Adidas & Nike Shoes Classification with Arduino Nano 33 BLE Sense

Jiani Che

Link to github repo with project work:https://github.com/wwCARww/shoes-classification-CASA0018

Link to Edge Impulse projects: https://studio.edgeimpulse.com/public/200120/latest

Introduction

Adidas and Nike shoe products have been popular with consumers for many years, however, in recent years consumers have faced the risk of potentially buying fake shoes. Identifying fake shoes with the naked eye is still risky and costly, so could automatic identification with high accuracy be accomplished if deep learning and AI technology were used? Before identifying the authenticity of shoes, we need to identify the brand and type of the shoes. There have been many attempts to use datasets of shoe images sourced from the web to train classification models for shoe brands and types[1]. But few projects have applied them to sensors for real-time recognition. Then the aim of our project is to train a classification model by using neural network based embedded AI to classify the brand of the shoes in the images. In this project, we tried two different sources of data as training sets for separate experiments:1) existent shoes images downloaded from Google images; 2) real-world shoes images taken by an Arduino Sense camera. The experimental results show that the two datasets do not have the same predictive effect. The final model using the Arduino camera data as the training set achieved a maximum of 95% accuracy in correctly identifying the brand of shoes as either Nike or Adidas.

Research Question

Can we build an embedded artificial intelligence project based on neural networks to identify whether the brand of shoes is Nike or Adidas?

Application Overview

The application is built using several components, including Edge Impulse, an Arduino Nano 33 BLE Sense board with a camera, and the Arduino IDE.

The application consists of three main building blocks:

1) Data collection, which involves capturing images of Nike and Adidas shoes manually using the Arduino Nano onboard camera. These images are then processed and pre-processed using Edge Impulse and stored as dataset;

2) Model training, involves creating a machine learning model that can accurately distinguish between Nike and Adidas shoes. This is done by conducting multiple experiments with different model architectures, learning blocks and other advanced training settings;

3) Model deployment, which involves running the best-trained model on new, unseen data to make predictions. In our application, the model is deployed on the Arduino Nano 33 BLE Sense board, which allows it to make real-time predictions on images captured by the onboard camera.

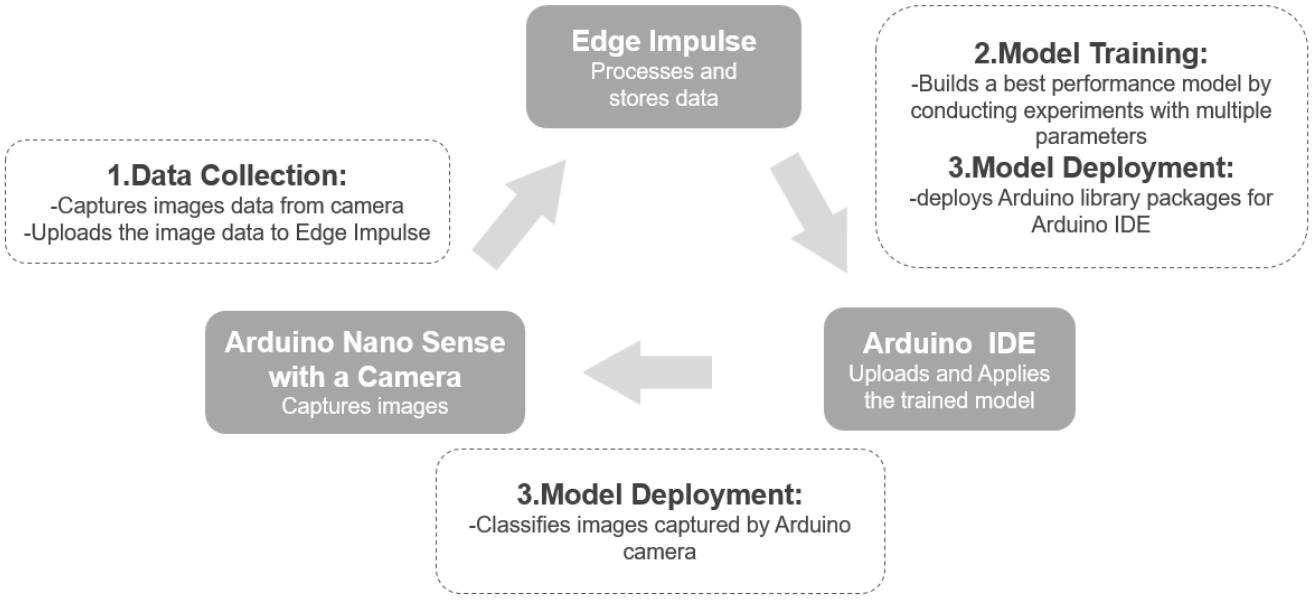

Figure1: Application structure of the building blocks

All three building blocks are connected through the use of Edge Impulse, which acts as a central hub for data collection, model training, and deployment. The Arduino IDE is used to upload the trained model to the Arduino Nano 33 BLE Sense board, enabling it to perform inference on new data in real time.

Data

Dataset

As mentioned in the introduction, I used 2 shoes images datasets from different sources:

- Dataset 1: existent Adidas and Nike shoes images downloaded from Google images. To create dataset 1, I used 2 image datasets from Kaggle which downloaded the shoe image from Google[2]. To improve the accuracy of the training model, I only used images with a white background. A total of 400 images were collected, with a 50/50 split between the two brands. And dataset 1 is used as one of the training datasets.

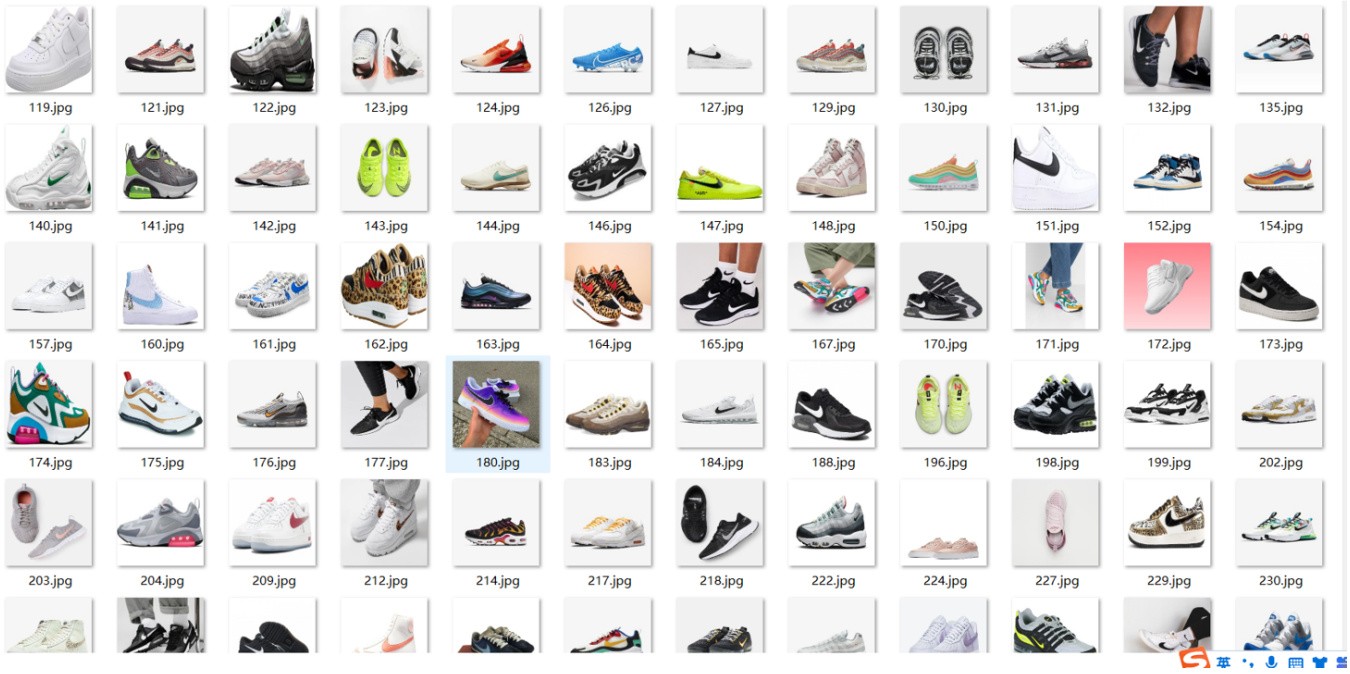

Figure2: Overview of dataset 1 image data

- Dataset 2: shoes images taken by an Arduino Sense onboard camera: Dataset 2 consists of 2 parts: 1) I took 80 images of 2 pairs of Nike and Adidas shoes in different angles as another training dataset; 2) 20 images of another 4 pairs of Nike and Adidas shoes were taken as our test dataset. All images were taken under almost the same light and background conditions.

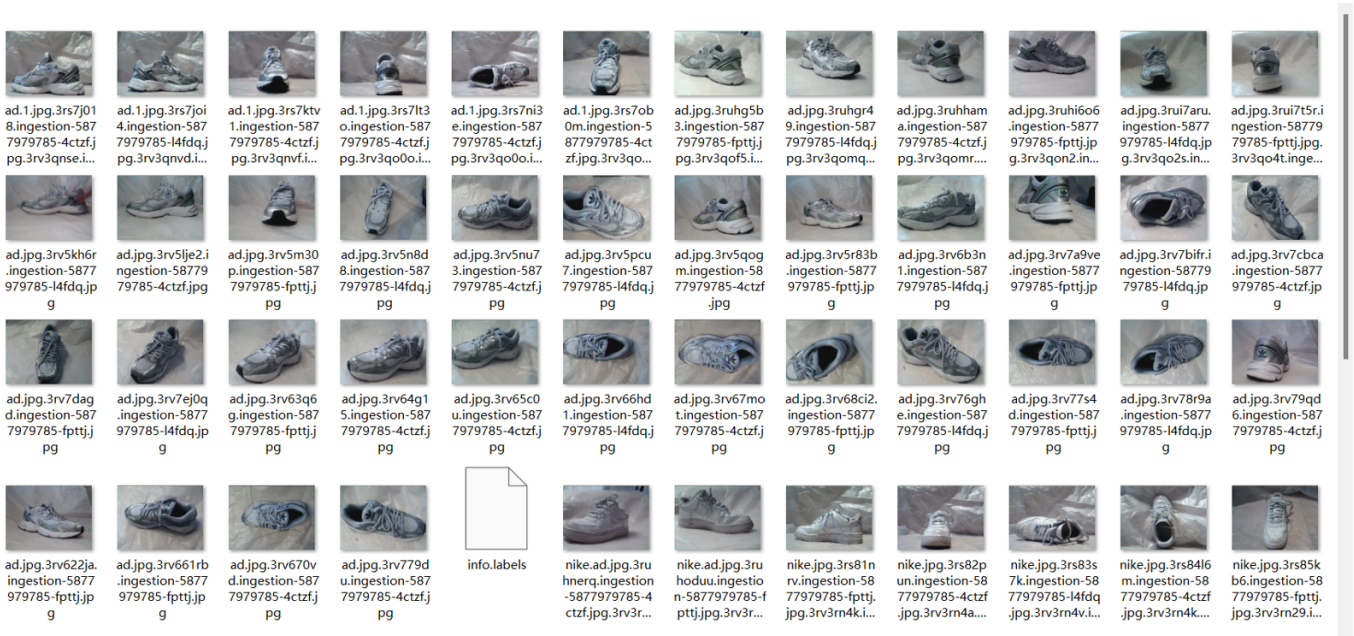

Figure3: Overview of dataset 2 image data

Figure4: Overview of dataset 2 data source

The following experiments were divided into 2 parts:

	Training	Testing	Total
Nike	200 (dataset 1)	50 (dataset 2)	250
Adidas	200 (dataset 1)	50 (dataset 2)	250
Total	400 (dataset 1)	100 (dataset 2)	500

Table1: Datasets description of experiment 1

	Training	Testing	Total
Nike	40 (dataset 2-1)	10 (dataset 2-2)	50
Adidas	40 (dataset 2-1)	10 (dataset 2-2)	50
Total	80 (dataset 2-1)	20 (dataset 2-2)	100

Table2: Datasets description of experiment 2

Data Processing

After uploading the raw image data, Edge Impulse then processed the images by resizing and changing the color depth. In this project, the image data was resized into 96*96 and then converted into grayscale, which has the best performance in the model.

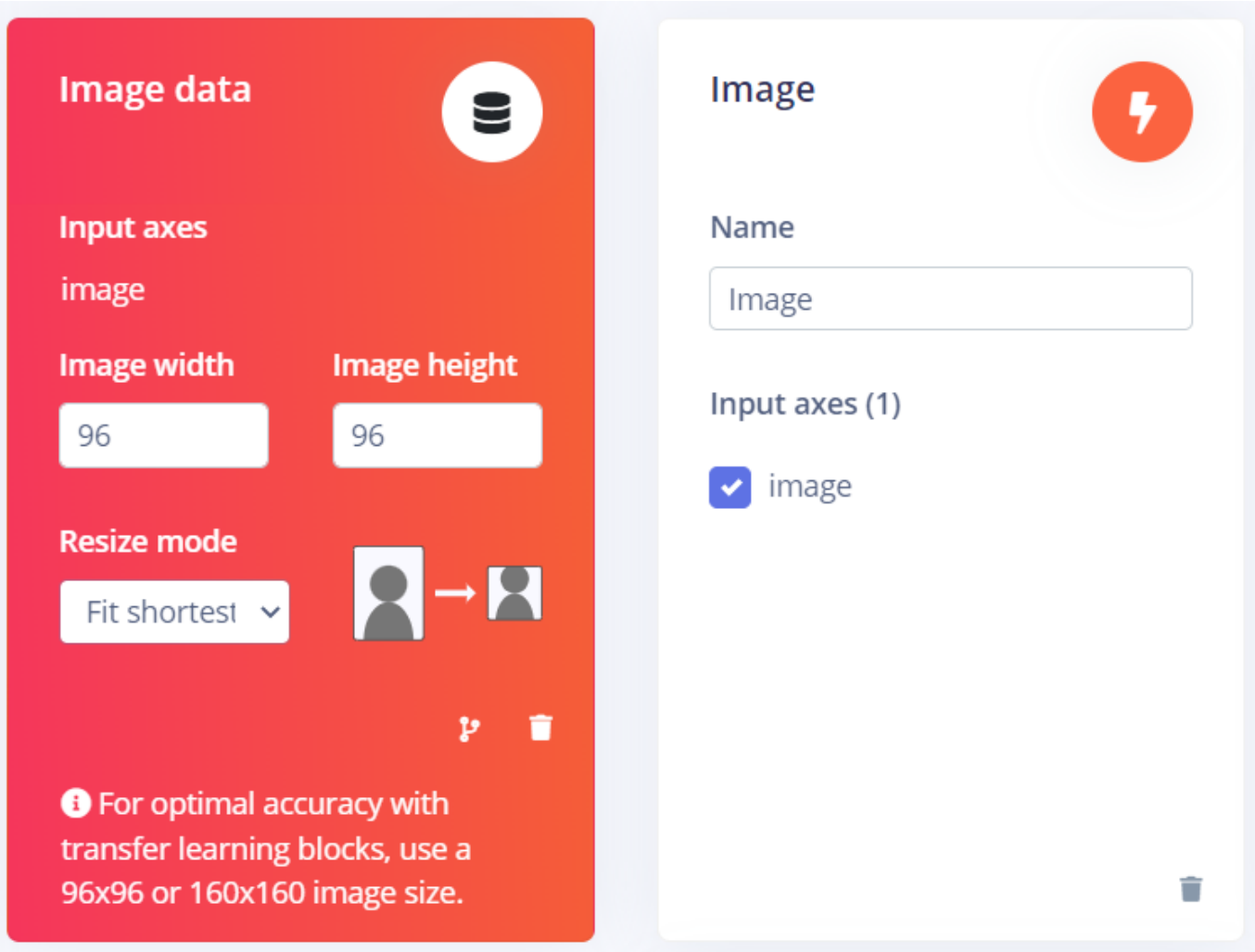

Figure5: Edge Impulse user interface of image data processing

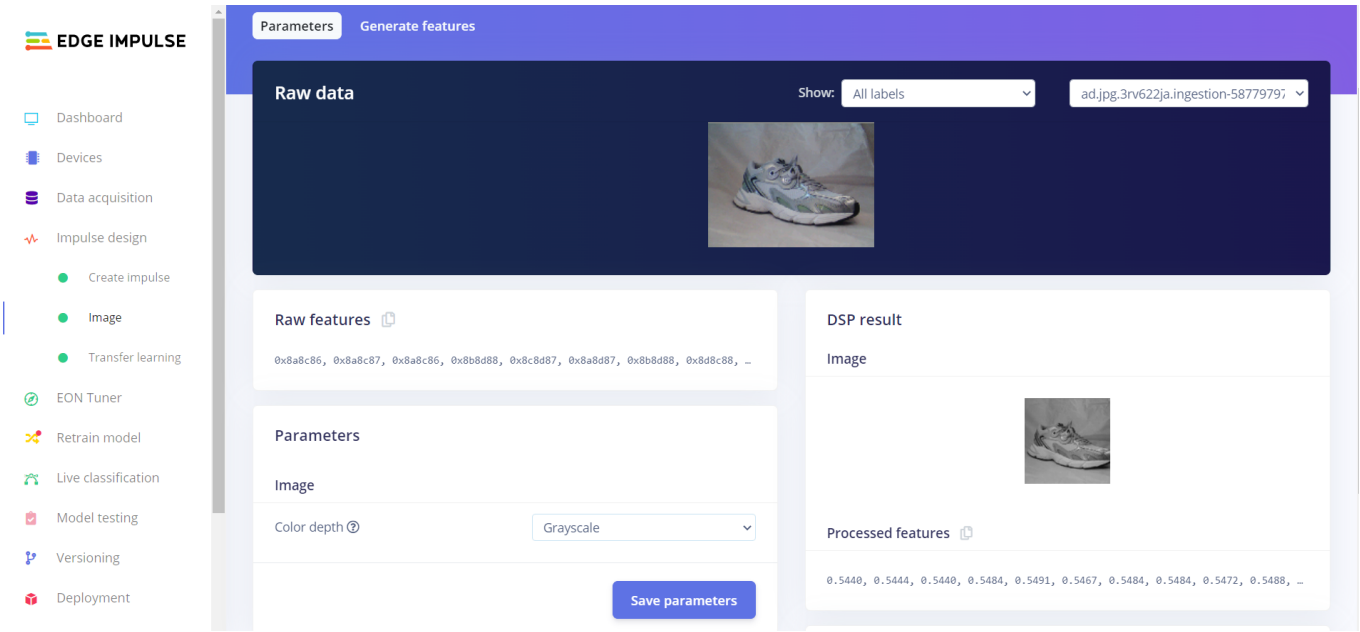

Figure6： Edge Impulse user interface of image data color depth

Model

After data collection, a processing block and a learning block were required for feature generation and machine learning. For image detection projects, Edge Impulse recommended image transfer learning as the learning block. Though different learning blocks were applied as well, after a series of tests, image transfer learning and dataset 2 were chosen as our final training/test dataset for better performance.

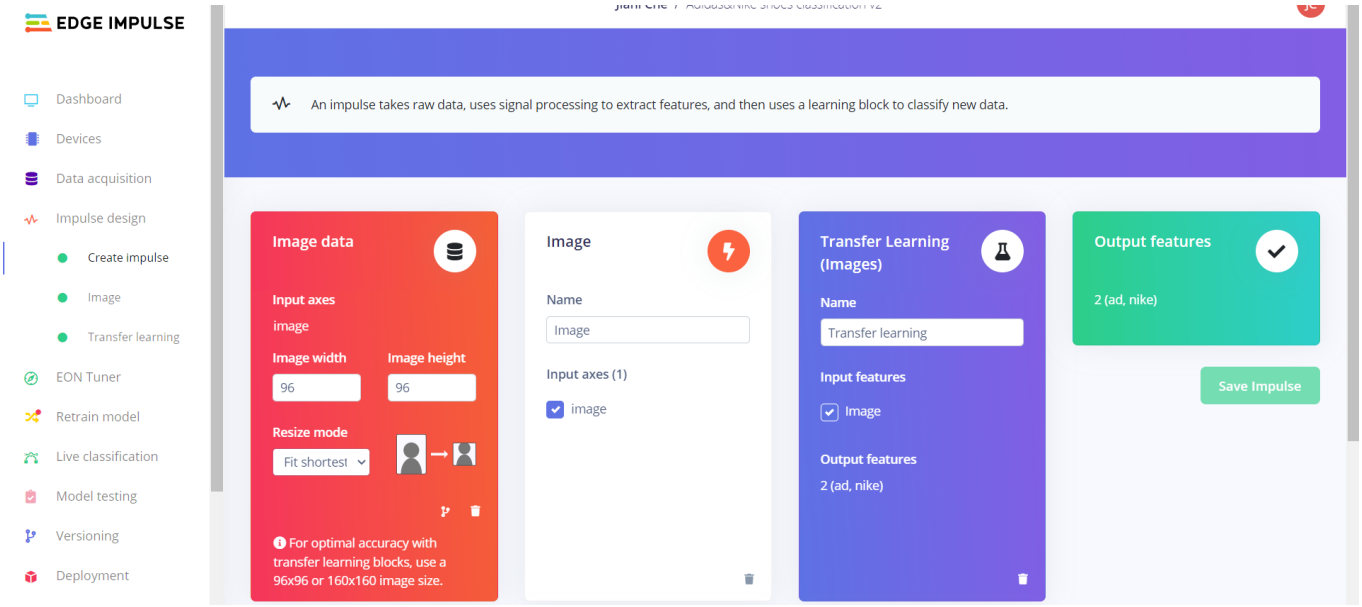

Figure7: Edge Impulse user interface of Impulse design

Edge Impulse allows users to adjust the model in different ways, including a simplified visual mode, Kera(expert) mode and editing Python files locally. To build a better performance model, I modified multiple training settings and model architectures in 35 experiments. Among all experiments, the following settings and architecture have the best performance.

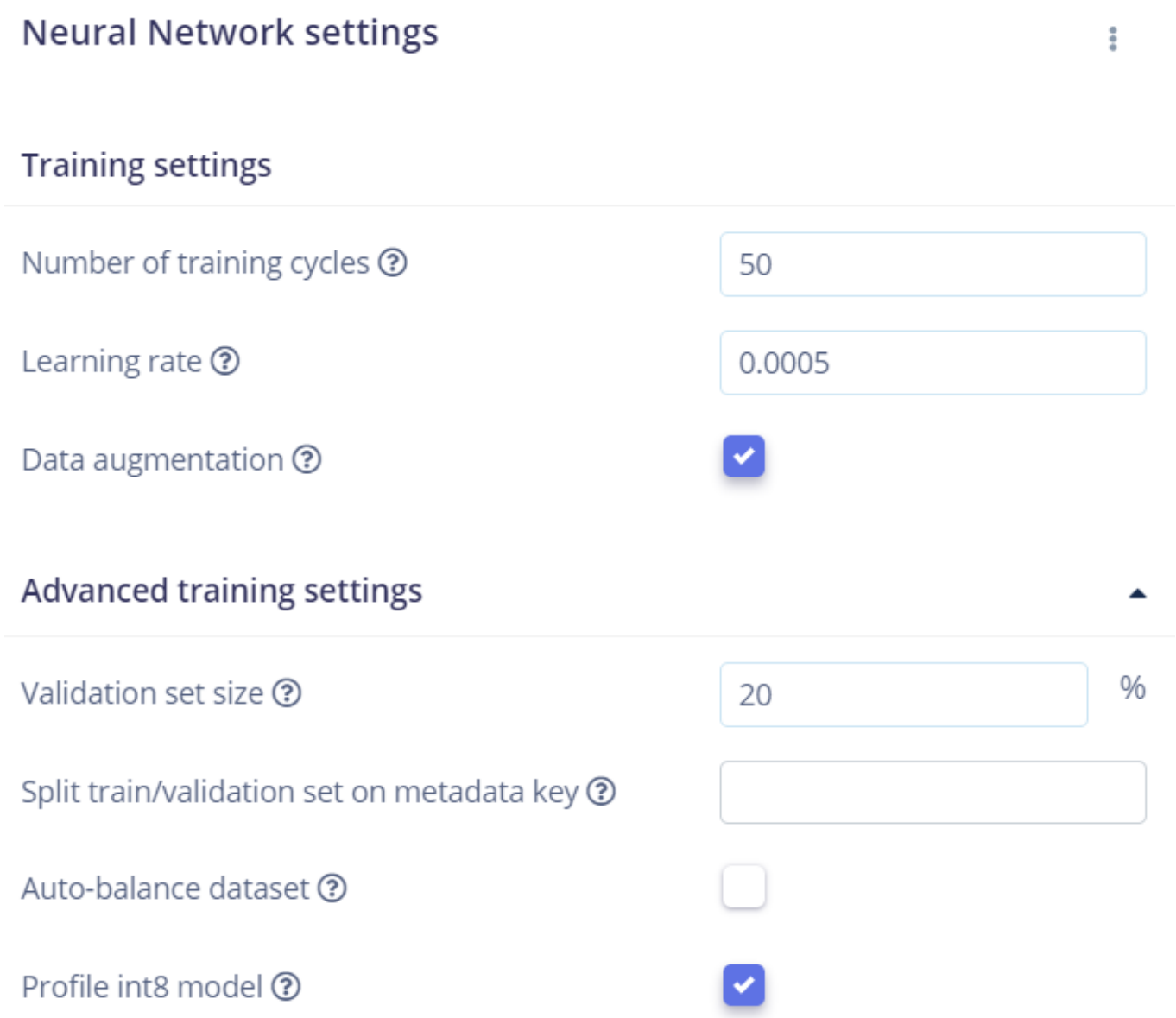

Figure8: Shoes Detection Model Neural Network Architecture and settings in the Edge Impulse user interface

The key parameters settings are shown below:

Key Parameters	**Final Settings**
Model	Transfer Learning MobileNetV2 0.35
Dense Layer	16 neurons
Drop out Layer Rate	0.1
Number of training cycles	50 Epochs
Validation Set Size	20%

Table3: The key parameters settings of the final model

Experiments

What experiments did you run to test your project? What parameters did you change? How did you measure performance? Did you write any scripts to evaluate performance? Did you use any tools to evaluate performance? Do you have graphs of results? Out of a total of 35 experiments, 13 experiments were

conducted on datasets sourced from the web and 22 experiments were conducted on datasets taken by sensor cameras. And 11 parameters were selected for different combinations in the experiments.

#	Parameters	Descriptions	Experiment Range
1	Training data(n)	Two training datasets from different sources	80, 216/400
2	Image Resize	Resize all images to equal dimensions before training	96*96, 120*120, 160*160
3	Epochs	How many times the model will run to train	10, 20, 30, 50
4	Learning Rate	How fast the neural network learns	0.0005, 0.001
5	Validation(%)	The percentage samples from training set held from validation	10, 20
6	Data Augmentation	Randomly transform data during running	yes or no
7	Color Depth	Different color spaces used to represent digital images	RGB or Grayscale
8	Neural Network Architecture	Use a present model recommended by Edge impulse or defining the model myself by changing the layers	1) MobilenetV2 2) 2 2Dconv,drop out and flatten 3)Reshape 5 columns, two 2Dconv,drop out and flatten 4)1 Dense layer, drop out and flatten
9	Dropout Rate	The percentage of randomly shout down neruons at each training step	0.1, 0.25,0.45
10	Neurons(n)	The numbers of nerurons in each layer/model	8, 16, 256
11	Learning Block	Different types of pre-defined learning blocks	Image Transfer Learning, Classification

Table4: Parameters changed during the experiments

As mentioned before, the experiments were divided into 2 parts:

- Training data: dataset 1(400 images)/Test Data: dataset 2 (100 images)

The experimental results show that the maximum accuracy of the machine learning model was only 64.36% when using image data from the network as the training set. Although the accuracy improved from 20.83% to 64.36% by adjusting the combination of parameters, the loss values remained high, which significantly reduces the usability of the deployment.

#	Training data(n)	Image Size	Epochs	Dropout Rate	Learning Rate	Validation(%)	Data Augmentation	Learning Block	Color Depth	Neurons(n)	Neural network architecture	Validation Accuracy	Testing Loss	Testing Accuracy(%)
1	216	96*96	20	0.1	0.0005	20	no	Image Transfer Learning	RGB	8	MobileNetV2 96x96 0.35	75	0.53	20.83
2	216	240*240	20	0.1	0.0005	20	yes	Image Transfer Learning	RGB	8	MobileNetV2 96x96 0.35	75	0.45	41.67
3	430	120*120	20	0.1	0.0005	20	yes	Image Transfer Learning	RGB	8	MobileNetV2 96x96 0.35	75.6	0.63	41.3
4	430	240*240	20	0.1	0.0005	20	yes	Image Transfer Learning	RGB	8	MobileNetV2 96x96 0.35	73.3	0.59	32.61
5	430	96*96	20	0.1	0.0005	20	yes	Image Transfer Learning	RGB	8	MobileNetV2 96x96 0.35	70.9	0.65	32.61
6	430	96*96	20	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	8	MobileNetV2 96x96 0.35	81.4	0.39	34.78
7	430	160*160	20	0.1	0.0005	20	yes	Image Transfer Learning	RGB	8	MobileNetV2 160x160 0.35	82.6	0.91	47.83
8	430	96*96	30	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	8	MobileNetV2 96x96 0.35	83.7	0.38	36.96
9	430	96*96	50	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	8	MobileNetV2 96x96 0.36	86	0.37	28.26
10	430	96*96	50	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	16	MobileNetV2 96x96 0.37	86	0.35	50
11	430	96*96	50	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	256	MobileNetV2 96x96 0.38	88.4	0.29	34.78
12	430	96*96	50	0.1	0.0005	20	yes	Image Transfer Learning	Grayscale	16	MobileNetV2 96x96 0.39	81.4	0.36	64.36
13	430	96*96	50	0.45	0.0005	20	yes	Classification	Grayscale	-	2D conv / pool layer	70.9	0.52	54.46

Table5: Experiments records of experiment 1

- Training data: dataset 2-1 (80 images)/Test Data: dataset 2-2 (20 images)

Experiments have shown that when using images sourced from the sensor as the training set, the accuracy of the model improves from 70% to a maximum of 95% after adjusting the combination of parameters.

Two learning blocks, image transfer learning and Classification, were applied in the experiments and the models based on them both performed well, and image transfer learning was chosen for the final model, which has a higher testing accuracy.

In addition, experiments have shown that models with image transfer learning blocks performed better with grayscale image data, however, models with classification blocks performed better with RGB image data.

#	Training data(n)	Color Depth	Learning Block	Features	Dropout Rate	Epochs	Learning Rate	Validation (%)	Nerurons (n)	Pre-trained Model	Validation Accuracy	Loss	Testing Accuracy(%)
1	80	Grayscale	Image Transfer Learning	9216	0.1	30	0.0005	20	8	MobileNetV2 96x96 0.35	81.3	0.39	70
2	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.0005	20	8	MobileNetV2 96x96 0.35	81.3	0.3	75
3	80	RGB	Image Transfer Learning	27648	0.1	50	0.0005	20	8	MobileNetV2 96x96 0.35	62.5	0.52	75
4	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.0005	20	8	MobileNetV2 96x96 0.35	81.3	0.26	70
5	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.0005	20	16	MobileNetV2 96x96 0.35	81.3	0.33	90
6	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.0005	10	16	MobileNetV2 96x96 0.35	87.5	0.48	70
7	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.001	20	16	MobileNetV2 96x96 0.35	75	0.42	90
8	80	Grayscale	Image Transfer Learning	9216	0.25	50	0.0005	20	16	MobileNetV2 96x96 0.35	75	0.49	85
9	80	Grayscale	Image Transfer Learning	9216	0.1	50	0.0005	20	16	MobileNetV2 96x96 0.35	87.5	0.19	95
11	80	Grayscale	Classification	9216	0.25	10	0.0005	20	-	two 2Dconv,drop out and flatten	62.5	0.66	25
12	80	Grayscale	Classification	9216	0.25	30	0.0005	20	-	two 2Dconv,drop out and flatten	81.3	0.44	65
13	80	Grayscale	Classification	9216	0.25	50	0.0005	20	-	two 2Dconv,drop out and flatten	87.5	0.3	70
14	80	Grayscale	Classification	9216	0.25	100	0.0005	20	-	two 2Dconv,drop out and flatten	87.5	0.24	75
15	80	Grayscale	Classification	9216	0.25	150	0.0005	20	-	two 2Dconv,drop out and flatten	87.5	0.23	75
16	80	Grayscale	Classification	9216	0.1	100	0.0005	20	-	two 2Dconv,drop out and flatten	87.5	0.23	70
17	80	Grayscale	Classification	9216	0.45	100	0.0005	20	-	two 2Dconv,drop out and flatten	87.5	0.21	80
18	80	RGB	Classification	27648	0.45	100	0.0005	20	-	two 2Dconv,drop out and flatten	93.8	0.15	90
19	80	RGB	Classification	27648	0.45	150	0.0005	20	-	two 2Dconv,drop out and flatten	93.8	0.11	90
20	80	RGB	Classification	27648	0.45	150	0.0005	20	-	Reshape 5 columns, two 2Dconv,drop out and flatten	failed	failed	failed
21	80	RGB	Classification	27648	0.45	150	0.0005	20	16	1 Dense layer, drop out and flatten	75	0.42	80
22	80	RGB	Classification	27648	0.45	150	0.0005	20	8,16	2 Dense layer, drop out and flatten	81.3	0.39	85

Table6: Experiments records of experiment 2

Results and Observations

Results

As mentioned in the experiment part, the accuracy of the model which used the image data from the network as the training set is not satisfactory. The model using the image data taken by the Arduino onboard camera as the training set was selected as the final model to be uploaded to the Arduino Nano sensor. The final model was obtained with a maximum testing accuracy of 95% and a validation accuracy of 87.5%.

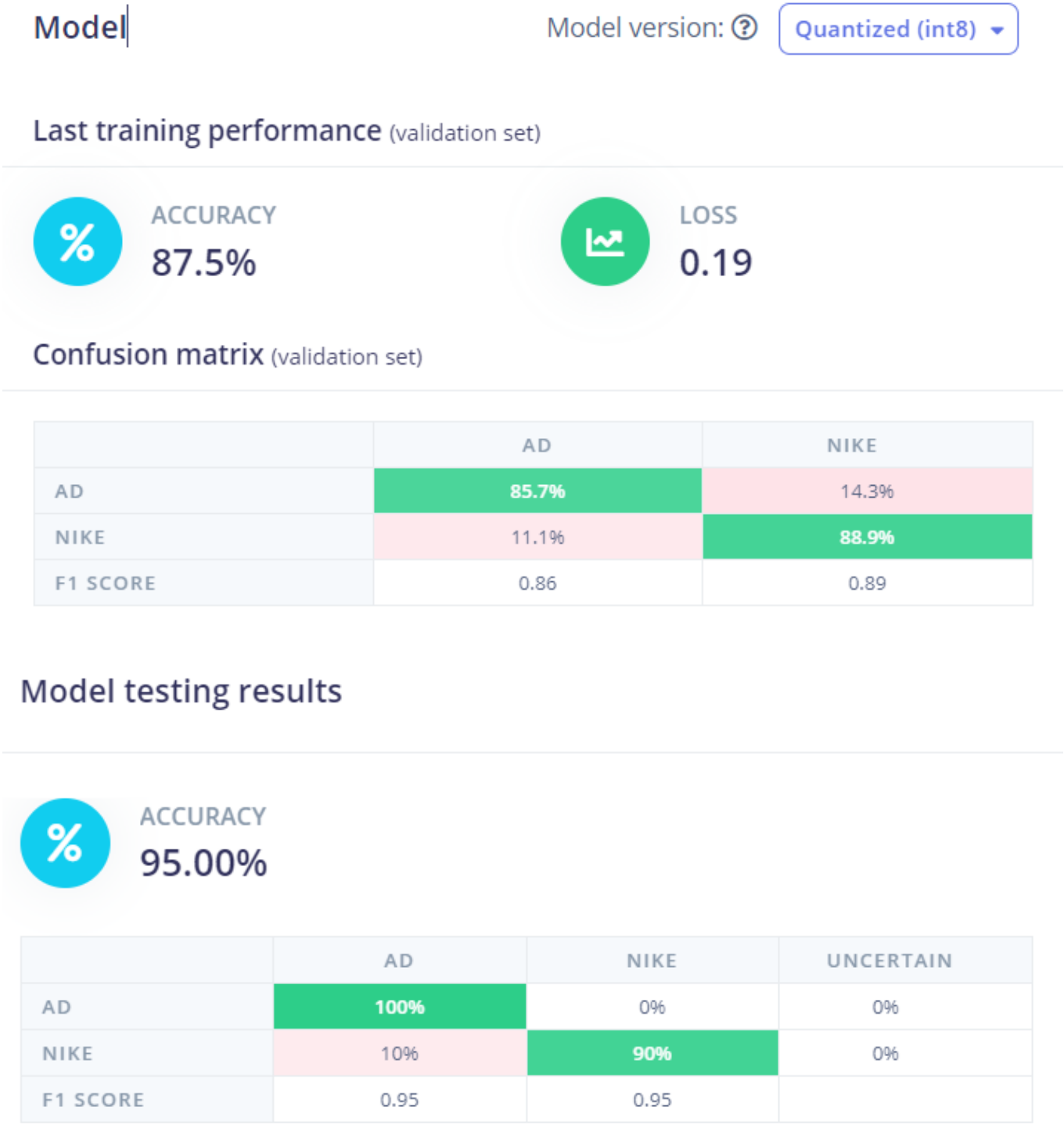

Figure9 & 10: Validation accuracy and Testing Accuracy of the final model

Observations

- Models performed better when using real-world images taken by Arduino Camera:

The images from the web may have different lighting conditions, angles, backgrounds, and resolutions compared to the images taken from the sensor camera. This can lead to differences in the features that the model learns to identify, as well as differences in the noise and variations in the images. In addition, the images from the sensor camera are likely to be more similar to the images that the model will encounter in real-world scenarios, as they are taken from the same type of camera that the model will be deployed on. This can help the model better generalize and make more accurate predictions on new, unseen data.

- Models with image transfer learning blocks performed better with grayscale image data, however, models with classification blocks performed better with RGB image data:

Grayscale and RGB are two different color spaces used to represent digital images. Grayscale images have a single channel that represents the intensity of the image, whereas RGB images have three channels that represent the intensity of red, green, and blue colors in the image[2]. In transfer learning, the pre-trained model is used to extract high-level features from the input image. Grayscale images have a lower dimensionality and therefore may require fewer computational resources for feature extraction, allowing the model to learn more efficiently. Additionally, the lack of color information may help the model focus on the distinctive shapes and textures of the shoe, which can be more important for brand detection. In classification, on the other hand, color information can be a crucial feature for distinguishing between Nike and Adidas shoes, as both brands have distinct color schemes that are often associated with their products. Using RGB color channels may provide the model with more information about these color differences, leading to better classification performance.

Reflections

Although the model has a theoretical accuracy of 95%, in practice the deployment accuracy is influenced by a number of factors.

- Angle, light and background

To improve the accuracy of the model when building the dataset, I controlled the lighting and background conditions when the volume was taken so that the majority of the data was in the same light and against a solid color background, which resulted in the Arduino sensor being susceptible to failure when performing real-time classification due to different lighting and backgrounds. Therefore, in order to improve the usability of the deployment in the future, it is necessary to increase the number of data under different conditions.

- Pixels from different cameras

Due to the low pixel count of the Arduino onboard camera, the results of real-time classification are also affected when using a higher pixel count camera such as a mobile phone or computer. Therefore it is necessary to add image data with different pixels, or to keep the training set image data and the camera conditions consistent at the time of application.

- The diversity of shoes

The appearance of Nike and Adidas shoes varies greatly between types and seasons. The fact that I only used my own shoes in the experiment means that my subjective aesthetic also influences the model training results. Therefore, more shoes from different collections and types need to be added to the dataset.

In the end, detecting the brands of shoes is just a basic step before identifying the authenticity of shoes. There is still a long way to go before it can be implemented into real-life applications. But Edge Impulse offers a low-cost, fast and easy platform and approach for we all.

Bibliography

Data Source

- https://www.kaggle.com/datasets/die9origephit/nike-adidas-and-converse-imaged
- https://www.kaggle.com/datasets/ifeanyinneji/nike-adidas-shoes-for-image-classification-dataset

Reference

1. Rukshan Pramoditha, 2021. How RGB and Grayscale Images Are Represented in NumPy Arrays. [online] Available at:https://towardsdatascience.com/exploring-the-mnist-digits-dataset-7ff62631766a[Accessed 24 April 2023].
2. kaggle.com, 2022. Shoes Classification Dataset | 13k Images |. [online] Available at:https://www.kaggle.com/datasets/utkarshsaxenadn/shoes-classification-dataset-13k-images[Accessed 24 April 2023].

Declaration of Authorship

Jiani Che, AUTHORS NAME HERE, 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.

Jiani Che

24th April 2023