UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Richard Aljaste

Vegetable Visual Quality Evaluation System Based on Artificial Intelligence

Bachelor's Thesis (9 ECTS)

Supervisor: Chinmaya Kumar Dehury, PhD

Tartu 2023

Vegetable Visual Quality Evaluation System Based on Artificial Intelligence

Abstract:
Thanks to the rapid development of neural networks in recent decades, applications for this technology have been found in various fields, from medicine to waste management. The same applies to agriculture, where artificial intelligence enables agribusinesses to make decisions based on objective statistical data and through that helps to increase the productivity of the businesses. An example of precision agriculture is the visual quality evaluation of vegetables with the use of machine learning based classifiers. This thesis aims to make such tools more accessible and affordable for small agribusinesses as existing solutions are generally too expensive or cannot be easily integrated into existing processing lines. A new system, Vegeval, is designed and developed to overcome these issues and to provide real-time statistics to agribusiness owners about the quality of their produce. With the use of edge computing, it is shown that a relatively inexpensive system can be built for a hassle-free adoption of precision agriculture processes in existing vegetable processing lines. Consequently, based on the results of the thesis, it can be observed that hardware with low computing resources can successfully be deployed for fulfilling computer vision and object detection tasks in the discussed use cases. The latter additionally indicates that applying artificial intelligence to make everyday tasks more efficient does not necessarily have to come at a large expense.

Keywords:
Vegetable, visual quality evaluation, artificial intelligence, computer vision, object detection, edge computing

CERCS:
P170 Computer science, numerical analysis, systems, control

Tehisintellektil põhinev juurvilja visuaalse kvaliteedi hindamise süsteem

Lühikokkuvõte:
Tänu neurovõrkude kiirele arengule viimastel aastakümnetel on üha enam leitud sellele tehnoloogiale rakendusi erinevates valdkondades, meditsiinist jäätmekäitluseni. Nii samuti ka põllumajanduses, kus tehisintellekt võimaldab põllumajandusettevõtjatel teha otsuseid, mis põhinevad objektiivsetel statistilistel andmetel, ning aitab seeläbi suurendada ettevõtete tootlikkust. Näide täppispõllumajandusest on juurviljade visuaalse kvaliteedi hindamine masinõppel põhinevate klassifikaatorite abil. Selle töö eesmärk on muuta taolised vahendid nii kättesaadavamaks kui ka taskukohasemaks väikepõllumajandustootjatele, arvestades varasemate lahenduste kallidust ja ebapraktilisust olemasolevatesse töötlusliinidesse integreerimisel. Uus süsteem, Vegeval, on loodud eesmärgiga nimetatud probleemid lahendada ja pakkuda põllumajandustootjatele reaalajas koostatud statistikat nende toodangu kvaliteedi kohta. Kasutades servtöötlust näidatakse, et on võimalik luua suhteliselt taskukohane süsteem, mida saab vähese vaevaga integreerida ka olemasolevatesse köögiviljatöötlusliinidesse. Sellest tulenevalt saab töö tulemuste põhjal täheldada, et vaatluse all olnud kasutusvaldkondades saab tehisnägemist ja objektituvastust rakendavaid ülesandeid edukalt lahendada ka väikese arvutusliku võimekusega riistvaral. Viimane ühtlasi viitab sellele, et tehisintellekti rakendamisega igapäevaste ülesannete efektiivsemaks täitmiseks ei pea tingimata kaasnema suuri kulutusi.

Võtmesõnad:
Juurvili, visuaalse kvaliteedi hindamine, tehisintellekt, tehisnägemine, objektituvastus, servtöötlus

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimis-teooria)

Contents

Acronyms

API Application Programming Interface

CAN Controller Area Network

CNN Convolutional Neural Network

COCO Common Objects in Context

CSS Cascading Style Sheets

DCNN Deep Convolutional Neural Network

FPS Frames Per Second

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IoT Internet of Things

IoU Intersection over Union

IP Internet Protocol

IR Infrared

JWT JSON Web Token

LAN Local Area Network

mAP Mean Average Precision

MQTT Message Queuing Telemetry Transport

MSRP Manufacturer's Suggested Retail Price

ORM Object Relational Mapping

RCNN Region-based Convolutional Neural Networks

REST Representational State Transfer

RFCN Region-based Fully Convolutional Networks

RPC Remote Procedure Call

RPM Revolutions Per Minute

SDK Software Development Kit

SR Softmax Regression

SSD Single Shot Detector

TFLite TensorFlow Lite

TLS Transport Layer Security

TPU Tensor Processing Unit

UV Ultraviolet

YOLO You Only Look Once

1 Introduction

The process of evaluating the visual quality of a vegetable is based on classifying the visual appearance of the vegetable into a predefined quality category or class, without physically examining it. With the use of object detection models as a subset of artificial intelligence, this process is automated and integrated into a system which provides the end user with real-time statistical data about the quality of their vegetables under evaluation.

Thanks to the rapid development of neural networks in recent decades, applications for this technology have been found in various fields such as medicine [1], the automotive industry [2] and waste management [3]. The same also applies to agriculture [4]. Precision agriculture, through machine learning, allows agribusinesses to make decisions based on statistical data that is less prone to human error, thus helping to increase the businesses' productivity. One specific method for achieving this is the visual evaluation of crop quality and planning subsequent actions in the production chain based on the statistical results, such as sorting out unsuitable produce or reorganising some preceding processes.

This thesis builds upon and adds to existing solutions by proposing a new all-in-one system design and implementation for evaluating the visual quality of vegetables that is cost and usability-wise more approachable for small agribusinesses than existing solutions. The system consists of two subsystems, a user-friendly web application and an on-premise Edge Device. From the end user's point of view, the former is used for managing and monitoring the output of the Edge Devices, while the Edge Devices themselves use object detection models to detect and classify vegetables on a conveyor line to generate statistical reports of the quality of the produce. A set of object detection models is trained and compared on a custom dataset of potatoes as a proof of concept for the solution.

In Section 2, an overview of the previous works along with a proposal for a new system is given based on the observations made on the analysis. Section 3 continues to describe the proposed system in detail by providing the functionality of the subsystems and communication between them. The web application's architecture is given in Section 4, where the structure of the frontend and backend are described along with the tools used for developing the web application. In Section 5, the dataset and the training procedure of the object detection models are described. Section 6 gives an overview of the system metrics, such as the throughput capabilities of an edge device and the performance of the object detection models. Finally, in Section 7, the work is concluded and thoughts for future improvements are discussed.

2 Related work and background

In order to get an overview of the existing solutions and their shortcomings, the related work must be analysed. The following subsections provide an overview of two previously developed machine learning based classification methods for visually evaluating vegetable quality and two conveyor-based object sorting systems. Based on the analysis of the previous works and improvements in technology since their publishing, an alternative new system is proposed.

2.1 Methods of visual vegetable quality evaluation

The research studies under review describe the possibilities of applying neural networks to assess the quality of potatoes visually, without physically examining them [5, 6]. In addition to describing the methods used for collecting data and training the classification model in each of the research, the performance metrics of the trained models are highlighted along with the authors' ideas for further development. Specifically, models trained for detecting and classifying potatoes are reviewed, as the proof of concept vegetable quality Evaluation Module presented in Section 5 of this thesis is also based on potatoes.

Detailed data from a depth camera. In 2020, a research group working on potato quality assessment [5] proposed using a depth camera [7] for mapping the shape of potatoes in order to capture irregularities on the surface of the potatoes in more detail than a regular colour camera could. The same technology had previously been used to calculate the length, width, height, and volume of potatoes for creating a 3D model and predicting mass [8]. The authors of the study [5] mention that previous potato classification models' accuracy had largely been dependent on the external environment, such as specific light sources, which highly influences the outcome of the input frames and therefore require more work to achieve a similar result when the environment changes. They point out that the results are also affected by the appearance and shape of the potatoes, which can change over time, so taking all of this into account, it is necessary for such a classification model to be updated and retrained periodically with new data. The research group created two machine learning models for predicting potato quality: a Softmax Regression (SR) model[1] and a Convolutional Neural Network (CNN) model[2]. For either of the models, the dataset was manually labelled into classes based on the quality and mass of the potatoes:

[1] Softmax regression is a linear model that predicts the probabilities of different classes in a classification problem using the softmax activation function, which takes the exponential of each element in the input vector and normalizes the resulting values between 0 and 1.

[2] Convolutional neural network is a type of neural network model designed for image processing that uses different layers to automatically learn complex patterns in image data.

- „Abnormal Big“ (misshapen, damaged or sprouting with a mass of >=300 g),
- „Abnormal Medium“ (misshapen, damaged or sprouting with a mass of 100-300 g),
- „Abnormal Small“ (misshapen, damaged or sprouting with a mass of <100 g),
- „Normal Big“ (with a standard shape and a mass of >=300 g),
- „Normal Medium” (with a standard shape and a mass of 100-300 g),
- „Normal Small“ (with a standard shape and a mass of <100 g).

After processing and combining initial depth images using mean calculation, a total of 7084 depth images with a resolution of 200 by 200 pixels were obtained of 296 randomly selected potatoes with different masses and appearances. In an attempt to improve the accuracy of the models, data augmentation techniques such as random rotations, reflections and translations were applied to existing images for all 500 training epochs separately. Data augmentation is a technique commonly used in machine learning to increase the size and diversity of the training dataset by generating modified versions of existing data [9]. Next, 5691 images (80% of the dataset) were selected for training and the other 1393 images (20% of the dataset) were left for validation. Among the models, the best results were obtained from the CNN model with an overall accuracy of 86.6% and a loss of 0.304 on the validation set after 500 epochs, while the SR model achieved an accuracy of 67.2% and 0.777 loss. The models predicted the size of potatoes with practically the same accuracy (94.5% for the CNN model and 94.4% for the SR model), however, the CNN model was ahead in classifying the appearance (91.6% accuracy compared to 70.6% for the SR model). It is important to note, however, that processing the validation data took eight times longer for the CNN model than the SR model (56 and 7 seconds respectively). The authors [5] admit that the accuracy could likely be increased with a larger dataset and using combined data from the depth and additional colour images in the models.

Using object detection for classifying potatoes. In an article published a year later [6], another research team using an industrial colour camera instead compared three different Deep Convolutional Neural Network (DCNN) based object detection models pre-trained on the Common Objects in Context (COCO) source dataset [10]. Transfer learning [11] was used to achieve more accurate results with a small target dataset, essentially building on top of an existing model trained on a source dataset as a starting point. The approach described in the article [6] differs from the previous [5], in addition to using a colour camera, by applying an object detection model on the input image instead of an image classification model, with potatoes belonging to three appearance categories as the objects:

- "Normal" (yellow, undamaged, edible),
- "Scratch" (artificially scratched),
- "Sprout" (sprouting).

Although object detection was applied to individual potatoes in this particular study [6], the same approach could be used to detect and classify a larger number of potatoes on a single image, which is an advantage of object detection models over classification models. To evaluate the three models, the research team captured 2770 images of 642 potatoes that were manually labelled into the before mentioned categories, reducing the resolution of the images on the short side to 600 pixels and dividing them into training and test sets in a ratio of 3:1. The models compared were Single Shot Detector (SSD) Inception V2, Region-based Fully Convolutional Networks (RFCN) ResNet101, and Faster Region-based Convolutional Neural Networks (RCNN) ResNet101[3]. From the selected models, RFCN ResNet101 achieved the best balance between accuracy and inference speed[4], 95.6% and 28.5 Frames Per Second (FPS) respectively, and the rest having similar accuracy, yet varying speed (with the smallest accuracy of 92.5% but the highest speed of 51.3 FPS from SSD Inception V2 and accuracy of 98.7% and speed of 21.2 FPS from RCNN ResNet101).

Performance of the models [6] was measured on a workstation configured with an Intel Core i7-8700 CPU, 16 GB of RAM and an Nvidia GeForce RTX 2070 GPU. The accuracy of RFCN ResNet101 was also evaluated with an additional 642 potato images collected from outside of the original dataset to see how the model would cope with data that was completely different from what it had seen before. By analysing the test results, it was found that it is important to capture the potatoes from different angles in order to avoid situations where the appearance features required for objectively classifying the objects are not visible. In addition, it was found that the quality of the image decreased in too dark or bright environments, making it difficult to effectively find the patterns sought from the images.

Observations. In conclusion, several useful observations can be made from the research [5, 6]. For one, convolutional neural network based models prove to be a good choice for accurately classifying potatoes of different size and appearance categories, while achieving inference speeds of above 20 and up to 50 FPS when using above-average performing hardware as in the case of the latter article. To achieve high model accuracies, however, a large training dataset of at least several thousand instances or the use of

[3]The models compared were no longer publicly accessible from the original source at the time of writing.

[4]Inference speed refers to the time it takes for a model to process an input image and generate a prediction.

transfer learning techniques is required. What is more, object detection models can be used to classify multiple objects at once, which is an effective means to increase the throughput of objects in the classification stage. As mentioned in both of the studies [5, 6], whenever colour images are used, it is essential to ensure that the lighting conditions are sufficient for recognising distinctive patterns on objects and that the conditions remain the same from the training to deployment stages of the model. The latter article also suggests that in order to guarantee an objective evaluation of a potato, it must be captured from different angles.

2.2 Conveyor-based object sorting systems

This subsection analyses research studies that propose two different conveyor-based systems for sorting fruit [12] and a variety of different objects, including tomatoes [13], respectively. In particular, the system architecture, the hardware used and the communication between different system components are reviewed. Object detection and classification in these systems only focus on the shape and colour of the objects and are not performed using modern machine learning based methods as required by the proposed system in Section 3 of this thesis.

High-speed fruit sorting. The authors of the article published in 2001 [12] were driven to their work by the demand for a more modular and cost-effective solution for sorting fruit than what was available and economically viable for small fruit packaging companies in Spain at the time. The development of the project was partially funded by an agricultural machinery company with other participants in the field. Architecturally, the system consists of a central control unit, a user interface and storage unit, a number of weight and vision modules as well as output control units. A high-level overview of the system structure can be seen in Figure 1. As the name suggests, the central control unit is at the centre of all the other components, being connected via a Controller Area Network (CAN) bus. This network is used for any time-critical communication such as control signals and classification data between the control unit and modules. Additional connections from the central control unit to the user interface, vision modules and any other components that do not require real-time communications are transmitted over Local Area Network (LAN).

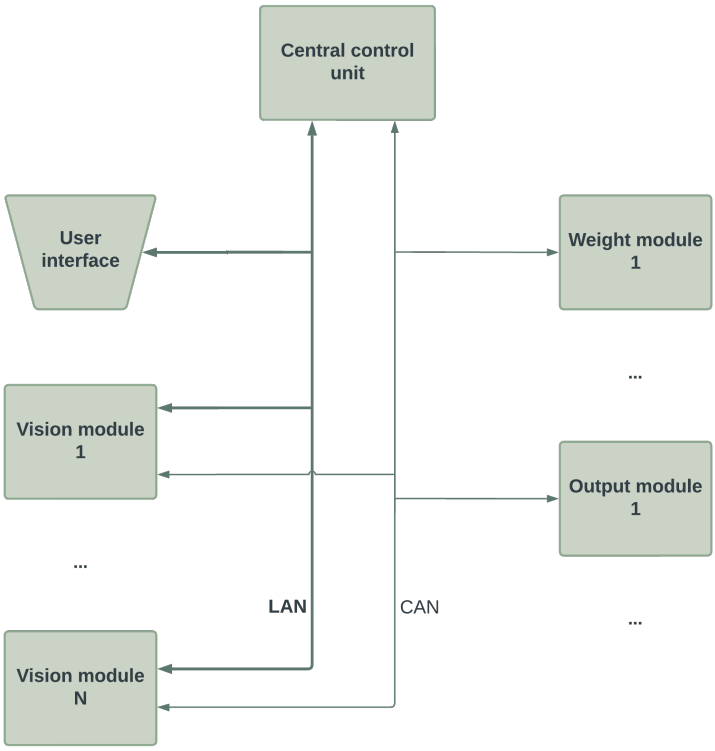

Figure 1. System modules and connections [12].

The vision modules used in the system [12] are individual embedded computer systems with their own operating system, each of which is used to capture images, classify the colour properties and estimate the size of fruit on two separate conveyor lines simultaneously. These systems can be configured to use different combinations of colour, Infrared (IR) and Ultraviolet (UV) cameras at a time, based on the requirements and budget of the end user. Results obtained from the vision modules are sent to the control unit over the CAN bus, while instructions from the LAN can also be accepted. Images of the fruit are captured in a synchronous manner when the fruit enters an illumination chamber specially designed to evenly distribute light on the fruit surface. As the fruit is singulated and rotated by transport rollers while moving through the illumination chamber, up to four different images and angles of each fruit are captured to make an objective decision about sorting the fruit by weight, size and colour. Another important component of the fruit grading system is an arguably easy-to-use graphical user interface, which amongst other features is used to perform an initial set up of the system, monitor statistics, configure classification parameters and calibrate vision modules. The collected statistical data appears to be stored locally, on-premise.

In terms of performance, the system [12] was measured to be able to process up to

15 fruits per second on each of 10 conveyor belts simultaneously. The speed is said to be mainly limited by the frame rate of the cameras used for capturing images, which is 30 FPS. Future improvements are focused on detecting and classifying smaller defects on the fruit surface and therefore implementing more advanced image processing methods.

Using IoT to sort objects. Narzary and Ashok from the National Institute of Technology in Calicut describe a similar system in their work published in 2019 [13], but on a smaller scale by using Internet of Things (IoT) devices to sort boxes and tomatoes based on a single characteristic such as their shape or colour. The use of IoT technology, as opposed to specialised hardware, is reasoned by the flexibility in the end user's location in relation to the devices, as well as being able to add new devices without worrying about wiring as the communication with other network devices is handled over WiFi. With that in mind, the proposed system consists of a consumer-grade web camera (with a 5 MP sensor), a Raspberry Pi 3 Model B+[5] and the hardware necessary for running the conveyor (a stepper motor and the conveyor belt itself). The total cost of the project [13] is reported to be approximately 10000 rupees (converts to 110 euros at the time of writing), where the conveyor belt and stepper motor make up the majority of the cost.

The Raspberry Pi [13], with the use of multithreading, is in charge of image capturing and processing, controlling the conveyor belt as well as transmitting data to the internet via an Message Queuing Telemetry Transport (MQTT)[6] message broker (server). MQTT is a widely used messaging protocol within networks of IoT devices due to its lightweight publish/subscribe messaging transport model, allowing for small client size and low network bandwidth. A simplistic Graphical User Interface (GUI) is used for operating and monitoring the system [13]. The GUI provides a real-time overview of the Raspberry Pi's temperature, the number of objects detected as well as the speed of the conveyor (in Revolutions Per Minute (RPM)). Additionally, the user interface provides buttons for changing the direction of the conveyor rotation and halting it. A separate GUI is said to be used for calibrating the colour profile and size criteria of the detection algorithms used.

Both the Raspberry Pi and the main GUI are subscribed to relevant topics available on the MQTT broker, waiting for messages to be published, while also acting as publishers themselves when it is necessary to communicate with the other [13]. This procedure is visualised in Figure 2. As multiple clients can be subscribed to and publish to the same topics on the MQTT broker at the same time, it therefore allows multiple users to use local copies of the GUI at the same time as well, irrespective of their location [13].

[5] https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/
[6] https://mqtt.org/

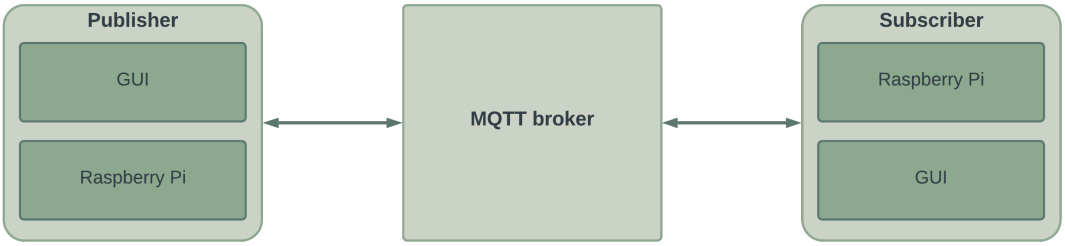

Figure 2. MQTT Communication [13].

The speed of the conveyor monitoring system's [13] image processing is reported to be 10–12 FPS with the resolution of input images set to 640 by 480 pixels. It is noted that the speed could be increased by lowering the resolution to 320 by 240 pixels, however, using a lower quality image would also increase the chance of false detections. The system performance is additionally limited by the conveyor rotation speed, as with a speed higher than 20–22 RPM the image is said to be too blurry for accurate object detection.

Observations. While the systems described above [12, 13] differ drastically in the number of components used, they were developed to serve a common purpose and with that, they also have many similarities. For example, both systems are controlled via a GUI for day-to-day operation and monitoring. This is essential for retrieving results from the system and intervening when necessary. In both cases, a central control unit is also required for routing and managing the communications between different system components. While the first system [12] utilises dedicated computers (vision modules) for capturing and processing images of fruit, in the latter [13] all of this is also handled by the central control unit, the Raspberry Pi. The systems additionally differ in the technologies used for communication between components, where one transmits its messages over CAN or LAN, depending on the latency requirements of a message type, while the other uses the lightweight MQTT protocol instead. Performance-wise, the two sorting systems are not directly comparable due to their differences in methods used for detecting and classifying objects. It is clear, however, that while the fruit sorting system was built with flexibility in mind, it is still very tightly integrated into the conveyor belt system when compared to the IoT based system. While the latter also has controls for operating the conveyor belt in real-time, it does not appear to be a requirement of the detection algorithm but rather a functionality for improving usability.

2.3 Advancements on existing solutions

Based on the above research and advancements in technology since their publishing, a new and improved system for real-time monitoring and virtual sorting of vegetables is designed and developed. Previous works have focused on limited areas of such a system and do not provide a suitable all-in-one solution. Additionally, existing commercial products aimed at large agribusinesses are generally too complex and thereby expensive for smaller producers to acquire. The purpose of the new system is to provide a cost-effective alternative to small agribusinesses for gathering statistics on the quality of their produce with the use of deep convolutional neural networks.

The concept of using an object detection model for classifying vegetables is applied for high throughput of produce and easier adaptability to existing conveyor systems as the objects under detection can appear anywhere in the frame and do not have to be singulated. This is an improvement over solutions where a limited number of objects can be classified at a time [5, 12]. Additionally, assuming that the objects under evaluation are constantly rolling on the conveyor line (applicable for round vegetables and fruit, for example) multiple angles of the objects can be captured for creating a detailed training dataset and making objective classification decisions. While this method has previously been used for classifying objects [12], capturing different angles of objects for training a classification or object detection model has been carried out manually [5, 6]. Capturing training data in an active work environment, therefore, reduces the time spent on creating a suitable dataset.

Cost efficiency is achieved by using readily available consumer-grade technology such as a colour web camera and a Raspberry Pi 4 as opposed to specialised hardware configurations by today's standard [12]. Statistics generated by the system are made available to the relevant users within a user-friendly web application, where the system can be controlled and monitored anytime and anywhere. Previous implementations of control panels for sorting systems have either been accessible only locally on-premise [12] or do not provide detailed statistics of the detections retrieved [13]. The new system additionally provides an added feature that enables users to choose from a variety of Evaluation Modules stored in the database and deploy them on their Edge Devices based on their needs at the time, whereas with the reviewed solutions [12, 13], reconfiguring the detection algorithm must be done manually. An in-depth overview of the system is given in Section 3.

3 Vegeval

The proposed system, Vegeval, for visually evaluating the quality of vegetables on a conveyor line, is designed to be used as an assisting tool for agribusinesses of small to medium size looking to integrate precision agriculture processes into their workflow. Vegeval consists of two subsystems, an on-premise Edge Device for gathering vegetable quality statistics in real-time and a web application in the role of a control and analytics panel, as can be seen in Figure 3. While the figures related to the system architecture present a single Edge Device, the system is in fact designed so that in theory, an unlimited number of Edge Devices can be added to the system. The output and main value of the system are the statistical data generated by the Evaluation Modules deployed on Edge Devices, giving an insight into the quality of produce and with that allowing business owners to make more accurate decisions. The name "Vegeval" is a derivation of the words "vegetable" and "evaluation". All of the source code files related to the project are available in a public GitHub repository [14]. The writing assistant software Grammarly[7] was used to prevent typographical errors in the writing of this thesis.

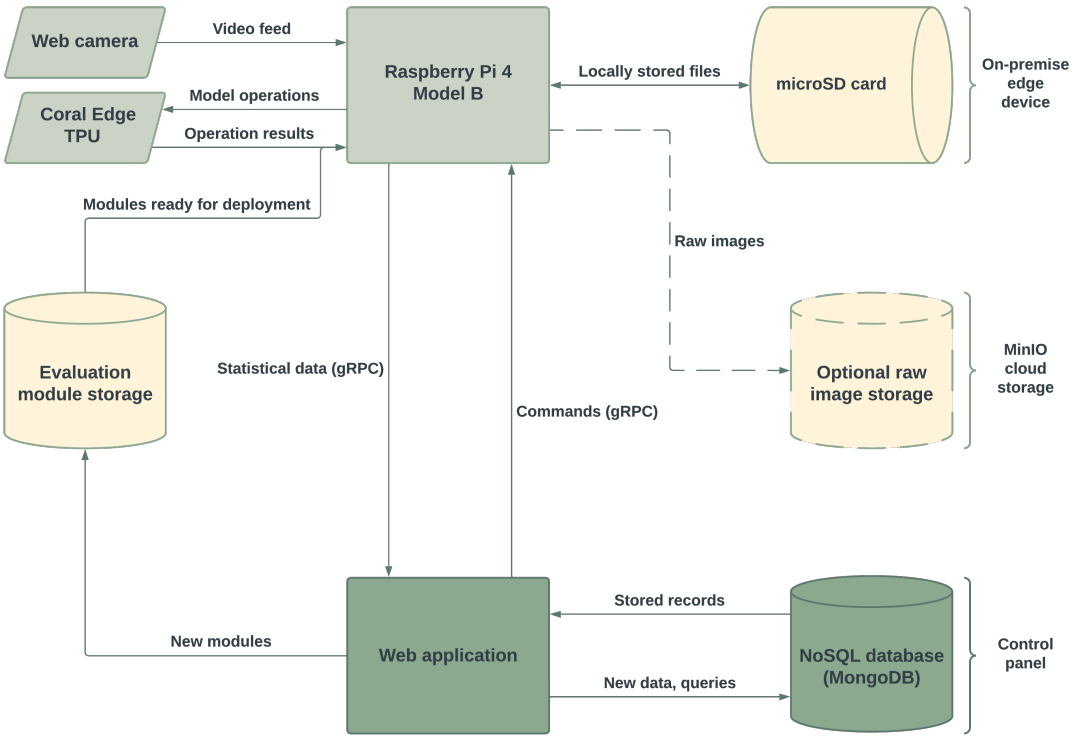

Figure 3. High level architecture of the Vegeval system.

[7] https://www.grammarly.com/

The following subsections describe the purpose as well as the core functionality of the before-mentioned subsystems and the supporting cloud storage layer, including how they operate separately but also in conjunction with each other to generate statistical quality reports of the vegetables assessed and make the reports available to the end user.

3.1 On-premise Edge Device

In order to evaluate the visual quality of vegetables in real-time, one or more dedicated physical devices are installed at a location where vegetables are normally processed (e.g., sorting station at a warehouse facility). The on-premise Edge Device, a subsystem of Vegeval, captures images from a web camera feed in real-time, applies an object detection model on the video frames, and based on the detections retrieved from the model, generates statistical data that is sent to the web application for analytical purposes. Communication with the web application is handled over gRPC[8], an open source Remote Procedure Call (RPC) framework, for fast and lightweight messaging. New devices are provisioned and configured for system users by administrators.

The Edge Device subsystem comprises a consumer-grade web camera Logitech Streamcam[9], a Raspberry Pi 4 Model B[10] with 4 GB of RAM, a Coral USB Accelerator[11], and a Samsung Evo Plus 32 GB microSD card[12], as shown in Figure 4. The Coral USB Accelerator adds an Edge Tensor Processing Unit (TPU) to the subsystem, which is used for running machine learning operations and shifting the processing load away from the Raspberry Pi's CPU. This is required for increased throughput of video frames within a period of time when compared to object detection models running on solely the CPU. It is important to note, however, that depending on the specific model's design and its compatibility with an Edge TPU, not all operations can be run on the Edge TPU and such operations need to utilise the computing resource provided by the CPU.

[8] https://grpc.io/

[9] https://www.logitech.com/en-us/products/webcams/streamcam.html

[10] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[11] https://coral.ai/products/accelerator/

[12] https://www.samsung.com/uk/memory-storage/memory-card/evo-plus-microsd-card-32gb-mb-mc32ga-eu/

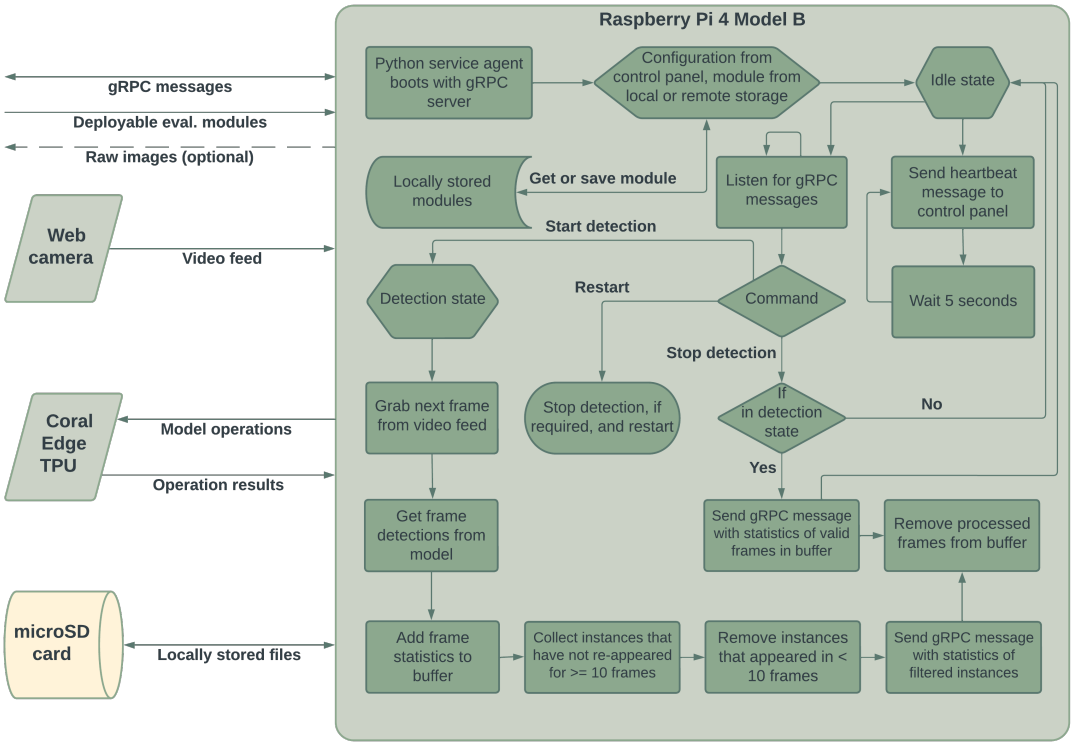

Figure 4. Architecture of the on-premise Edge Device.

Object detection models used for vegetable quality evaluation originate from a dedicated S3 cloud object store based on the Edge Device configuration provided by the system's web application. More specifically, the object detection models are integrated into Python Evaluation Modules, which include the supporting functions for object detection models to provide statistics in the universal format accepted by the web application. Additionally, if the Edge Device's assigned owner (system user) has opted for the device to save the captured raw images for later use, the images are automatically uploaded to a dedicated S3 storage bucket. All the necessary credentials for accessing external resources are configured for new devices by a system administrator. The cloud storage functionality is described in further detail in Section 3.2.

3.1.1 Working principle of the Edge Device

Algorithmically, the Edge Device's Python service agent[13] by default boots to an idle state, preceding which, an updated configuration is requested from the system's web application. Based on the configuration received, an Evaluation Module is retrieved from

[13] `https://github.com/35grain/vegeval/tree/master/edge-agent`

the S3 module store, if no local copy is present, and the object detection model within the module is loaded into memory for quick access at a later stage. While in the idle state, the service agent sends a heartbeat message to the web application every five seconds, letting the system know that the Edge Device is online and ready to accept commands. When a command for starting object detection is received, the service agent switches to the detection state, where input frames from the web camera are processed by the object detection model and the results of which are used to keep track of the detected objects and send relevant statistics to the web application over gRPC. In the detection state, heartbeat messages are only sent when no statistics report has been sent for more than five seconds. When a command for stopping object detection is received, the service agent sends the remaining statistics to the web application and returns to the idle state. A restart command first similarly switches the program to an idle state, if required, and issues a reboot command for the Raspberry Pi so that a new configuration for the Edge Device can be loaded or simply for troubleshooting an issue. The described processes run in parallel with the use of multithreading.

The statistics reports generated in the example Evaluation Modules include amongst others the final class name for each object that has passed the conveyor and is considered a valid result (the object appeared in at least 10 frames). The final class name is retrieved by selecting the class into which an object was on average classified the most, with appropriate weight multipliers applied to the classes. The weight multiplier is required in quality control to ensure that when one side of the object is considered low quality, but the other appears to be high quality, a low quality object must not be classified as high quality.

3.2 Cloud storage

Versions of deployable vegetable Evaluation Modules are kept in a dedicated S3-compatible MinIO[14] cloud object store bucket, readable by system users and their devices, and modifiable by administrators and the web application. Optionally, the raw images captured by the Edge Devices can also be uploaded to a cloud storage space divided into private sections based on system users and their devices. This means that one user (if not an administrator) cannot access another user's data and one device does not have access to another device's storage bucket. The web application only has privileges for uploading new versions of Evaluation Modules and creating new storage buckets for new device registrations. All existing storage buckets must be deleted manually by an administrator, if and when required.

Any communication with the MinIO server is handled over Hypertext Transfer Protocol (HTTP) with Transport Layer Security (TLS) protocol encryption using the

[14] `https://min.io/`

MinIO Software Development Kit (SDK)[15] for Python and JavaScript. The object store can be accessed either with a username and password combination, which is provided for each system user by an administrator, or an Access Key and Secret Key pair for programmatic access from the web application and Edge Devices, for the latter of which a unique combination is generated for each device. Issuing separate credentials for each user and device ensures that if one of the credential combinations were to be compromised, it can simply be revoked and the rest of the system is not affected.

3.3 Web application

The web application acts as a control and analytics panel for operating and monitoring statistics of the Vegeval system. Functionally, the web application provides registered users with a list of available Evaluation Modules, their Edge Devices and the devices' configurations, and statistics collected by the Edge Devices presented as graphs. Users with the administrator role can additionally register new users, Evaluation Modules and Edge Devices. System users are authenticated against the system by local authentication (username and password combination), while Edge Devices are authorised access to the web application based on an Access Key and Secret Key combination (separate from the MinIO key pair). All authenticated users can change their password at any time (after their first login, for example). Public registration for the application is not available. A detailed description of the web application architecture is given in Section 4.

[15] `https://min.io/docs/minio/linux/developers/minio-drivers.html`

4 Web application

On a high level, the web application is divided into a presentation layer (frontend)[16] and a data access layer (backend)[17], as shown in Figure 5. All web application data is stored within a NoSQL (or key-value based) MongoDB[18] database for quick retrieval of stored records. The following is a detailed description of the inner workings of the web application. Precisely, the technologies used for authenticating users and Edge Devices are given and the tools used for developing the frontend and backend applications are presented in regard to the web application functionality as a whole.

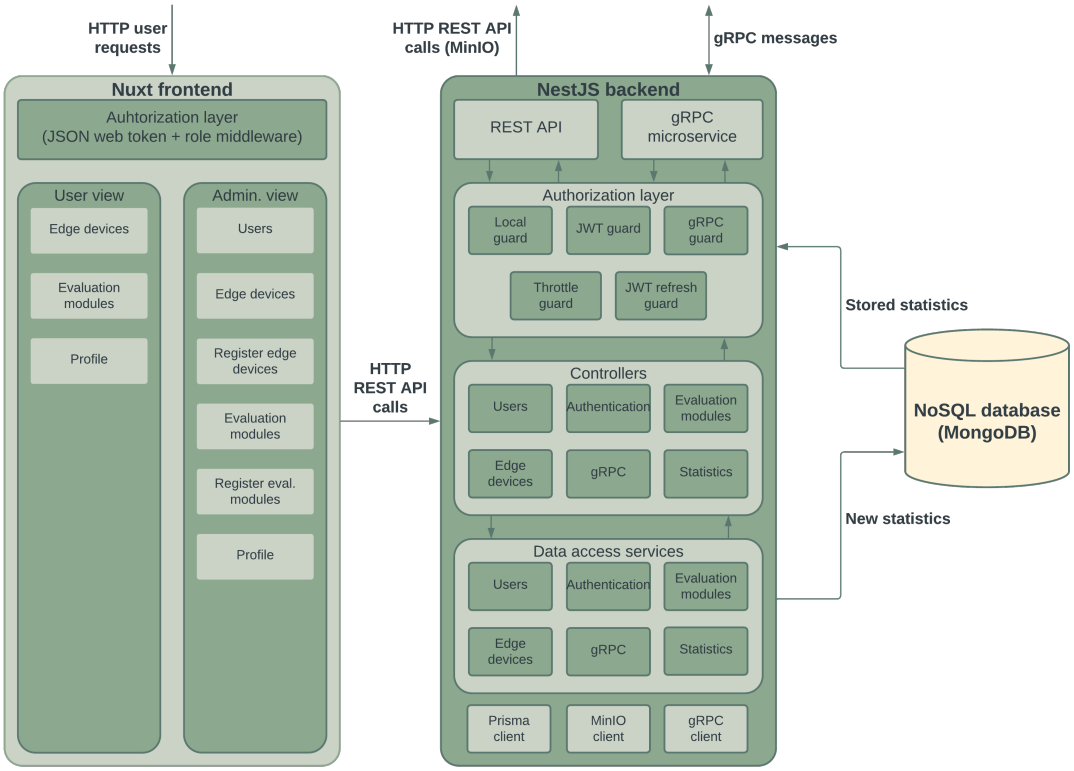

Figure 5. Architecture of the web application.

4.1 Authentication

For authenticating users and their Edge Devices, as well as authorising them access to otherwise private system resources, an authentication layer is put in place on both the

[16] https://github.com/35grain/vegeval/tree/master/frontend
[17] https://github.com/35grain/vegeval/tree/master/backend
[18] https://www.mongodb.com/

frontend and backend of the system. All passwords and Secret Keys that are used for authenticating system users or devices are stored in the web application's database in a hashed format to mitigate the risk of user credentials being compromised in case of a data breach. The credentials are hashed and verified with a random salt using the bcrypt library[19] for NodeJS. On the frontend, an authentication module nuxt-auth[20] acts as a middleman for forwarding local authentication requests from users to the backend and retrieving the JSON Web Token (JWT) for convenient authorisation purposes in future requests.

A JWT consists of three parts: a header describing the token type (in this case it is always JWT) and the algorithm used for signing the token, the payload with data claims about the authenticated user (in this case, the user's ID, email and role), and the signature for verifying that the message has not changed along the way. Precisely two JWT are received upon successful local authentication against the backend server. The first JWT acts as an Access Token that is stored within the nuxt-auth module's session cookie (which also uses the JWT format). The Access Token is used for accessing the backend resources and services as an authorisation method and expires after 15 minutes of issuing as a countermeasure for session hijacking attempts. Based on the role of a user stored in the JWT Access Token, the user is either allowed or denied access to the administrative views of the web application and related backend services. When a user logs out of the web application, all session cookies and stored tokens are destroyed.

Due to the short lifetime of the Access Token, it must be renewed for a smooth user experience. This is the core purpose of the second JWT, the Refresh Token. This JWT is hashed and stored in the web application's database as an alternative to local authentication. The structure of the Refresh Token is identical to the Access Token, however, the Refresh Token is valid for up to 24 hours after issuing. This means that whenever the Access Token has expired but the Refresh Token is still valid, the Refresh Token is used for authenticating against the backend server and issuing a new pair of access and Refresh Tokens. In order to mitigate attempts of stealing the Refresh Token, it is stored within the frontend application's server-side session and all requests for refreshing the tokens are sent to the backend server from the nuxt-auth module. The flow of exchanging the tokens is visualised in Figure 6. Should the Refresh Token also expire, the user is required to log in with their username and password.

[19] `https://www.npmjs.com/package/bcrypt`
[20] `https://github.com/sidebase/nuxt-auth`

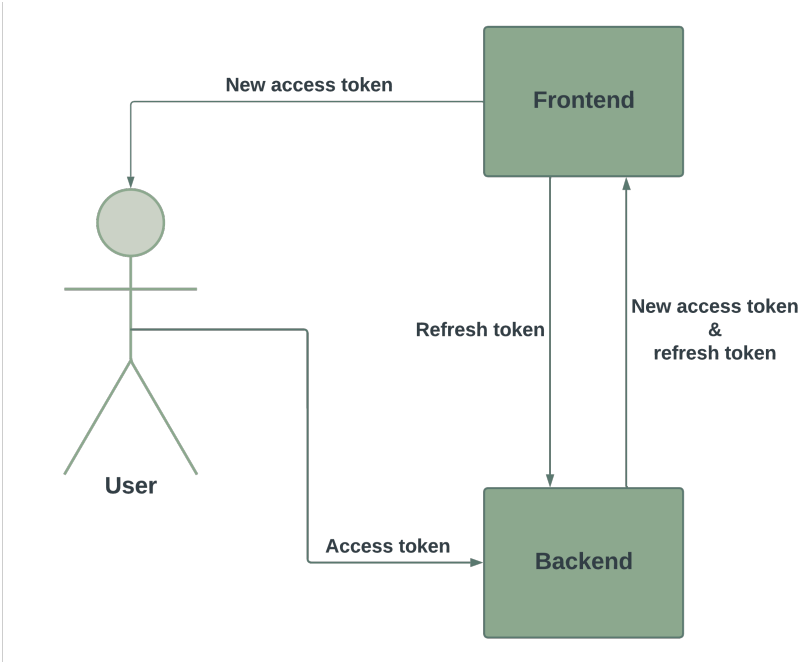

Figure 6. Flow of JWT Access and Refresh Tokens.

Edge Devices are granted or denied access to the web application services based on an Access Key and Secret Key combination. The key pair is generated automatically when a new device is registered and the Secret Key is displayed only once to the system administrator in plaintext for proper configuration of the Edge Device and is stored in the database in a hashed format. The Access Key is stored in the database as plaintext and acts as an alternative identifier of the device. Incoming gRPC requests to the Edge Device itself are accepted or rejected based on the web application's TLS certificate verification.

4.2 Frontend

The presentation layer of the web application is built using a Vue.js[21] based JavaScript framework Nuxt[22]. As the framework provides a component-based programming model for building user interfaces, the Vegeval web application's functionality is presented as different pages (views) and components. Large components such as the navigation bar or login form are built using and have been inspired by individual components such as buttons or input fields from the DaisyUI Tailwind Cascading Style Sheets (CSS)

[21] https://vuejs.org/
[22] https://nuxt.com

components library plugin[23]. Components and regular Hypertext Markup Language (HTML) elements also make use of the wide range of utility classes of the CSS framework Tailwind CSS [24]. The use of such frameworks allows to build robust applications in less time when compared to developing and testing the underlying logic from scratch. The before-mentioned frameworks were chosen for this project as they are widely adopted and therefore thoroughly tested modern tools, which also fulfil the requirements of the web application.

Depending on a system user's role ("user" or "admin"), some of the web application pages are not accessible to all users due to insufficient privileges or will only display data available in the scope of the user role. Unauthenticated (guest) users have access to the default Landing page and the Login page. The pages accessible to all authenticated system users are the following: User profile page, User edge devices page, Evaluation modules page and Analytics page. Users with the administrator role are additionally provided with an overview of all system users, all registered Edge Devices and the functionality to register new Edge Devices and Evaluation Modules to the system. The functionality offered by each page is described in the corresponding subsections below. In order to easily distinguish the regular user pages from the administrative pages, a different colour scheme is used on the sidebar navigation menu, as can be seen in Figure 7.

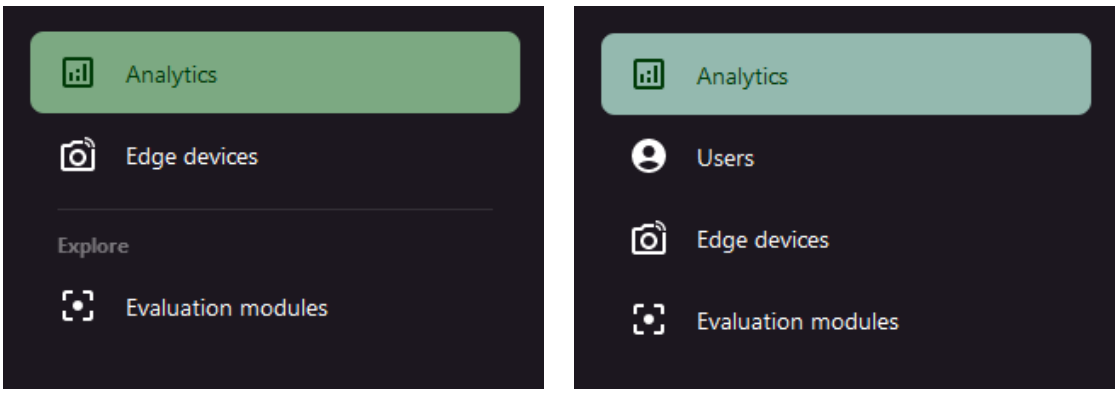

Figure 7. Sidebar on regular user pages (left) and sidebar on administrative pages (right).

[23] https://daisyui.com/
[24] https://tailwindcss.com/

4.2.1 Landing page

At the root index of the web application is the Landing page, which provides a minimal description of the application and points unauthenticated users toward the Login page with call-to-action buttons "Get Started" in the main body of the page and "Login" on the top navigation bar. In the background, there is a video of the output of one of the object detection models playing, as shown in Figure 8.

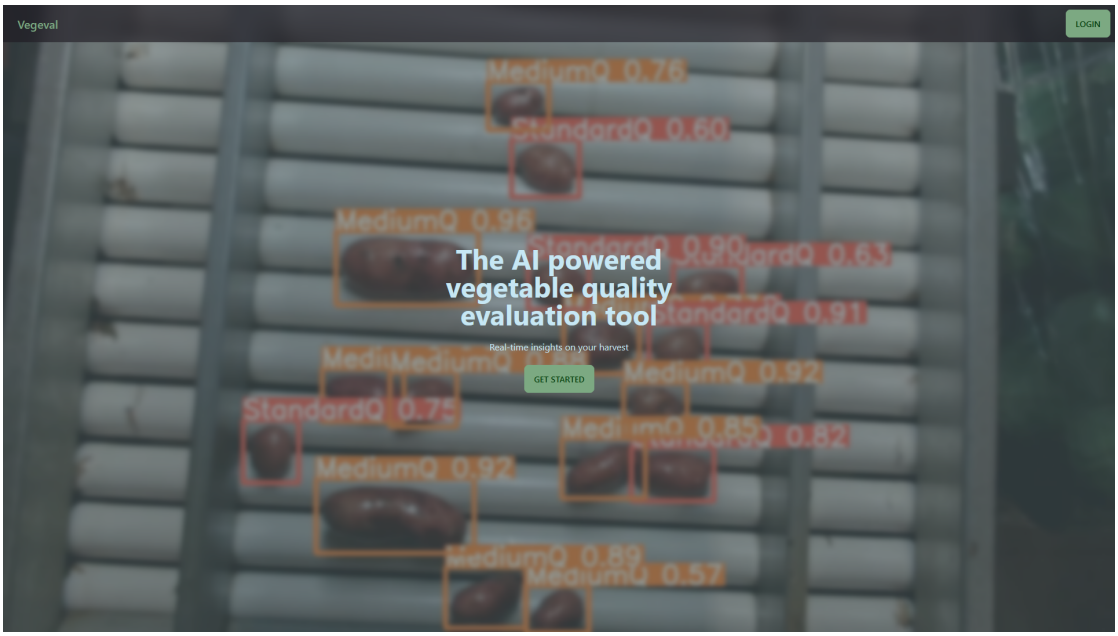

Figure 8. The default Landing page of the web application.

4.2.2 Login page

The Login page provides a simple form with input fields for a username (email address) and password along with a disabled "Register" button to indicate that public user registration is unavailable, and a "Login" button, as shown in Figure 9. Upon successful authentication, the user is redirected to the Analytics page.

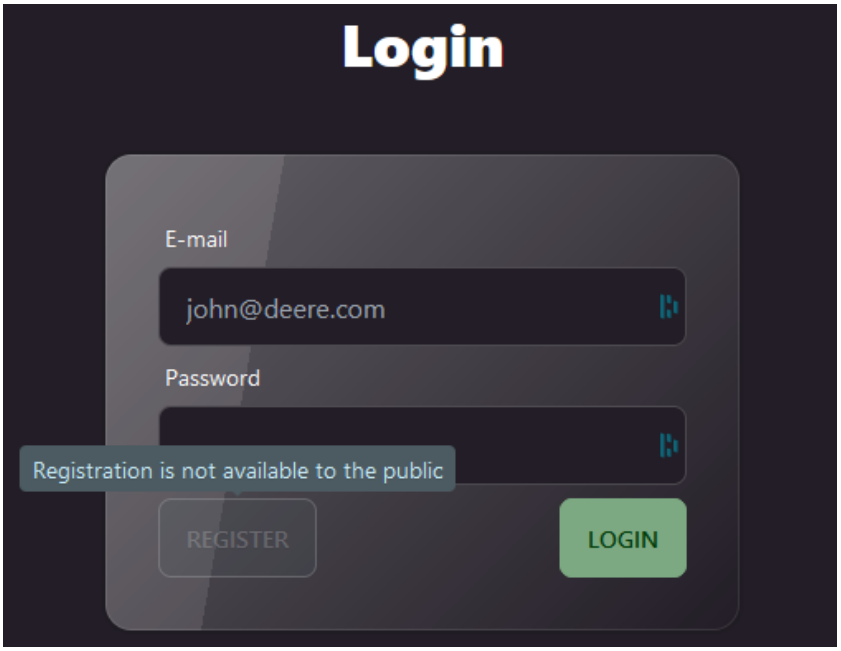

Figure 9. The Login page with relevant input fields and buttons.

4.2.3 Analytics page

General system statistics and statistics generated by the Edge Devices are visualised and displayed on the Analytics page of the web application and are updated automatically every 10 seconds. General system statistics displayed to administrators can include, for example, the number of statistical records collected in total, the total number of users and the total number of Edge Devices registered to the system. Administrators are additionally given an overview of the distribution of Evaluation Modules deployed on Edge Devices as well as the number of statistics collected by each module in the form of a graph. The graphs used for visualising data are generated using the Chart.js[25] JavaScript library.

Users without administrative privileges are provided with an overview of the collected statistics in the user's scope, meaning that only data related to their devices can be seen. Such statistics can include but are not limited to the observed class distribution of each Evaluation Module with existing statistical records, the number of frames an object was tracked for and the number of statistics collected by each device, as shown in Figure 10.

[25] https://www.chartjs.org/

Figure 10. The Analytics page displayed to regular users.

4.2.4 Users page

The full list of system users can be read from the Users page, accessible to administrators only. The page also offers the functionality for registering new users to the system via a modal window with a syntactically valid email address and password combination, as can be seen in Figure 11. The user password is required to be at least 16 characters long as a precautionary measure against attempts of guessing a user's password. If the user has opted to upload the raw images collected by their Edge Devices to the MinIO cloud storage, a MinIO user account must be created manually by a system administrator due to the technical limitations of the system.

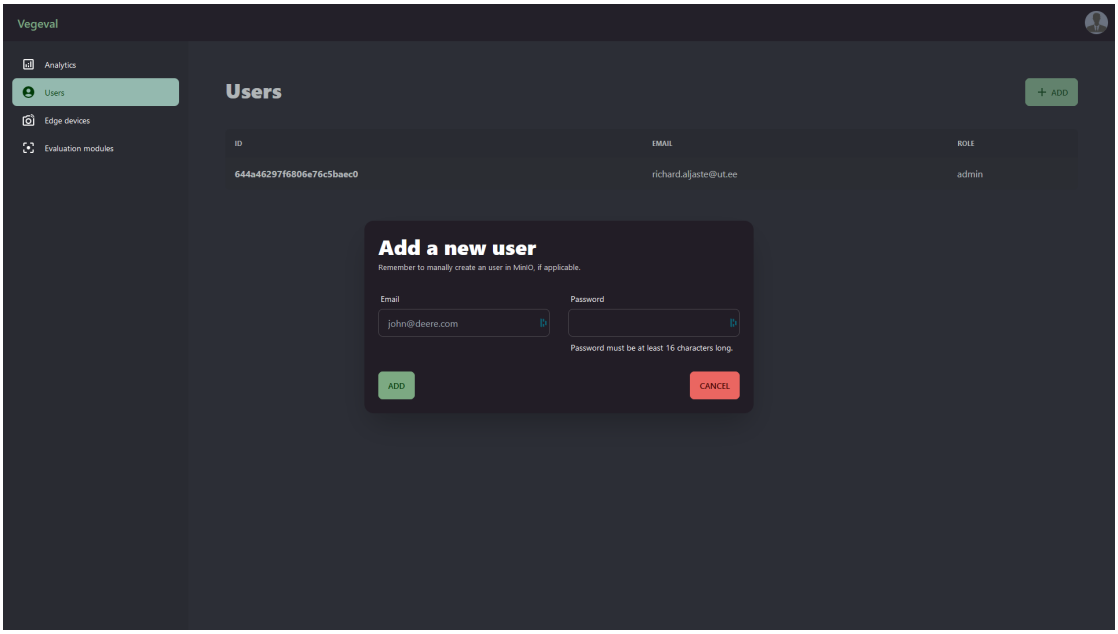

Figure 11. The Users page with registration modal open in the foreground.

4.2.5 User profile page

The User profile page is populated with user-specific information, which currently is limited to the user's email address. The user can change their email address and password of their own account from the Profile page, as shown in Figure 12. The same requirements apply to the email and password values as in the user registration procedure and the existing valid password must be provided in order to make changes to the account.

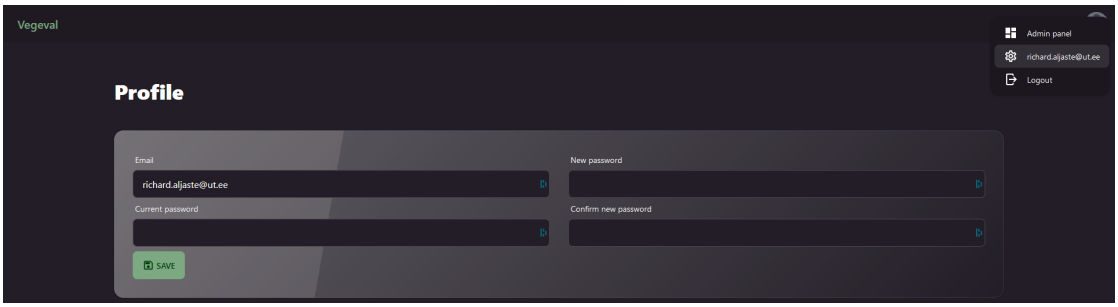

Figure 12. The User profile page.

The profile page can be accessed from the dropdown menu that is toggled when clicking on the user avatar circle on the right-hand side of the top navigation bar. This dropdown menu also contains navigational buttons to the Administrator panel (the button

is only displayed to users with the "admin" role) and the Logout route (redirects to the Login page once logged out).

4.2.6 Edge devices page

The Edge devices page displays a table of Edge Devices registered under the user or all Edge Devices registered to the system in the case of users with administrative privileges. Each row in the table represents a single device, for which its human-readable identificatory label, current status (either "Offline", "Idle" or "Detecting"), deployed Evaluation Module name, Access Key, Internet Protocol (IP) address and action buttons are shown. Depending on the current state of a device, the action buttons allow the user to start and stop the object detection model or restart the device. The table contents are updated every 10 seconds to reflect the status of the Edge Devices, however, a separate button for manually refreshing the table is also provided. In the case of administrator users, for each device in the table, the user's email address is displayed under whom the device is registered. Administrators also have the ability to register new Edge Devices to the system via a modal window, as shown in Figure 13.

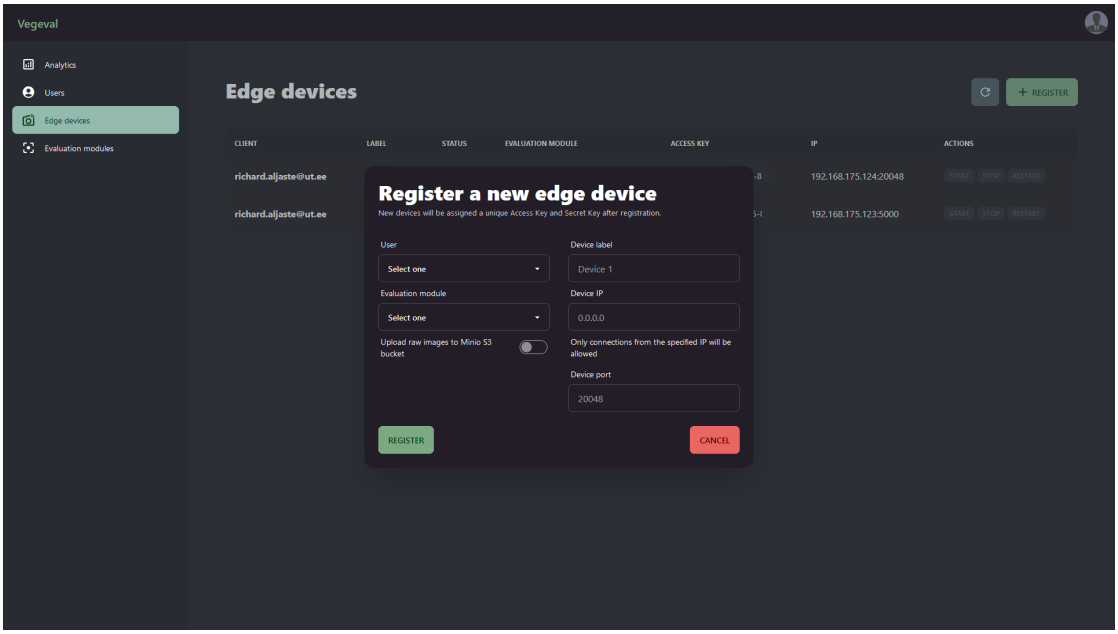

Figure 13. Table of Edge Devices with the device registration modal open in the foreground.

Upon successful registration of a new device, a new S3 storage bucket is automatically created if the toggle for uploading raw images was turned on. The necessary connection details to be used on the Edge Device for communicating with the rest of the system are

displayed within a modal window, as can be seen in Figure 14. This includes the created MinIO storage bucket name as well as the Access Key and Secret Key for authenticating against the web application. Once the modal has been closed, the plaintext version of the Secret Key can no longer be retrieved.

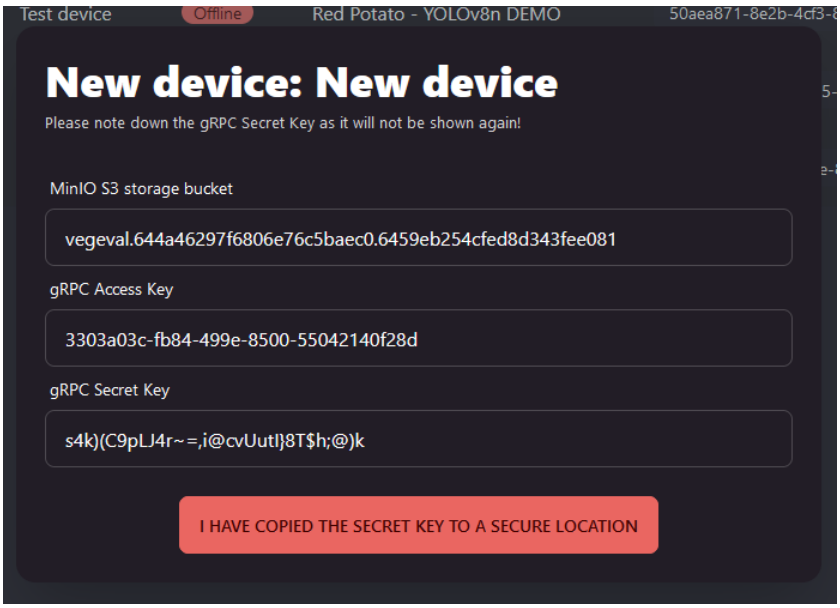

Figure 14. The connection details displayed on successful Edge Device registration.

4.2.7 Evaluation modules page

A full list of deployable modules is displayed on the Evaluation modules page, including the name, version and date of publishing for each module. The system administrators additionally can see the object name in the MinIO S3 modules storage bucket for each module as well as have access to the module registration functionality via a modal window for publishing new modules to the system, as can be seen in Figure 15. In order to register a new module, the module's descriptive name, version and a ZIP file archive containing the module files must be provided. The file archive is automatically uploaded to the S3 storage bucket for access by Edge Devices.

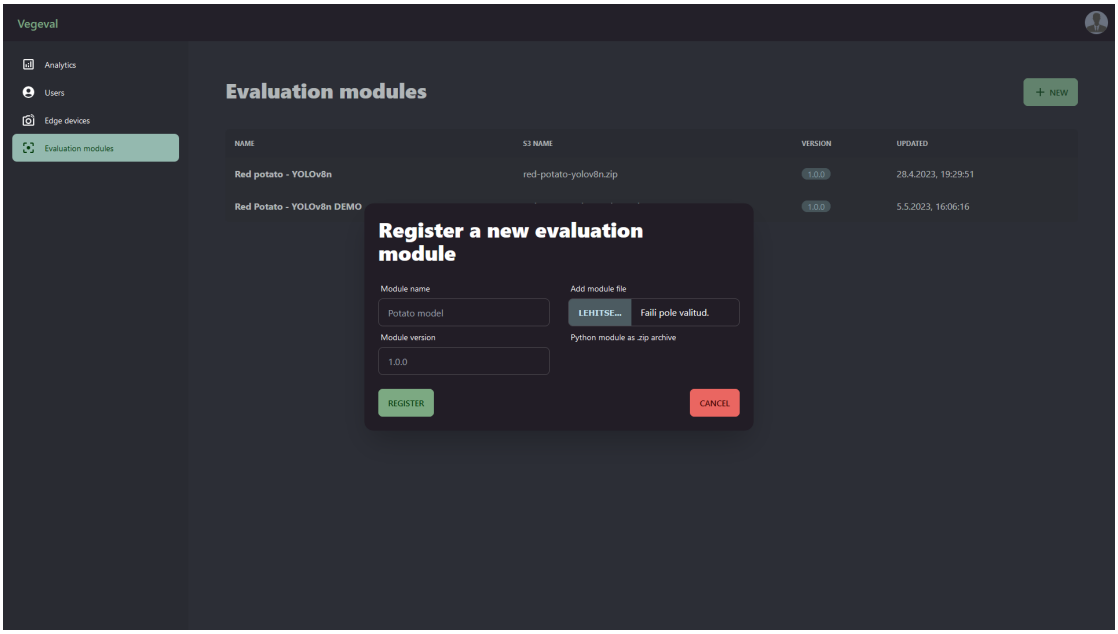

Figure 15. List of Evaluation Modules with the module registration modal open in the foreground.

4.3 Backend

For building the data access layer of the web application, the NodeJS (an open-source JavaScript runtime environment)[26] framework NestJS[27] was used. NestJS was selected for the backend development for its great customisability while providing detailed documentation for common use cases. Additionally, having the frontend and backend both written in the same language (JavaScript), makes it easier to navigate in code as opposed to needing to switch back and forth between two or more languages. While the majority of backend services are served from a REST API, in order to accept messages from the Edge Devices, a gRPC server is also started together with the NestJS application as a microservice.

4.3.1 Authentication guards

Both the Representational State Transfer (REST) Application Programming Interface (API) and gRPC server make use of an authentication layer (or a number of "guards") to prevent unauthorised access to system services, as seen in Figure 5. The "JWT guard" and "JWT refresh guard" are in charge of JWT based authentication and authorization as described previously. The "Local guard" verifies authentication requests using a

[26] https://nodejs.org/
[27] https://nestjs.com/

username and password combination, while the "gRPC guard" authorises Edge Devices access to system resources based on the Access and Secret key pair, and the "Throttle guard" limits the number of requests allowed per client within a defined period of time.

4.3.2 Data transactions

For efficiently designing the database structure and querying the database in a type-safe manner, the Prisma Object Relational Mapping (ORM) software[28] for Node.js and Typescript[29] is used for accessing data from the service classes. For interacting with the MinIO S3 storage server, a dedicated MinIO SDK client is called. Outgoing gRPC messages are also sent out via a separate gRPC client.

All backend services are implemented using controllers as the endpoint for routing authorised requests to the respective data access services and returning or updating relevant data. The Users controller and service are in charge of serving all system users' related data and registering or updating users. As the name suggests, the Authentication controller and service ensure that only authorised users can log into the system, are safely logged out of the system and are issued new Access Tokens. The Edge devices controller and service enable the functionality for registering new and retrieving existing Edge Devices as well as routing device-related commands to the gRPC client. Requests for registering new Evaluation Modules or retrieving existing modules are handled by the Evaluation modules controller and service. The gRPC controller accepts all incoming gRPC messages for processing by the Edge devices service or Statistics service. All outgoing gRPC requests are routed through the gRPC service, however, which forwards the final requests to the gRPC client. Finally, the Statistics controller and service gather various system statistics and data collected by edge devices into a suitable format for use on the frontend, while also taking requests for adding new statistics to the database.

[28] `https://www.prisma.io/`
[29] `https://www.typescriptlang.org/`

5 Vegetable detection models

In this thesis, the visual quality of a vegetable is considered as the difference or compliance of the visual appearance of the vegetable with a standard. This is evaluated by object detection models trained on a custom dataset of a specific vegetable type. Next, a description of the dataset used to train the proof of concept models and the necessary steps taken prior to training are given. *Red Potatoes* were chosen as the example vegetables under quality evaluation mainly due to their great availability and the opportunity to access and use specialised industrial equipment set up for sorting potatoes in a real work environment. The latter is consequently useful for improving the usability of the designed system for its target users.

5.1 Dataset

The raw dataset comprises 597 images of red potatoes on a conveyor, captured from a web camera feed with a resolution of 1280 by 720 pixels, where individual instances of potatoes are on average approximately 75 pixels wide and 55 pixels tall. An example of a raw image is shown in Figure 16.

Figure 16. A raw image of potatoes on the conveyor.

Both the potatoes and the industrial facilities (conveyor line, sorting station etc. shown in Figure 17, excluding the hardware used to capture the images) required for visually evaluating the potatoes were provided by the agribusiness Saaresepa OÜ located

in Pärnu County, Estonia. While the results of this thesis may be integrated into the workflow of the company in the future, it is important to note that the research and development were conducted independently and without bias towards any particular business agenda or outcome.

Figure 17. Vegetable sorting station at Saaresepa OÜ's produce processing facility.

The dataset collected in this thesis will not be made publicly available as this has yet to be agreed upon with Saaresepa OÜ at the time of writing, however, it can be provided per request on an on-demand basis and upon approval of all related parties.

5.1.1 Data collection

For collecting images of potatoes on the conveyor line, an image capturing system consisting of the Logitech Streamcam web camera, the Raspberry Pi 4 Model B with 4 GB of RAM and a USB flash drive Sony USM32GXB[30] with 32 GB of storage was used. The webcam was mounted on the aluminium ceiling of the sorting station to make

[30]Product has been discontinued.

optimal use of the integrated fluorescent light as well as a suitable view angle, and the Raspberry Pi 4 along with the flash drive were enclosed in a nearby electrical box, as shown in Figure 18. The latter was mainly a precautionary measure for protecting the devices against possible dust and moisture in the air.

Figure 18. Mounted webcam (left) and Raspberry Pi 4 with flash drive enclosed in the electrical box (right).

Internally on the Raspberry Pi 4, a custom Python script[31] was configured to run at system boot. The script uses multi-threading to simultaneously detect motion in the webcam feed and save frames to the USB flash drive when motion is detected. When no motion has been detected for at least 5 minutes, the script automatically issues a system command to shut down the Raspberry Pi. The system and algorithm structure can be seen in Figure 19. Although the Raspberry Pi 4 could process a larger number of frames within a given time period in terms of processing power, it was configured to save the images to the flash drive at 10 FPS, as this speed was deemed sufficient for capturing different angles of individual potatoes. Additionally, higher FPS appeared to have caused the process to eventually crash, as the write speed of the flash drive was limited and the system memory was gradually filling up. This custom image capturing system was used instead of a commercial digital camera, for example, to allow for finer control over the timing of frame capture and to keep the number of environmental variables, such as the camera used, in the lifecycle[32] of the object detection models at a minimum.

[31] `https://github.com/35grain/vegeval/tree/master/data-collection`

[32] A machine learning model's lifecycle commonly consists of six main stages: data collection, data preparation and analysis, model training, model testing, model deployment and model monitoring.

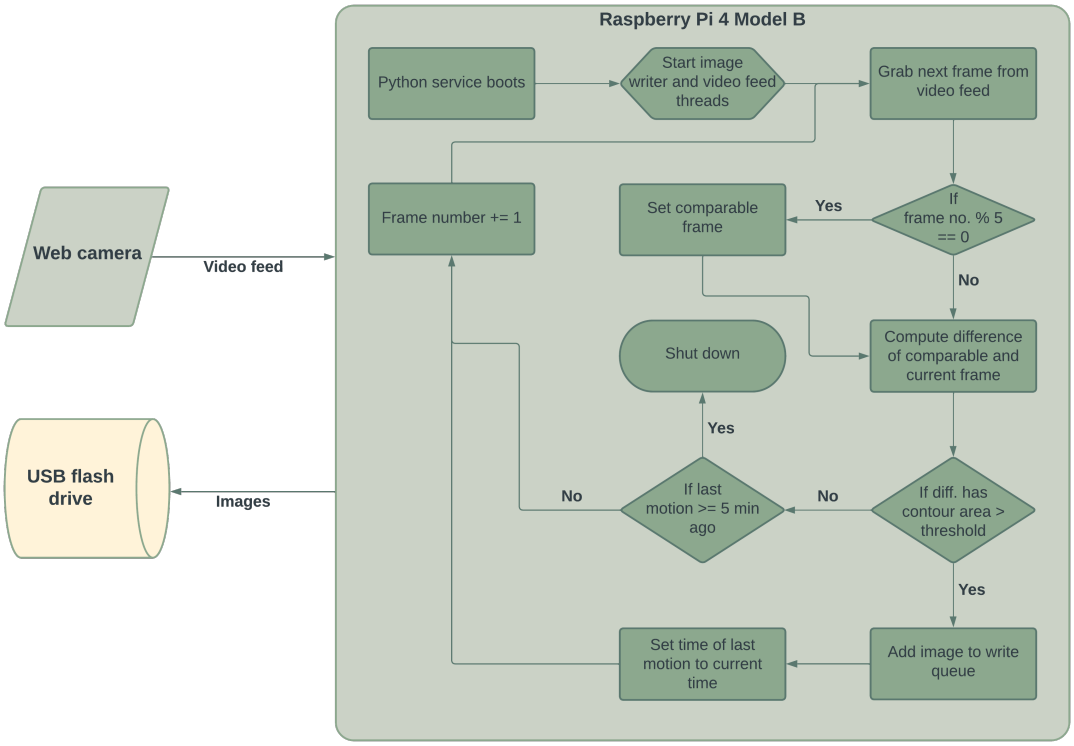

Figure 19. Structure of the image capturing system and algorithm.

5.1.2 Data labelling

Data labelling (or annotation) is a process where individual unlabeled data points (text, audio clips, images or other classifiable object types) in a candidate training dataset are assigned a label (or class). In the case of an object detection or segmentation dataset, the bounding box coordinates of the object of interest are also attached to the label. This allows machine learning models to understand and learn into which class of a predefined set of classes an object belongs to and what features identify it. In this case, instances of potatoes in the captured images were categorised into three classes based on their visual appearance ("Q" in the class names stands for Quality):

- "StandardQ" (uniform red colour, without mechanical injuries, sparsely distributed small blemishes are allowed),
- "MediumQ" (densely distributed small to medium-sized blemishes covering the minority of surface, slight discolouration allowed),
- "LowQ" (extensive discolouration, mechanically injured, majority of surface covered in medium to large-sized blemishes).

Initially, the class names were "Edible", "Visual defects" and "Inedible" respectively, however, the labels were renamed during the course of the project for clarity. An example of an instance from each class in the dataset is shown in Figure 20.

StandardQ	MediumQ	LowQ
		

Figure 20. Instances of each dataset class.

The dataset used in this thesis was labelled using an open source data labelling platform Label Studio Community Edition[33] hosted on a private virtual machine. Labelling was performed in a random sequence generated by the platform based on the uploaded dataset consisting of 597 images. The first 50 images were labelled completely manually, requiring the bounding boxes and classes to be assigned by hand as shown in Figure 21.

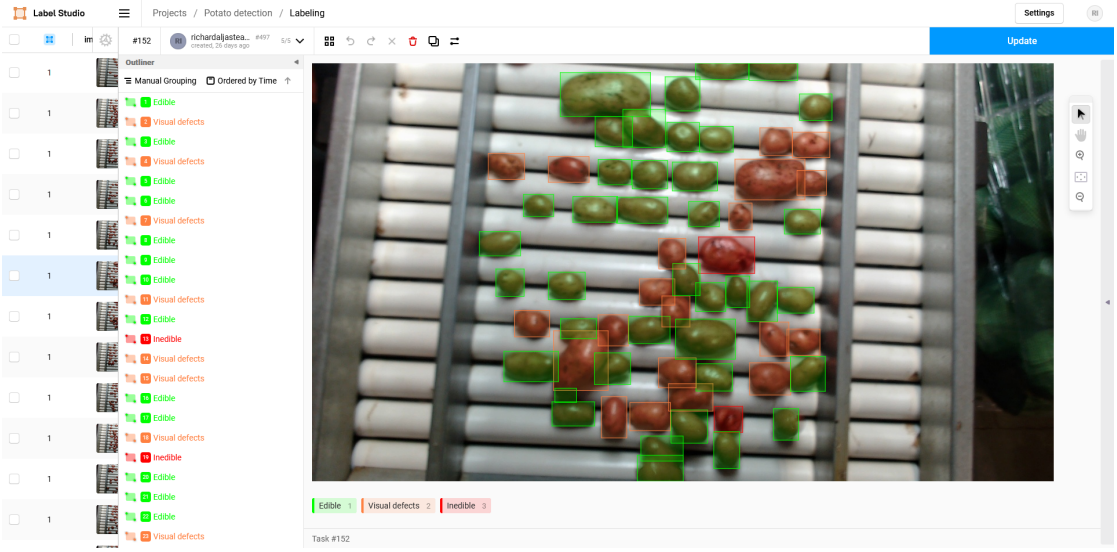

Figure 21. An image with labelled potatoes in Label Studio.

With 50 images labelled, an initial version of an object detection model was trained using transfer learning on the YOLOv5n object detection model [15] with default settings.

[33] https://labelstud.io/

The model training process is described in Section 5.2. The initial YOLOv5n model was used to set up a machine learning backend[34] for the Label Studio platform, which assists in pre-labelling the images by providing predictions of the objects' bounding boxes and classes. In most cases, this means that the human annotator has to refine the predictions where necessary and manual work is reduced as the model grows and becomes more accurate in its decisions. The object detection model used in the machine learning backend was thereafter re-trained with additional data every time a new batch of 100 images was labelled. The YOLOv5n model was replaced with the YOLOv8n pre-trained object detection model[35] at around 250 labelled images, however, the labelling and model training process remained the same besides technical changes in code. The change was made due to the improved output precision of YOLOv8n over YOLOv5n. Due to time constraints of the thesis, 501 images (or 18169 instances of potatoes) were labelled out of the total 597 images of the raw dataset.

5.2 Object detection models

In order to measure the capabilities of the Vegeval system and to find the best-performing object detection model for evaluating the visual quality of vegetables, a number of modern object detection models were trained and tested on the custom potatoes dataset, and the results are compared. The following introduces the compared models and describes the training and testing processes.

5.2.1 Model selection

The selection of object detection models under review was based on the requirement that the model should be able to run in real-time on devices with low computational resources while not losing a lot of performance in terms of detection precision. This, therefore, calls for a balance between inference speed and accuracy from the models.

At the time of writing, most of the popular general-use and rapidly developing models that fit the set criteria belong to the You Only Look Once (YOLO) model family. "You Only Look Once" stands for the implementation of predicting results in the single neural network of the model. As the name suggests, the neural network is passed only once to predict the bounding boxes and class probabilities of objects all at the same time. This is done by dividing the input image into a grid of cells, where for each cell the bounding boxes, object class probabilities, and confidence scores are predicted. Based on the confidence score of cells, the model indicates the likelihood of objects belonging to a bounding box. Class probabilities, however, show how certain the model is about an object belonging to a particular class. When compared to other model architectures,

[34] https://labelstud.io/guide/ml.html

[35] https://github.com/ultralytics/ultralytics

where classifiers need to run hundreds of times or implement a two-stage algorithm to first generate region proposals and then predict classes for these regions, the YOLO architecture is able to achieve higher speeds while remaining accurate [16]. This does not mean, however, that more accurate and faster models cannot exist, but the YOLO models are undeniably highly competitive in terms of balance between speed and accuracy.

The first YOLO model was published in 2016 by Joseph Redmon et al. [17] and has since been updated and revised in multiple versions, with the latest version, at the time of writing, being the YOLOv8 model. It is important to note that only some of the published model versions have common authors, while most do not. Four YOLO model versions published in the last three years (2020–23) are compared in this thesis, where from each version the mobile-oriented "nano" or "tiny" model was selected (with an input image resolution of 640 pixels on the longer side). The compared YOLO models are:

- YOLOv8n (640 px);
- YOLOv7-tiny (640 px) [18];
- YOLOv6-N (640 px) [19];
- YOLOv5n (640 px).

Another set of object detection models that have previously proven to be comparably fast and accurate on mobile devices belong to the EfficientDet-Lite model family developed by Google[36]. EfficientDet-Lite is a derivation from the EfficientDet architecture that has been considered to achieve high accuracy with limited computational resources [20]. EfficientDet-Lite models are based on a two-stage detection process that first generates a set of candidate object region proposals and then refines these proposals to improve accuracy. From the EfficientDet-Lite family, the EfficientDet-Lite2 model (with an input image resolution of 448 px on the longer side) was selected for comparison as it is the largest of the models that fit into the memory of a single Edge TPU used for measuring the model's inference speed on the Edge Device of the Vegeval system.

5.2.2 Training and testing

The labelled dataset was randomly divided into a training and validation set by 80% and 20% respectively. A separate test set was not used as this would have limited the number of instances available for training, which is not a desirable tradeoff with a small dataset. Due to the small size of the compared object detection models, the dataset images are automatically downscaled to the respective resolutions (640 px or 448 px on the longer side). The pre-trained object detection models under comparison were re-trained using default parameters with the exception of the number epochs (training

[36] `https://github.com/google/automl/tree/master/efficientdet`

iterations) set to 250 and batch size set to 32, as higher batch sizes gained accuracy at a slower rate and no significant improvements were observed after 250 iterations. Some of the models also implement early-stopping to automatically stop training when no performance improvements are observed for a set number of epochs (e.g., 100). The YOLO models additionally apply data augmentation techniques to the images on each iteration of training. All compared models were pre-trained on the COCO dataset.

The results from the epoch with the best metrics are used in the comparison of the models, where the industry standard Mean Average Precision (mAP) metric is used for comparing the precision of object detection models. mAP is an average across the Average Precision metric of all classes of a model, where the value ranges from 0 (all predictions are incorrect) to 1 (all predictions are correct). The mAP is calculated at an Intersection over Union (IoU) threshold of 0.5 (or 50%). IoU represents the required amount of overlap between the predicted bounding box of an object and the ground truth bounding box (the label). IoU is also often used as a standalone metric for measuring how well a model is able to locate objects within an image. It is calculated by finding the area where the bounding boxes overlap and diving it by the total area of both bounding boxes combined. The models were trained on Google Colaboratory[37], a hosted Jupyter Notebook[38] platform, using hardware acceleration from the NVIDIA A100-SXM4-40GB GPU. The precision of the models was evaluated on the validation set.

For deploying models on the Edge TPU, they must first be converted to a TensorFlow Lite (TFLite) 8-bit quantized format[39], from which they are compiled[40] for use on the Edge TPU. The quantization conversion is commonly used for reducing the model size with an expected degradation of accuracy as a tradeoff for reducing the object detection latency. After converting and compiling the models into a suitable format, the models' precision was evaluated again on the Google Colaboratory platform and the inference speed was measured on the hardware of the Edge Device. Even though all compared models were compiled into an Edge TPU compatible format, some did not successfully run on the hardware and for such models, the relevant metrics are not provided in the results section of the thesis.

As a proof of concept, only the YOLOv8n model combined with an integrated ByteTrack[21] object tracker was used for building a deployable Evaluation Module[41] to be deployed on the Edge Device. The process of creating an Evaluation Module generally follows the same procedure for all models and differs by the technical implementation. The YOLOv8n model was chosen as it is the most straightforward of the models to integrate into a fully functional module.

[37] https://colab.research.google.com/

[38] https://jupyter.org/

[39] https://www.tensorflow.org/lite/performance/quantization_spec

[40] https://coral.ai/docs/edgetpu/compiler/

[41] https://github.com/35grain/vegeval/tree/master/edge-agent/modules/red-potato-yolov8n

6 System performance

This section gives an overview of the measurable results of the set objectives and the performance metrics of the system, including the throughput of an Edge Device as well as the precision and speed of the compared object detection models.

Cost of Vegeval. With the assumption that Vegeval's web application and cloud storage solution are managed by a single organisation or enterprise, the total cost for an end user (e.g., an agribusiness owner) looking to integrate the system into their workflow would, as an example, consist of the cost of acquiring the necessary hardware for a set of Edge Devices and presumably an added fixed monthly fee for compensating the operating costs of the rest of the system. The total hardware costs for a single Edge Device in its exact configuration at Manufacturer's Suggested Retail Price (MSRP) is approximately 300 euros. However, largely impacted by the ongoing global semiconductor chip shortage [22], the actual cost of the hardware purchased for the project is approximately 400 euros. A large part of the total cost is taken up by the price of the Logitech Streamcam web camera, which could be replaced with a cheaper alternative that is capable of outputting a video stream with a minimum resolution of 640 by 640 pixels, as this is the maximum resolution used in the compared object detection models.

Edge Device's performance. Highly dependent on the inference speed of the object detection models, the Edge Device subsystem presented in this thesis is able to process and generate statistics from 9–11 input frames per second, where 50–60 objects can be detected and tracked comfortably within a single frame when the objects are moving down the conveyor line at a normal speed. This converts to about 150–180 potatoes that can be processed per minute. In cases where an object temporarily moves at an increased speed, the tracker often loses track of the object and a new identifier is assigned to the object once stable. It is important to note, however, that the measured detectable number of objects is limited by the collected dataset and the true maximum could be higher. It is also possible that with highly optimised object detection models, the throughput could be improved further on the same hardware.

Performance of object detection models. For each compared object detection model, its name, precision (mAP) in its native format, number of model operations compiled to be run on the Edge TPU, precision (mAP) in the TFLite int8 quantized format and inference speed on the Edge Device hardware are provided in Table 1. The metrics that were unobtainable without extensive research and coding or did not successfully run in the testing environment are redacted from the results. It is valuable to note that the YOLOv7-tiny and YOLOv6-N with missing values are also the only models from the compared set with no official support for TFLite nor Edge TPU compatible formats.

Model name	**Native precision (mAP@0.5 IoU)**	**Operations compiled for Edge TPU**	**TFLite int8 precision (mAP@0.5 IoU)**	**Raspberry Pi 4 & Edge TPU avg. inf. time**
YOLOv8n (640px)	0.8	231/257 = 89.9%	0.75	69 ms
YOLOv7-tiny (640 px)	0.81	321/330 = 97%	No evaluation script available	Did not run
YOLOv6-N (640 px)	0.79	164/170 = 96%	No evaluation script available	58.9 ms
YOLOv5n (640 px)	0.74	258/261 = 99%	0.7	32.7 ms
EfficientDet-Lite2 (448 px)	0.42	354/357 = 99%	0.31	189 ms

Table 1. Compared object detection model metrics.

Considering the small size of the dataset, the precision (mAP at IoU 0.5) for all models, with the exception of EffiicientDet-Lite2, is satisfactory, with values at or around 0.8. When taking a closer look at the test results, it is clear that in the case of all models, the average precision is brought down by the low precision of the "LowQ" class, which is also a severely underrepresented class in the dataset. For comparison, in the case of the YOLOv8n model, for example, the "StandardQ" and "MediumQ" classes achieved precisions of 0.92 and 0.81 respectively, while the precision for "LowQ" was only 0.67.

The precision degrades slightly for all models with the relevant data when converted to the TFLite int8 (8-bit quantized) format, however, this is also the expected behaviour of the model size reduction process. In terms of inference speed on the Edge Device, the YOLOv5n is ahead of all others with an average latency of 32.7 milliseconds. Do note that the inference speeds measured do not include the added overhead of an object tracker and other supporting services. The best balance between precision and inference speed could be found with the YOLOv6-N or YOLOv5n model as the precision with the TFLite int8 format for the former remains to be unknown, although based on the results of other models, the expected value could be in the range of 0.7 to 0.75.

7 Conclusion and future work

In this thesis, a new vegetable visual quality evaluation system was designed and developed. The new system, Vegeval, adds to and improves on previous solutions for virtually sorting objects with the use of object detection models and edge computing.

Description of the developed system. Vegeval consists of two subsystems, the control panel in the form of a web application and the Edge Device, as well as a supporting cloud storage layer. The control panel provides system users with an analytical overview of the quality of their produce in real-time and allows them to configure and control their Edge Devices. Statistics about the quality of the produce are uploaded to the web application from Edge Devices, which are deployed on-premise at the user's vegetable processing facility. The Edge Devices capture images of objects on a conveyor line using a web camera and apply a preconfigured object detection model on the images to categorise the objects and generate a statistical report based on the results. The raw images captured can optionally be uploaded to an S3-compatible cloud storage server, where the deployable Evaluation Modules are also stored. For training the object detection models, a custom dataset of *Red Potatoes* was collected in collaboration with an agribusiness Saaresepa OÜ and labelled into three quality classes.

Results. The output of the work is a functional vegetable quality evaluation tool that implements a proof of concept Evaluation Module based on the YOLOv8n object detection model with the integrated ByteTrack object tracker for deployment on Edge Devices and displays the generated statistics within the web application. The current implementation of the system was measured to be able to process up to 9–11 input frames per second, where each frame may include 50–60 individual objects, however, the true maximum throughput of the system could be higher and practically increased with a well-optimised object detection model. The total cost of hardware for a single Edge Device can range from 300 to 400 euros due to the ongoing semiconductor chip shortage. The cost can be reduced, however, by opting for a cheaper alternative to the web camera.

Based on the described results, it can be observed that hardware with low computing resources can successfully be deployed for fulfilling computer vision and object detection tasks in the discussed use cases. The latter additionally indicates that applying artificial intelligence to make everyday tasks more efficient does not necessarily have to come at a large expense.

Future of Vegeval. Due to the time constraints of the project, there are many aspects of the system with room for further development and improvement. First and foremost, given sufficient resources, the precision of object detection models can be improved by expanding the dataset and fine-tuning model parameters. Additionally, as this thesis

presents an object detection model for only one vegetable type, models for other vegetables can similarly be trained and made available for use in the Vegeval system. What is more, continuous training techniques could be implemented to create an automatic flow for improving already existing and deployed models in time as the dataset grows [23]. Allowing users to opt into uploading the collected raw images from Edge Devices to a cloud storage space is an example of the preconditions for this to be possible.

Another aspect of the system that can be improved is the fault tolerance of the Edge Device service agent. With the current implementation, the Edge Device assumes that the web application and MinIO storage server are always available and internet connectivity is stable. This may not always be the case, however, and fallback methods should be implemented to guarantee that the system can continue operating even when connectivity between the subsystems is temporarily lost. One way to mitigate downtime would be to save the collected statistics and raw images into local storage until a stable connection is restored and the data can be offloaded to the web application and storage server. The same applies to retrieving the Edge Device configuration from the web application on boot, where the previously retrieved configuration could be saved locally.

In terms of object detection speed, alternative hardware solutions with built-in GPUs should be considered for applications of the system where objects must be detected in minimal time, as the Raspberry Pi 4 Model B paired with the Coral USB Accelerator Edge TPU will only perform well with optimised object detection models. However, multiple Edge TPUs can also be run in parallel for load distribution and increased performance. A dedicated GPU, on the other hand, would allow the models to be run in their native format and not become less accurate due to conversions as is the case with most general-purpose models running on the Edge TPU. It would consequently greatly reduce the amount of time spent on fiddling with runtime and model export settings to get everything running as expected. As an example, one such alternative to consider is the Nvidia Jetson Orin Nano Developer Kit[42]. It is important to bear in mind, though, that increased performance also comes at an increased cost.

In the long term, in order to compensate for the operating costs of the system, it could only exist as a commercialised product. For that, the system should additionally be thoroughly tested and implement methods for scaling the system to increase the throughput of user and Edge Device requests. As a starting point, however, the developed system is a great fit.

[42] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

References

[1] T. R. Undru, U. Uday, J. T. Lakshmi, A. Kaliappan, S. Mallamgunta, S. S. Nikhat, V. Sakthivadivel, and A. Gaur, "Integrating artificial intelligence for clinical and laboratory diagnosis - a review," *Maedica*, vol. 17, no. 2, p. 420—426, June 2022.

[2] R. Espinosa, H. Ponce, and S. Gutiérrez, "Click-event sound detection in automotive industry using machine/deep learning," *Applied Soft Computing*, vol. 108, p. 107465, 2021.

[3] M. Abdallah, M. Abu Talib, S. Feroz, Q. Nasir, H. Abdalla, and B. Mahfood, "Artificial intelligence applications in solid waste management: A systematic research review," *Waste Management*, vol. 109, pp. 231–246, 2020.

[4] F. Bal and F. Kayaalp, "Review of machine learning and deep learning models in agriculture," *International Advanced Researches and Engineering Journal*, vol. 5, pp. 309–323, 08 2021.

[5] Q. Su, N. Kondo, D. firmanda al Riza, and H. Habaragamuwa, "Potato quality grading based on depth imaging and convolutional neural network," *Journal of Food Quality*, 2020.

[6] C. Wang and Z. Xiao, "Potato surface defect detection based on deep transfer learning," *Agriculture*, vol. 11, no. 9, 2021.

[7] R. Horaud, M. Hansard, G. Evangelidis, and C. Menier, "An overview of depth cameras and range scanners based on time-of-flight technologies," 12 2020.

[8] Q. Su, N. Kondo, M. Li, H. Sun, and D. F. Al Riza, "Potato feature prediction based on machine vision and 3d model rebuilding," *Computers and Electronics in Agriculture*, vol. 137, pp. 41–51, 2017.

[9] C. Shorten and T. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 07 2019.

[10] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014.

[11] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*. Springer, 2018, pp. 270–279.

[12] F. Pla, J. Sanchiz, and J. Sanchez, "An integral automation of industrial fruit and vegetable sorting by machine vision," in *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*, vol. 2, 2001, pp. 541–546 vol.2.

[13] A. Narzary and S. Ashok, "Real-time monitoring of conveyor using computer vision and iot," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, vol. 1, 2019, pp. 73–78.

[14] R. Aljaste, "35grain/vegeval." [Online]. Available: https://github.com/35grain/vegeval

[15] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Z. Yifu, C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022.

[16] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," 2023.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[18] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[19] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022.

[20] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[21] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022.

[22] W. Mohammad, A. Elomri, and L. Kerbache, "The global semiconductor chip shortage: Causes, implications, and potential remedies," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 476–483, 2022, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

[23] I. Prapas, B. Derakhshan, A. Mahdiraji, and V. Markl, "Continuous training and deployment of deep learning models," *Datenbank-Spektrum*, vol. 21, 11 2021.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Richard Aljaste**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Vegetable Visual Quality Evaluation System Based on Artificial Intelligence, supervised by Chinmaya Kumar Dehury.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe on other persons' intellectual property rights or rights arising from the personal data protection legislation.

Richard Aljaste **May 9, 2023**