FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
BRNO UNIVERSITY OF TECHNOLOGY

Digital Electronics (MPA-DIE 25/26Z)

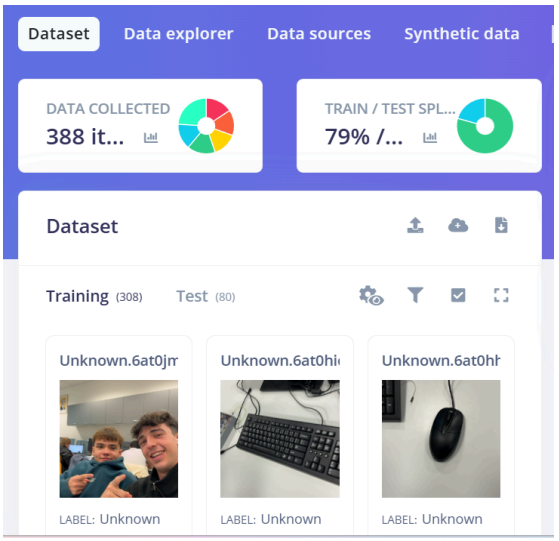
Index:

Project Report: Coin Recognition with Edge Impulse

1. Introduction

Before starting this project, we completed a tutorial to familiarize ourselves with Edge Impulse and understand how to work with image recognition models. We wanted to choose a topic that could have practical applications in everyday life, so we decided on recognizing different coins.
For the project, we collected approximately 50 images of each coin type and around 70 images of random objects to help the model distinguish coins from non-coin items. To capture the images, we used a mobile phone, and the first step of the project was to connect the mobile device to Edge Impulse using a QR code. Once connected, we captured the images directly through the mobile camera.

After capturing the images, the dataset was split so that 80% of the photos went to the training set and 20% to the test set.
During the development, we encountered some setbacks that initially resulted in the model achieving only about 65% accuracy. The main issue was related to the transfer learning configuration settings, which were not optimized for learning rate, number of epochs, and layer adjustments. By identifying and correcting these configuration problems, we were able to significantly improve the model's performance, ultimately reaching a final accuracy of 90%. The detailed explanation of these challenges and the steps taken to resolve them will be presented later in this document.



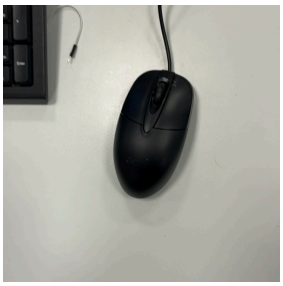
Dataset collecting and automatic separation of the photos

2. Image Capture and Preparation

2.1. Initial Data Collection

Photographs of each of the coins were captured in different positions, distances, and backgrounds. An attempt was made to obtain a relatively balanced quantity of images for each class, (more or less 50 pictures for each coin and 70 for random items).

Attached here is a photo of each item.



Random item

2kc

5kc

10kc

20kc

50kc

2.2 Data labeling and Preprocessing:

Following the initial collection of raw images of the dataset, we had to start the label phase. This process is essential because it is how we teach the supervised learning model to distinguish between different categories of all the coins that we have.

We had to assign a unique label to every single photograph. For instance, every image containing the 2KC coin was tagged with the '2KC' label, every 50KC coin image with the '50KC' label, and so on with all the types of coins that we had. The purpose of this was to assign a specific label to each image. This allowed the EfficientNet model to learn the unique visual features associated with each label and begin to differentiate the various coin classes. Without this initial labeling step, the model could not learn to identify one coin from another.

2.3 Initial Problems Detected

During the dataset review, we observed:

- **Blurriness in Images:**
  Some images were blurry or out of the focus of the camera, this would suppose that the model would have learned with low-quality and noisy features, reducing its ability to learn the fine distinct details like the edges or the reliefs that would differentiate all the classes of the coins. Making a model that wouldn't be reliable and wouldn't work correctly in a real-world scenario.

- **Poor lighting:**
  Certain photos had too much or poor lighting conditions, affecting the color features of the model and making it to be very susceptible to changes in a real-world lightning, drastically reducing the accuracy of the model.

- **Limited variation:**
  There was a limited variation in angles and backgrounds which could affect the model in its accuracy.

Example of poor lightning

Example of good lightning and not blurry

2.4. First Correction: Data cleaning and refinement.

To ensure that the system is trustworthy and had the best accuracy with the limitation of having so few images for every label, we performed a critical data cleaning and refinement step to ensure the integrity of the training process.

- Eliminate every blurry, mislabeled or unrecognizable photo were removed from the dataset.

- We deliberately preserved images that were technically difficult but correctly labeled to ensure that the model would work in difficult conditions and situations that the coin weren't perfectly placed.

- As we removed low-quality images, we added new more accurate images to ensure there was no missing data, and we had the number of data needed for every label.

3. First Model Training (Results near 65%)

With all the process of the data acquisition correctly done we can pass to the creation of the blueprint for our entire machine learning pipeline, now we will define how the input data is transformed, processed and ultimately turned into a coin class prediction. And then we will train and see how the model works in a real-case scenario.

3.1 Creating the impulse:

The first step for making the blueprint for our machine learning project, we will have to create the impulse design. We had to combine different blocks of Edge Impulse to make them work together and make our image classification project.

3.1.1 Block 1 - Image data (Acquisition and Preprocessing Block):

This is the entry point and the standardization for our data, its function is to take the raw input of the images and adapt it to the exact format required to be processed by all the blueprint of the machine learning project.

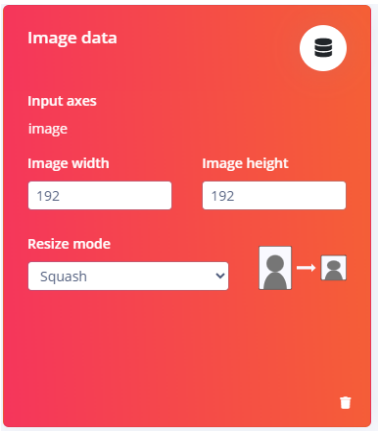


Image block of combination

Specific actions:

- **Input axes:** it is defined the input as the image
- **Dimensions:** it ensures all images are uniform size for the model. We put the width and the high of 192 pixels because it's the optimal size for ensuring the precision and needed.
- **Resize mode (Squash):** stretches the image to fit the 192 x 192 frame, a process that ensures dimensional consistency.

3.1.2 Block 2 - Image (Feature Processing Block):

This block prepares the standarization image for the machine learning model, its function is to convert the image pixels into numerical representation that the Convolutional Neural Network (CNN) can process. It's a necessary intermediary step before the deep learning stage.

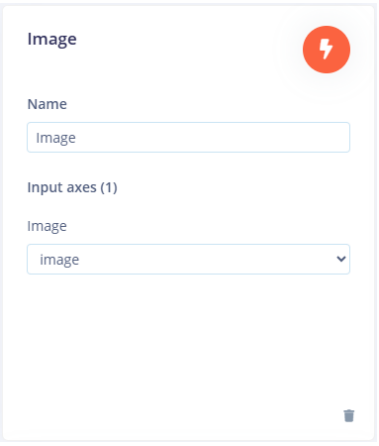
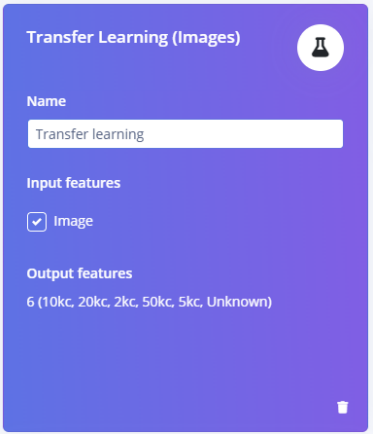


Image feature block

3.1.3 Transfer learning block:

This is the core of our project, it's the learning block where the model will understand how to differentiate the characteristics of the different classes of the coins. In this step, it generates the confidence scores for each class, indicating how probable it is that the input image belongs to each coin denomination.

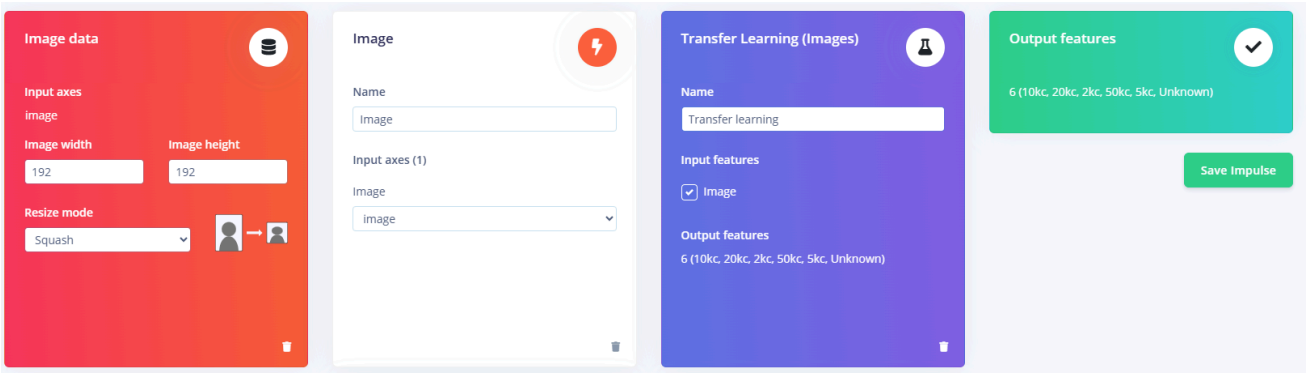


Transfer learning block

3.1.4 Output features:

This block defines the final layer and the response of the entire combination of the blueprint of the machine learning project. It is here where the model classifies the input into 6 different categories, 2kc, 5kc, 10kc, 20kc, 50kc and Unknown.

Output features block



Combination of all the blocks of the blueprint of the machine learning project

3.2 Image parameters:

Here we had to decide if we want the model to take the input raw data with the color depth of grayscale or with RGB, and in our case, because we have the color of the coin as a very important feature of the image classification, so we had to use the RGB color depth.

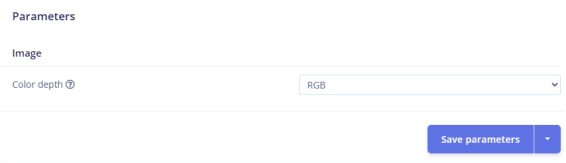


Image parameters of the raw data

3.3 Architectural configuration:

With this first model, we used the default architecture that Edge Impulse proportioned to us, that us the MobileNetV2. Which is a fast and lightweight architecture, but with limited capacity for subtle detail distinction, that could affect the accuracy of the model.

- **Epochs (20):**

  We used 20 epochs because it was the optimal for the CPU mode to make it able to train the model within a short time period, like 15 minutes. But the thing is that you want the higher epochs as possible because the model will learn all subtle detail better and not only learn the superficial features of the classes.

- **Learning rate (0.001):**

  With 0.001 of LR we had the model to arrive for a solution quicker as well, but with similar types of coin a higher LR generates instability and fluctiations and can’t find the best optimized solution.

- **Batch size (16):**

  With this batch size we ensured that the model could train on a wide variety of machines, but the problem was that a small batch size produces noisy and unstable gradients. This creates random oscillations in accuracy, making it difficult for the model to follow a smooth, consistent learning trajectory.

- **Freeze layers (100%):**

  By freezing nearly all layers of the pre-trained architecture (MobileNetV2), only the last few dense layers are trained, which is significantly faster.

- **Data augmentation (Disabled):**

  It is done to accelerate iteration speed. However, the major problem is that the model can overfit the images to the exact conditions under which the initial photographs were, taken like the same lighting and background. This severely compromises the model's generalization ability, leading to complete failure in any real-world deployment.

4. Results of the 65% model

The accuracy reached was approximately 65%, with notable confusion between different classes of the coins. With this final accuracy and a Loss of 1.17, we had a model that couldn't reach the required standard of reliability and a trustworthy system for a real-world case scenario, with the minimum accuracy of 80-85% of acuracy.

Matriz de confusión (conjunto de validación)

	10KC	20KC	2KC	50KC	5KC	DESCON
10KC	54,5%	9,1%	0%	18,2%	0%	18,2%
20KC	0%	100%	0%	0%	0%	0%
2KC	63,6%	9,1%	0%	0%	18,2%	9,1%
50KC	0%	7,7%	0%	61,5%	7,7%	23,1%
5KC	0%	14,3%	0%	0%	85,7%	0%
DESCONC	0%	0%	0%	0%	0%	100%
PUNTUAC	0,50	0,75	0.00	0,70	0,75	0,83

Confusion matrix of the model

4.1 Overall performance and reliability:

A 65% accuracy means that we have a error rate of 35%, making the model unreliable. The high loss of 1.17 indicates that the model failed to converge properly during the limited training cycles and the high initial Learning Rate. We can see a notable confusion between the different classes

The matrix shows very unbalanced classes and confusion between similar coins (10KC ↔ 2KC, 2KC being very confused).

4.2 Best performing class (20KC and Unknown):

The system's most reliable classification is the 20KC coin, achieving a perfect 100% accuracy in the confusion matrix. That's because the 20KC coin exhibits the most distinct visual features within our dataset. Its unique characteristics (specifically, its gold color and particular shape/texture). The model is therefore able to perfectly separate and differentiate the 20KC class from all others.

The unknown class is also with a perfect accuracy and that's because the model is capable of distinguishing the very different samples that we proportioned to him with the random objects and for him it is a simple task.

4.3 Weakest Performing Class (2KC)

The class with the absolute lowest performance was the 2KC coin, achieving 0% of accuracy on the validation set. This represents a catastrophic failure of the initial model configuration.

The model predicted the majority of the 2KC samples were 10KC coins, with a misclassification rate of 63.6%. This severe error occurred due to the insufficiency of the default hyperparameters:

- **The Insufficient Training:** The lack of epochs (only 20) and the high Learning Rate 0.001 made the model not learn the critical features of the 2Kc class and made it not classify any of the sample we put on.

- **Resulting Flaw:** the model could not **extract or correctly distinguish the subtle features** of the 2KC coin, leading it to aggressively group them with the visually similar 10KC features.

This failure confirms that the model was structurally **incapable** of learning the specific features of the 2KC coin under the default, unoptimized training conditions.

5. Problem Analysis of the first model:

The main causes of low performance were:

- Lack of augmentation.
- Base model with limited capacity to distinguish small details.
- Unoptimized hyperparameters.

The main problem we initially faced was related to the transfer learning configuration settings. The default settings, including parameters such as the learning rate, the number of epochs, and adjustments to certain layers, were not optimized. This caused the model to perform poorly, resulting in only about 65% accuracy at first. By carefully identifying these issues and adjusting the settings, we were able to significantly improve the model's performance.

6. Architecture Improvement (second model):

6.1 Critical limitation of the first model:

Our project began by utilizing MobileNetV2, which was the default architecture provided by the Edge Impulse program for image classification. However, after training on our collected dataset, we encountered a critical limitation: the model's accuracy peaked at around 65%.

The problem with this model was that we couldn’t accept such a low accuracy. The unreliability of the project with 35% of error rate, that meant that one-third of the decisions would not be correct and that we would compromise functionality and practical utility in a real-world scenario. So we had to improve this model to guarantee a minimum of 80% of accuracy for the image classifier to be considered trustworthy.

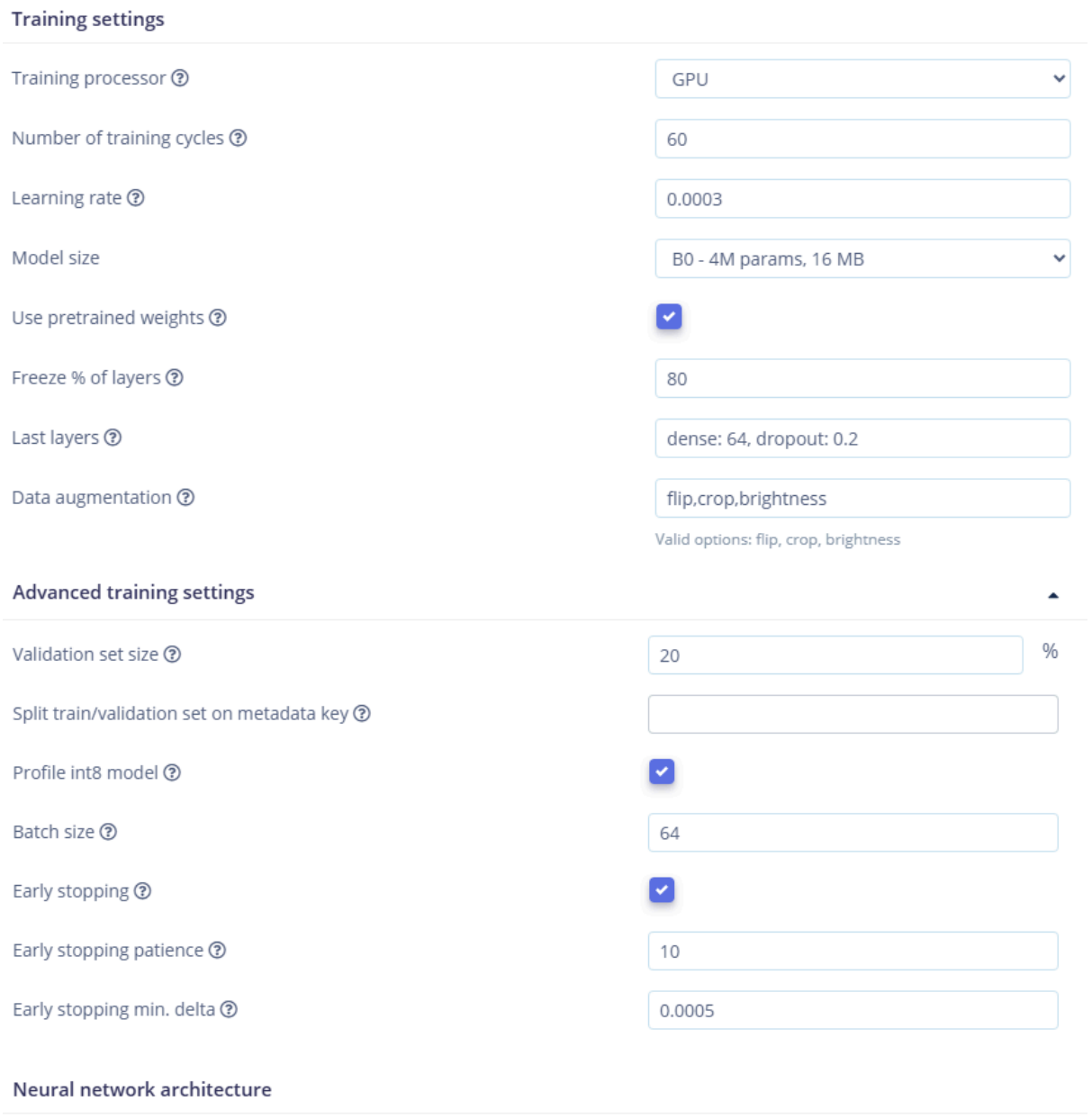
6.2 The problem with the default architecture (MobileNetV2):

When we tried to improve the model with the same architecture we found out that to increase the accuracy, we needed to increase the input image resolution to capture all the subtle details. However we couldn’t improve the image resolution because it inflated the interference latency and the model size, violating the speed and memory limits imposed by the edge device.

6.3 Architectural change and configuration:

To resolve the reliability issue we replaced the default architecture of EdgeImpulse (MobileNetV2) to another architecture that could use the principle of Compound Scaling.

Compound scaling is an important feature of image classification because it uses a uniform scaling coefficient to optimize all three dimensions (depth, widht, resolution) simultaneously, wich results in the best accuracy-to-efficiency ratio, achieving the higher accuracy that we needed. The only architecture that we found in EdgeImpulse that used this principle was the EfficientNet-Lite B0, and now we had to configurate everything with this architecture to optimize the model and improve the accuracy.



All training settings in the new model

The principal changes in the configuration of this model with the new architecture were:

- **Use of the GPU instead of the CPU and improvement of volume of epochs:** When using the CPU the model trining was much slower and used few epochs due to the time constraint and the less stable training. With the GPU we could train much faster the model and use x3 times of the epochs used before. With this improvement of the volume of the epochs helped the model to understand better all the subtle details of the samples and appreciate more the characteristics.

- **Learning rate, from 0,001 to 0,0003:**
  When using a higher learning rate (LR), the model learns faster but in an imprecise and unstable way. As a result, the model became unstable and frequently confused different classes, especially 2kc and 10kc. Once we switched to the GPU configuration and the model was able to process all samples much faster, we reduced the learning rate. This allowed the model to learn more slowly but in a much more precise and stable manner, enabling it to capture subtle details and correctly distinguish between the different coin classes.

- **Freeze layers from 100% to 80%:**
  With the initial configuration of 100%, almost all the layers were frozen wich indicated that the model was barely updating its internal feature extractor, it was only training a simple classifier without learning the specific features of the coins. So the network couldn't adapt to the unique characteristics of out dataset. By reducing to 80% of freeze layers, it allowed the last portion to be trained and the model was able to fine-tune deeper features of the coins.

- **Batch size, from 16 to 64:**
  With the initial configuration the batch size was of 16, a small batch size lead to unstable training because the gradients vary too much from one update to the next, this creates fluctuations in accuracy. With a way bigger batch size we were able to product more stable gradients allowing the model to learn in a smoother and more consistent way, reducing random oscillations.

- **Data augmentation (flip + crop + brightness):**
  With the initial data augmentation all the photos had the same conditions, with the same background, same illumination and the model could not identificate new images and resulted that in the real world we had many errors with the image classification. To improve this, we introduced realistic data augmentation, The flip augmentation simulates rotated coins, crop recreates coins that appear off-center or near the edge of the frame, and brightness variation allows the model to adapt to different lighting conditions. These transformations expose the network to a wider variety, making the model improve in its real-world accuracy.

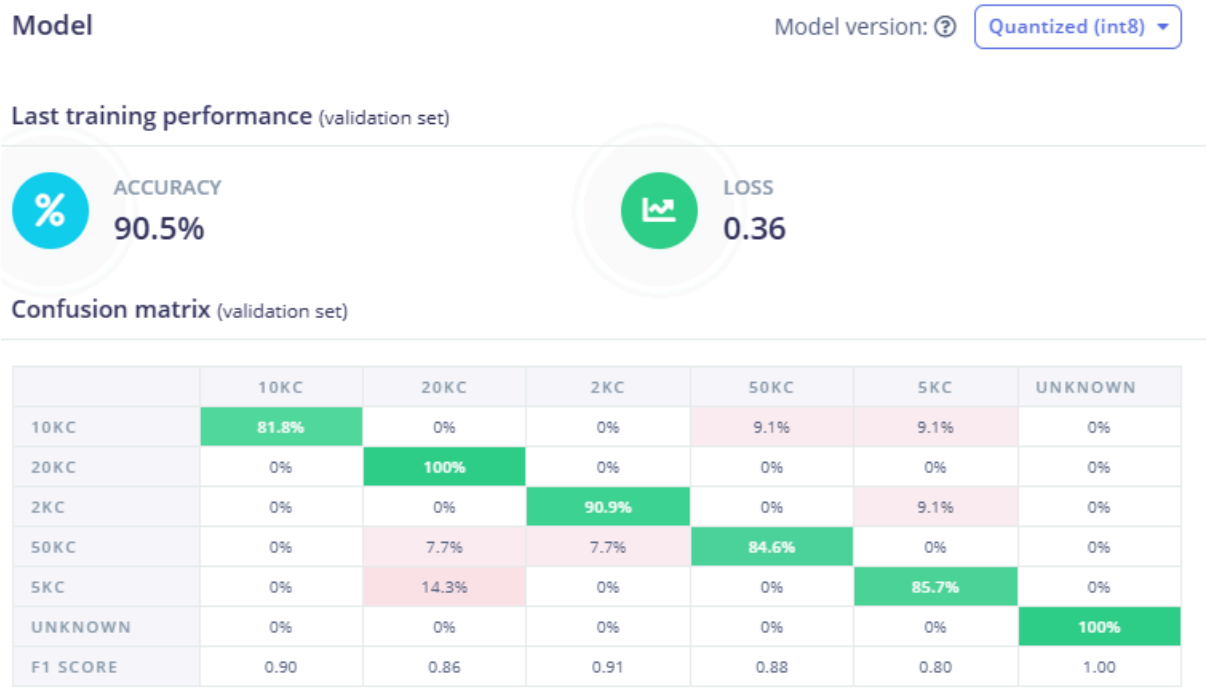
- **Early stopping (now activated):**
  In the initial configuration with the early stopping deactivated, the model could continue training after reaching its optimal performance, causing overfitting to the machine learning project. When activating the early stopping, the process automatically stopped when the model stops improving, preventing overtraining or avoiding having less accuracy.

7. Final Results of the improved model:

After applying all improvements to the old model:

1. Switch to EfficientNet
2. Data Augmentation
3. Hyperparameter optimization
4. Dataset cleaning

We obtained this confusion matrix:



Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 90.5%

LOSS 0.36

Confusion matrix (validation set)

	10KC	20KC	2KC	50KC	5KC	UNKNOWN
10KC	81.8%	0%	0%	9.1%	9.1%	0%
20KC	0%	100%	0%	0%	0%	0%
2KC	0%	0%	90.9%	0%	9.1%	0%
50KC	0%	7.7%	7.7%	84.6%	0%	0%
5KC	0%	14.3%	0%	0%	85.7%	0%
UNKNOWN	0%	0%	0%	0%	0%	100%
F1 SCORE	0.90	0.86	0.91	0.88	0.80	1.00

And obtained a data explorer graphic:

7.1 Overall performance and reliability:

The validation set achieved an Overall Accuracy of 90.5% and a Loss of 0.36. This high accuracy confirms that the architectural and configuration improvements successfully elevated the model's performance with a final accuracy of 90.5%

7.2 Best performing class (20KC):

The system's most reliable classification is the 20KC coin, achieving a perfect 100\% accuracy in the theoretical value.

The 20KC coin exhibits the most distinct visual features within our dataset. Its unique characteristics (specifically, its gold color and particular shape/texture). The model is therefore able to perfectly separate and differentiate the 20KC class from all others.

7.3. Weakest Performing Class (10KC)

Conversely, the coin that our project classifies with the lowest individual accuracy is the **10KC** coin. The difficulty lies in the **high visual similarity** the 10KC coin shares with the 50kc and the 5kc.

- **Color/Texture Similarity**: the 50KC coin (leading to 9.1% misclassification).
- **Size Similarity:** with the 5KC coin (leading to 9.1% misclassification).

7.4 Metric values and its values:

With the area under Roc Curve = 0.99, we have a model that has a great capacity of distinction of the classes.

With an weighted average precision = 0.92, this indicates that when the model asserts a class, it is correct 92% of the time.

With a weighted average recall = 0.9, this indicates that the model is able to find and correctly classify all true instances of the classes.

With a weighted average F1 score = 0.91, we have a model that is robust and trustworthy because

7.5 Data explorer:

This plot shows how the model separates the visual sample into the feature space.

· **Green Dots:** Represent the samples that were **classified correctly**.

· **Red Dots:** Represent the samples that were **classified incorrectly**.

We can observe two large and separated groups of data, this suggests that the model has successfully learned to separate the count features into at least two primary visual categories, based on the size and the color. The incorrect points of the classes that correspond to the errors in the confusion matrix are shown in the plot with red dots, wich are found in the overlap zones between the different clusters. We can confirm that the errors of the 10Kc and 5Kc because we can see the red dots in the plot that confirm the difficulty of the model with the visual features of these two coins.

8. Conclusions

The image classification project successfully demonstrated the development of a trustworthy system by achieving a final accuracy of 90.5% overcoming the initial 65.1% failure. So the final system is capable of recognizing coins with an adequate level of precision for a real-case scenario.

The key conclusions of the process and performance are:

- **Priority on the dataset quality:**

  With all the data cleaning and refinement stage, we understood that it's a critical step for the classifier reliability. With removing noisy and ambiguous data we can eliminate model erroneous features, and we ensured the model focused on the true discriminative features of the coins, not photographic defects.

- **Data augmentation greatly improves generalization capacity:**

  The application of realistic transformations like flip, crop or brightness was the key of the project to prevent the model to overfit the specific initial capture conditions. By simulating unpredictable scenarios like tilted coins or lightning variation, we artificially increased the dataset diversity, forcing the model to learn invariant features and reaching the expected accuracy.

- **Architectural superiority of EfficientNet vs MobileNetV2:**

  For projects that need a high number of epochs and the profound learning of the data, the EfficientNet architecture was much more powerful. Because of its compound scaling principle, it proved to be significantly superior with the subtle patterns of the deferent classes of the coins and solving the accuracy problem.

- **Optimization and Edge variability, with GPU, Epochs and LR:**

  The use of the GPU enabled us to implement a significantly higher number of epochs and a lower Learning Rate. This combination stabilized the training process and resulted in a much deeper learning model that could adapt robustly to the different images, backgrounds, and coin classes in a trustworthy way.