

Project Report

Damaged vs Undamaged Egg Classifier — an AI image classifier that flags cracked egg shells

LRN International GCSE Artificial Intelligence (7923) — Paper 2, Component B

PRACTICE DRAFT - the accuracy figures below are placeholder values, not measured results

Candidate name	Muhammad Usama
Candidate number	2345
Centre name	_____
Project title	Damaged vs Undamaged Egg Classifier
Tool used	Edge Impulse (WebAssembly export, int8 quantised)
Date	2 September 2026

1. Explanation of the AI System and Objective

My system is an image classifier that looks at a photograph of an egg and decides whether the shell is damaged or not. I built it with Edge Impulse and deployed it as a web page: the user selects a photograph, the page resizes it to 120 x 120 pixels, and the model returns damaged or notdamaged with a confidence score. Results below a 0.60 threshold are reported as uncertain rather than being shown as an answer.

I chose this problem because a hairline crack lets bacteria into the egg, so spotting it matters, but writing a fixed rule for 'a crack' is very hard: cracks vary in length, angle, width and shadow, and a dark line on a shell can equally be dirt or a natural marking. A model that learns from examples can pick up that difference where a rule cannot. It would be useful at a packing station or on a smallholding, as a first-pass check that flags suspect eggs for a person to look at.

2. Step-by-Step Process

Step 1: Problem definition

The model had to take one still photograph and return one label from a fixed set, together with a confidence score between 0 and 1. There are two labels, damaged and notdamaged, and the output feeds a decision that has a real cost attached, so a confidence threshold of 0.60 was added to hold back weak answers. The output is a single winning label plus the confidence, which the web page shows as a bar for every class so that a close call is visible rather than hidden.

Step 2: Choosing the tool

I used **Edge Impulse**. I chose it for three reasons. First, it trains models in the browser with no coding, so all of my time went into the data and the testing rather than into setting up a Python environment. Second, it lets me upload my own dataset instead of only using its sample projects, which the syllabus expects. Third, and most importantly for this project, it exports the finished model as WebAssembly, so the classifier runs on my own web page rather than on somebody else's site.

Step 3: Collecting the data

The dataset is a collection of photographs of eggs, some with damaged shells and some sound. Source and licence: _____. There are two labels: **damaged** and **notdamaged**.

Step 4: Preparing the data

I removed photographs where the damage was implied by the caption rather than actually visible, and any image too unclear for me to classify by eye myself. I then split the images **80 / 20** into a training set and a separate test set. The split matters because a model scored on the images it was trained on would only be proving that it can memorise; the test set is the only honest measure.

Step 5: Training the model

Edge Impulse trained a **MobileNetV2** image model using transfer learning. Every image is resized to 120 x 120 pixels in colour, which is 14,400 pixels and 43,200 values going into the network. Transfer learning was used because the pre-trained network already recognises edges and fine lines, which is exactly what a crack is. Only the last layers were retrained on my two classes. The model was exported int8 quantised, which makes it smaller and faster to run in a browser at a small cost in precision.

Step 6: Testing the model

The trained model was scored on the held-back test images, which it had never seen during training. Test accuracy: **79%**. Accuracy alone is not the whole story here, because the two kinds of mistake do not cost the same - see section 3.

Step 7: Improving the model

Testing showed a clear pattern of failure: the model missed faint, low-contrast damage - hairline cracks, and shells that were shadowed or dirty - while catching obvious breaks easily. I added more examples of exactly that faint damage, photographed under different lighting, and retrained. I then ran the same hard images through the new version and compared the confidence scores, rather than only looking at the headline accuracy.

Step 8: Deploying the model

In Edge Impulse I used **Deployment - WebAssembly - Build**, then unzipped the download. I copied `edge-impulse-standalone.js`, `edge-impulse-standalone.wasm` and `run-impulse.js` next to my own `index.html`. The page calls `Module.init()`, reads `get_properties()` to discover the label names, the input size and the number of features, then packs each pixel of the uploaded image into a single integer as (red << 16) | (green << 8) | blue before calling the classifier. Because the page reads everything from the model, nothing about the classes is hard-coded. The `.wasm` file has to be served over http with the content type `application/wasm`, so the site is hosted on a real web host (`uzlabz.com`) rather than opened as a local file.

3. Evaluation of Results

Test accuracy: **79%**

Test accuracy on the unseen images: **79%**. For this system, though, the headline accuracy is less important than which way the errors fall. Calling a sound egg damaged wastes one egg; calling a cracked egg sound sends contaminated food onward. The two errors are not equal, and a single accuracy figure hides that completely.

It handled well: clear, well-lit photographs of a single egg where the damage is an obvious break or a visible dark line across the shell. It found hard: hairline cracks, damage in shadow, and dirty or speckled shells, where marks that are not damage look like damage.

This is why the 0.60 threshold exists: anything below it is reported as uncertain rather than as an answer. A high confidence still does not mean correct - the model can be confidently wrong on a shadow that looks like a crack - so the honest use of this system is as a first-pass filter that flags eggs for a person to check, not as a final decision.

4. Screenshots and Evidence

Each screenshot below is captioned with the input, the predicted label, the confidence and the time it was taken.

[paste screenshot here — a clear, easy example]

Caption: A correct result — input _____, predicted _____, confidence ____%, taken _____.

[paste screenshot here — a different class from the first]

Caption: A second correct result — input _____, predicted _____, confidence ____%, taken _____.

[paste screenshot here — one of the failures described in Step 7]

Caption: A hard or failed case, before improvement — input _____, predicted _____, confidence ____%, taken _____.

[paste screenshot here — the improvement, for direct comparison]

Caption: The same case after retraining — input _____, predicted _____, confidence ____%, taken _____.

5. Ethics and User Consideration

Privacy: the model runs locally in the browser through WebAssembly, so photographs of stock never leave the device and no image is uploaded or stored. That matters commercially as well as personally: a producer would not want photographs of their damaged stock sitting on somebody else's server.

Limitation: there are only two labels, so a dirty, speckled or oddly shaped but perfectly sound egg can still be pushed into 'damaged'. The model also judges only what is facing the camera - damage on the far side of the egg is invisible to it.

Bias: it was trained on mostly one shell colour, so brown or speckled eggs score worse than white ones. A producer with brown hens would get a worse product than one with white hens, purely because of my data. The fix is to count the images by shell colour and deliberately collect the missing ones.

Note on Tools and Honesty

The dataset, the training, the testing and the judgements in this report are my own work. I used an AI assistant to help with the layout and CSS of the web page that displays the model, and to check the spelling and grammar of this document. The model itself was trained by me in Edge Impulse, and every number quoted here is one I measured.