Neural network on an ESP32Cam microcontroller for fruit and vegetable

Madiop DIOUF, **Cheikh Sidy Mouhamed CISSE, Birahime DIOUF, Ibra DIOUM, Idy DIOP**

Abstract— **Thanks to machine learning (ML), particularly deep learning, agriculture is undergoing a profound transformation. Convolutional neural networks (CNN), particularly suited to image analysis, have made it possible to develop computer vision systems capable of accurately identifying a wide variety of fruits and vegetables. Our project explores this avenue by using an ESP32-CAM module to create a deep learning-based fruit recognition system. The goal is to automate harvesting by leveraging the ability of CNNs to extract complex visual features, such as fruit shape, color and texture.**

We used the Edge Impulse platform to train our model and deployed it on the ESP32cam module, the model results are displayed as output on the serial monitor with 98% accuracy orange and 87% banana. This work aims to provide a solid foundation for future research exploring the application of deep learning (DL) to fruit detection and recognition in the context of automated harvesting.

***Index Terms*—Computer Vision (CV), Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL.**

I. INTRODUCTION

The fruit and vegetable processing sector perfectly illustrates the vitality of this type of activity who are mainly invested by women through GIEs. Some of these operators, through the development of their own commercial strategies, have also shown the way forward for better positioning of local products on the domestic market where imports are well established. The causes of the fragility and instability of fruit and vegetable processing companies, from the artisanal to the industrial level, are to be sought first in the difficulties of supplying raw materials, perishable, fragile and seasonal products.

Image processing can be used to accelerate, control and optimize processes. When artificial intelligence comes into play, the range of applications expands further. This is particularly true for agriculture and the food industry [1].

Detection and recognition of fruits and vegetables play a crucial role in various applications such as precision agriculture, food supply chains, and supermarket self-checkouts. The emergence of deep learning has revolutionized this field with models capable of learning complex features from visual data RPNs, fully convolutional networks, excel at object localization by generating region proposals. Coupled with Fast R-CNN for classification, they provide a powerful framework for object detection. For the very deep VGG-16 model [1] our detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007 (73.2% mAP) and 2012 (70.4% mAP) using 300 proposals per image. Belgian retailer Colruyt uses AI robot to automatically recognize fruits and vegetables [2]. we optimized inference speed without compromising accuracy. Inspired by industrial applications, such as the Colruyt robot, we aim to deploy this technology on mobile devices for fruit classification, paving the way for new applications in agriculture and distribution. In this paper we attempt to create a mobile device that can classify a variety of species of fruit and legumes by using Edge Impulse platform for acquisition de données Preprocessing of acquired data, Feature extraction, Model training, Model testing, and deployment on ESP32Cam.

The paper is structured as follows: in the first part we will shortly discuss a few outstanding achievements obtained using deep learning for fruits recognition, followed by a presentation of the concept of deep learning. In the second part we describe the Fruits-360 dataset: how it was created and what it contains. In the third part we will present the different framework used in this project [3] and the reasons we chose it. Following the framework presentation, we will detail the structure of the neural network that we used. We also describe the training and testing data used as well as the obtained performance. Finally, we will conclude with a few plans on how to improve the results of this project. Source code is listed in the Appendix.

II. RELATED WORK

In this section we review several previous attempts to use neural networks and deep learning for fruits recognition. A method for recognizing and counting fruits from images in cluttered greenhouses is presented in [4].

The plants studied in this work are peppers with fruits presenting complex shapes and varied colors, often similar to those of their plant environment. The main objective of the application is to locate and count red and green fruits on dense pepper plants grown in a greenhouse. For this, a dataset of 28,000 images, collected on more than 1,000 plants and their fruits, was used for training and validation of the model.

In the article [5], an innovative approach based on deep neural networks is proposed to detect fruits from images. The authors adapted a region-based convolutional network, optimized for fast and accurate detection. This network aims to be integrated into autonomous robots capable of

Madiop DIOUF, Assistant Professor, Department Mathematics, Computer Science, Elhadj Ibrahima NIASS University, Kaolack Senegal.

Cheikh Sidy Mouhamed CISSE, Iba Der THIAM University, THIES, Senegal.

Birahim DIOUF, Alioune DIOP Universite, Bambey, Senegal.

harvesting the fruits. Training is performed using RGB and NIR (near infrared) images, with two types of fusions: early, where RGB and NIR data are combined as input across 4 channels, and late, where they are merged at later stages.

CNNs, key elements of modern computer vision systems, are particularly effective at extracting local features from images. Early models like AlexNet and VGGNet demonstrated their effectiveness in simple image classification tasks, notably for categorizing fruits [6], [2]. More advanced models like ResNet [7] and Inception [8] provide higher performance through architectural innovations such as residual connections. These models not only classify, but also detect several fruits and vegetables simultaneously, paving the way for robust applications in automated agriculture. Object detection allows multiple fruits/vegetables to be located and classified simultaneously: R-CNN, Fast R-CNN, Faster R-CNN: These models are suitable for tasks requiring precise localization [9]. YOLO (You Only Look Once) is an object detection architecture known for its execution speed, is particularly suited to real-time environments. Recent versions, such as YOLOv4 and YOLOv5, have demonstrated excellent results in fruit detection, as shown in studies [10] and [11]. Another popular approach for object detection is the SSD detector.

The rest of the document is organized as follows: Section 1 deals with the dataset and deep learning and convolutional neural networks, Sections 2 presents and discusses the results obtained, the final Section concludes the paper and announces the authors' future work.

A. Deep Learning

Deep Learning (DL), a sophisticated subset of machine learning (ML), has revolutionized artificial intelligence (AI), particularly in image recognition. By employing multi-layered neural networks, such as convolutional neural networks (CNNs), DL algorithms extract complex features from images, enabling highly accurate identification and classification. These networks consist of multiple layers of interconnected nodes, each layer learning to transform input data into increasingly abstract representations [6, 7]. This hierarchical approach has enabled DL models to surpass the performance of traditional machine learning methods in various image recognition tasks.

Deep Learning: it is a machine learning technique based on the neural network model: tensor even hundreds of layers of neural are stacked to bring greater complexity to the establishment of rules [8].

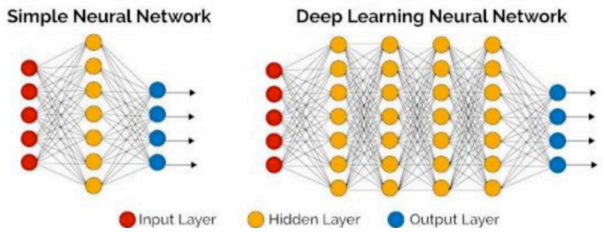


Fig. 1. (a) Simple Neural Network and Deep Learning Neural Network

B. Convolutional Neural Network

Convolutional neural networks (CNNs) are a class of deep learning models. A typical CNN architecture may include convolution layers, pooling layers, ReLU activation layers, fully connected layers, and loss layers. Typically, a convolution layer is followed by a ReLU layer, then a pooling layer, and so on, before concluding with one or more fully connected layers.

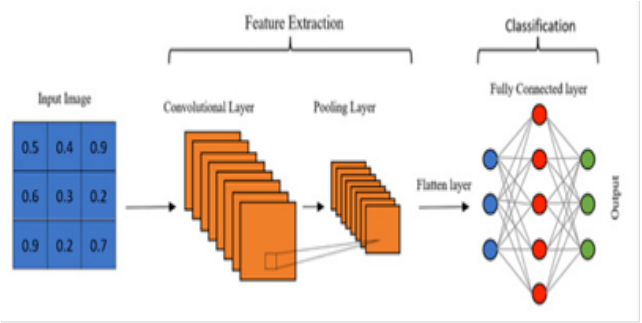


Fig. 1. (b) CNN ARCHITECTURE

Deep learning has demonstrated remarkable success in pattern recognition, achieving superhuman performance in certain domains [9]. This achievement underscores its potential as a crucial step towards Strong AI. Furthermore, deep neural networks, particularly convolutional neural networks (CNNs), have proven highly effective in image recognition. The following sections will provide a brief overview of various deep learning models and their successful applications in related problems. Traditional neural networks flatten the input data into a one-dimensional array, which can diminish their sensitivity to positional shifts. However, multi-column deep neural networks have achieved state-of-the-art results on the MNIST dataset [10] by employing multiple feature maps within each layer and incorporating numerous layers of non-linear neurons, as detailed in [9]. While the complexity of such networks presents training challenges, the advent of GPUs and optimized code has facilitated their development. These networks often utilize winner-take-all neurons with max-pooling for feature extraction.

Further evidence for the superiority of convolutional networks in computer vision is provided in [11]. Notably, [12] introduces an all-convolutional network that achieves exceptional performance on the CIFAR-10 dataset [13, 14] by replacing pooling and fully connected layers with equivalent convolutional operations. Although this approach may increase the number of parameters and introduce inter-feature dependencies, these concerns can be mitigated through the use of smaller convolutional filters and can even act as a form of regularization. The following sections will delve into the details of each layer within a typical CNN architecture.

III. Materials and Method

In this section, we will detail the main tools that we used for this work as well as the deployment of the solution.

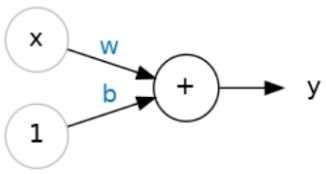
A. How do they work?

The Neural networks consist of layers of interconnected nodes, also known as neurons. Neurons (or nodes). Each node receives input from its predecessors, processes it, and

passes its output to succeeding nodes. The processing involves weighted inputs, a bias (threshold), and an activation function that determines whether and to what extent the signal should progress further through the network.

B. *The Linear Unit*

Let's begin by examining the fundamental building block of any neural network: the individual neuron. A simplified representation of a neuron with a single input can be visualized as follows:"



The linear Unit: y=wx+b

The input is x. Its connection to the neuron has a weight which is w. Whenever a value flows through a connection, you multiply the value by the connection's weight. For the input x, what reaches the neuron is w * x. A neural network "learns" by modifying its weights.

The 'b' term represents a special type of weight known as the bias. Unlike other weights that are multiplied by input data, the bias has a constant input value of 1. This ensures that the bias term 'b' directly contributes to the neuron's output, allowing it to shift the activation function independently of the input values.

The y is the value the neuron ultimately outputs. To get the output, the neuron sums up all the values it receives through its connections. This neuron's activation is y = w * x + b, or as a formula y=wx+b.

IV. Edge Impulse

Edge Computing, also known as AI on Edge, enables the execution of artificial intelligence algorithms directly on local devices, such as microcontrollers or microprocessors, instead of relying on centralized cloud servers. This approach aims to reduce costs and improve system performance by processing data locally, rather than sending it to remote servers [15]. Although cloud computing has become popular due to its lower initial installation and maintenance costs, technological advancements now make edge computing a viable and efficient option. Essentially, edge computing brings powerful processing capabilities directly to where data is generated, making systems faster and more efficient.

By combining image processing with machine learning and deep learning techniques, computer vision systems are revolutionizing many sectors, from industry to autonomous driving technologies, including sustainable cities [16] recognize objects in an image.

V. ESP32 CAM

Data Acquisition: The process begins with data acquisition. Collect multiple images of the object you want the system to recognize [17]. The data acquisition phase will be done with the ESP32-CAM IoT Module based on an ESP32 offering a WiFi interface associated with a miniature camera. This set is ideal for creating connected miniature projects requiring video or photo capture, here are the characteristics:

- Using a low-power dual-core 32-bit processor, which can be used as an application processor.
- Using a low-power dual-core 32-bit processor, which can be used as an application processor
- The main frequency is up to 240 MHz and the computing power is up to 600 DMIPS
- ESP32-CAM-MB camera module
- 1 USB-TTL serial adapter module

Edge Impulse has revolutionized the field of machine learning with the recent release of “Bring Your Own Model” (BYOM) represents a significant advancement in the field of artificial intelligence. This innovative tool empowers developers by streamlining the development and deployment of business models. By facilitating the seamless integration of custom AI models onto cutting-edge devices, BYOM not only represents a technological breakthrough but also marks a crucial step towards democratizing access to advanced machine learning capabilities.

Bring Your Own Model is an innovative feature that allows ML engineers to integrate their own models into the Edge Impulse platform.

With BYOM, users can take their pre-trained models, developed in frameworks such as TensorFlow o PyTorch, and convert them into optimized code that runs on a wide variety of cutting-edge hardware. [12]

Additionally, BYOM makes it easier to transition machine learning models from the development environment to actual deployment. This process is essential for applications that depend on the speed and efficiency of real-time data processing.

VI. Dataset Used

After uploading the code to the arduino IDE we go to the serial monitor we select the frequency 115200, then we enter the IP address on the browser for collecting the images as shown in Figure 2. (A)

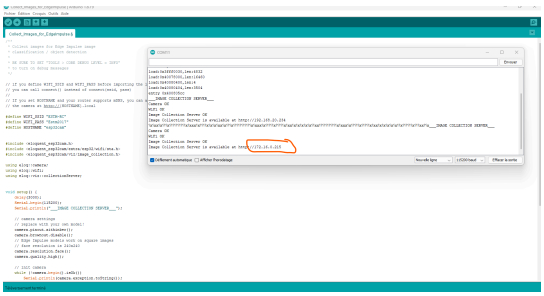


Fig. 2. (a) IP address server on arduino monitor

This step is important for learning to go well. In the event of poor results in the next step, you will have to redo this adjustment and possibly rework the data using Crop sample or/and Split sample (Data acquisition menu then click on the 3 small dots) to remove unnecessary data or better frame

useful data. After entering the IP address on the navigator, the camera of the ESP32 Cam module to acquire the three fruits offered apple, banana and orange.

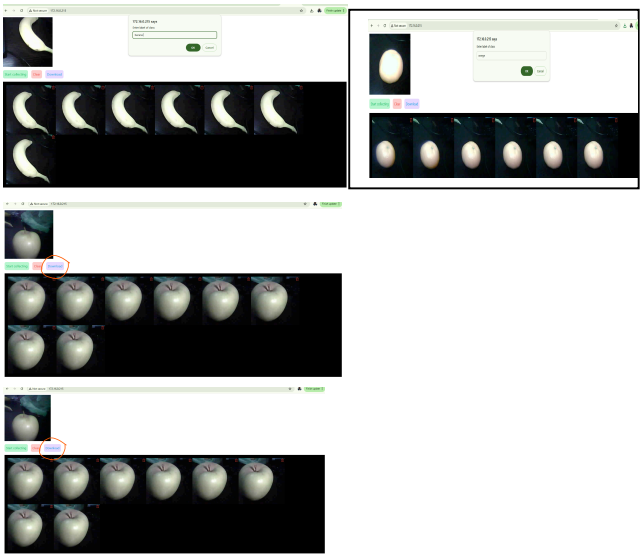


FIG. 2. (B) DATA ACQUISITION

Prior to further analysis, the acquired images undergo a preprocessing stage. This crucial step typically involves noise reduction techniques to improve image quality, contrast enhancement to enhance visual features, and potentially upscaling to increase image resolution. In this particular dataset, we have 24 images representing three distinct fruit varieties.

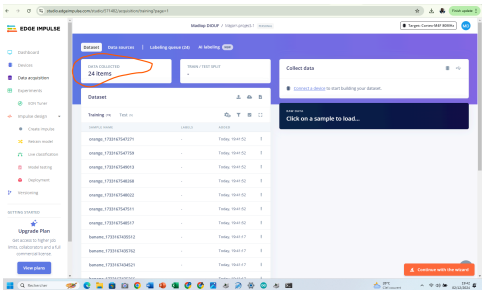


FIG. 2. (C) DATA PRETRAITMENT

1). Feature Extraction: Key features of the image, such as texture, shape, color, and edges, are extracted for use in training the system.

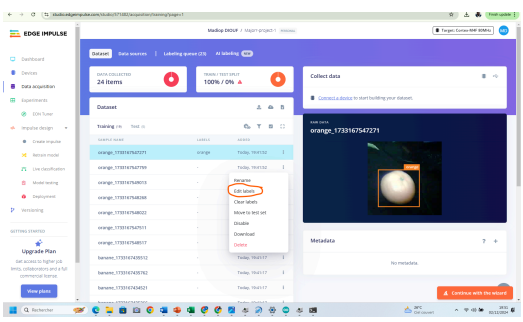


FIG. 2. (E) DATA EXTRACTION

2). Training Model: Machine learning models, such as convolutional neural networks (CNN), are trained on a large dataset of labeled images. The model learns to associate the extracted features with specific objects.

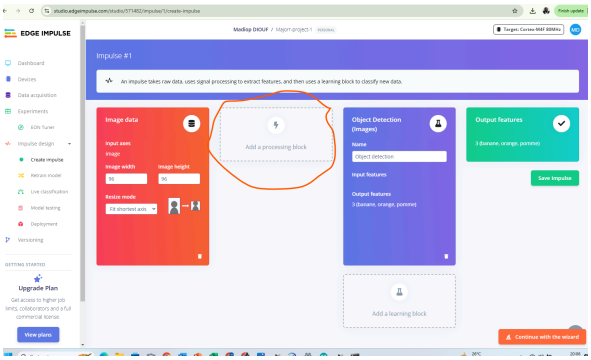


FIG. 2. (G) DATA LEARNING AND TRAINING

3). Testing Model: After training, the model is ready to be tested and deployed. The model recognizes individual objects and compares them to features extracted from the training dataset. It then plots the location and names the detected objects. To improve accuracy, train the model on a larger dataset.

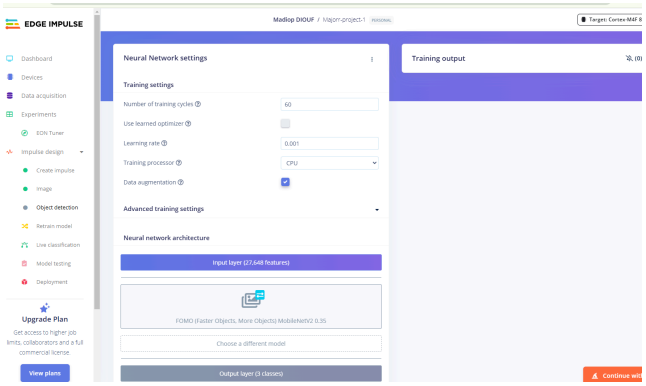


FIG. 2. (H) DATA NEURAL NETWORK MODEL

5). Deployment Model: Based on the system constraints, the format of the trained model is converted and executed on the system. Here, higher accuracy in detection is expected, otherwise, we need to focus more on data acquisition and model training [18].

In the following menu (NN classifier) choose the number of iterations for learning. Choosing a large enough number will increase the calculation time but will, if possible, improve the accuracy of the predictions [19]. Choose the number of neurons in the hidden layers, then start learning. After calculations we can check the learning performance [20].

VII. RESULTS AND DISCUSSION

The following figures show the results obtained after the deployment phase on the ESP32 CAM module. "The results show that the orange and apple classes were detected with 100% confidence, indicating a high probability." "When no classes are detected, the model returns very low confidence or 'no objects detected.

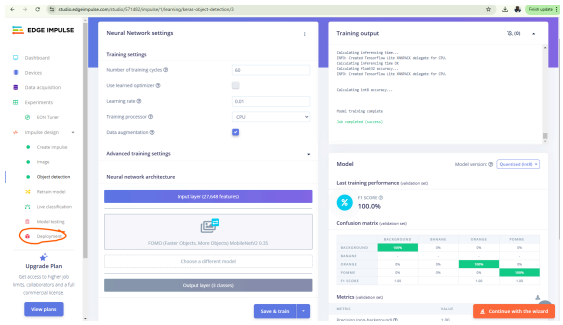


Fig. 2. (I) data Deployment

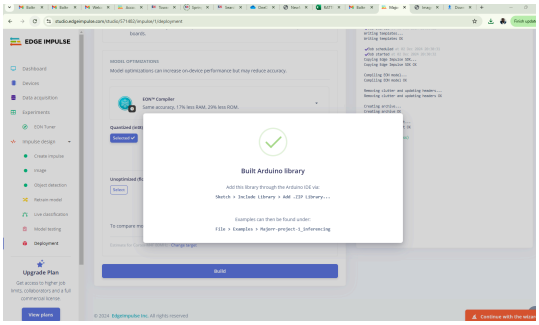


FIG. 2. (J) DATA BUILDING T ESP32 CAM

On the side of the Arduino IDE serial monitor, after the deployment phase, the ESP32 Cam camera detects the three fruits with good position of the object in the field of view of the camera, see figures below.

```
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  orange (0.656250) [ x: 16, y: 72, width: 8, height: 8 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
```

FIG. 2. (K) ORANGE DATA DISIPLAY ON SERIAL MONITOR

```
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  pomme (0.675781) [ x: 16, y: 48, width: 16, height: 8 ]
  pomme (0.656250) [ x: 48, y: 56, width: 8, height: 8 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  pomme (0.601562) [ x: 16, y: 48, width: 16, height: 8 ]
```

Fig.2. (L) Pomme DaTa Disiplay on serial Monitor

```
Object detection bounding boxes:
  banane (0.898438) [ x: 48, y: 16, width: 8, height: 16 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  banane (0.875000) [ x: 48, y: 16, width: 8, height: 16 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  banane (0.820312) [ x: 48, y: 16, width: 8, height: 16 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  banane (0.804688) [ x: 48, y: 8, width: 16, height: 16 ]
Predictions (DSP: 8 ms., Classification: 751 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
```

FIG. 2. (K) BANANE DATA DISIPLAY ON SERIAL MONITOR

We scanned the QR code after deploying Edge impulse, so we obtained a mobile application that can run without internet, we scanned the information with the phone. We obtained 87% accuracy for the banana and 98% for the orange without internet.

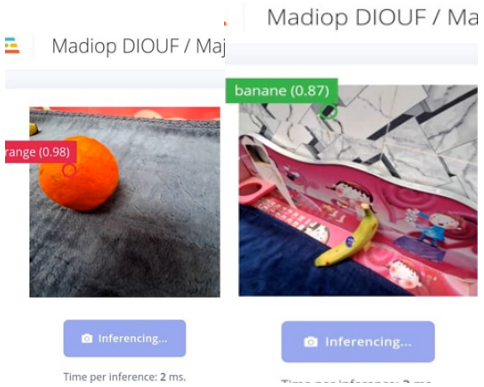


FIG. 2. (L) ACCURACY DISPLAY ON MOBILE PHONE QR CODE WITHOUT INTERNET

VIII. CONCLUSION

Image detection performed with Edge Impulse and an ESP32-CAM demonstrates the power of embedded artificial intelligence technologies, even on low-cost, resource-constrained devices. Here are the key takeaways: Edge Impulse offers a user-friendly interface to collect data, train models, and quickly deploy them to the ESP32-CAM.

Optimization for on-board use: The generated models are lightweight and well adapted to the calculation and memory constraints of the ESP32-CAM. This setup makes it possible to run tasks such as object detection, pattern recognition, or other computer vision applications on cost-effective and portable devices. For Improvement Opportunities, adding diverse training data and increasing the dataset size could improve detection accuracy, Model Optimization by further reducing model size or adjusting hyperparameters via Edge Impulse to improve speed of inference.

Combine the ESP32-CAM with additional sensors (lighting, IR sensors) or more efficient modules like a Raspberry Pi for even better results.

REFERENCES

[1] K. Simonyan et A. Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », 10 avril 2015, arXiv: arXiv:1409.1556. doi: 10.48550/arXiv.1409.1556.

[2] « Le distributeur belge Colruyt utilise l'IA pour reconnaître automatiquement les fruits et légumes ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://www.usine-digitale.fr/article/le-distributeur-belge-colruyt-utilise-l-ia-pour-reconnaitre-automatiquement-les-fruits-et-legumes.N874055

[3] « Edge Impulse lance « Bring Your Own Model » pour les ingénieurs en apprentissage automatique - IA Blog ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://iartificial.blog/fr/d%C3%A9veloppement/Edge-Impulsion-lance-apportez-votre-propre-mod%C3%A8le/

[4] Y. Song, C. A. Glasbey, G. W. Horgan, G. Polder, J. A. Dieleman, et G. W. A. M. van der Heijden, « Automatic fruit recognition and counting from multiple images », Biosyst. Eng., vol. 118, p. 203□215, 2014, doi: 10.1016/j.biosystemseng.2013.12.008.

[5] « DeepFruits: A Fruit Detection System Using Deep Neural Networks ». Consulté le: 8 décembre 2024. [En ligne]. Disponible sur: https://www.mdpi.com/1424-8220/16/8/122

[6] J. Schmidhuber, « Deep Learning in Neural Networks: An Overview », Neural Netw., vol. 61, p. 85□117, janv. 2015, doi: 10.1016/j.neunet.2014.09.003.

[7] « (PDF) Flexible, High Performance Convolutional Neural Networks for Image Classification. » Consulté le: 8 décembre 2024. [En ligne]. Disponible sur: https://www.researchgate.net/publication/220812758_Flexible_High_

Performance_Convolutional_Neural_Networks_for_Image_Classification

[8] « Deep Learning - an overview | ScienceDirect Topics ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://www.sciencedirect.com/topics/computer-science/deep-learning

[9] « Ciresan, D., Giusti, A., Gambardella, L. and Schmidhuber, J. (2012) Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. Advances in Neural Information Processing Systems, 25, 2843-2851. - References - Scientific Research Publishing ». Consulté le: 8 décembre 2024. [En ligne]. Disponible sur: https://www.scirp.org/reference/referencespapers?referenceid=3806722

[10] « MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges ». Consulté le: 8 décembre 2024. [En ligne]. Disponible sur: https://yann.lecun.com/exdb/mnist/

[11] M. Liang et X. Hu, « Recurrent convolutional neural network for object recognition », in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), juin 2015, p. 3367□3375. doi: 10.1109/CVPR.2015.7298958.

[12] M. Liang et X. Hu, « Recurrent convolutional neural network for object recognition », in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), juin 2015, p. 3367□3375. doi: 10.1109/CVPR.2015.7298958.

[13] « Machine learning - How to increase accuracy of All-CNN C on CIFAR-10 test set - Cross Validated ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://stats.stackexchange.com/questions/198463/how-to-increase-accuracy-of-all-cnn-c-on-cifar-10-test-set

[14] « CIFAR-10 and CIFAR-100 datasets ». Consulté le: 8 décembre 2024. [En ligne]. Disponible sur: https://www.cs.toronto.edu/~kriz/cifar.html

[15] « How Edge Computing is Transforming the Future of Technology ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://www.thinslices.com/insights/edge-computing-transforming-the-future-of-technology

[16] A. B. Rashid et M. A. K. Kausik, « AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications », Hybrid Adv., vol. 7, p. 100277, déc. 2024, doi: 10.1016/j.hybadv.2024.100277.

[17] « ESP32 cam-based Object Detection & Identification using Edge Impulse ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://circuitdigest.com/microcontroller-projects/object-recognition-using-esp32-cam-and-edge-impulse

[18] A. Mumuni et F. Mumuni, « Automated data processing and feature engineering for deep learning and big data applications: A survey », J. Inf. Intell., janv. 2024, doi: 10.1016/j.jiixd.2024.01.002.

[19] « Learning process of a neural network ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://resources.experfy.com/ai-ml/learning-process-of-a-neural-network/

[20] « Neural Network Classification | solver ». Consulté le: 11 décembre 2024. [En ligne]. Disponible sur: https://www.solver.com/xlminer/help/neural-networks-classification-intro