using YOLOv4's SPP and PAN, it uses two different scales feature maps that are 13×13 and 26×26 to predict the detection results.

The adopted methodology for the study is based on a YOLOv4-tiny network, a regression-based one-step object detection algorithm, and the CSPDarknet53-tiny model backbone.

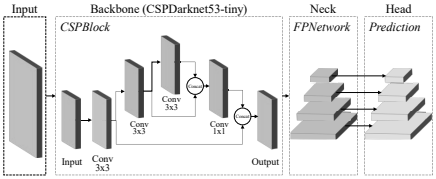

Fig. 5. Architecture of proposed YOLOv4-tiny model with CSPDarknet53-tiny

2) Training Configuration

To build the CSPDarknet53-tiny environment some dependencies need to be fulfilled such as – OpenCV, Cuda Toolkit, cuDNN, and GPU architecture. To enhance this process, Google Colab with Tesla K80 GPU is used. The pre-trained weights of YOLOv4-tiny are used for accelerating the training process and getting better accuracy. A few other configurations have been done to train the model. This optimal configuration has been made based on the official GitHub repository of YOLOv4. Hence, the model architecture is adjusted depending on the number of defined classes in the dataset. The configurations are:

- Batch = 64
- Subdivision = 16
- Max batches = classes×2000, but not less than the number of training images and not less than 6000.
- Steps = 80% and 90% of the max batches i.e., steps=4800; 5400.
- Filters = (classes + 5) × 3 i.e., 18 for our one class.
- Width = 416 & Height = 416.
- Learning rate = 0.00261

The model is trained for 6000 epochs with this configuration. There are 9 weight files created once the training process is completed. One for every 1000 epochs, hence 6 in total. The other 3 weight files are – the best, final, and the last. We used the best weights file for our inference which provides the highest accuracy.

C. Model Evaluation

For the model evaluation, confusion matrix-based metrics are used. There are four main terms in the confusion matrix, namely True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), which are used for calculating the accuracy, precision, recall, and F1-score. The accuracy presents the number of correct classification over the total number of data (1). The precision represents what percentage of the positive prediction were true (2). The recall calculates what percentage of the actual positive were predicted correctly (3). The F1-score combines the precision and recall into a single metric (4). In addition, IoU score is used as an evaluation metric (5) measures the detection accuracy based on the ground truth and the prediction.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (5)$$

Fig. 6 illustrates the calculated loss and mean average precision (mAP) of the proposed model during the training period. The combination of the precision and recall curve is called the precision-recall curve (PRC). The area under the PRC expresses the mAP. The higher the value of mAP, the better the accuracy of the model. The red line represents the mAP over the training iteration, while the blue line represents the loss. The mAP was 93,7 % after 6000 iterations. The binary cross-entropy is used for the loss function and the average loss was 0.0702.

In order to evaluate how the model performs, three different unseen datasets were prepared. The first dataset is collected from the inspection videos, which are recorded by Yuneec E90 high-resolution camera, while the second dataset is taken with FLIR Blackfly S camera. The third dataset is

gathered from an indoor setup with a FLIR camera. In order words, the first two datasets are collected from the same environment with different cameras, while the second and third datasets are collected from a different environment with the same camera. Fig. 7 presents the confusion matrix of the trained model under different IoU thresholds on the first dataset.

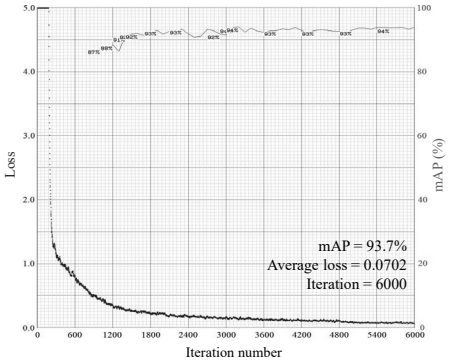

Fig. 6. Loss and mAP graph, red line mAP curve, blue line Loss curve

The trained model has been evaluated on the prepared datasets under different IoU thresholds. For the evaluation 0.5, 0.75, 0.85, and 0.9 IoU thresholds were used to measure the defined metrics. The model evaluation result is summarized in 0 By analyzing the data, it is possible to conclude that the current model performs excellently on 0.5 IoU threshold with mAP of 78.6% – 98.9% along with all datasets. However, as the IoU threshold increases, the model accuracy drops significantly. It is possible to achieve a higher mAP on a larger IoU threshold by increasing the dataset and longer training. But that will affect the inference time.

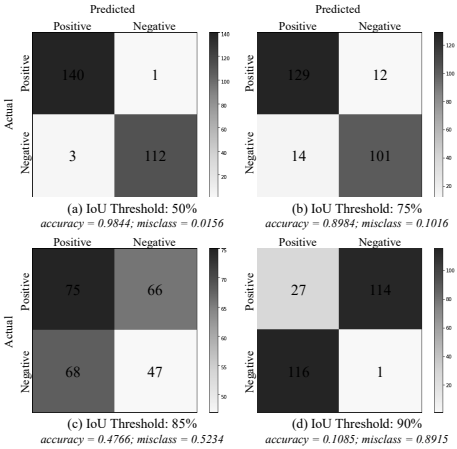

Fig. 7. Confusion matrix results for Dataset 1, (a) IoU threshold at 50%, (b) IoU threshold at 75%, (c) IoU threshold at 85%, and (d) IoU threshold at 90%.

TABLE I. MODEL EVALUATION RESULT

	Dataset 1				Dataset 2				Dataset 3			
IoU thr.	0.50	0.75	0.85	0.90	0.50	0.75	0.85	0.90	0.50	0.75	0.85	0.90
Accuracy	0.98	0.89	0.47	0.11	0.90	0.36	0.16	0.14	0.84	0.51	0.28	0.17
Precision	0.98	0.90	0.52	0.19	0.85	0.43	0.17	0.08	0.82	0.44	0.18	0.06
Recall	0.99	0.91	0.53	0.19	0.98	0.49	0.20	0.09	0.82	0.44	0.18	0.06
F1-score	0.98	0.66	0.53	0.19	0.91	0.46	0.19	0.09	0.82	0.44	0.18	0.06
IoU	0.83	0.77	0.46	0.17	0.64	0.36	0.15	0.07	0.61	0.37	0.16	0.05
mAP (%)	98.9	89.7	32.4	5.01	97.0	24.9	3.96	0.85	78.6	27.0	4.80	0.62

IV. EXPERIMENTATION

A. Hardware setup

The experiment has been done on the copter which was developed according to the AREIOM architecture. Besides the copter itself, the main hardware components are the companion computer and the camera setup. To make the UAS lightweight, compact, portable, and low-power consuming, the vision subsystem is realized as an embedded system. For this reason, the FLIR industrial cameras have been selected for navigation and inspection purposes due to their low weight, compact size, and reasonable performance. The navigation camera is a 3.2 Megapixel global shutter FLIR Blackfly S camera with a Sony IMX252 sensor, which is equipped with Tamron 3-14 mm focal length zoom lens. The camera maximum resolution is 2048 x 1536 pixels, the maximum frame rate is 55 fps, and the lens field of view (horizontal × vertical) is from 105.4°×77.6° to 33.0°×24.8°. The camera uses a USB 3.0 interface, and the maximum throughput is limit to 380MBps.

The experiment has been carried out on two different SBCs, which are Nvidia Jetson Nano and Nvidia Jetson Xavier NX. Both of them have the necessary CUDA cores, high-speed interface, and the same 64-bit architecture. The Jetson Nano has a quad-core ARM Cortex-A57 CPU, 128 CUDA core Nvidia Maxwell GPU, 4 GB 64-bit LPDDR4 memory, and four USB 3.0 interface. Due to the GPU performance, USB3.0 interface, and small form factor, this device is the best low-cost option. On the other hand, the Jetson Xavier has a 6-core Nvidia Carmel ARM CPU, 384 CUDA core Nvidia GPU, 8 GB 128-bit LPDDR4, and four USB 3.1 interfaces, which makes it a better performing alternative with a higher budget.

B. Experiment on the embedded platform

In this section, the trained model's inference time is mainly discussed. The insulator detection model has been tested on the two target platform in different power modes. On the Jetson Nano 5W and 10W power modes, and on the Jetson Xavier NX 10W, 15W, and 20W power modes were used for the experiment (see TABLE II.). The image processing can be divided into image acquisition and image processing steps.

1) Image acquisition

In order to acquire frames from the FLIR industrial cameras, the Spinnaker Software Development Kit is used. Spinnaker is a Generic Interface Camera (GenICam) based Application Programming Interface, which supports FLIR's USB vision cameras and provides necessary functionalities.

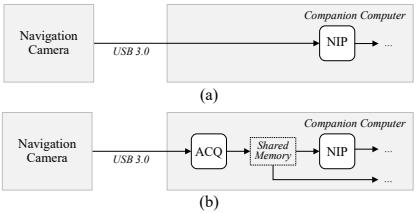

Fig. 8. Image acquisition solutions, (a) Integrated acquisition, (b) Modular acquisition

Fig. 8 illustrates the two image acquisition methods, which are proposed for the system. Each of them has its advantages and disadvantages.

The first option (Fig. 8(a)), the integrated acquisition is a straightforward solution that does not require any additional component and is relatively simple to implement. The NIP acquires images from the camera and directly applies the object detection model to it as a single process. However, this integrated implementation of the image acquisition and image processing would not allow other components to access the same image feed simultaneously.

On the other hand, the second option (Fig. 8(b)), the modular acquisition has relatively complex implementation with separate components and provides a number of possibilities to the system compared to the other alternative. The main difference is the data flow of the image data. The first component, Acquisition (ACQ) configures the camera, reads the video frames with the help of the Spinnaker library, and writes them to the Shared Memory as a byte array. The shared memory is used for sharing the data for other processes, due to some advantages like no data are explicitly moved, high-speed data access, and no overhead. The blackboard architecture is used to implement this data-sharing mechanism [27]. In a nutshell, the blackboard architecture has three main components, namely the Blackboard, which is a common memory space where the data are stored, and the Knowledge Sources, which are the processes that read and write data into the memory. Lastly, the Control component selects the knowledge sources to interact with the blackboard. There can be any number of knowledge sources. According to the AREIOM architecture, every software component is a knowledge source. The second component NIP reads the frames from the blackboard and applies the object detection model to it. This makes the second solution more modular by separating the image acquisition part from the image consumer components. In addition, there can be multiple detection modules and other components which are reading from the shared memory at the same time.

The comparison of the both acquisition approaches results is presented in the TABLE II. In the integrated acquisition approach, the average image acquisition times were 2.042 ms on Jetson Nano and 1.811 ms on Jetson Xavier NX. In contrast, in modular acquisition implementation with shared memory, the average acquisition times were 0.056 ms on Jetson Nano and 0.077 ms on Jetson Xavier NX. The shared memory based implementation decreased the acquisition time by 1.7 ms to 1.9 ms. However, there is no big advancements depending on the power modes.

TABLE II. EMBEDDED PLATFORM TEST RESULT

Device	Jetson Nano (128 CUDA cores)			Jetson Xavier NX (384 CUDA cores)			
Integrated Acquisition							
Power mode	5W	10W	Avg.	10W	15W	20W	Avg.
Acquisition [ms]	2.246	1.839	2.042	2.057	1.723	1.655	1.811
Detection [ms]	111.1	75.67	93.39	24.09	19.739	18.42	20.75
FPS	9.0	**13.21**	11.10	41.50	50.66	**54.28**	48.81
Modular Acquisition							
Power mode	5W	10W	Avg.	10W	15W	20W	Avg.
Acquisition [ms]	0.062	0.050	0.056	0.083	0.079	0.070	0.077
Detection [ms]	112.8	73.48	93.17	24.26	19.45	18.82	20.84
FPS	8.859	**13.60**	11.23	41.20	51.39	**53.13**	48.57

2) Image Processing

There are two possible approaches to predict the target object with our trained YOLOv4 model: the OpenCV DNN module and the Darknet module. Since OpenCV DNN module is optimized for Intel processors, the inference time of the insulator detector was 130.0 ms per frame on the ARM processor. In contrast, the inference time of the Darknet module was 73.48 ms per frame. Accordingly, the Darknet module is selected for further test. TABLE II. presents the measured inference time of the same insulator detector model on different embedded devices. The same trained model performed differently on different circumstances. For example, the detection time was 112.8 ms on Jetson Nano in 5W mode, while it was 20.84 ms on Jetson Xavier NX in 20W mode.

In addition, under the APOLI project, the traditional image processing algorithm has been developed for insulator detection, which is compared with the current DL-based approach on TABLE III. The traditional image processing approach was tested on the Odroid XU4 and Odroid H2 devices, particularly on a CPU. This algorithm is developed without parallelism techniques for running on the CPU cores. Likewise, the DL model is designed to run on the GPUs, and the experiment has been done only on the devices equipped with GPU. The Google Colab experiment result is also included in TABLE III. , which is performed on the Tesla K80 graphic card.

TABLE III. COMPARISON OF THE INSULATOR DETECTION ALGORITHM ON EMBEDDED PLATFORM

Detection Method	Traditional Image Processing		Deep-Learning based Model YOLOv4 tiny		
Device	Odroid XU4	Odroid H2	Jetson Nano	Jetson Xavier	Google Colab
Processing unit	CPU	CPU	GPU	GPU	GPU
CUDA cores	-	-	128	384	4992
Image source	Camera	Camera	Camera	Camera	File
Image resolution	1024x768	1024x768	1024x768	1024x768	416x416
Acquisition [ms]	7.140	1.140	0.050	0.070	0.106
Detection [ms]	175.330	65.030	73.489	18.820	14.743
FPS	5.480	15.110	**13.607**	**53.134**	67.344

The total frame rate of the application is defined by the summation of the acquisition and detection time. Though the detection time takes the biggest portion, as it decreases from 111.1 ms to 18.42 ms (see TABLE II.), the acquisition time ratio increases from 1.98% to 8.24% of the total time.

However, the shared memory-based implementation has shown significant advancement by decreasing the acquisition time by 96.33% on average.

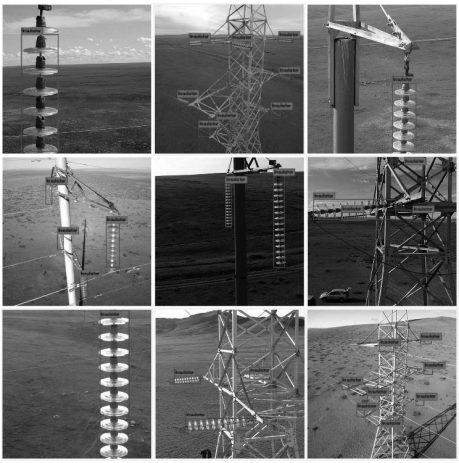

Fig. 9. Detection result of high voltage power line insulators

Fig. 9 shows the detection result of the YOLOv4-tiny insulator detector in different conditions.

CONCLUSION

In this study, the DL-based insulator detector solution developed, and the performance measured and compared on different edge computing devices. In the training phase, we have carefully collected the training dataset to reach higher accuracy and robustness with fewer training data. Furthermore, YOLOv4-tiny architecture is selected for the implementing the real-time insulator detector for the autonomous system.

The lightweight YOLOv4-tiny detector showed promising results on the selected platforms even without any optimization technique. The proposed methodology was able to reach a maximum frame rate of 13.607 fps on a Jetson Nano board. Meanwhile, it recorded a frame rate of 53.134 fps on the Jetson Xavier NX SBC. In order to reach the aforementioned frame rate, shared memory based data sharing method proposed in the image acquisition stage, and it outperformed the conventional method. Based on the obtained data, the hardware configuration, such as the number of CUDA core and the enabled power mode, have a significant impact on the model's performance.

On the other hand, the training approach, the optimization method, and the dataset are also relevant factors that influence the final model accuracy and the detection time. For this reason, our ongoing research will be targeting the usage of advanced training methods, and the optimization of the current model to decrease the inference time and increase the accuracy. In addition, the ongoing research will compare different DNN architectures to find the optimal model respecting the speed/accuracy tradeoff for the autonomous aerial inspection system.

REFERENCES

[1] U. Tudevdagva, B. Battseren, W. Hardt, S. Blokzyl and M. Lippmann, "UAV Based Fully Automated Inspection System for High Voltage Transmission Lines," 12th International Forum on Strategic Technology (IFOST 2017), Ulsan, Korea, May 2017

[2] R. Harradi, B. Battseren, A. Heller and W. Hardt, "AREIOM: Adaptive Research Multicopter Platform," 14th International Forum on Strategic Technology (IFOST 2019), Tomsk, Russia, pp. 219-223, Oct 2019

[3] V. Kühn, R. Harradi. and W. Hardt, "Expert System for Adaptive Flight Missions," Chemnitzer Informatik-Berichte, Chemnitz, Germany, June 2019

[4] M. Stephan, B. Battseren and U. Tudevdagva, "Autonomous Unmanned Aerial Vehicle Development: MAVLink Abstraction Layer," International Symposium on Computer Science, Computer Engineering and Educational Technology (ISCSET-2020), Lauta Germany, pp. 45-49, Oct 2020

[5] B. Battseren, U. Tudevdagva, W. Hardt and A. Banerjee, "Image Processing Based High Voltage Transmission Line Insulator Fault Detection," Proceedings of 13th International Forum on Strategic Technology (IFOST 2018), Harbin, China, pp. 955-960, June 2018

[6] M. S. Harras, "Detection of Physical Damages of High Voltage Power Line Insulator by Image Processing," M.S Thesis, Faculty of Computer Science, TU Chemnitz, 2020

[7] U. Tudevdagva, B. Battseren, W. Hardt and G. V. Troshina, "Image Processing Based Insulator Fault Detection Method," 2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE), Novosibirsk, Russia, pp. 579-583, Nov 2018

[8] T. Jabid and M. Z. Uddin, "Rotation invariant power line insulator detection using local directional pattern and support vector machine," 2016 International Conference on Innovations in Science, Engineering and Technology (ICISET), 2016, pp. 1-4, doi: 10.1109/ICISET.2016.7856522.

[9] S. Saleh, C. Rellan, S. P. Surana, J. Nine and W. Hardt, "Collision Warning Based on Multi-Object Detection and Distance Estimation," International Symposium on Computer Science, Computer Engineering and Educational Technology (ISCSET-2020), Lauta Germany, pp. 68-72, 2020

[10] S. Saleh, H. Saleh, M. A. Nazari and W. Hardt, "Outdoor Navigation for Visually Impaired based on Deep Learning," Proceedings of the 6th International Conference Actual Problems of System and Software Engineering, pp. 397-406, Nov 2019

[11] Ch. Surenjav, D. Byambaa, and U. Tudevdagva, "Deep-Learning-Based Insulator Detection Algorithm Study," Proceedings of MMT 2020 conference, Ulaanbaatar, Mongolia, pp. 28-32, 2020

[12] A. J. Chaudhry, "Burn-Mark Detection Based on Active Deep Learning" M.S Thesis, Faculty of Computer Science, TU Chemnitz, 2021

[13] A. Naeem and S. Peter, "Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections" Electronics, vol. 10, pp. 1091, 2021

[14] A. Bochkovskiy, C.Y. Wang and H.Y.M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," ArXiV, arXiv:2004.10934, 2020

[15] M. W. Adou, H. Xu and G. Chen, "Insulator Faults Detection Based on Deep Learning," 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2019, pp. 173-177, doi: 10.1109/ICASID.2019.8925094.

[16] Z. Wang, X. Liu, H. Peng, L. Zheng, J. Gao and Y. Bao, "Railway Insulator Detection Based on Adaptive Cascaded Convolutional Neural Network," in IEEE Access, vol. 9, pp. 115676-115686, 2021, doi: 10.1109/ACCESS.2021.3105419.

[17] D. Sadykova, D. Pernebayeva, M. Bagheri and A. James, "IN-YOLO: Real-Time Detection of Outdoor High Voltage Insulators Using UAV Imaging," in IEEE Transactions on Power Delivery, vol. 35, no. 3, pp. 1599-1601, June 2020, doi: 10.1109/TPWRD.2019.2944741.

[18] X. Tao, D. Zhang, Z. Wang, X. Liu, H. Zhang and D. Xu, "Detection of Power Line Insulator Defects Using Aerial Images Analyzed With Convolutional Neural Networks," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 4, pp. 1486-1498, April 2020

[19] Y. Wang, J. Wang, F. Gao, P. Hu, L. Xi, J. Zhang, Y. Yu, J. Xue and J. Li, "Detection and Recognition for Fault Insulator Based on Deep Learning," 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1-6, 2018

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," ArXiV, arXiv:1506.01497v3, Jan 2016

[21] "Jetson Benchmarks," Accessed on: Aug. 10, 2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-benchmarks

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, and A.C. Berg, "SSD: Single Shot MultiBox Detector," European Conference on Computer Vision (ECCV 2016), Springer International Publishing, pp. 21-37, 2016

[23] P. Warden, How many images do you need to train a neural network?, Dec 14, 2017, Accessed on: June. 2, 2021. [Online]. Available: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/

[24] "Jetson Nano: Deep Learning Inference Benchmarks" Accessed on: June. 2, 2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-dl-inference- benchmarks

[25] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," ArXiv, arXiv:1804.02767, Apr. 2018, Accessed: Aug. 26, 2021. [Online]. Available: http://arxiv.org/abs/1804.02767

[26] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny," ArXiv, arXiv:2011.04244, Dec. 2020, Accessed: Sep. 14, 2021. [Online]. Available: http://arxiv.org/abs/2011.04244

[27] B. Hayes-Roth, "A Blackboard Architecture for Control," Artificial intelligence, vol.26, pp. 251-321, 1985