

Project Report

Sports Ball Identifier — an AI image classifier for five kinds of sports ball

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	Sports Ball Identifier
Tool used	Edge Impulse (WebAssembly export)
Date	2 September 2026

1. Explanation of the AI System and Objective

My system is an image classifier that looks at a photograph of a ball and decides which sport it belongs to: basketball, football, American football, cricket ball or bowling ball. I built it with Edge Impulse and deployed it as a web page: the user chooses an image, the page resizes it to 120 x 120 pixels, and the model returns the winning label plus a confidence score for all five classes.

I chose this problem because sports balls are a good test of what a model actually learns. Several of them are the same basic shape and differ mainly by surface pattern and colour, and both of those change with the manufacturer, the lighting and how worn the ball is - so fixed rules like 'orange means basketball' fail immediately. Learning the combination of shape, texture and pattern from examples is what makes it work. It would be useful for tagging sports photography automatically, or in a stock-checking app for a sports shop.

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 five labels: basketball, football, cricket ball, bowling ball and American football. There is no 'neither' class, which I return to in section 5. 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 was downloaded from Kaggle:

kaggle.com/datasets/samuelcortinhas/sports-balls-multiclass-image-classification. Licence: **CC0 1.0 Public Domain**, so it can be reused freely for coursework. The full dataset holds over 9,000 images across 15 sports, scraped from Google Images, manually reviewed to remove wrong labels and de-duplicated with a dHash algorithm by the dataset author; I used five of those fifteen classes. There are five labels: **basketball, football, cricket ball, bowling ball** and **American football**.

Step 4: Preparing the data

I removed images that showed more than one type of ball, and any that were too small or blurry to identify by eye. 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 MobileNetV2 already recognises shape, texture and pattern from a very large image collection, and only the final layers had to be retrained on my five classes. Training from scratch on a dataset this size would not work.

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: **85%**. That is measured on the 20% held back from training, so it is a score on images the model had never seen.

Step 7: Improving the model

Testing showed a clear pattern of failure: the model lost confidence where colour and shape overlapped between classes - for example a dark football against a bowling ball, or a worn cricket ball against a plain background. I added more varied training examples covering those overlaps 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: **85%**

Test accuracy: **85%** on the 20% of images held back from training. With five classes, random guessing would score about 20%, so 85% is a genuinely strong result - though it still means roughly one image in seven is wrong.

It handled well: a single ball filling most of the frame, in reasonable light, with its distinctive pattern visible - the panels of a football, the seam of a cricket ball, the finger holes of a bowling ball. It found hard: balls whose colour and shape overlap with another class, such as a dark football against a bowling ball, and worn or dirty balls whose surface pattern has faded.

The confidence score shows how strongly the image matched the winning class, and my page displays all five bars so a close call between two classes is visible. It is not proof of correctness: because there is no 'neither' class, the five scores always add to 1, so a photograph of a tennis ball - or of no ball at all - still comes back as a confident answer for one of my five. Those confident wrong answers are the real weakness, not the 15% error rate itself.

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 the photograph the user chooses is classified on their own device and is never uploaded, stored or logged.

Limitation: there are five labels and no 'neither' class, so the model is forced to assign every image to one of the five balls even when the image contains no ball at all. It also identifies only one ball per photograph.

Bias: the Kaggle dataset was collected by someone else, and I did not choose which brands, colours or conditions of ball went into it. If it is dominated by clean, professional-standard equipment photographed in studio light, then scuffed school equipment in a sports hall will score worse. The fix is to check the balance of the dataset before trusting it, and add my own photographs of ordinary worn equipment.

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.