

Project Report

Cat vs Dog Classifier — an AI image classifier that decides whether a photo shows a cat or a dog

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	Cat vs Dog Classifier
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 one photograph and decides whether it shows a cat or a dog. I built it with Edge Impulse and deployed it as a web page: the user picks an image file from their own device, the page shrinks it to 100 x 100 pixels, and the model returns the more likely of the two labels together with a confidence score for each.

I chose this problem because it is a task humans do instantly but cannot write rules for. There is no single feature that separates a cat from a dog: not colour, not size, not ear shape, not pose. That is exactly the situation where machine learning is the right tool rather than an if-statement, because the model can learn the combination of patterns from examples instead of being told what to look for. A working version of this is genuinely useful to a shelter or a pet-photo site that needs to sort thousands of incoming images without a person tagging each one.

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, cat and dog, and no third option, so every image it is given is forced into one of the two classes. 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 cat and dog photographs. Source and licence: . There are two labels: **cat** and **dog**. I checked that both labels had a similar number of images, because a class with far fewer examples is learned worse.

Step 4: Preparing the data

I removed images that were mislabelled, duplicated, or too blurry to identify by eye - on the principle that if I cannot tell what the picture shows, the model has no chance either. 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 100 x 100 pixels in colour, which is 10,000 pixels and 30,000 values going into the network. Transfer learning was used because MobileNetV2 has already learned general visual features such as edges, fur texture and shape from a very large image collection. Only the final layers are retrained on my data, which is the only practical option when my dataset is a few thousand images rather than a few million.

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: **80%**. That means roughly four out of every five unseen photographs were labelled correctly. I recorded the result rather than the impression, because the model always looks better on the images it was trained on.

Step 7: Improving the model

Testing showed a clear pattern of failure: the model was correct on clear, well-lit, front-on photographs but lost confidence on unusual breeds, on dark or blurry pictures, and on animals that were only partly in frame. I added more varied training examples covering those cases - different breeds, different backgrounds and 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: **80%**

A test accuracy of **80%** means about four in every five unseen photographs were labelled correctly, and roughly one in five was wrong. That is a usable result for a first model but clearly not good enough to trust unsupervised.

It handled well: clear, well-lit photographs of a single animal taken from the front or the side, filling most of the frame. It found hard: unusual breeds (particularly small dogs with cat-like faces and long-haired cats), dark or blurry photographs, animals partly out of frame, and pictures containing both a cat and a dog, where only one label can win.

The confidence score is the model's own estimate, not a guarantee. Because there are only two classes the two scores always add up to 1, so an image of something that is neither - a rabbit, or an empty sofa - still produces a confident-looking answer. A high

confidence therefore tells me the image looks like one class much more than the other; it does not tell me the model is right. The cases I care about most are the confident wrong ones, because those are the ones a user would never think to check.

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 is compiled to WebAssembly and runs inside the visitor's own browser. The photograph they choose is read into a canvas element, turned into numbers and classified on their machine; it is never uploaded to a server, never stored and never logged. Someone can classify a picture of their own pet without that picture leaving their device.

Limitation: the model has exactly two labels and no 'neither' class, so it is forced to answer cat or dog for any image at all. Give it a photograph of a horse and it will confidently pick one of the two. This is a design flaw in my class list rather than a training problem, and it is the first thing I would change.

Bias: the model can only be as fair as the breeds, coat colours and lighting conditions in my training set. If most of my cat photographs are of light-coloured short-haired cats indoors, then black cats photographed outdoors will be classified worse, and a user with a black cat gets a worse product than a user with a ginger one. The fix is to count the images per breed and coat colour and deliberately collect more of the under-represented ones, rather than simply adding more images overall.

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.