

DETECTION DU CANCER DE PEAU EN UTILISANT ARDUINO, LA CARTE NANO 33 SENSE BLE AINSI QUE LE MODULE CAMERA FOURNIT AVEC LA CARTE

Enoncé du projet : [Projet Final TinyML - Google Docs](#)

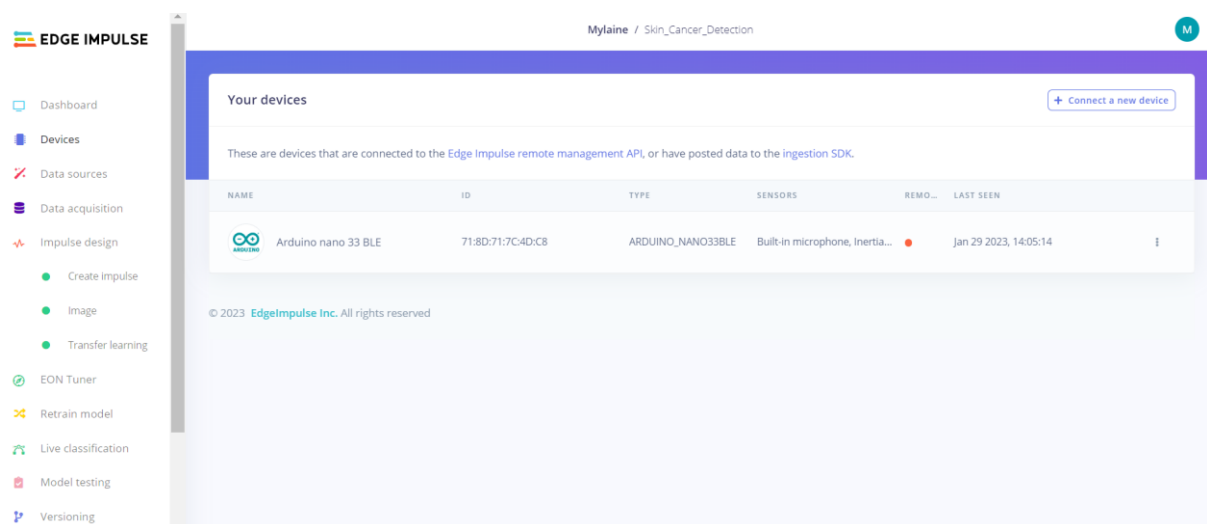
Logiciels utilisé : edge impulse (créer un compte sur edge impulse)

PREMIERE PARTIE : EDGE IMPULSE

A : Entrainement sur Edge impulse

Se connecter sur la carte : <https://docs.edgeimpulse.com/docs/development-platforms/fully-supported-development-boards>

Une fois connecté, on va dans l'onglet devices et celui-ci s'affiche comme suit :



- Data acquisition :

Le dataset étant déjà fourni, nous devons dans un premier temps les télécharger et ensuite les importer dans notre projet.

Datasets des peaux atteintes de cancer [Skin Cancer Dataset | Kaggle](#)

Datasets peau normale et images inconnu :

https://figshare.com/articles/dataset/Texture_Patch_Dataset_zip/6091007 les datasets étant en .tif, nous devons les convertir en une extension d'image en utilisant le lien suivant :

<https://www.iloveimg.com/fr/convertir-en-jpg/tiff-en-jpg>

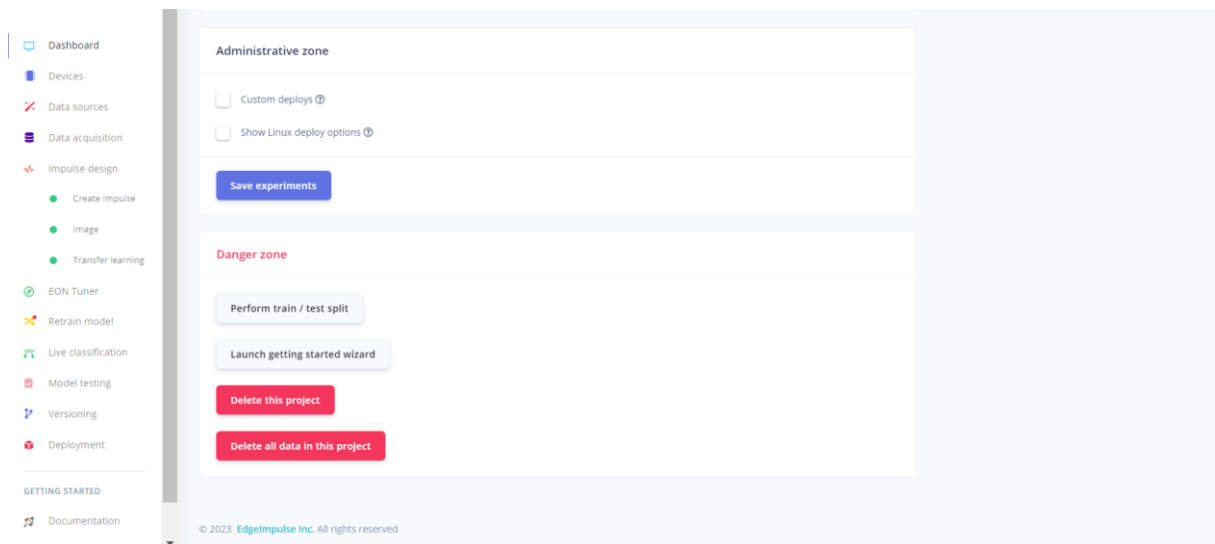
Upload data : ici, nous pouvons directement donner un nom à notre label en sélectionnant la case label -> enter label et donner une appellation à nos données par catégories.

Ensuite select files -> upload

The screenshot shows the 'Upload existing data' form in the EdgeImpulse interface. The form is titled 'Upload existing data' and includes instructions: 'You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG, PNG, AVI or MP4 files.' Below this, there is a 'Select files' section with a button labeled 'Sélect. fichiers' and the text 'Aucun fichier choisi'. The 'Upload into category' section has three radio buttons: 'Automatically split between training and testing' (selected), 'Training', and 'Testing'. The 'Label' section has three radio buttons: 'Infer from filename', 'Leave data unlabeled', and 'Enter label:' (selected). Below the 'Enter label:' radio button is a text input field containing the text 'normal_skin'.

Pour repartir les données entre l'entraînement et le test

- Dashboard -> perform train / test split



Training dataset :

EDGE IMPULSE

Training data | Test data | Data explorer | Upload data | Export data

Did you know? You can capture data from any device or development board, or upload your existing datasets - [Show options](#)

DATA COLLECTED: 507 items

TRAIN / TEST SPLIT: 80% / 20%

Record new data [Connect using WebUSB](#)

No devices connected to the remote management API.

RAW DATA: Click on a sample to load...

SAMPLE NAME	LABEL	ADDED
ISIC_0024485	Cancer_skin	Jan 31 2023, 09:19:54
ISIC_0024489	Cancer_skin	Jan 31 2023, 09:19:54
ISIC_0024482	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024483	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024487	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024475	Cancer_skin	Jan 31 2023, 09:19:53

Testing dataset

EDGE IMPULSE

Training data | Test data | Data explorer | Upload data | Export data

Did you know? You can capture data from any device or development board, or upload your existing datasets - [Show options](#)

DATA COLLECTED: 130 items

TRAIN / TEST SPLIT: 80% / 20%

Record new data [Connect using WebUSB](#)

No devices connected to the remote management API.

RAW DATA: Click on a sample to load...

SAMPLE NAME	LABEL	ADDED
ISIC_0024484	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024480	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024486	Cancer_skin	Jan 31 2023, 09:19:53
ISIC_0024472	Cancer_skin	Jan 31 2023, 09:19:52
ISIC_0024469	Cancer_skin	Jan 31 2023, 09:19:51
1 (565)	normal_skin	Jan 31 2023, 09:13:17

Create impulse : choisir images comme type de données et Transfer Learning comme méthode d'apprentissage. Et enfin enregistrer

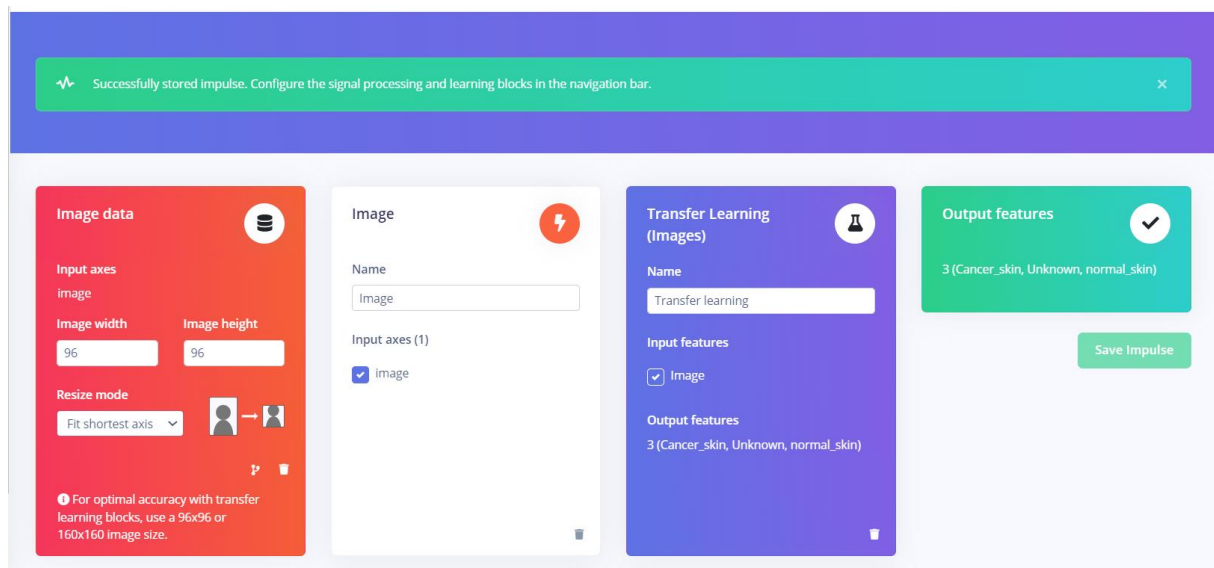
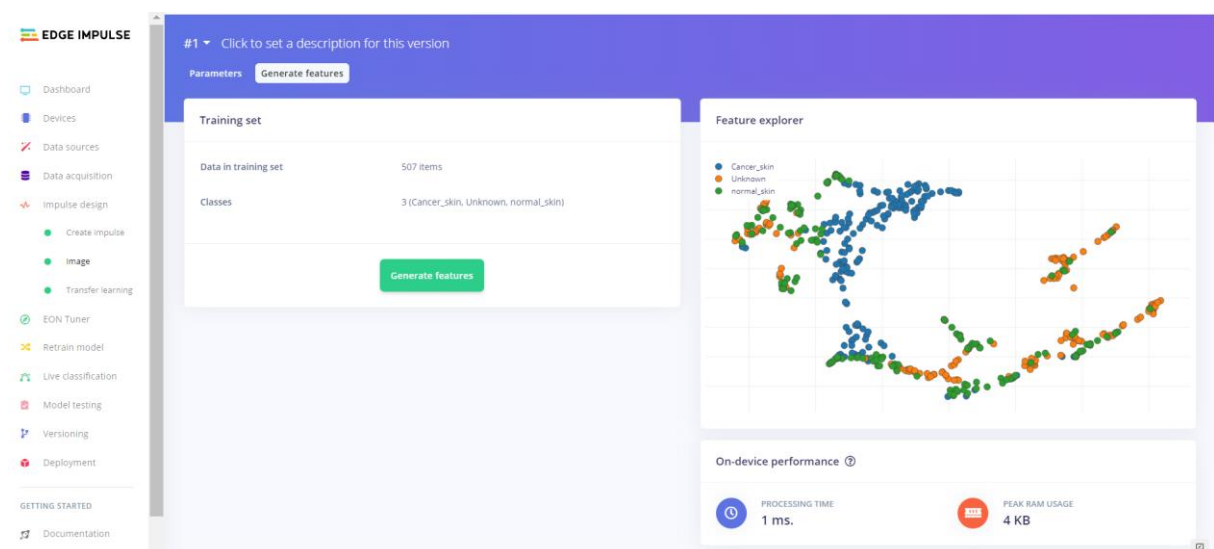
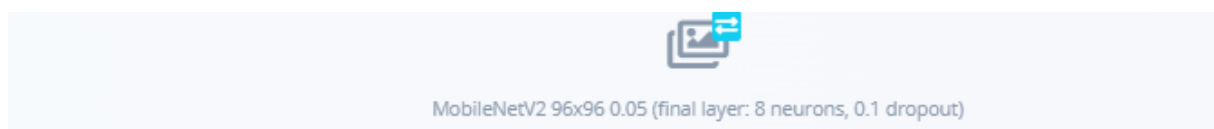


Image: dans ce bloc, on laisse les paramètres tel qu'ils sont et on génère des features.



Transfer learning : ici, on modifie le pourcentage de validation et on le met à 70%, on augmente le nombre d'époch (cycle d'entraînement du modèle) à 80 et comme model d'entraînement on choisit



Pour optimiser la taille mémoire et enfin on lance l'entraînement du modèle

EDGE IMPULSE

Dashboard

Devices

Data sources

Data acquisition

Impulse design

Create impulse

Image

Transfer learning

EDN Tuner

Retrain model

Live classification

Model testing

Versioning

Deployment

GETTING STARTED

Documentation

Forums

#1 Click to set a description for this version

Neural Network settings

Training settings

Number of training cycles

80

Learning rate

0.0005

Validation set size

70

%

Auto-balance dataset

☐

Data augmentation

☒

Neural network architecture

Input layer (27,648 features)

MobileNetV2 96x96 0.05 (final layer: 8 neurons, 0.1 dropout)

Choose a different model

Output layer (3 classes)

Start training

Training output

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY

83.4%

LOSS

0.43

Confusion matrix (validation set)

	CANCER_SKIN	UNKNOWN	NORMAL_SKIN
CANCER_SKIN	94.9%	0.9%	4.3%
UNKNOWN	0.9%	76.5%	22.6%
NORMAL_SKIN	1.6%	19.5%	78.9%
F1 SCORE	0.96	0.77	0.77

Data explorer (full training set)

On-device performance

INFERRING TIME

49 ms.

PEAK RAM USAGE

283,4K

FLASH USAGE

159,3K

Training output :

Training output

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY

83.4%

LOSS

0.43

Confusion matrix (validation set)

	CANCER_SKIN	UNKNOWN	NORMAL_SKIN
CANCER_SKIN	94.9%	0.9%	4.3%
UNKNOWN	0.9%	76.5%	22.6%
NORMAL_SKIN	1.6%	19.5%	78.9%
F1 SCORE	0.96	0.77	0.77

Data explorer (full training set)

On-device performance

INFERRING TIME

49 ms.

PEAK RAM USAGE

283,4K

FLASH USAGE

159,3K

On constate que la taille de ram qu'occupera le model sur la carte est de 283 ,4k ce qui est supérieur à celle de la carte donc aller dans create impulse et modifier la taille des images

Image data

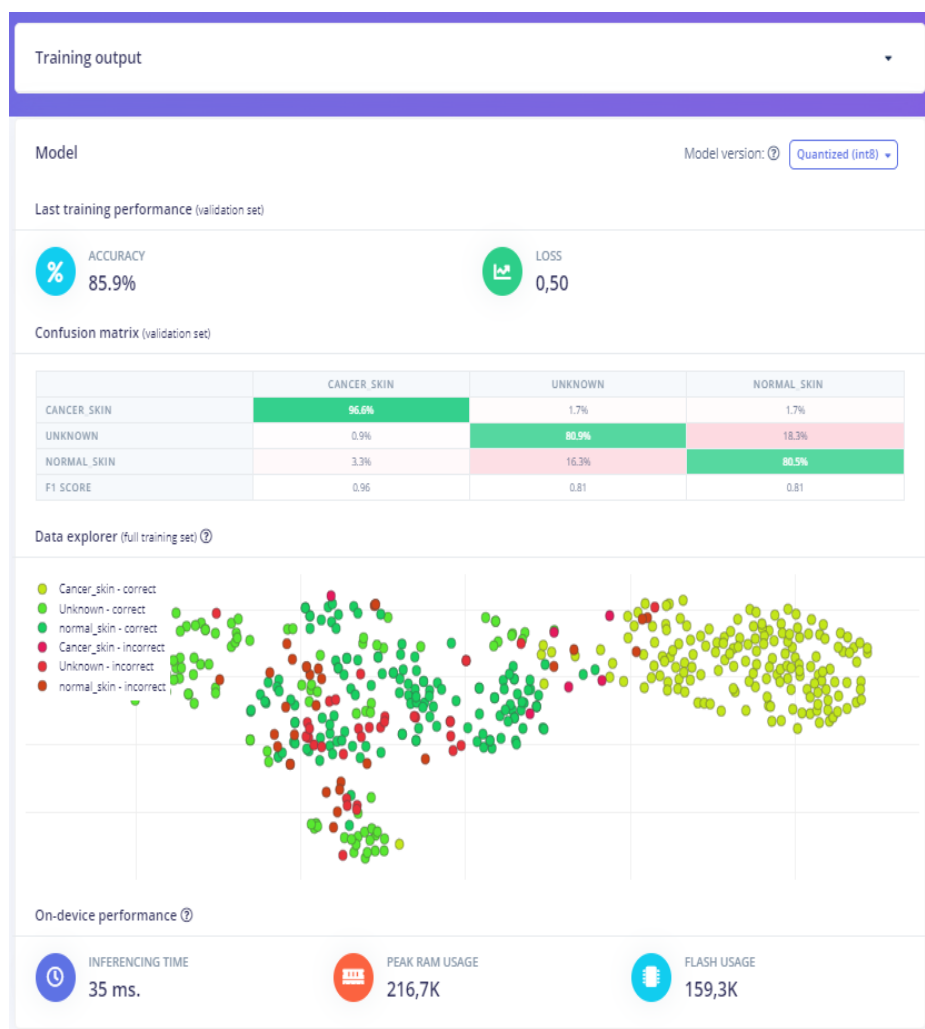
Input axes
image

Image width: 80 Image height: 80

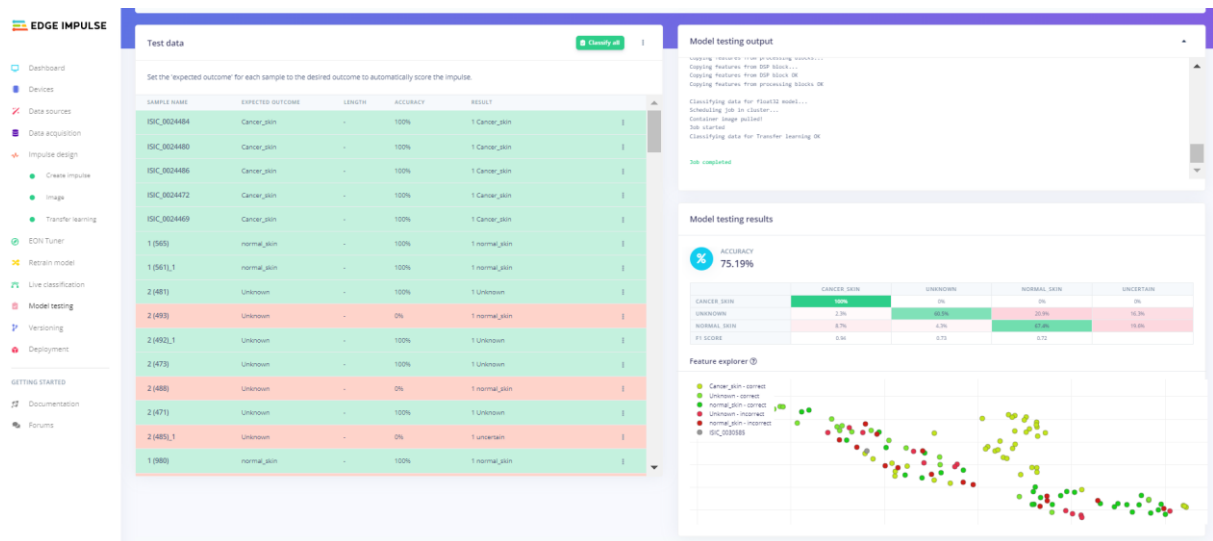
Resize mode: Fit shortest axis

For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size.

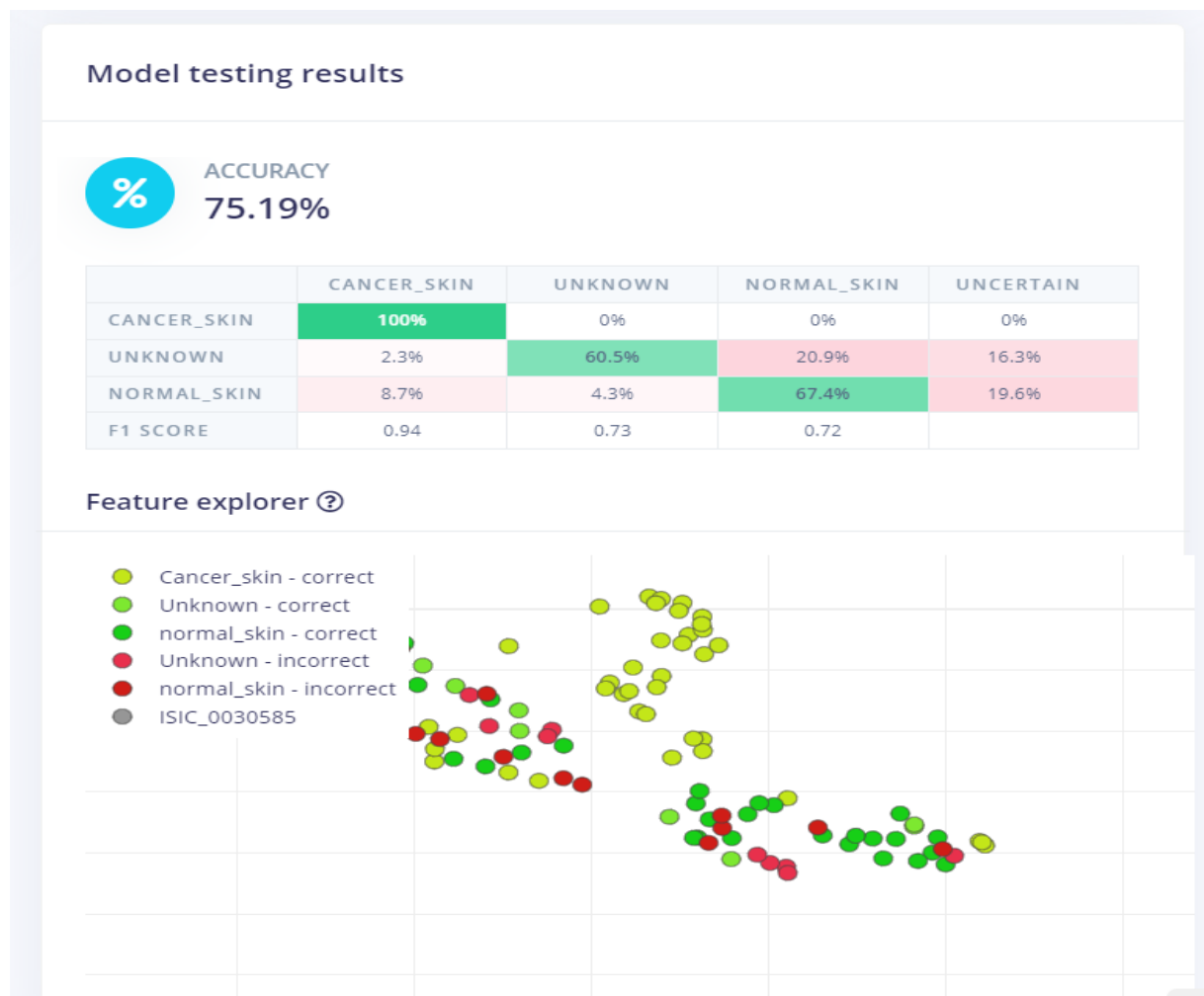
Enregistrer les nouveaux paramètres et relancer l'entraînement : on a



- Model testing

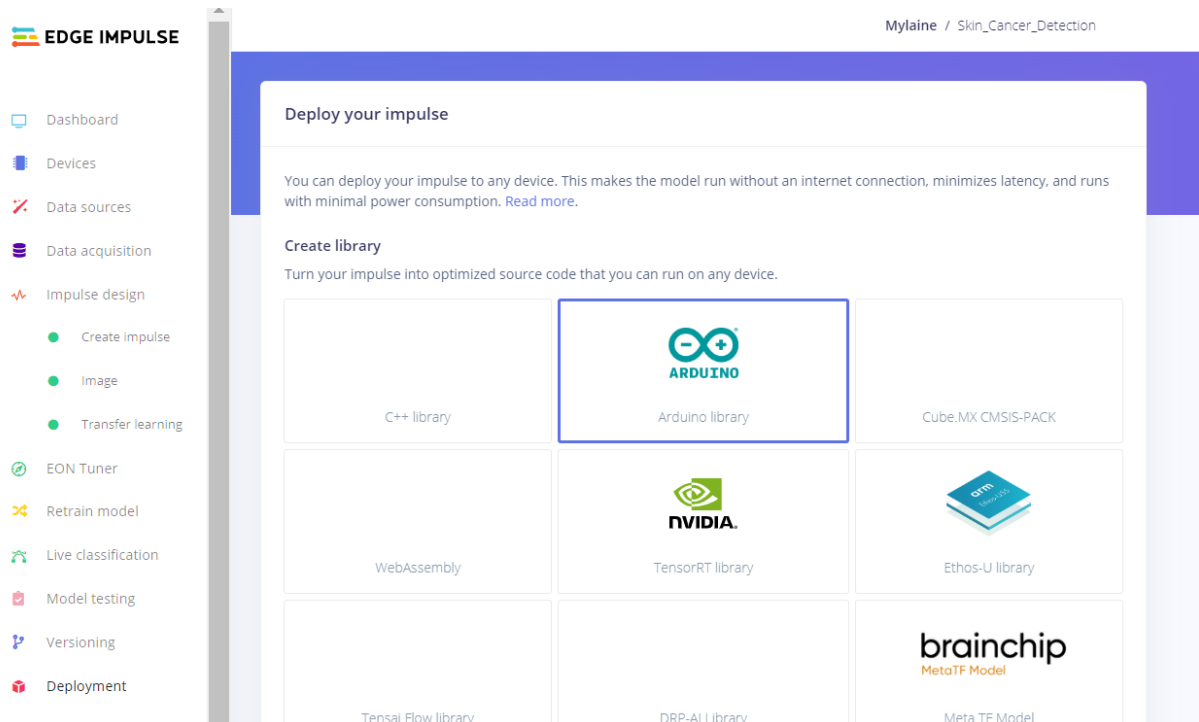


Model testing results :



- Deployment

Aller dans deployment et sélectionner arduino (arduino Library)



Ensuite analyse optimizations

Select optimizations (optional)

Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.



Enable EON™ Compiler

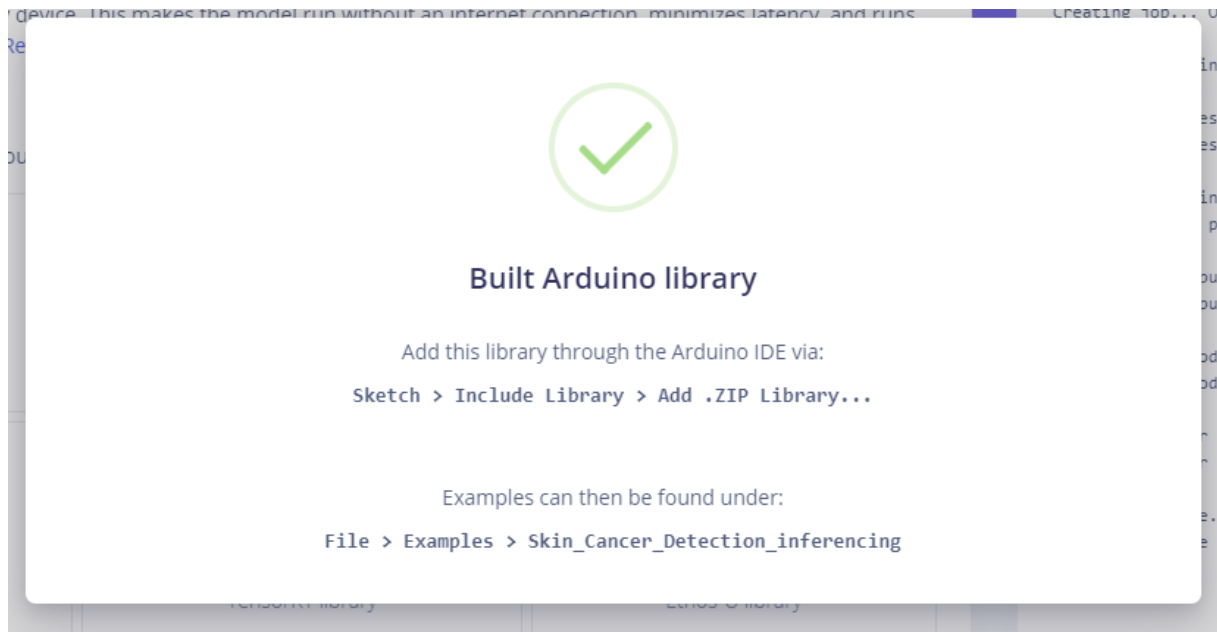
Same accuracy, up to 50% less memory. Open source.



Available optimizations for Transfer learning

Quantized (int8) ★ Currently selected This optimization is recommended for best performance.	RAM USAGE 216,7K	LATENCY 35 ms	CONFUSION MATRIX ⓘ			
	FLASH USAGE 159,3K	ACCURACY 77.52%	100	0	0	0
Unoptimized (float32) Click to select	RAM USAGE 662,7K	LATENCY 101 ms	2.3	72.1	16.3	9.3
	FLASH USAGE 244,7K	ACCURACY 75.19%	6.5	10.9	63.0	19.6

Et enfin build



B : Arduino IDE

Ouvrir l'IDE arduino et exporter le fichier .zip qui a été créé.

Aller dans les exemples et choisir celui avec une utilisation de la camera et y rajouter des morceaux de code suivant :

Entête :

```
|  
#define LED_BUILTIN 0
```

Void setup :

```
void setup()  
{  
    // put your setup code here, to run once:  
    Serial.begin(115200);  
  
    //*****  
    // initialize digital pin LED_BUILTIN as an output.  
    pinMode(LED_BUILTIN, OUTPUT);  
    //*****  
}
```

Void loop :

```

else
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label,
                  result.classification[ix].value);

//*****

        if ( result.classification[ix].value > 0.7){
            digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
            delay(1000); // wait for a second
            digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
            delay(1000); // wait for a second
        }
//*****
    }
}

```

Deuxième partie : utilisation de tensorflow lite pour l'entraînement

<https://medium.com/@a.ayyuced/image-classification-models-on-arduino-nano-33-ble-sense-60bf845fd2aa>

<https://gist.github.com/gheesung/eb0076e040ba53d5be2ad2db1c70cf82>

première partie : <https://colab.research.google.com/drive/1yxY0-w9UXJB-2SEM1xBbVAaUPe7-xvO>

L'entraînement par méthode de classification fini par générer un fichier model.tLite de 3.5M ce qui est très volumineux car la carte ne peut que supporter jusqu'à 256k.

Nous devons donc utiliser une autre méthode optimisant au maximum le model :

https://colab.research.google.com/drive/15_OdL8mj1ls-C-Tg_0StuMAeaZwjnNq3?usp=sharing

à la suite de celui-ci, un fichier model.kmodel est généré contenant les données d'entraînement et de validation de notre model. Nous n'avons plus qu'à le tester.