BananAI

A deep learning project to detect the ripeness of bananas using an Arduino

Why apply Artificial Intelligence (AI) to bananas?

Artifical Intelligence (AI) is everywhere. From search bots, to cars, to marketing services, AI is experiencing a profound and significant moment in the Zeitgeist. The unregulated and unknown power of tools such as ChatGPT has led to many industry figures calling for a halt to further innovation to prevent catastrophic consequences (Narayan, Hu, &, Mukherjee, 2023). While such tools are immensely complex and almost impossible to reproduce in civilian hands, targeted domain-level AI solutions can be trained and deployed by novices. This student project chose to attach AI to a unique domain: that of the banana .

Bananas are an instantly recognisable fruit. Undeniably unique, bananas are easy-to-hold, hangable, and come with their own protective wrapper. Bananas, like many other fruit, provide value from their ripeness and availability to eat, which diminishes over time (Rizzo et al., 2023). As such, it becomes important for both supply chain efficiency and personal consumption to know when a banana is at the ideal point to eat.

BananAI is the application of deep learning to banana ripeness, using image recognition to determine when a banana is best to eat. In combination with an Arduino BLE 33 Sense, BananAI is deployed on a banana stand to provide an immediate indication of when to consume, and for monitoring ripeness in a shop setting. The following document describes how BananAI came to be, and outlines the steps taken in its development. Instructions on how to deploy are also provided.

The GitHub repository can be found here.

The final two category Edge Impulse model can be cloned from here.

An Excel spreadsheet with links to each version of the Edge Impulse projects can be found here.

Contents

- Model training and iteration.
- Software, device design, and deployment
- Findings and conclusion

Research question

Given the need for a way to detect banana ripeness, the following research question was determined:

Can we use an onboard, transfer learning model on an Arduino to understand when bananas become overripe, and how much training data do we need for an acceptable accuracy level (> 90%)?

A project outline is provided below.

Project outline

Ordered steps were taken to design BananAI. These steps included:

- Deciding on the physical and software/AI architecture.
- Gathering images.
- Processing and transforming images.
- Iterative model training and data gathering through experimentation.
- Design of the software to deploy on Arduino.
- Construction of the physical device(s).
- Deployment and testing.

Deciding on the physical and software/AI architecture

BananAI was split into two architectures: software (including AI) and hardware.

Software

BananAI utilises Transfer Learning. Transfer learning is an AI model architecture that uses large, existing models to act as a basis for the training of smaller collections of domain-specific data, which can save time (Tensorflow, 2022). Transfer learning was a good choice for BananAI, as it doesn't necessitate the gathering of thousands of images of bananas.

The AI model would be trained on Edge Impulse. Edge Impulse is a Platform-as-a-Service (PaaS) for training and deploying AI models on a variety of hardware (including Arduinos) (Edge Impulse, 2022). Edge Impulse is streamlined and easier to use for fast iteration of Arduino-focused AI models, as it can deploy to these devices easily. Edge Impulse also offers compression to produce models that are compatible with low-memory devices.

Hardware

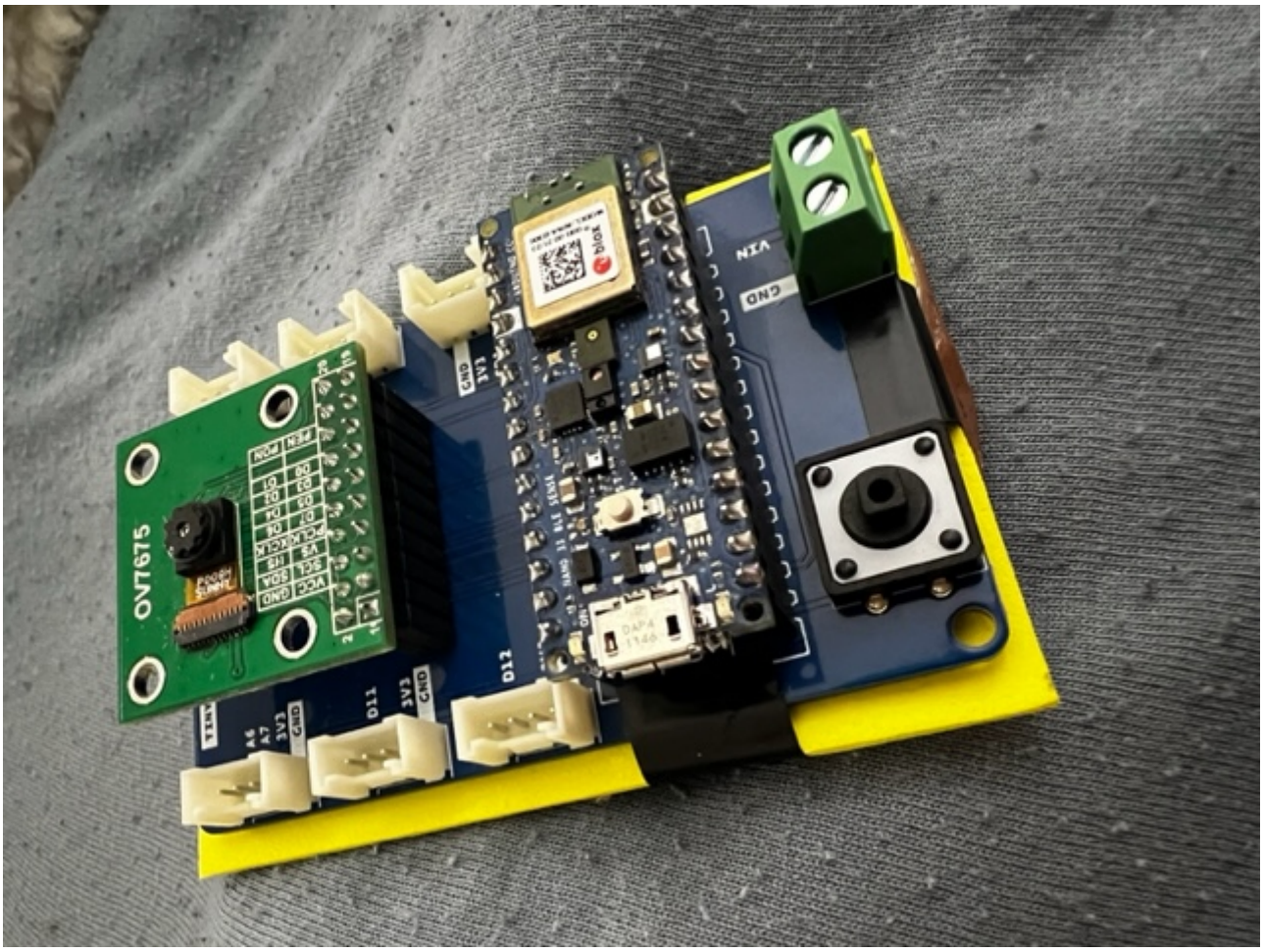


Figure 1: Arduino Nano 33 BLE Sense.

The device selected for this AI is the Arduino Nano 33 BLE Sense (Arduino.cc, n.d.). The Nano 33 BLE Sense is capable of running machine learning models with inputs from a camera shield, which would suit a live classification of bananas. The device is also small and easy to deploy into an enclosure. However, it has several specification limitations, which are:

- 256KB of SRAM.
- 1MB of storage.
- A 64Mhz processor.

These limitations constrained the complexity and accuracy of the AI model, as will be described later in the readme.

Data domain, gathering and transformation

Banana ripeness is a sliding scale, and past research into statistical and AI banana classification shows varying opinions on how to approach the problem. For example, Mazen and Nashat define four categories of classification for an Artificial Neural Network (ANN), based on the percentage value of brown spots and other features (2018). Marimuthu and Roomi (2017) used three categories for a fuzzy model (unripe, ripe, overripe). Saragih and Emanuel (2021) define four categories: unripe, yellowish-green, mid-ripen, and overripe.

Rizzo et al. (2023, p.46) note that categories of ripeness "can be arbitrarily large". Marimuthu and Roomi (2017, p.4095) note that it is "ambiguous to quantify the ripening levels with strict boundaries" since banana

ripeness is "fuzzy in nature". Another factor in this classification decision is the Arduino's limited memory of 256KB. Models with more categories would be too large. Furthermore, the "green" stage of a banana is challenging to get data for, as it occurs before shipping commences (Marimuthu & Roomi, 2017).

Hence, images of bananas were split into three categories, being:

- Underripe.
- Ripe.
- Overripe.

However, these categories were eventually condensed into ripe and overripe for reasons described below.

Photos of bananas were captured on an iPhone 13 Pro in various positions, bunches of bananas, lighting circumstances, and with varying backgrounds. Different distances were captured to ensure that the deployed devices can recognise bananas from varying positions (see below: "Software, device design, and deployment"). The three category data set can be obtained here. A selection of images is displayed below to illustrate the data set's diversity.

Figure 2: banana images sample.

Images were then transferred to a personal computer and converted from HEIC to JPG format. Following conversion, a batch image processing tool was used to transform images to 288x384 resolution. Images were then stored in categorised folders for ease of access and low-quality or irrelevant images were removed.

Model training and iteration

Three Category Model

Initially, three banana categories were trained across a range of parameters. The parameters included:

- Number of images.
- Epochs (20 or 50 training cycles).
- Data augmentation (enabling algorithmic transformation of image data to produce a larger image set).
- Final layer neurons (0 or 32).

Training was split by number of images, then each sub-category was trained (e.g., number of epochs) at a learning rate of 0.001. MobileNetV1 96x96 0.25 was used as the transfer learning model, as its output size was small enough to fit within 256KB of RAM.

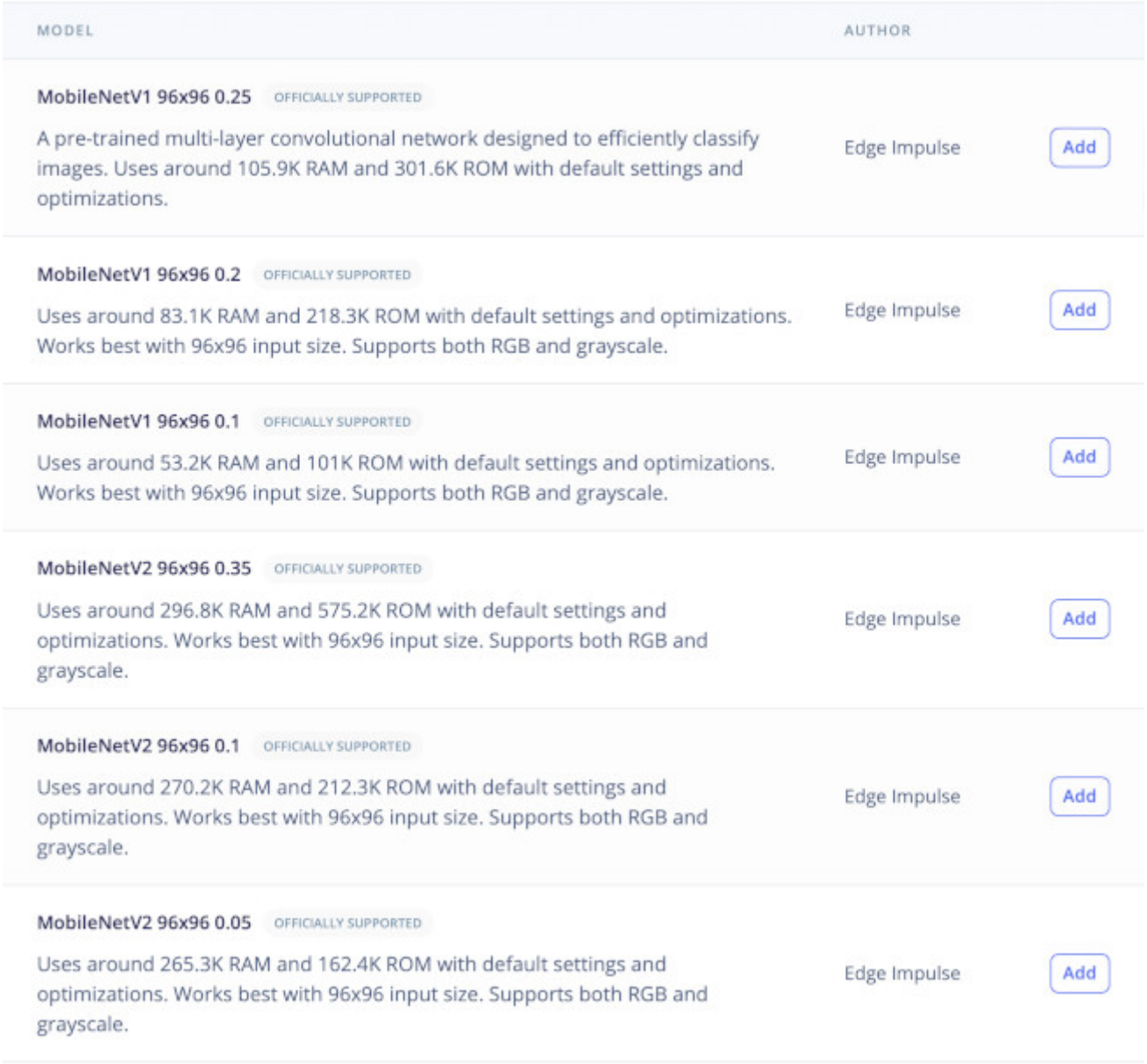


Figure 3: model complexity.

1,186 images were collected in total. Note: images were split in a ~80:20 ratio between training and validation data, meaning image counts must be multiplied by 0.8 to get the true training and testing image count.

Accuracy declined marginally as images were added to the model.

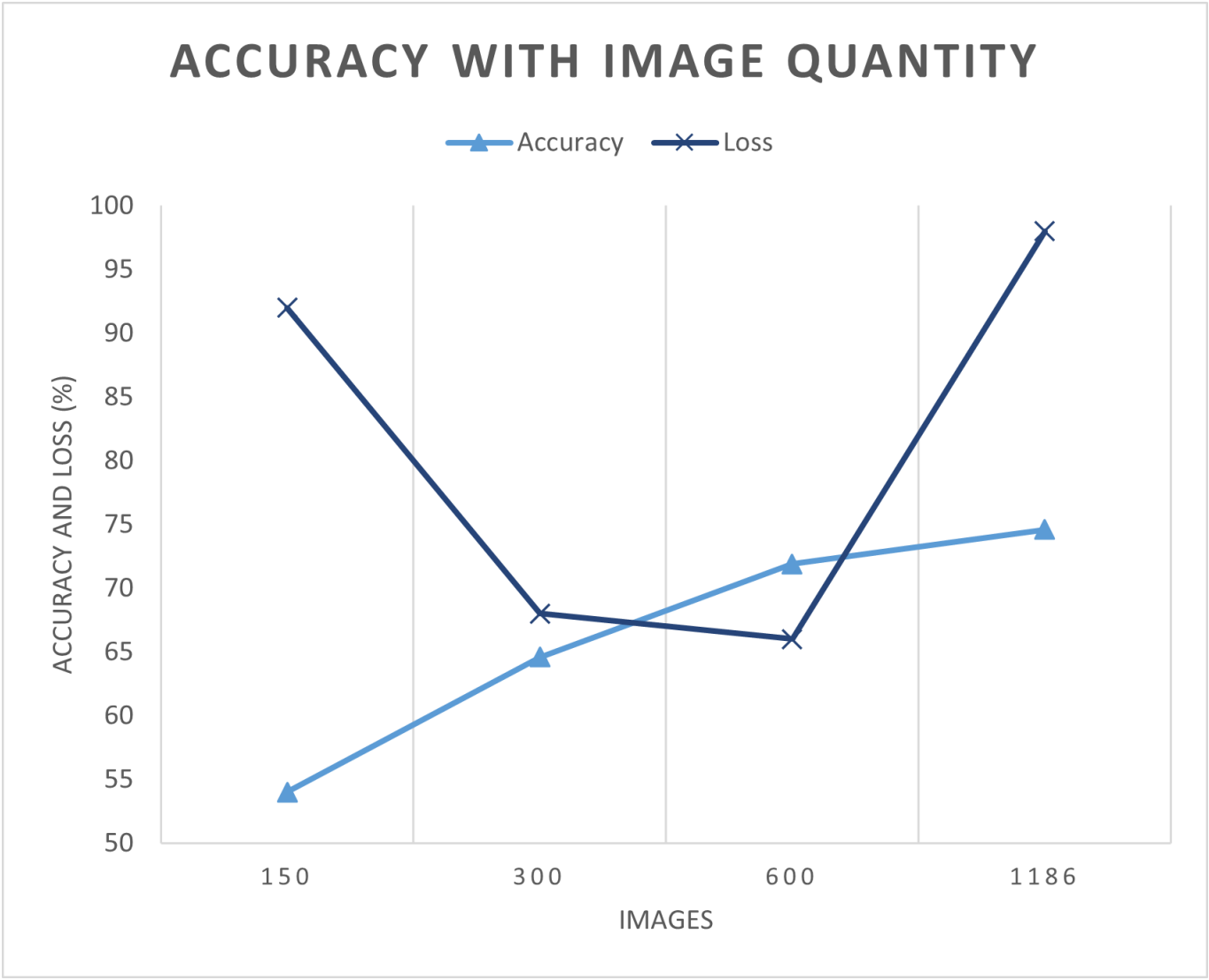


Figure 4: three category accuracy with images.

An interesting finding was that both data augmentation and the addition of final layer neurons became less effective at increasing accuracy as the model's images increased in number - even leading to a loss in accuracy.

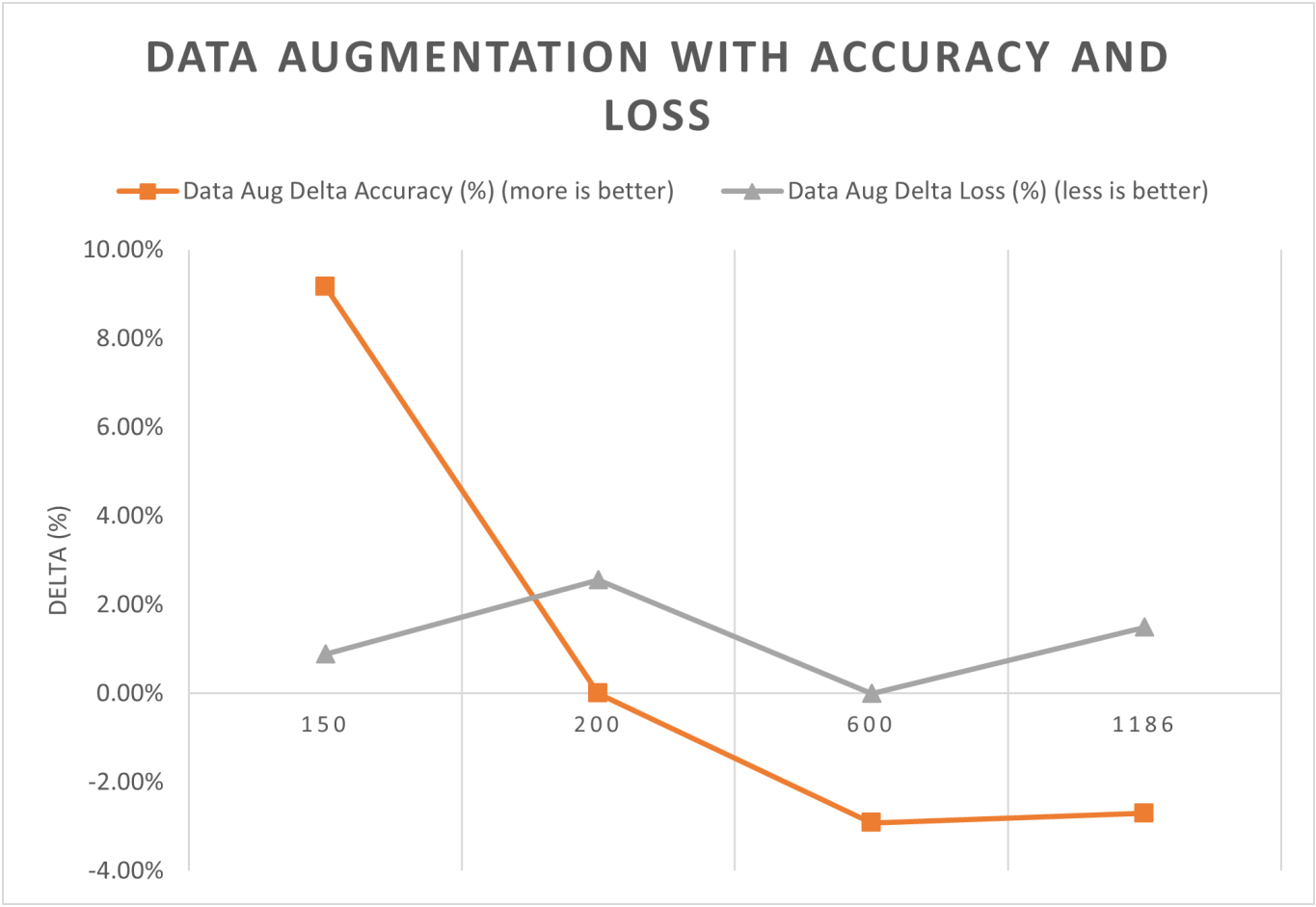


Figure 5: three category accuracy with data augmentation.

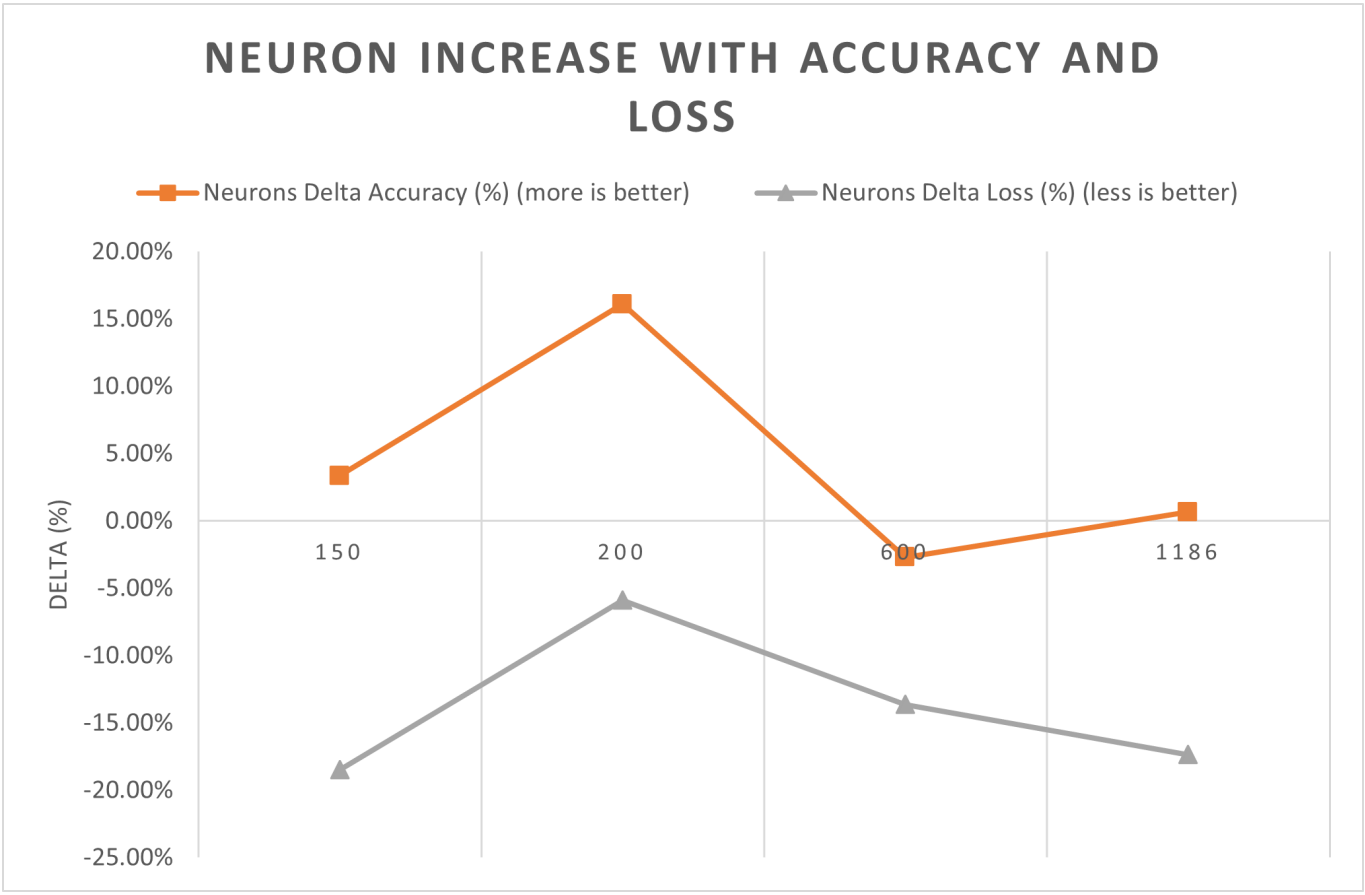


Figure 6: three category accuracy with final layer neurons.

The most successful model was able to achieve a validation data accuracy of 71% based on a training accuracy of 74.6%, which was far below optimal.

	OVERRIPE	RIPE	UNDERRIPE
OVERRIPE	92.4%	1.5%	6.1%
RIPE	15.8%	49.1%	35.1%
UNDERRIPE	10.6%	10.6%	78.8%
F1 SCORE	0.85	0.60	0.73

Figure 7: three category model accuracy.

A large amount of the accuracy loss came from the model's inability to distinguish between ripe and underripe bananas, which have a subtle difference in colour and texture. Another study had similarly degraded accuracy between these categories (Mazen & Nashat, 2018).

Figure 8: ripe and underripe comparison.

As such, a decision was made to narrow the model's categories to ripe and overripe.

Two Category Model

The two-category model proved significantly more accurate. It is surmised that this increase in accuracy comes from the removal of ambiguity between unripe and ripe image categories, which were quite similar. Two category accuracy only increased marginally with an increase in images.

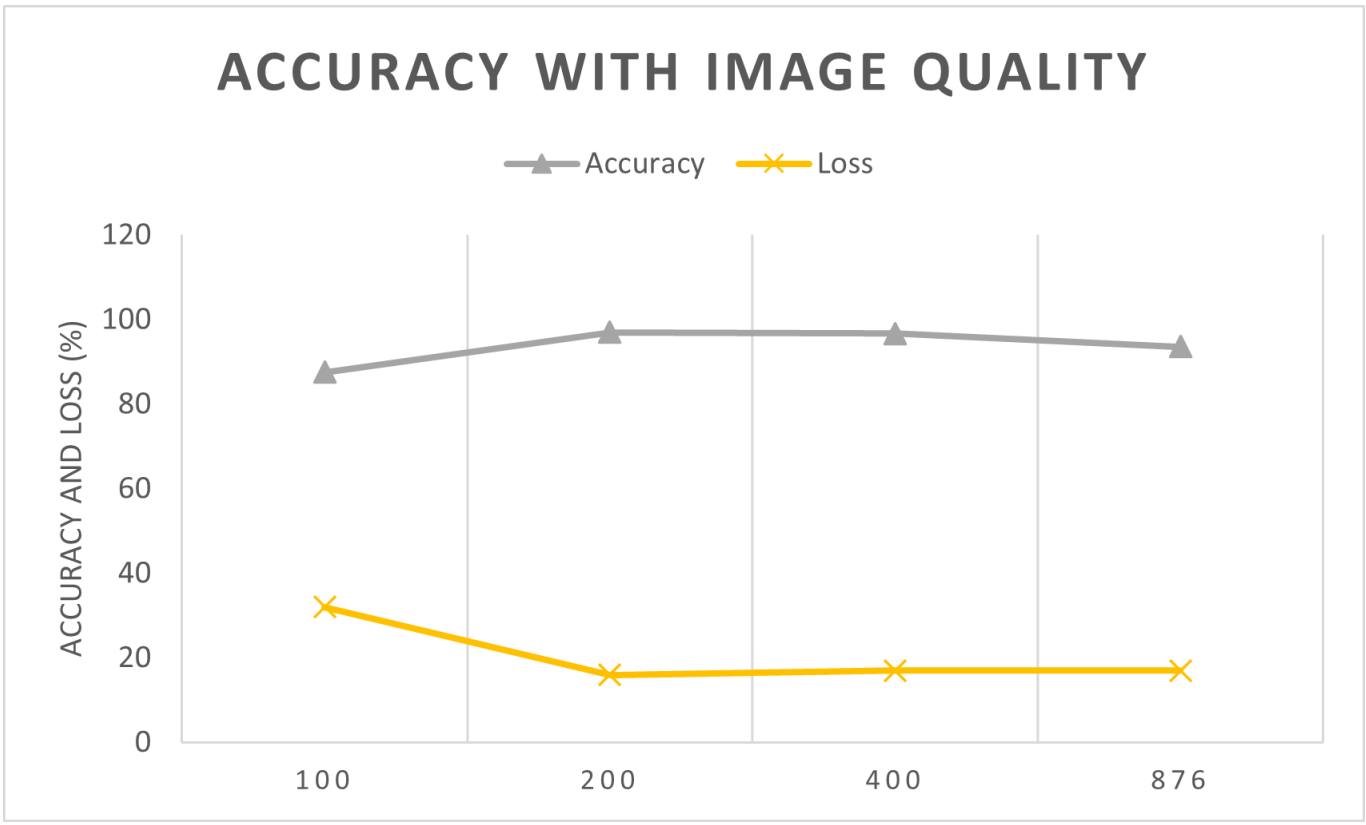


Figure 9: two category accuracy with images.

The most accurate model produced 100% accuracy on validation data:

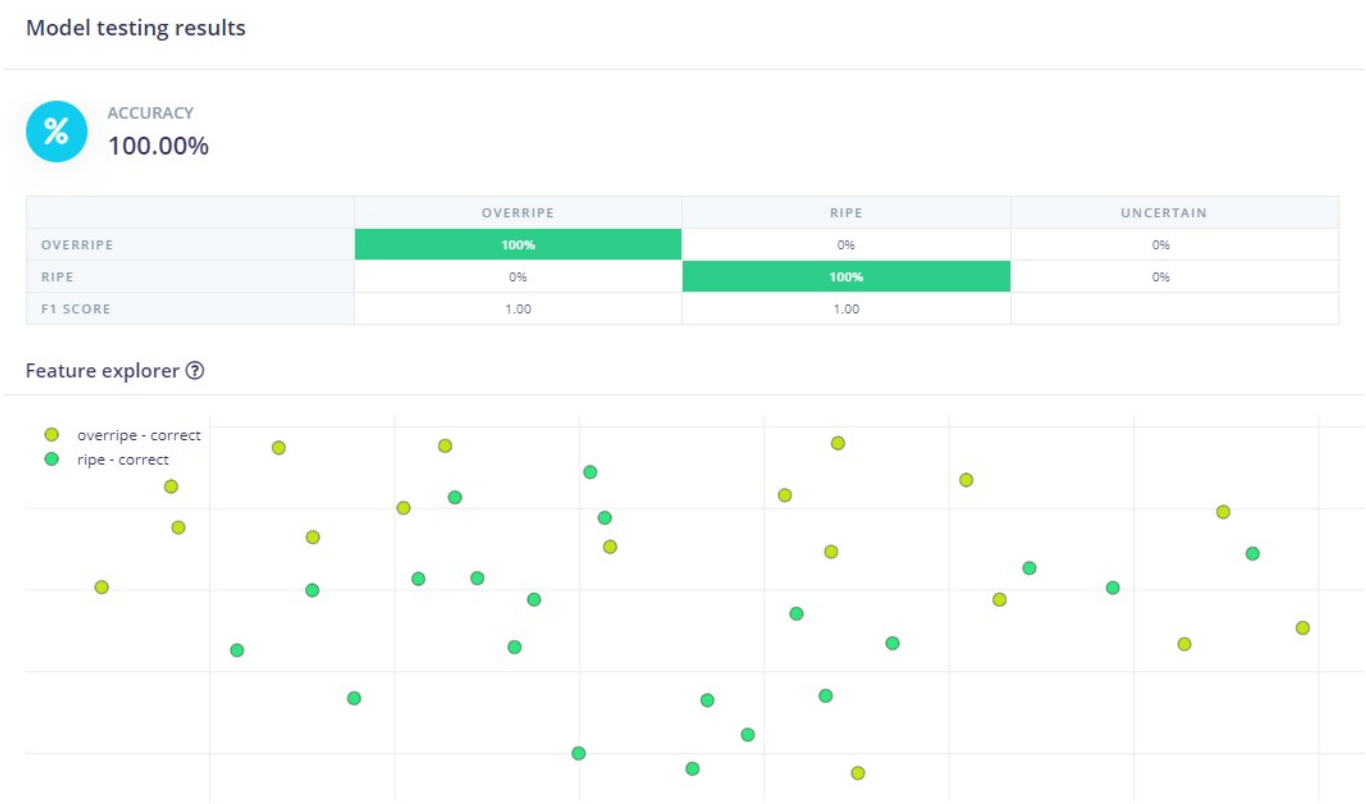


Figure 10: two category model accuracy.

While this model only scored 87.8% accuracy on training data, other models achieved up to 97%:

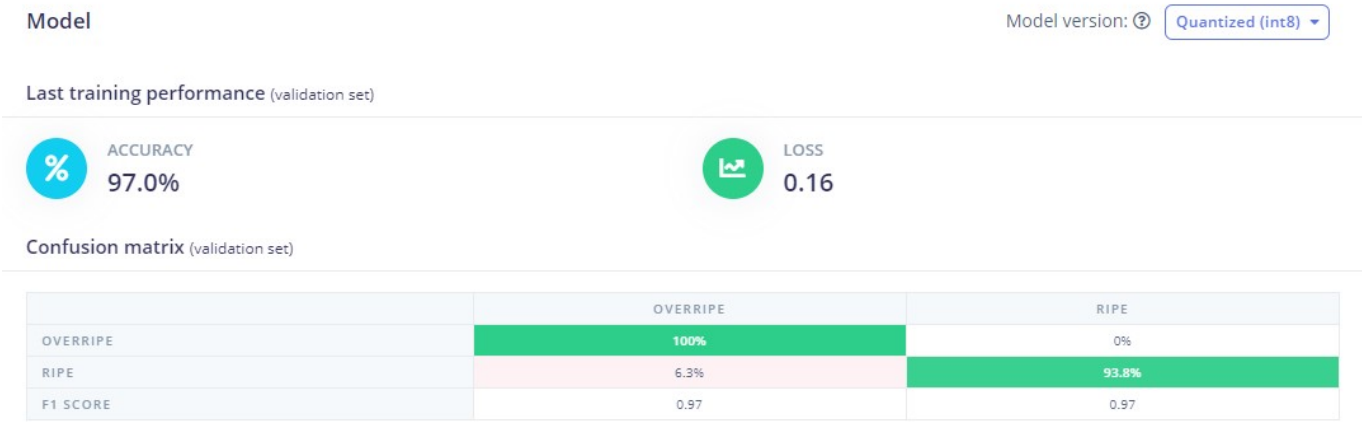


Figure 11: two category training accuracy.

As with the three-category model, both data augmentation and final layer neurons led to a general decrease in accuracy as model images increased in number:

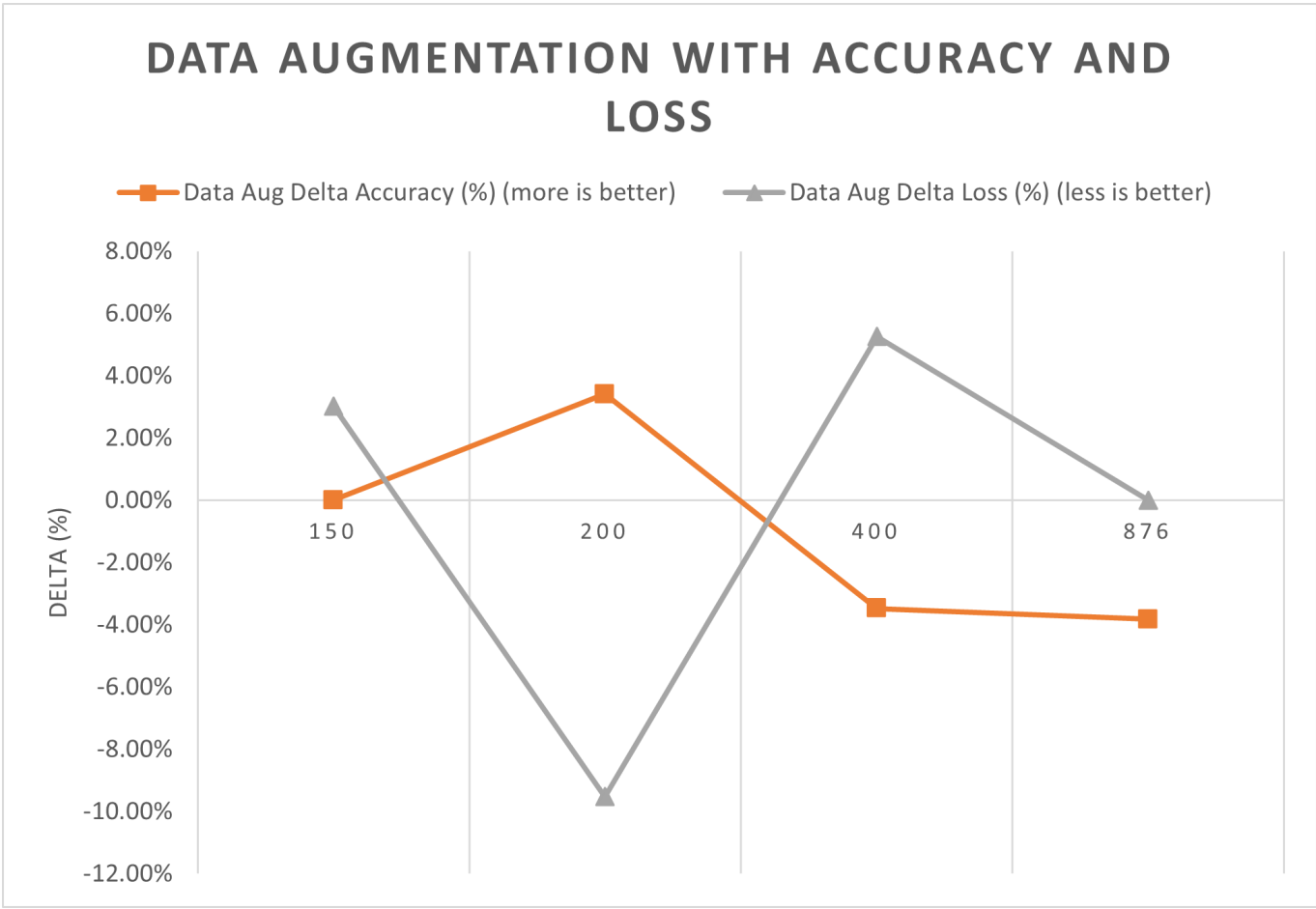


Figure 12: two category accuracy with data augmentation.

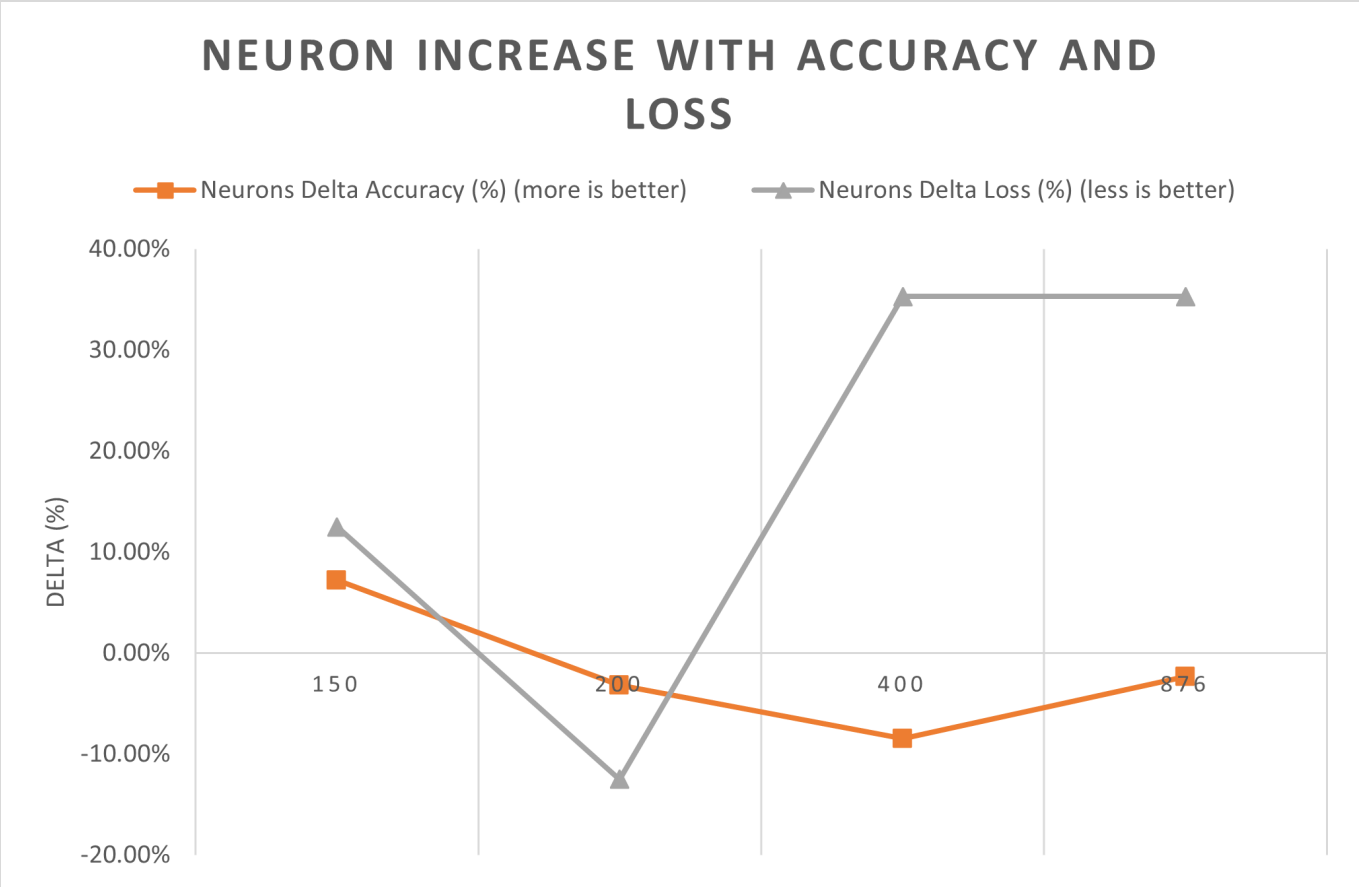


Figure 13: two category accuracy with final layer neurons.

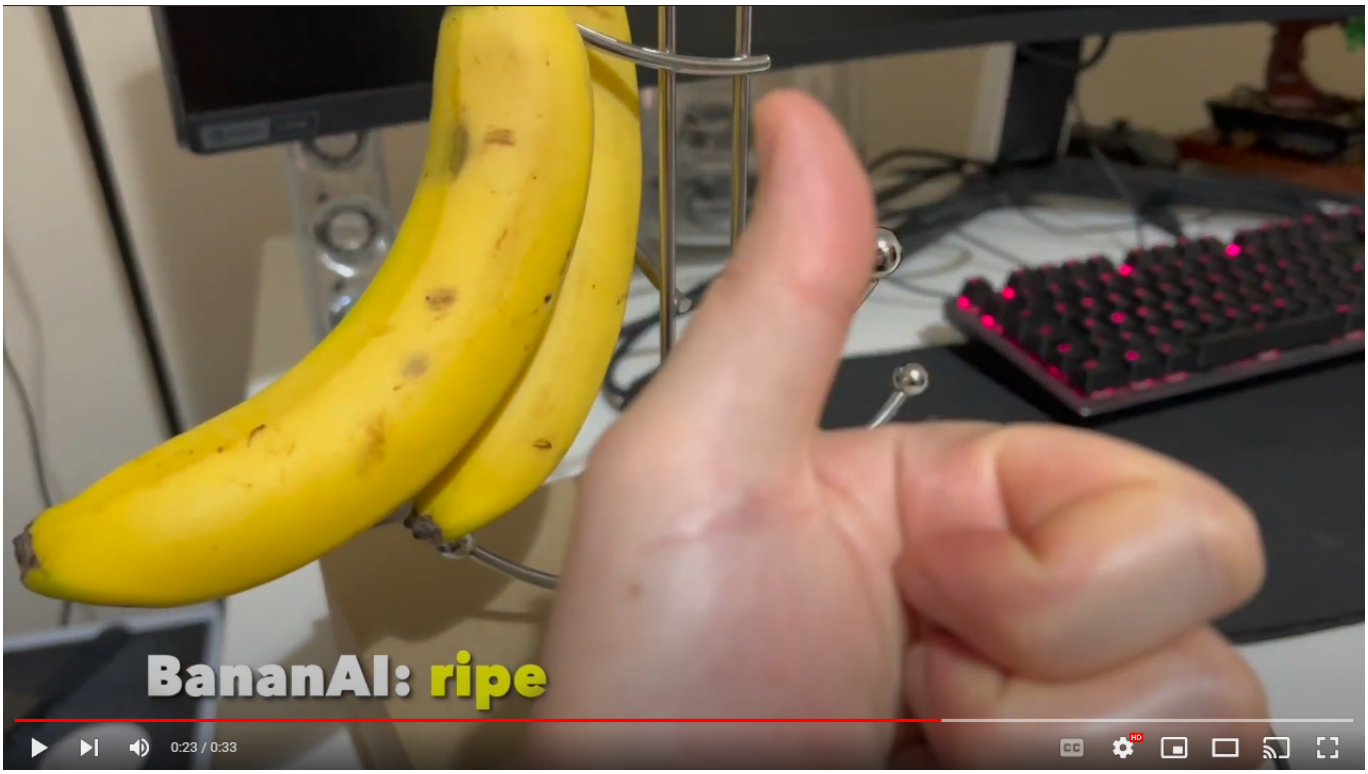
Ultimately, the model with 95% training and 96% validation accuracy was selected, as it represented the most generally accurate version.

Software, device design, and deployment

Software was exported from Edge Impulse as an Arduino library, which was subsequently modified. The library draws on image data from the Arduino's camera shield and runs this data in the model. Inferences are then drawn and logged on the serial console. Originally, the development plan included coding an alert for when fruit becomes unripe. However, time constraints meant this was not possible. Furthermore, the chosen Arduino model did not possess Wi-Fi capabilities, making networking a challenge. The device could be set to act as a low-energy Bluetooth® peripheral for remote reading, using some code from Arduino.cc (2023).

Code was deployed on the Arduino, and a prototype banana stand was constructed out of a box and a kitchen hanger.

Figure 14: prototype banana stand.



A YouTube video of this prototype can be found here.

However, this protoype was flimsy and didn't have a design suitable for use in the home. A second prototype was then developed.

Store monitoring device construction instructions

A version of the banana stand was prototyped for use in stores. This prototype is intended to be suspended above bananas, using a mechanical arm. The arm can be adjusted as is necessary to face specific targets.

Figure 15: store prototype

A YouTube video of this prototype can be found here.

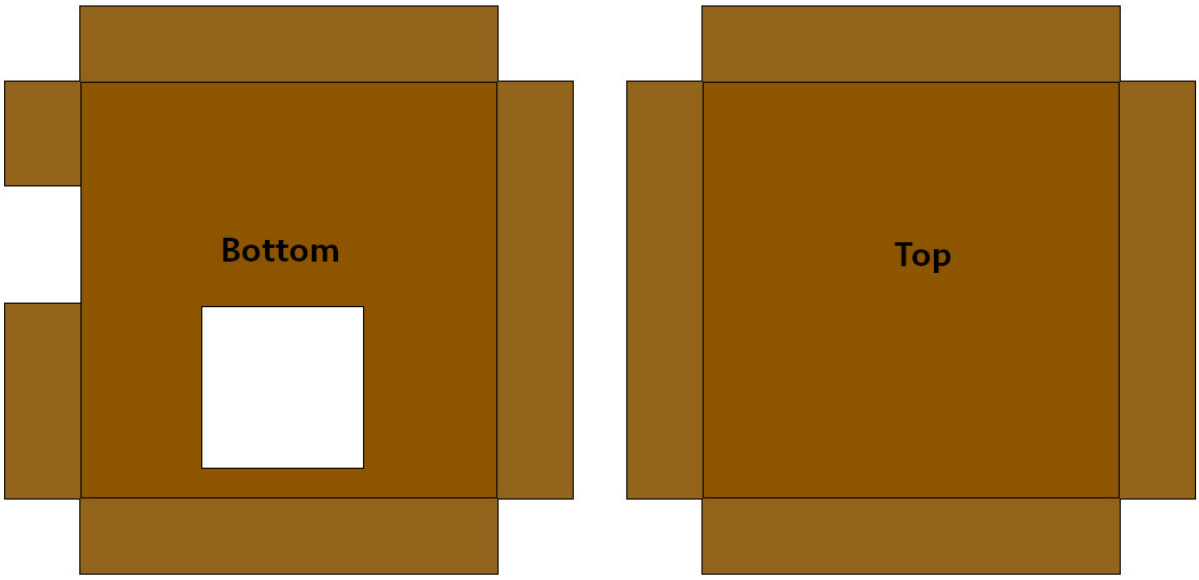
Requirements for deployment:

- A soldering arm kit: Amazon.
- A small cardbox box (measurements 10 * 2 * 10 cm).
- An Arduino Nano 33 BLE Sense kit: Arduino.

- A generic 5 volt power supply and USB-A cable: Amazon.
- Tape.

Instructions:

- Attach the camera shield to the Arduino Nano.
- Cut the box as illustrated below:



- Place the Arduino, still connected to the USB cable, within the box so as to align with the opening.
- Fold the box and seal with tape.
- Attach one arm to the soldering arm base and further attach a clamp to its end.
- Attach the clamp to the box and wrap the cable around the arm.

Deployment Instructions

Deployment requires that the user clone this repository and compile the MainAI.ino file onto an Arduino Nano 33 BLE Sense via the Arduino IDE. Additionally, the user will need to compile the Arduino library on the Edge Impulse project and install it via the Arduino IDE (see below). If the user wishes to train this model themselves, image data can be downloaded from here.

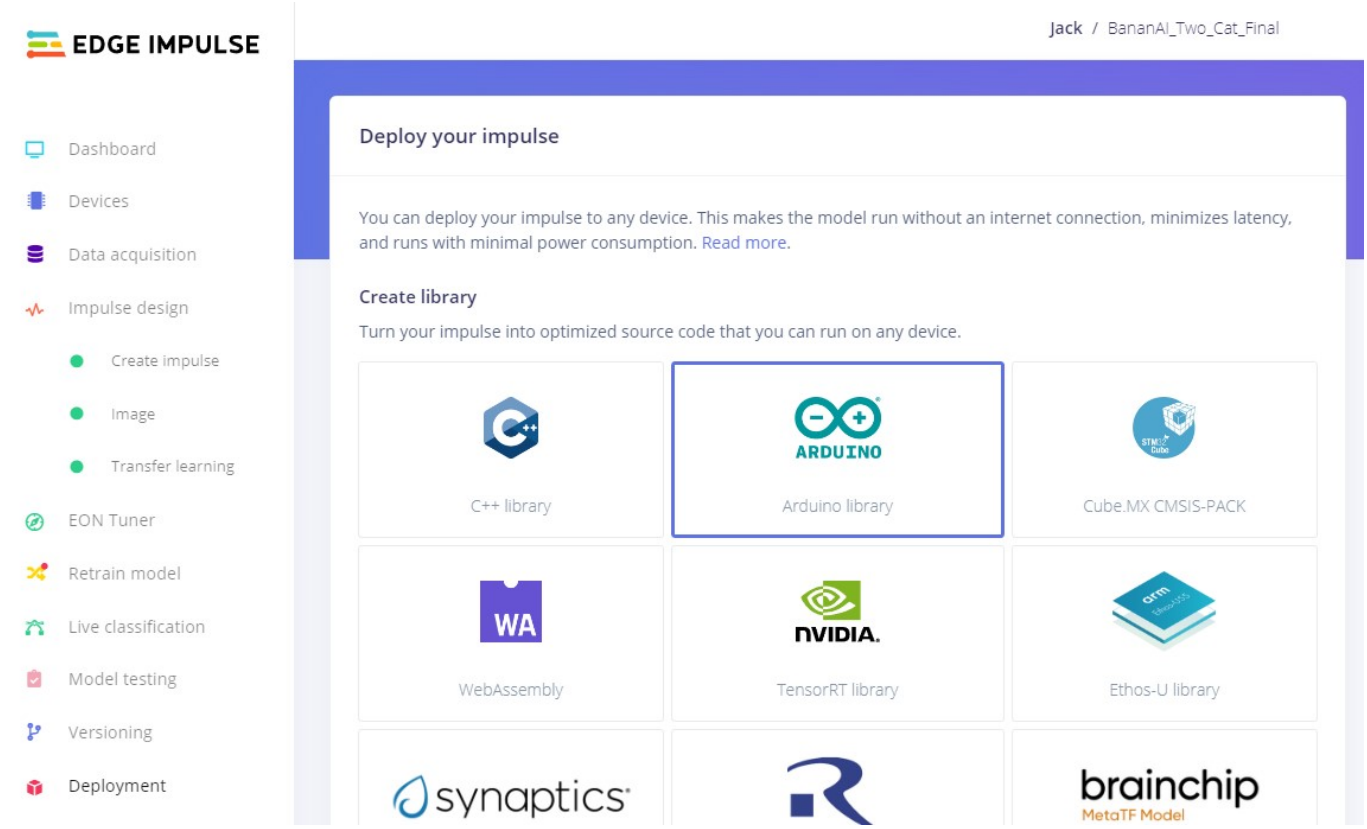


Figure 16: deploy library

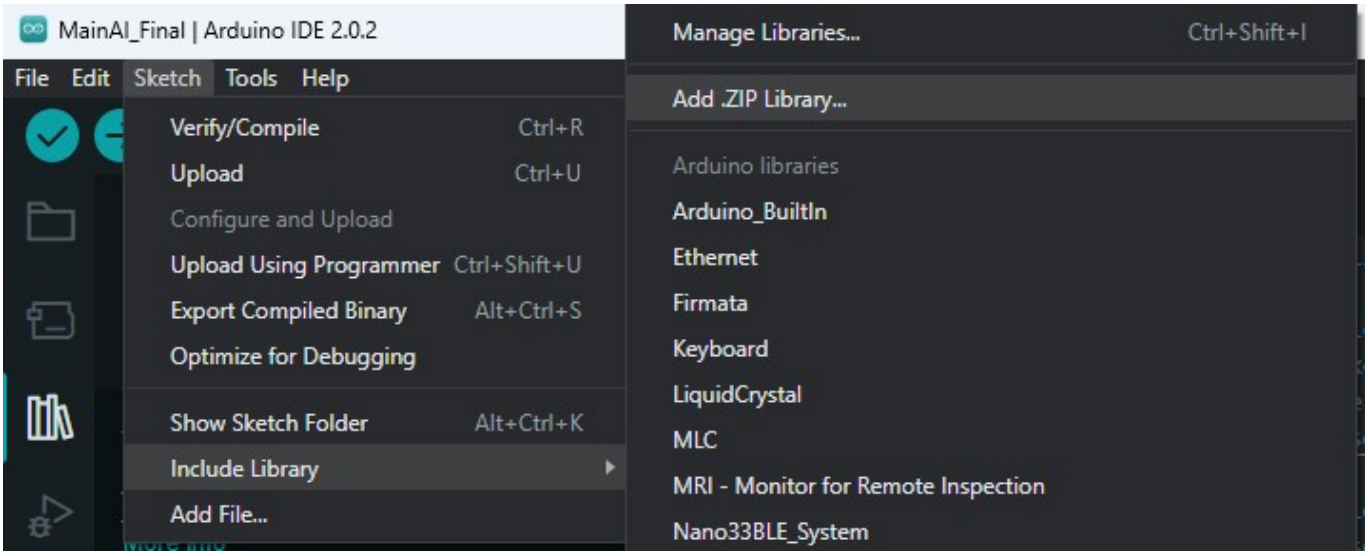


Figure 16: install library

The final two category Edge Impulse model can be cloned from here.

Findings and conclusion

Reflections and future work

Several key findings were drawn from this small study into AI:

- Achieving a 90% level of validation and test accuracy was possible with as little as 200 images in a two category transfer model. However, nearly 1,200 images were insufficient for a three-category model with >90% accuracy.

- This two-category model was deployable on an Arduino device despite RAM limitations.
- Accurate three-category models appear challenging to achieve with a deep learning approach, most likely due to the subtle differences between banana ripeness stages. Other approaches, such as ANNs, seem more effective at discrete ripeness categorisation (Mazen & Nashat, 2018).
- Increasing images tended to increase accuracy, but with diminishing returns. Time constraints meant that it was not possible to gather more than the ~ 1,200 obtained during this study, and a more accurate three-category model would take a lot longer to develop.
- Both data augmentation and final layer neurons can increase the accuracy of smaller data sets, but harm accuracy as the number of images increases.

Future work could expand on this project by:

- Attempting to build a very large banana image set to allow the classification of more categories.
- Utilising other machine learning methods, such as ANNs, in conjunction with deep learning image recognition.
- Building a more robust enclosure for commercial deployment.
- Expanding on the software in this project to connect the Arduino to a distributed network for in-store fruit monitoring.
- Attempting to build a more generalised ripeness model that uses image data from other species of fruit (e.g., strawberries, peppers, oranges, apples, and more).

Conclusion

This project sought to determine how many images were needed to develop a deployable AI model for recognising banana ripeness and unripeness of above 90% accuracy. The mini study concluded that approximately 200 images are needed to achieve >90% accuracy. Furthermore, several insights were gained into the training process and viability of an Arduino Nano 33 BLE sense for banana ripeness monitoring. Future work could expand the model, choose a more powerful Arduino device, and extend the training process to include other fruit. By continuing this research, food wastage may be tackled in a cost-effective and intelligent way.

References

Amazon (2023). '*Model fit: underfitting vs. overfitting*'. Available at: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html (accessed 1 April 2023).

Arduino.cc (2023). '*Connecting Nano 33 BLE Devices over Bluetooth®*'. Available at: https://docs.arduino.cc/tutorials/nano-33-ble-sense/ble-device-to-device (accessed 25 April 2023).

Arduino.cc (n.d.). '*Nano 33 BLE Sense*'. Available at: https://docs.arduino.cc/hardware/nano-33-ble-sense (accessed 13 March 2023).

Edge Impulse (2022). '*Getting started*'. Available at: https://docs.edgeimpulse.com/docs (accessed 7 April 2023).

Marimuthu, S. and Roomi, S. M. M. (2017). 'Particle swarm optimized fuzzy model for the classification of banana ripeness'. *IEEE Sensors Journal*, 17(15), pp.4903-4915. doi: 10.1109/JSEN.2017.2715222.

Mazen, F. M. A. and Nashat, A. A. (2019). 'Ripeness classification of bananas using an artificial neural network'. *Arabian Journal for Science and Engineering*, 44, pp.6901–6910. doi: 10.1007/s13369-018-03695-5.

Narayan, J., Hu, K. and Mukherjee, S. (2023). *'Elon Musk and others urge AI pause, citing 'risks to society'*'. Available at: https://www.reuters.com/technology/musk-experts-urge-pause-training-ai-systems-that-can-outperform-gpt-4-2023-03-29/ (accessed 3 April, 2023).

Rizzo, M., Marcuzzo, M., Zangari, A., Gasparetto, A. and Albarelli, A. (2023). 'Fruit ripeness classification: A survey', Artificial Intelligence in Nature, 7, pp. 44-57. doi: 10.1016/j.aiia.2023.02.004.

Saragih, R. E. and Emanuel, A. W. R. (2021). 'Banana ripeness classification based on deep learning using convolutional neural network', *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*. Surabaya, Indonesia, 9-11 April. Piscataway, NJ: IEEE. pp. 85-89. doi: 10.1109/EIConCIT50028.2021.9431928.

Tensorflow (2022). '*Transfer learning and fine-tuning*'. Available at: https://www.tensorflow.org/tutorials/images/transfer_learning (accessed 20 March 2023).

Declaration of Authorship

I, Jack Shiels, confirm that the work presented in this assessment is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Jack Shiels

27/04/2023