INSTITUTO POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA

Departamento de Engenharia Eletrotécnica

Demonstrador de Inteligência Artificial para Sistema Ciber Físico baseado em Arduíno (Dem4AI)

Relatório final da Unidade Curricular de
Projeto em Engenharia Eletrotécnica e de Computadores
Licenciatura em Engenharia Eletrotécnica e de Computadores, ramo de Energia e Automação

Autores:

Francisco António Lisboa Guarda Nº 2201675
Samuel Domingos Lourenço Nº 22000904

Orientadores do Projeto:

Professor Luís Manuel Conde Bento
Professora Mónica Jorge Carvalho de Figueiredo

Leiria, 3 de setembro de 2023

1. Aplicações de Redes Neuronais Profundas para alunos do Ensino Básico e Secundário

No presente capítulo, serão apresentados os tutoriais e exemplos de redes neuronais criados, utilizando diferentes métodos, assim como a explicação de como estes funcionam.

1.1. Classificação de frutas – Raise AI Playground

O tutorial seguinte, foi realizado recorrendo ao Raise AI Playground, mas utilizando uma versão diferente, de um projeto chamado "*Hackeduca*". que adiciona novos blocos necessários para o exemplo pretendido. O exemplo criado visa obter imagem em vídeo da camara do computador e permitir que o algoritmo distinga entre uma maçã e uma banana.

Para tal, como dito anteriormente, foram utilizadas as extensões do Raise AI Playground na plataforma Scratch e adicionada a extensão Tensor Flow K-NN, como se pode ver na Figura 5.3.

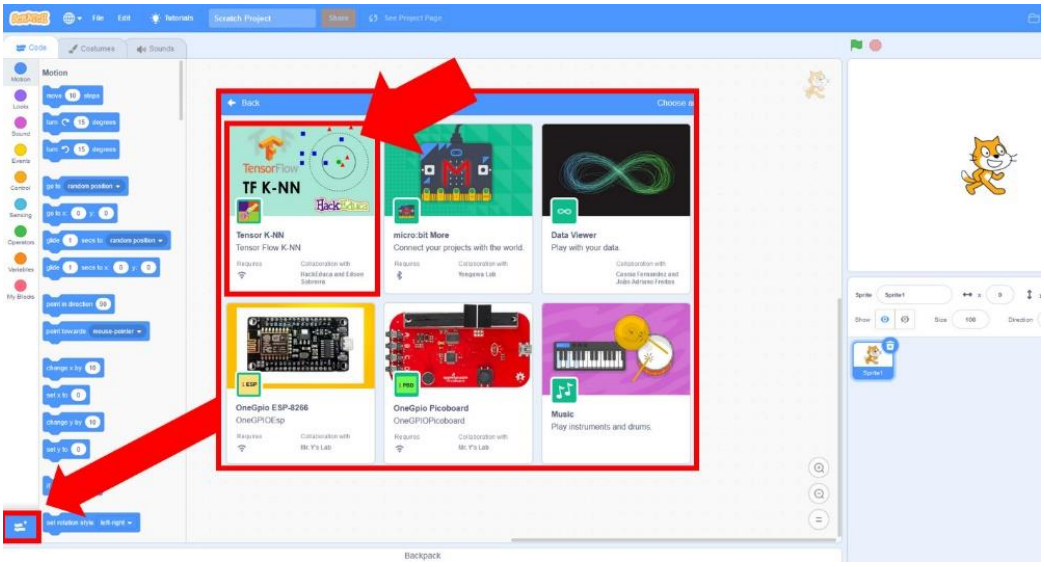

Figura 1 - Instalação da extensão Tensor Flow K-NN

Ao ser adicionada esta extensão, ficarão disponíveis novos blocos para utilização, como se pode ver na Figura 5.4.

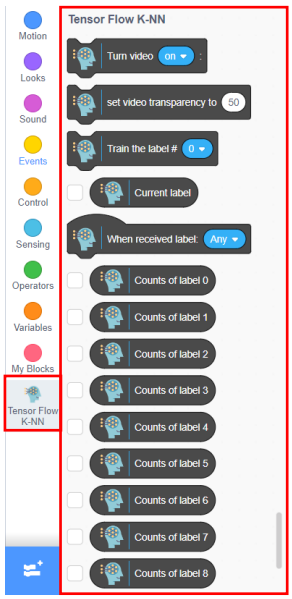

Figura 2 - Exemplos de blocos da extensão Tensor Flow K-NN

1.1.1. Funcionamento da rede

As montagens apresentadas nestes dois blocos representados na figura têm como objetivo treinar o algoritmo manualmente (Figura 5.5).

Figura 3 - Blocos usados para criar/treinar a rede

Para treinar, ao mostrar uma das frutas na camara é necessário clicar na tecla correspondente à fruta que se encontra a segurar de maneira ao programa tirar foto e associar a palavra à fruta na imagem de forma que com vários exemplos, esta possa identificar as frutas de forma correta, seja qual for a posição em que se encontrem.

Sendo assim, quanto mais fotografias forem tiradas de diferentes posições, diferentes bananas e em locais diferentes, mais facilmente será para o algoritmo de reconhecer a fruta de forma correta.

Depois de treinado o algoritmo, recolhendo várias amostras/imagens, através da câmara, com o objetivo de analisar se o algoritmo funciona corretamente, é ainda criada a junção de blocos seguinte (Figura 5.6), que permite que a camara esteja continuamente ligada e que o programa esteja sempre a correr, de forma que, quando mostrada uma das frutas na camara, esta vai ser identificada por este, como se verifica nos resultados obtidos (Figura 5.7).

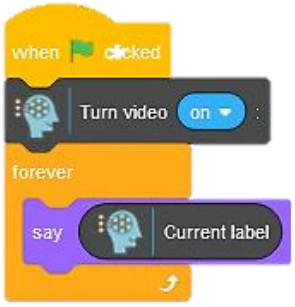

Figura 4 - Bloco que permite ligar a câmara

Resultados:

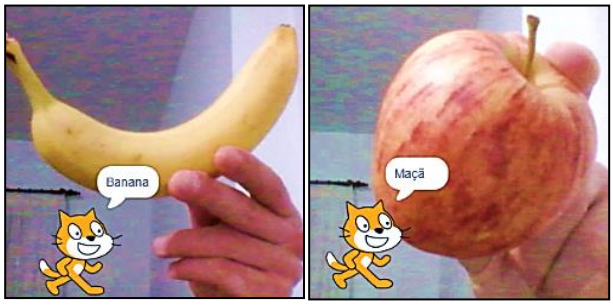

Figura 5 - Resultados obtidos na classificação das frutas

O exemplo pode facilmente ser expandido adicionando outros blocos como os apresentados inicialmente, que permitem treinar o modelo. Dessa forma, seria possível treinar a rede a identificar todas as frutas que pretendêssemos apresentar, assim como qualquer outro objeto.

Como mostrado na imagem seguinte, foi adicionada uma laranja, e o algoritmo foi capaz de identificar corretamente (Figura 5.8).

Figura 6 - Resultados obtidos com uma nova fruta

No entanto, adicionando mais frutas será necessário recolher relativamente muito mais amostras, pois existem frutas com características semelhantes. Uma laranja e uma maçã, quando atendendo ao critério de "arredondamento" da fruta, poderão ser confundidas pelo programa. Por isso é necessário treinar a rede com diversos exemplos da fruta diferentes, de maneira que esta seja capaz perceber quais as características mais importantes na distinção entre frutas.

De forma a obter os melhores resultados possíveis pode ainda alterar-se as condições de luz e imagem em que a fruta é revelada na câmara, permitindo ao algoritmo perceber como a aparência de cada fruta pode alterar dependendo das condições de luz apresentadas, ou qualidade de imagem.

1.2. Classificação de objetos em imagens usando um ESP-32 (Edge Impulse)

Neste tutorial, é usado o conceito conhecido como *Machine Learning* para construir um sistema que possa reconhecer objetos por meio de uma câmara – tarefa conhecida como classificação de imagens – conectada a um microcontrolador. Adicionar visão aos dispositivos incorporados pode fazer com que eles vejam a diferença entre caçadores furtivos e elefantes, façam controlo de qualidade nas linhas de fábrica ou deixem seus carros RC (*Remote Control*) dirigirem sozinhos.

1.2.1. Recolha de dados – criação de um *Dataset*

Neste exemplo, iremos continuar a ideia apresentada anteriormente, de classificação de frutas, e será então construído um modelo que consiga distinguir uma maçã de uma banana, no entanto, poderiam ser quaisquer dois objetos à escolha. Para fazer com que o modelo de *Machine Learning* a criar consiga "ver", é importante capturar muitas imagens de exemplo dos objetos pretendidos. Ao treinar o modelo, essas imagens de exemplo são usadas para permitir que o modelo analise as suas diferenças e aprenda a distingui-las. Como poderão existir muito mais objetos na imagem capturada com as frutas, é necessário também, obter imagens que não contenham nem a maçã nem a banana, de maneira que esta perceba que nem tudo o que está nas imagens corresponde a estas.

De maneira a criar um *Dataset* razoável é recomendado capturar a seguinte quantidade de imagens:

- ✓ 50 fotos de uma maçã.
- ✓ 50 fotos de uma banana.
- ✓ 50 imagens que não sejam de maçã nem de banana - deverão ser capturados diversos objetos aleatórios no mesmo ambiente que são capturadas as imagens das frutas.

É importante capturar uma ampla variedade de ângulos e níveis de zoom ao obter as imagens, de maneira que estas variáveis não interfiram na classificação das frutas, no futuro.

É possível capturar os dados (imagens) necessários diretamente para o projeto através dos seguintes dispositivos:

- ➢ Smartphone
- ➢ Microcontroladores suportados mencionados anteriormente
- ➢ Câmara do computador

É crucial, existir boa iluminação, seja na captura de dados, ou futuramente, aquando do teste da rede no microcontrolador, visto que, o ESP32 necessita de boas condições de iluminação para a sua câmara funcionar da melhor forma.

É ainda possível recolher dados com outra câmara, podendo depois estes serem enviados para o projeto no Edge Impulse através da opção 'Upload' no separados *'Data acquisition'*.

No final de recolhidas as imagens, é, então, concluída a etapa de criação de um D*ataset* equilibrado. No entanto poderão ser adicionados mais dados, desde que seja feito de forma equilibrada. É ainda possível observar a divisão feita automaticamente entre os dados de treino e de teste no projeto através da opção *'Data Collected'* no separador *'Data acquisition'* (Figura 5.9).

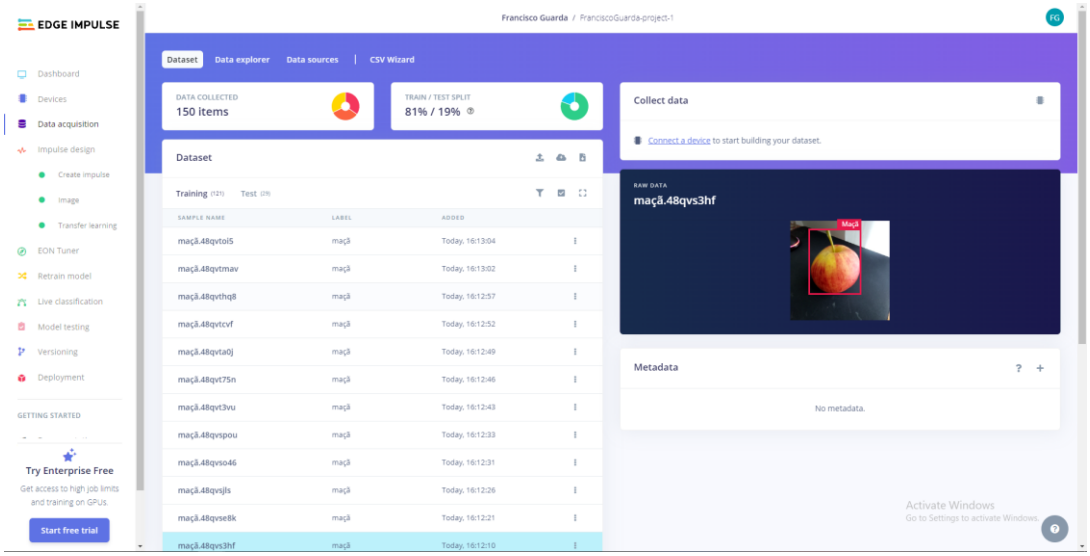

Figura 7 - Separador "Data acquisition"

1.2.2. Criação de um "impulso"

Com o treino definido, é possível agora criar um "impulso". Um "impulso" pega nos dados de entrada obtidos, ajusta o tamanho da imagem, usa um bloco de pré-processamento para manipular a imagem e, em seguida, usa um bloco de aprendizagem para classificar novos dados. Os blocos de pré-processamento sempre retornam os mesmos valores para a mesma entrada (por exemplo, convertem uma imagem colorida em uma imagem em tons de cinza), enquanto os blocos de aprendizagem aprendem com experiências anteriores.

Para este exemplo, é usado o bloco de pré-processamento 'Images'. Este bloco captura a imagem colorida (é possível, opcionalmente, colocar a imagem em tons de cinza) e, em seguida, transforma os dados em uma matriz de recursos (mapa de recursos). Em seguida, é utilizado um bloco de aprendizagem com o nome

‘Transfer Learning’, que recebe todas as imagens e aprende a distinguir entre as três classes (maçã, banana, desconhecido).

De maneira a criar o “impulso”, depois de abrir o projeto, é necessário escolher o separador ‘Create Impulse’. Neste separador, é necessário definir a largura e a altura da imagem, ambos para 96 e adicionar os blocos 'Images' e 'Transfer Learning', referidos anteriormente, e por fim clicar em ‘Save Impulse’ (Figura 5.10).

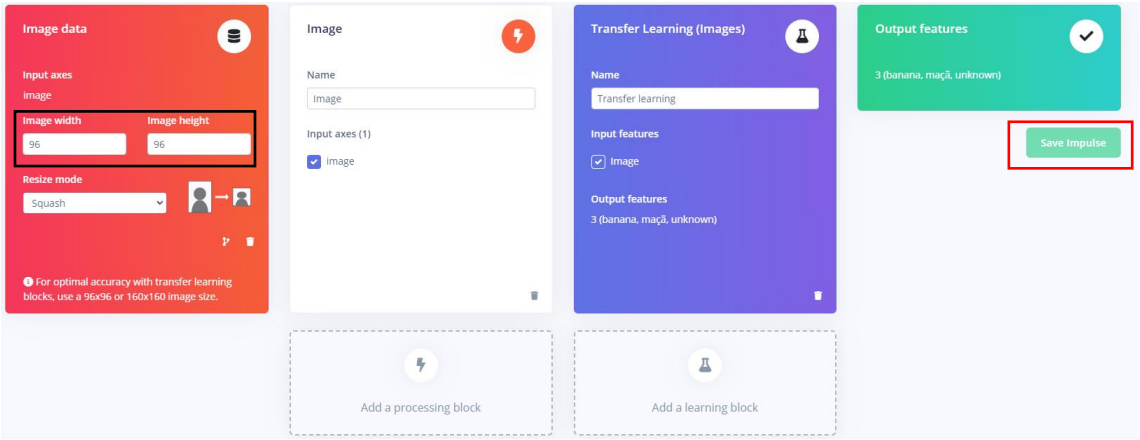

Figura 8 - Impulse Design

1.2.2.1. Configuração do bloco de processamento

Para configurar o bloco de processamento, é necessário clicar em ‘Images’ no menu à esquerda. Isto mostrará os dados obtidos inicialmente na parte superior da tela e os resultados da etapa de processamento à direita.

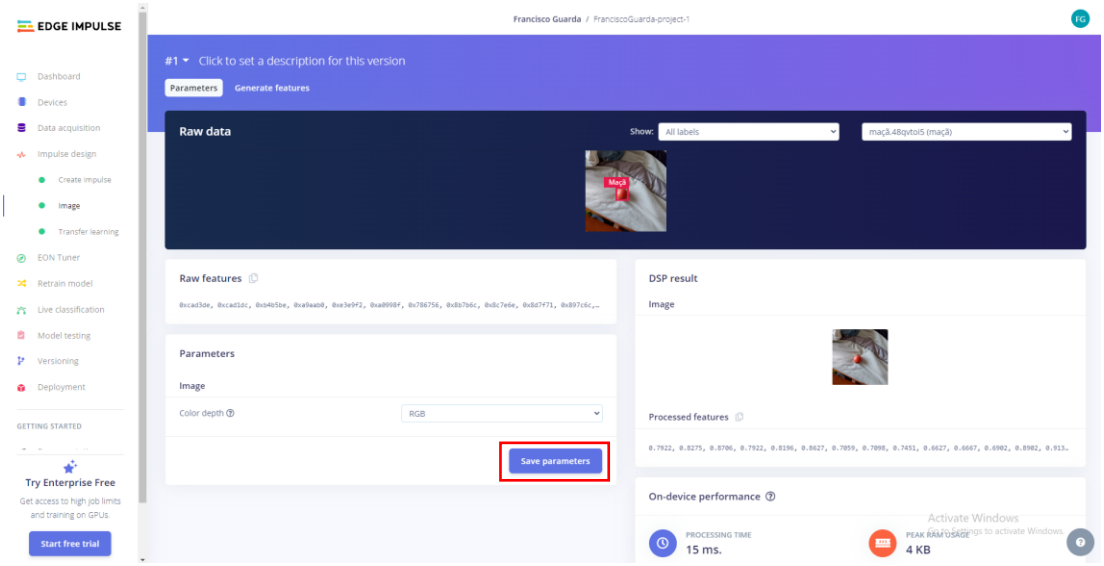

Figura 9 - Separador "Data acquisition"

É possível, ainda, usar as opções para alternar entre os modos 'RGB' e 'Escala de cinza', mas neste exemplo será usado o modo padrão, com a profundidade de cor em 'RGB'. Por fim, basta clicar em “Save pararameters” (Figura 5.11).

Terminando este processo, no separador seguinte ‘Feature generation’, ainda dentro da “Data acquisition”, é possível neste:

- Redimensionar todos os dados;
- Aplicar o bloco de processamento em todos os dados;
- Criar uma visualização 3D do conjunto de dados completo.

Para iniciar este processo, é apenas necessário criar em “Generate features” (Figura 5.12).

Depois disso, o “Feature explorer” irá aparecer do lado direito do ecrã. Este, corresponde a um gráfico de todos os dados do conjunto de dados. Como as imagens têm muitas dimensões (Neste caso: 96x96x3=27.648 características), é realizado um processo chamado 'redução de dimensionalidade' no conjunto de dados antes de visualizá-lo. Aqui, as 27.648 características são compactadas em apenas 3 e depois agrupadas com base na similaridade.

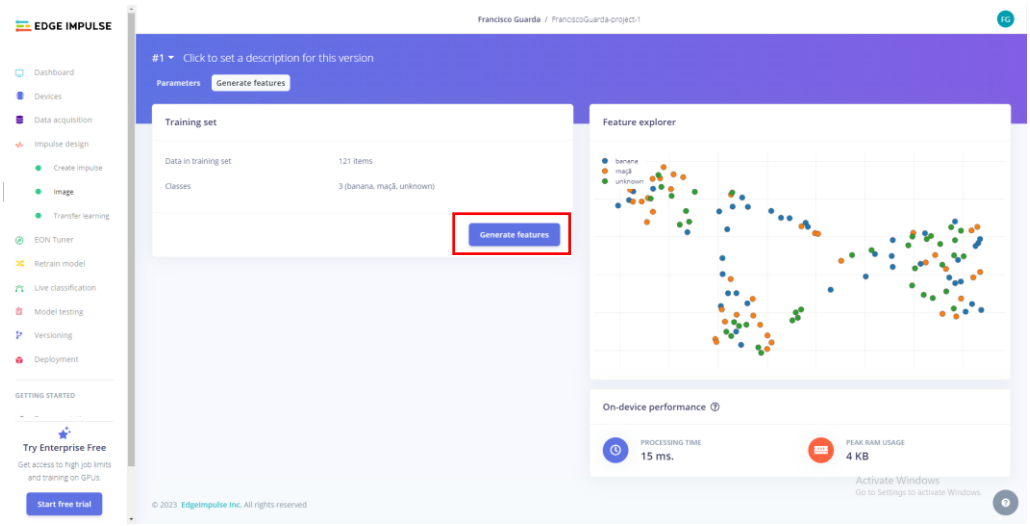

Figura 10 - Separador "Generate features"

1.2.2.2. Configuração do modelo "Transfer Learning"

Com todos os dados processados, o próximo passo é treinar a rede neuronal. A rede que será treinada irá receber todos os dados da imagem como entrada e tentará mapeá-los para uma das três classes.

É muito difícil construir um modelo de visão computacional que funcione bem de início, pois é necessária uma grande variedade de dados de entrada para treinar bem o modelo, e o tempo para treinar tais modelos poderá levar dias, dependendo da informação recolhida. Para tornar isso mais fácil e rápido, é usada a aprendizagem por transferência. Esta permite que a rede que pretendemos treinar seja "adicionada" a um modelo já bem treinado, sendo necessário apenar treinar novamente as camadas finais da rede neuronal, resultando em modelos muito mais confiáveis que treinam em uma fração de tempo muito inferior e trabalham com conjuntos de dados relativamente menores.

Para configurar o modelo de "Transfer Learning", é necessário clicar em "Transfer learning" no menu à esquerda. Aqui será possível selecionar o modelo base (o selecionado por padrão funcionara, mas existe a hipótese de a alterar com base nos requisitos de tamanho), opcionalmente ativar a opção de "Data Augmentation" (as imagens são manipuladas aleatoriamente para fazer o modelo funcionar melhor no mundo real, como explicado anteriormente no capítulo 4.2) e a taxa na qual a rede aprende.

Para este exemplo, será definido:

- ✓ Número de ciclos de treinamento para 20.
- ✓ Taxa de aprendizagem para 0,0005.
- ✓ Aumento de dados: ativado.

Por fim, basta clicar em "Start training**"**. Quando este processo terminar o modelo ficará pronto e será possível ver os números de precisão de classificação, uma tabela que demonstra a percentagem de precisão para cada uma das frutas e alguns números que preveem a performance da rede em microcontroladores (Figura 5.13).

Figura 11 - Dados de precisão do modelo

1.2.3. Validação do modelo

Com o modelo treinado, este deverá agora ser testado. Ao recolher os dados, estes foram divididos dois Datasets, o de treino e o de teste, como dito anteriormente. O modelo foi treinado usando apenas os dados do Dataset de treino e, portanto, podemos usar os dados do conjunto de dados de teste para validar quão bem o modelo funcionará no mundo real. Isto irá ajudar a garantir que o modelo não ficou demasiado ajustado aos dados de treino (Overfitting), o que é comum.

Para validar o modelo, no separador “Model testing”, deverá ser selecionada a caixa ao lado de 'Nome da amostra' e depois clicar em “Classify selected”. Neste exemplo, foi atingido 92% de precisão, o que é ótimo para um modelo com tão poucos dados (Figura 5.14).

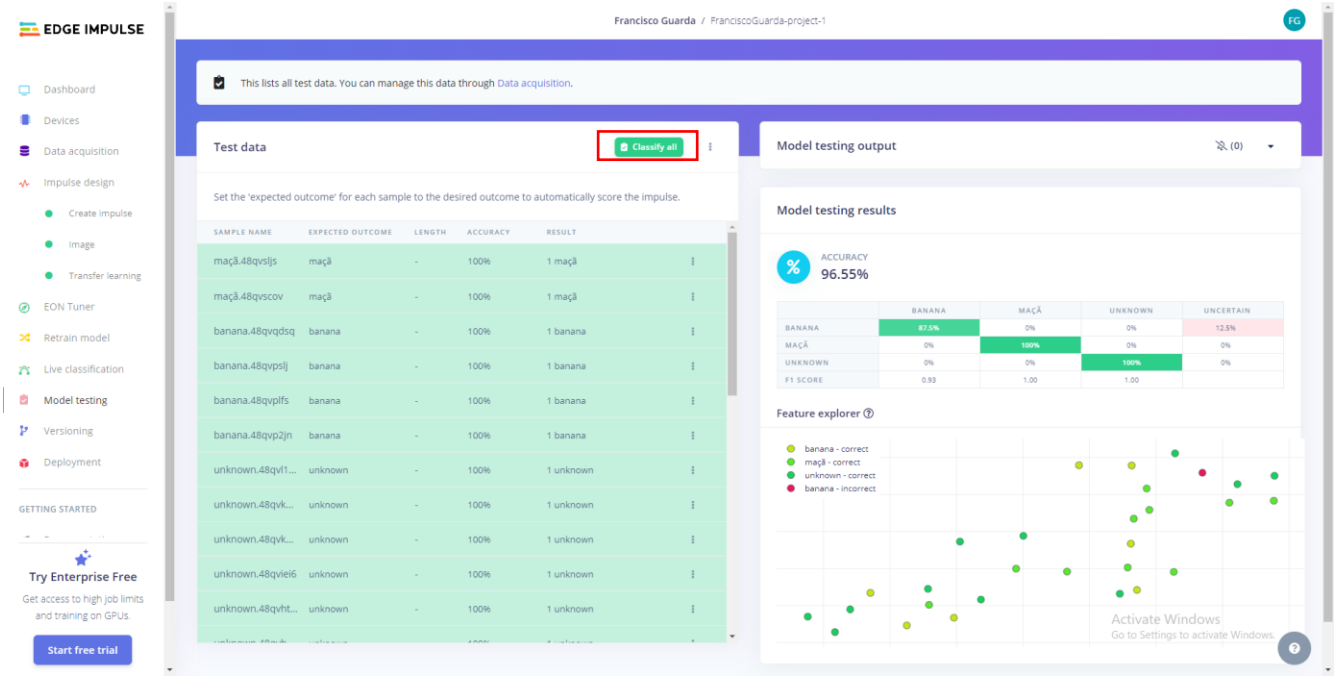

Figura 12 - Separador "Model testing"

Para ver uma classificação mais detalhadamente, é possível clicar nos três pontos ao lado de uma imagem e selecionar “Show classification”. Esta opção levará ao separador “Live classification” com muito mais detalhes sobre a imagem (também é possível capturar novas imagens com o smartphone neste separador). Este separador poderá ajudar a determinar o porquê de as imagens terem sido classificadas incorretamente.

1.2.4. Teste da rede neuronal

No anexo C, é possível analisar o processo necessário para utilizar o Arduino IDE. Assim que tudo estiver em ordem, a rede neuronal estará pronta a funcionar. Na Figura 5.21, é possível ver como a rede funciona.

É possível ver, no Serial Monitor, que aparecem as três classes criadas (banana, maçã e "unknown" - desconhecido), assim como o valor da probabilidade de cada classe (Figura 5.22).

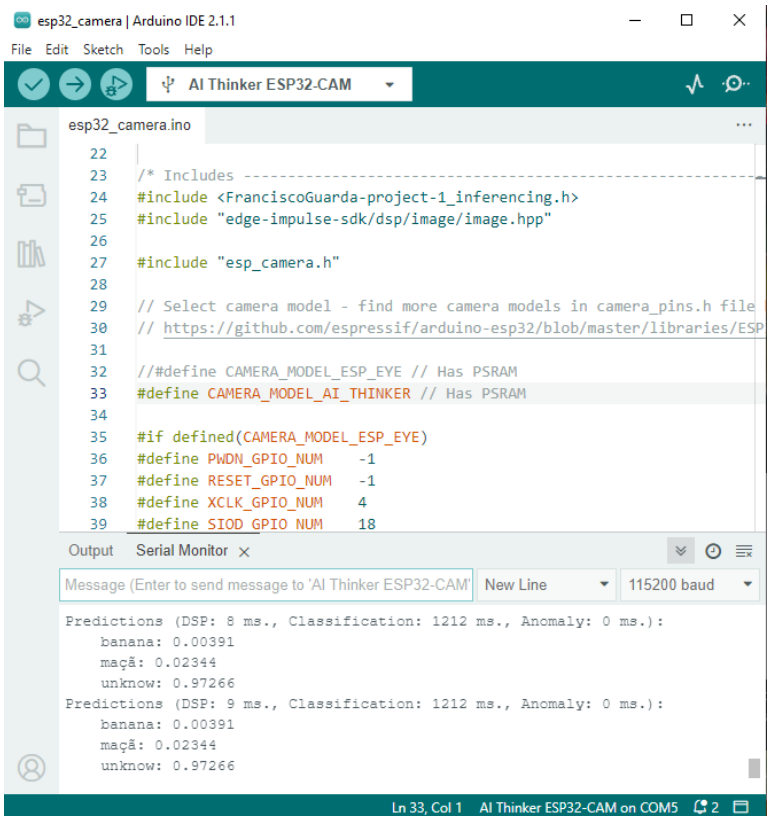

Figura 13 - Funcionamento da rede

```
Predictions (DSP: 8 ms., Classification: 1212 ms., Anomaly: 0 ms.):
    banana: 0.00391
    maçã: 0.02344
    unknow: 0.97266
Predictions (DSP: 9 ms., Classification: 1212 ms., Anomaly: 0 ms.):
    banana: 0.00391
    maçã: 0.02344
    unknow: 0.97266
```

Figura 14 - Apresentação dos valores no Serial Monitor

É importante relembrar que, como a rede não tem um dataset muito extensivo, é normal que o valor da precisão das frutas não seja muito elevado, como demonstra a Figura 5.23, retirada em tempo real durante o teste do exemplo, utilizando a maçã.

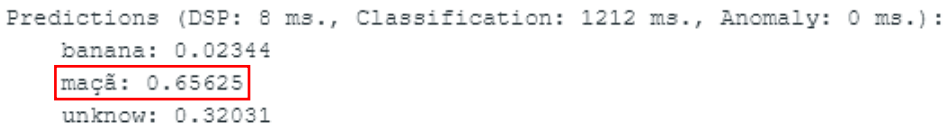

Figura 15 - Classificação de uma maçã

De maneira a aumentar a precisão da classificação seria necessário aumentar substancialmente o dataset de todas as classes.

No caso de se fazerem alterações, seja, adicionando dados ao dataset, ou adicionando novos objetos à rede, é necessário voltar ao passo do capítulo 5.4.2.1 e repetir todo o resto do processo para reimplementar a rede com os novos dados no microcontrolador.