

Project Report

Chess Piece Classifier — an AI image classifier for the six standard chess pieces

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	Chess Piece 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 a photograph of a single chess piece and decides which of the six types it is: bishop, king, knight, pawn, queen or rook. A seventh label, **neither**, lets it reject anything that is not a chess piece at all. I built it with Edge Impulse and deployed it as a web page: the user chooses an image, the page resizes it to 105 x 105 pixels, and the model returns the winning label with a confidence score for each of the seven classes.

I chose this problem because chess pieces differ almost entirely by the shape of the top - a cross, a crown, a horse's head, a mitre, a round ball, a tower - while everything else about them varies freely: colour, height, material, style of set, and the angle they are photographed from. Writing fixed rules for 'has a horse's head' is not realistic, but learning that shape from examples is exactly what an image model does well. It is useful for anyone digitising a physical game, and it is a stepping stone towards reading a whole board from a photograph.

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 seven labels: bishop, king, knight, pawn, queen, rook and **neither**. The seventh label is deliberate - without it, every photograph would be forced to be a chess piece. 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 individual chess pieces. Source and licence: [https://www.kaggle.com/datasets/ashleythornley/chess-pieces](#). There are seven labels: **bishop, king, knight, pawn, queen, rook** and **neither**. I checked the counts per label, because a class with far fewer images is learned worse - and with six real classes that is easy to get wrong.

Step 4: Preparing the data

I removed blurry images, duplicates, and photographs where more than one piece was visible, since the model is only meant to judge one piece at a time. 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 105 x 105 pixels in colour, which is 11,025 pixels and 33,075 values going into the network. Transfer learning was used because the network already knows general shape and edge features from a very large image collection, and only the final layers need to be retrained on my seven classes. Training from scratch would need far more images than I have.

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: **76%**. I recorded the figure from the test set rather than the training set, because the training score always flatters the model.

Step 7: Improving the model

Testing showed a clear pattern of failure: the model confused pieces with similar outlines once the image was shrunk - bishop against pawn, and queen against king - because at 105 x 105 the difference is only a few pixels at the very top of the piece. I added photographs of different sets, materials and camera angles for exactly those pieces, 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: **76%**

Test accuracy on the unseen images: **76%**. With seven classes, random guessing would score about 14%, so that figure has to be judged against 14% and not against 50%.

It handled well: pieces photographed upright, alone, against a plain background, where the top of the piece is clearly visible. It found hard: bishop against pawn, and queen against king - the two pairs whose silhouettes are closest once the picture is reduced to 105 x 105 pixels. It also weakened on themed or ornamental sets, which is unsurprising given what it was trained on.

The confidence score is the model's estimate, not proof. The useful signal is not the top score on its own but the gap between the top two: when bishop reads 0.51 and pawn reads 0.44, the model is effectively saying it cannot tell, and my page shows every class bar so that this is visible instead of hidden behind one answer. A confident wrong answer is the dangerous case, and with pieces that genuinely look alike it does happen.

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: inference runs locally in the browser through WebAssembly. The photograph is processed on the user's own machine and no image is uploaded, stored or logged.

Limitation: the model reads one piece at a time and cannot interpret a whole board. It also has no idea about colour of side - it will not tell you whether the knight is white or black, only that it is a knight.

Bias: it was trained mostly on one style of set, so themed, ornamental or unusually shaped sets are misread. That is a real fairness problem for a user whose set does not look like mine. The fix is to photograph several genuinely different sets - wooden, plastic, glass, themed - and add them to the training data, rather than assuming one standard set represents all of them.

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.