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]

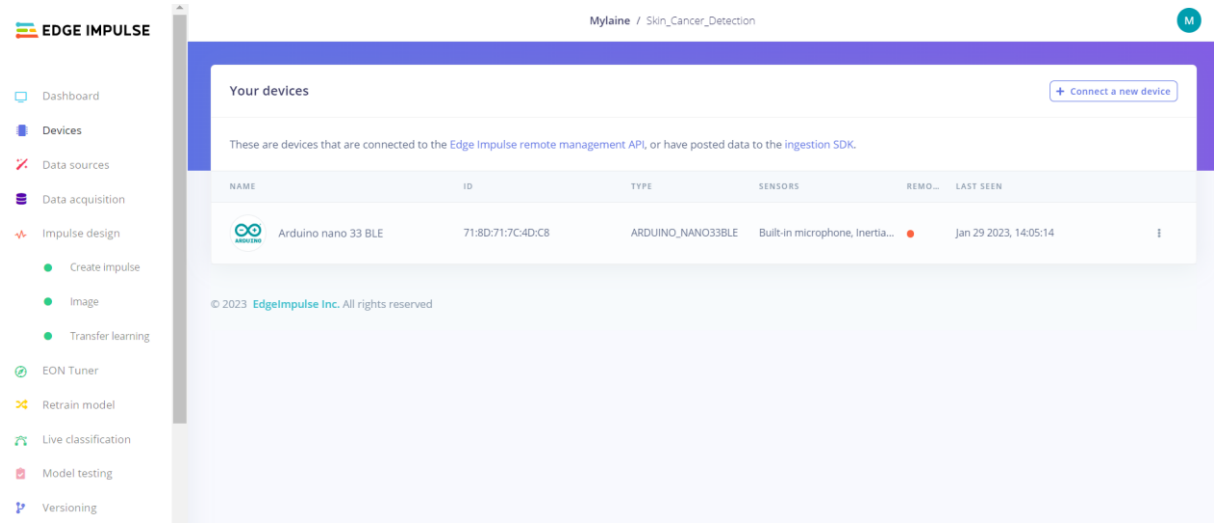
Logiciels utilisé : edge impulse (créer un compte sur edge impulse)

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

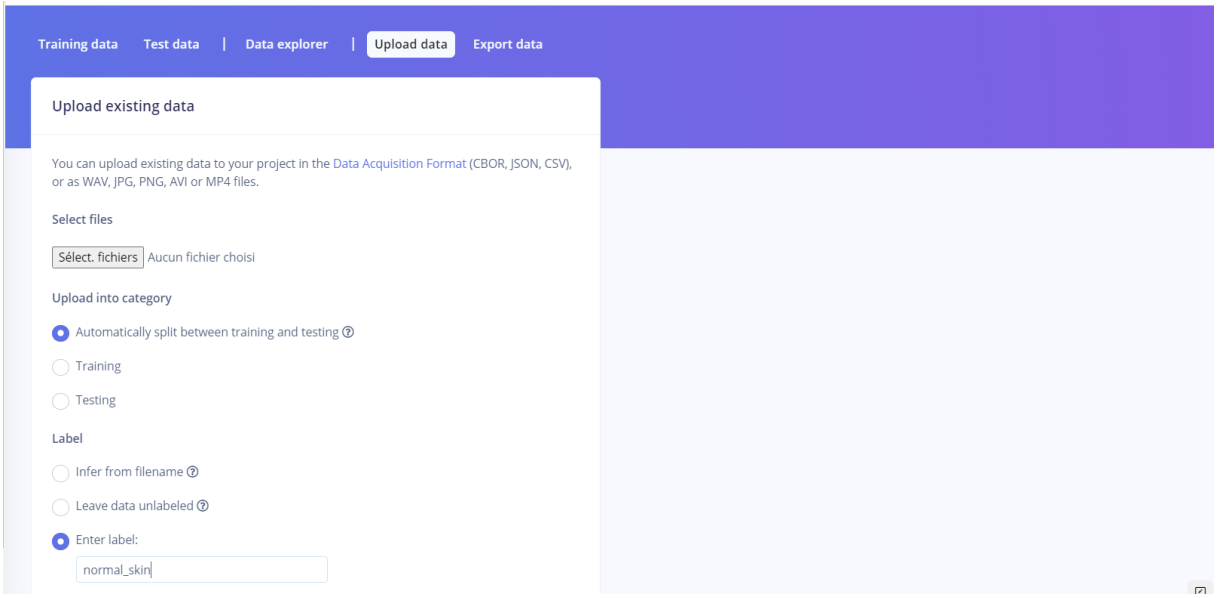
Datasets des peaux atteintes de cancer Skin Cancer Dataset | Kaggle

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

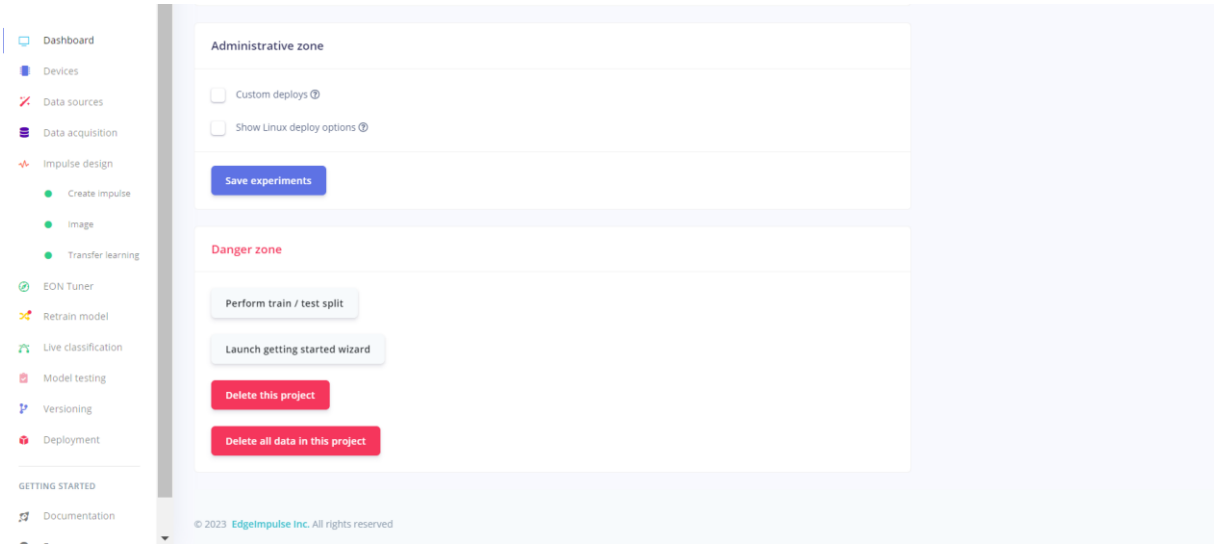
Upload data : ici, nous pouvons directement donner un nom à notre label en sélectionnant la case label -> enter label et donner une appellation à nos données par catégories.

Ensuite select files -> upload

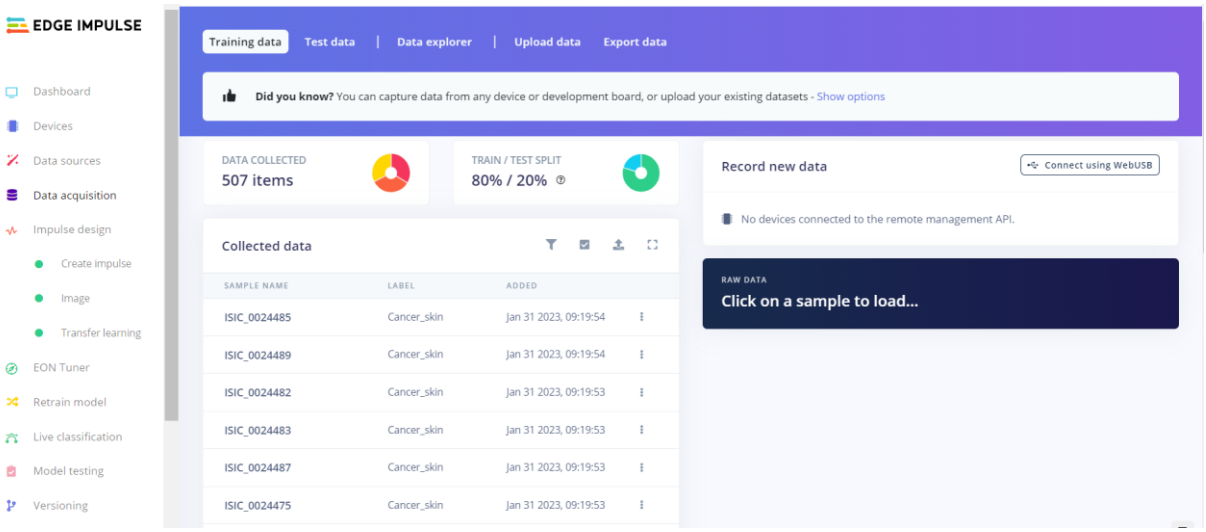


Pour repartir les données entre l'entrainement et le test

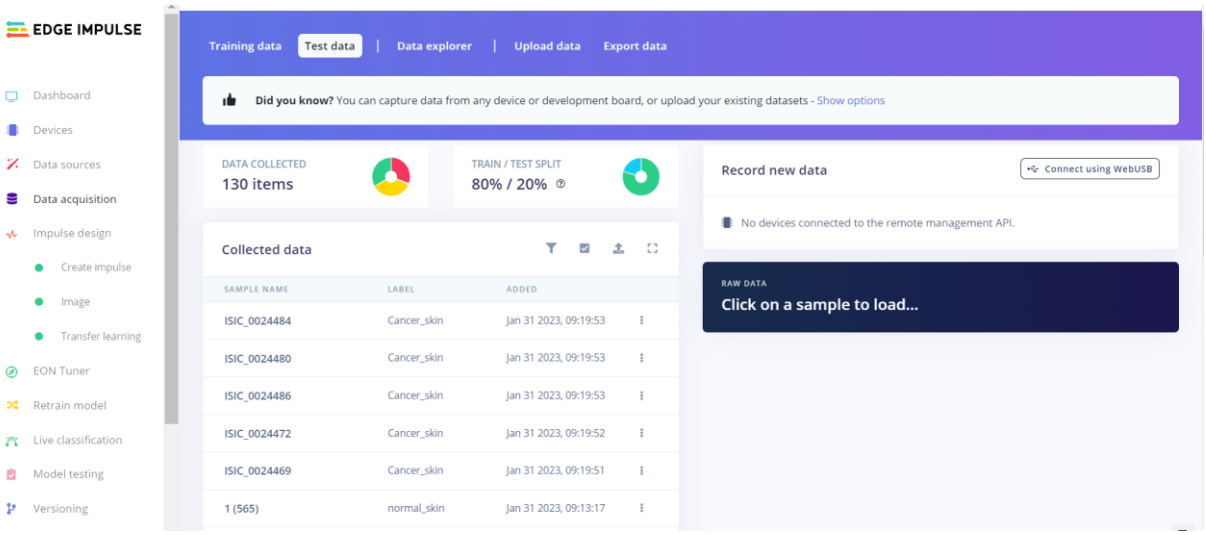
- Dashboard -> perform train / test split



Training dataset :



Testing dataset



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

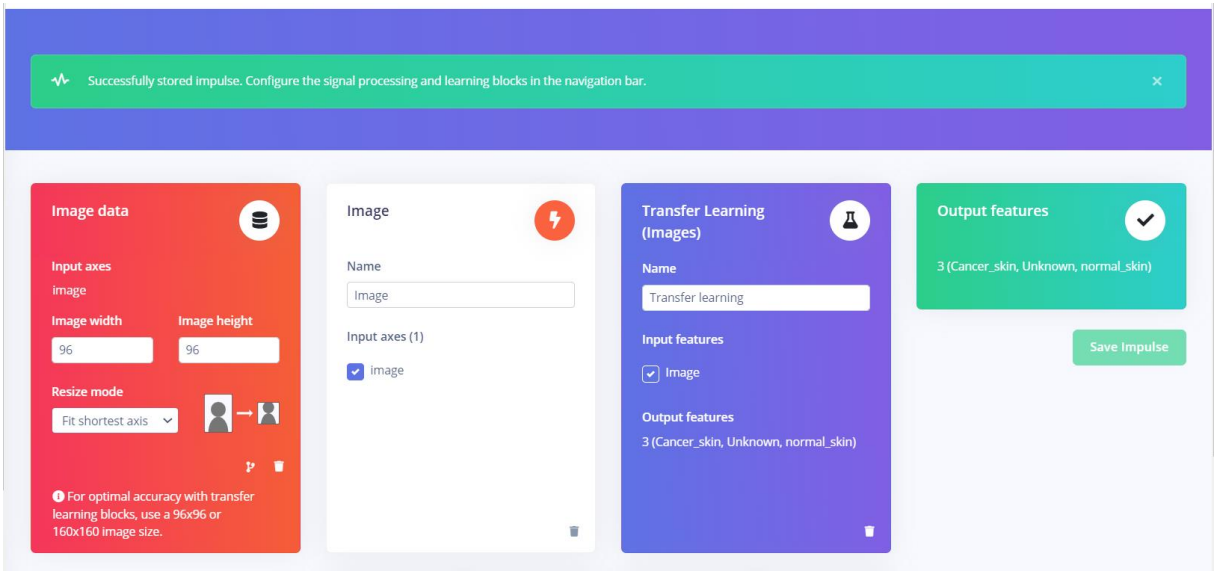
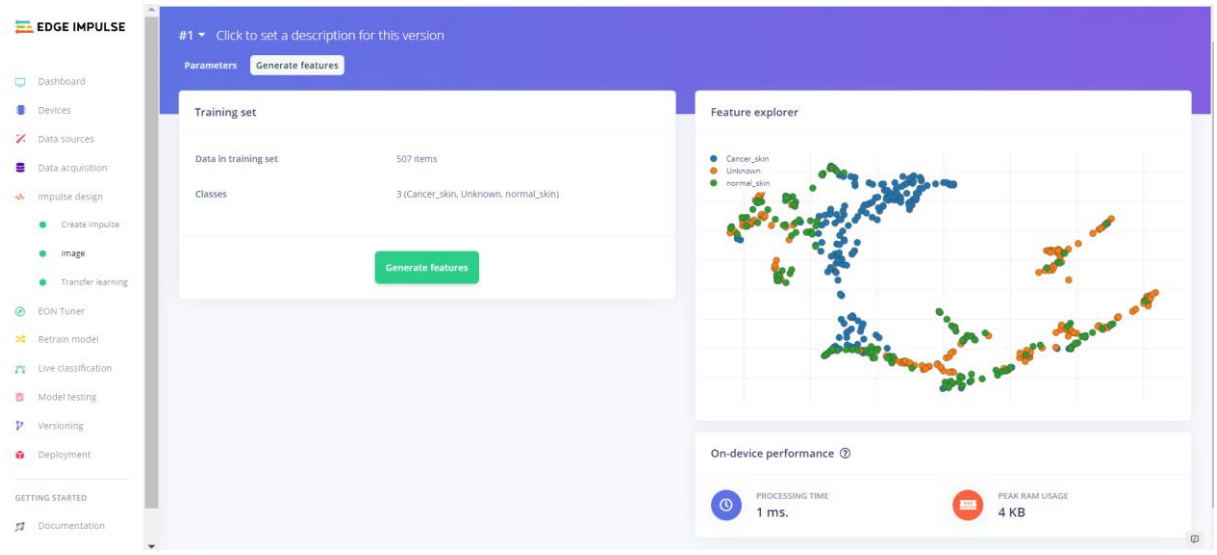
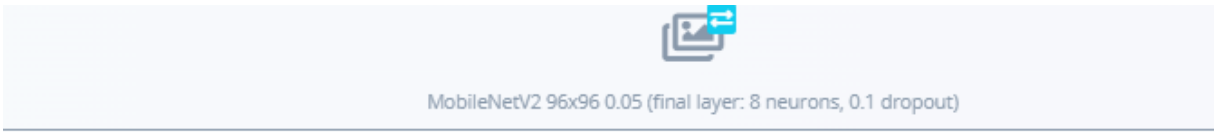


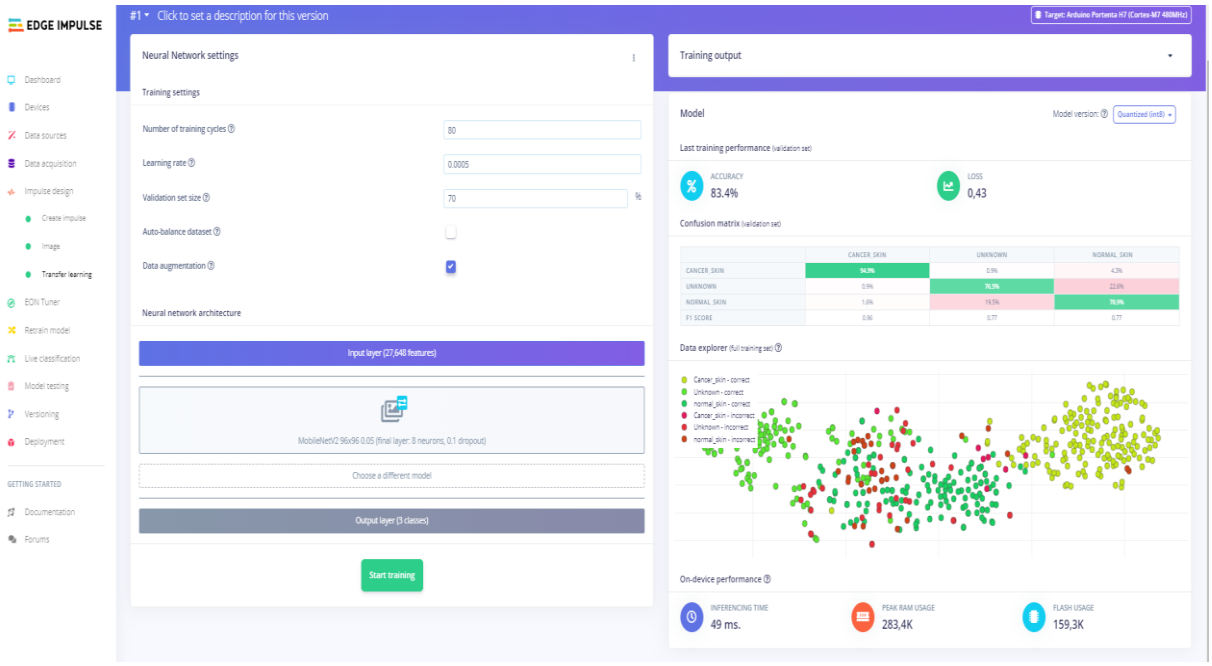
Image: dans ce bloc, on laisse les paramètres tel qu'ils sont et on génère des features.



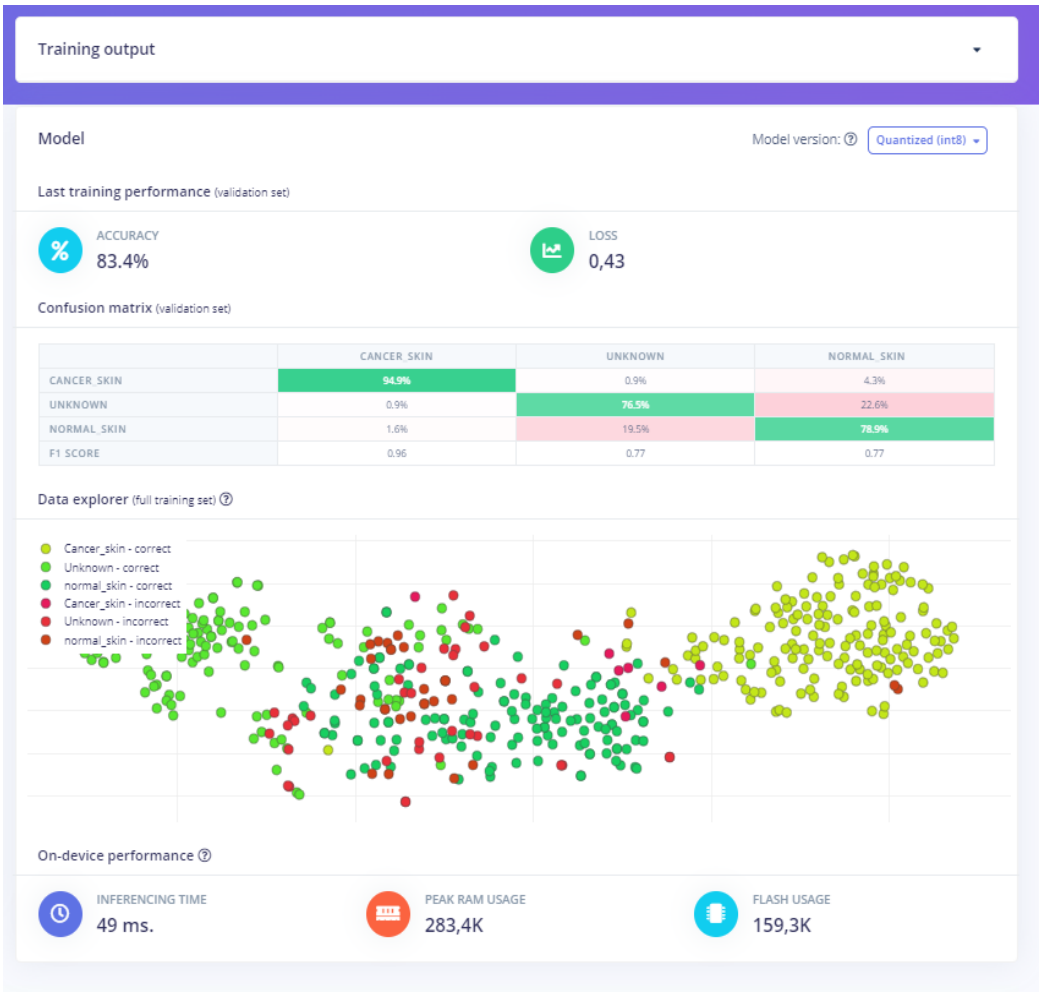
Transfer learning : ici, on modifie le pourcentage de validation et on le met à 70%, on augmente le nombre d'epoch (cycle d'entrainement du modèle) à 80 et comme model d'entrainement on choisis



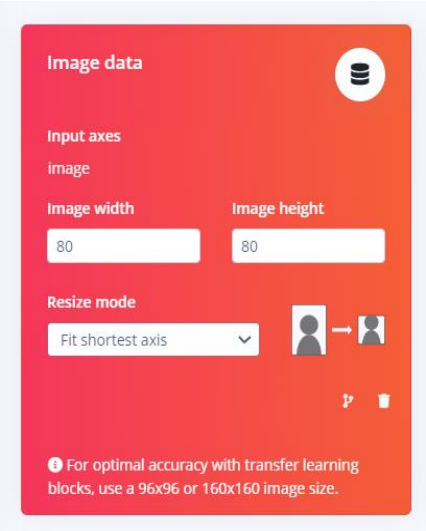
Pour optimiser la taille mémoire et enfin on lance l'entrainement du modèle



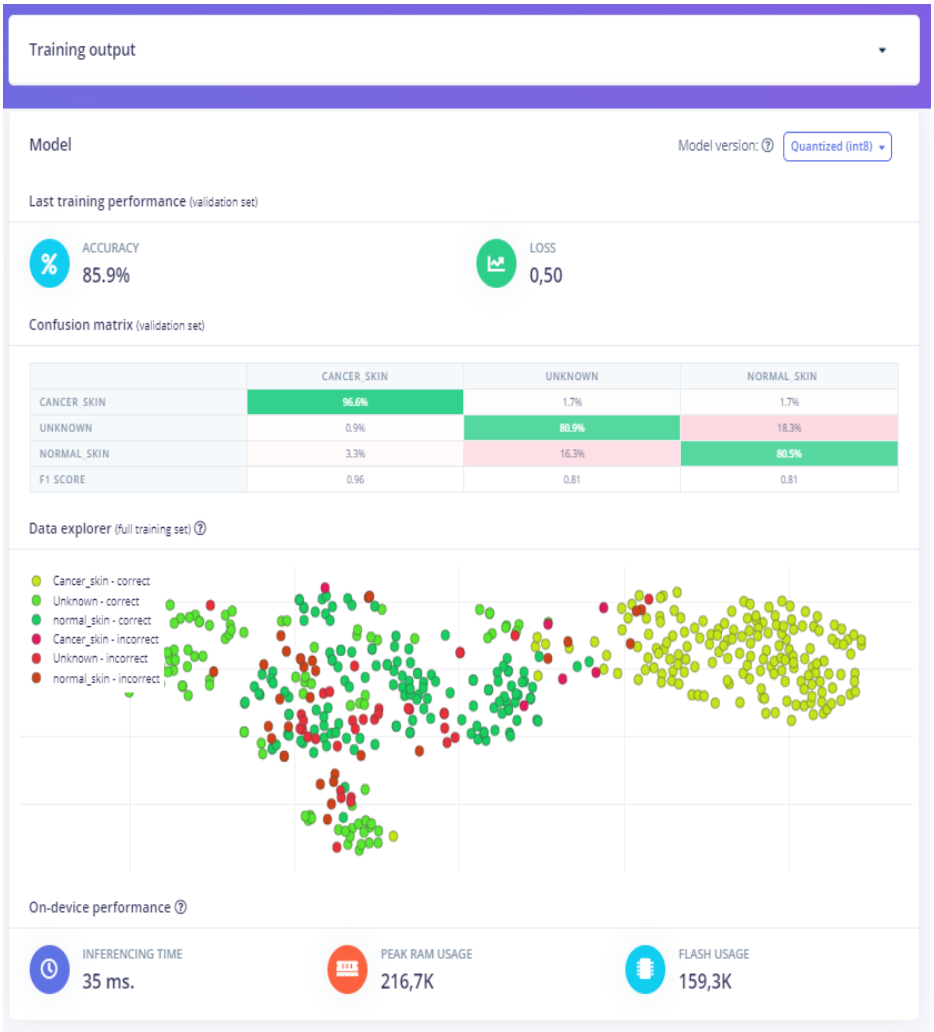
Training output :



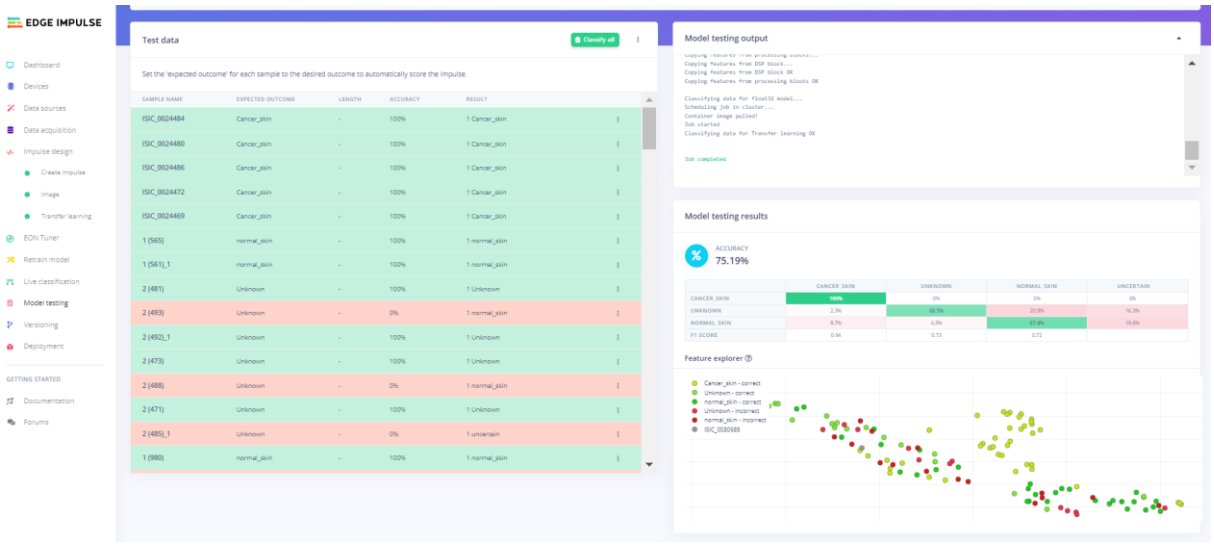
On constate que la taille de ram qu'occupera le model sur la carte est de 283 ,4k ce qui est supérieur à celle de la carte donc aller dans create impulse et modifier la taille des images



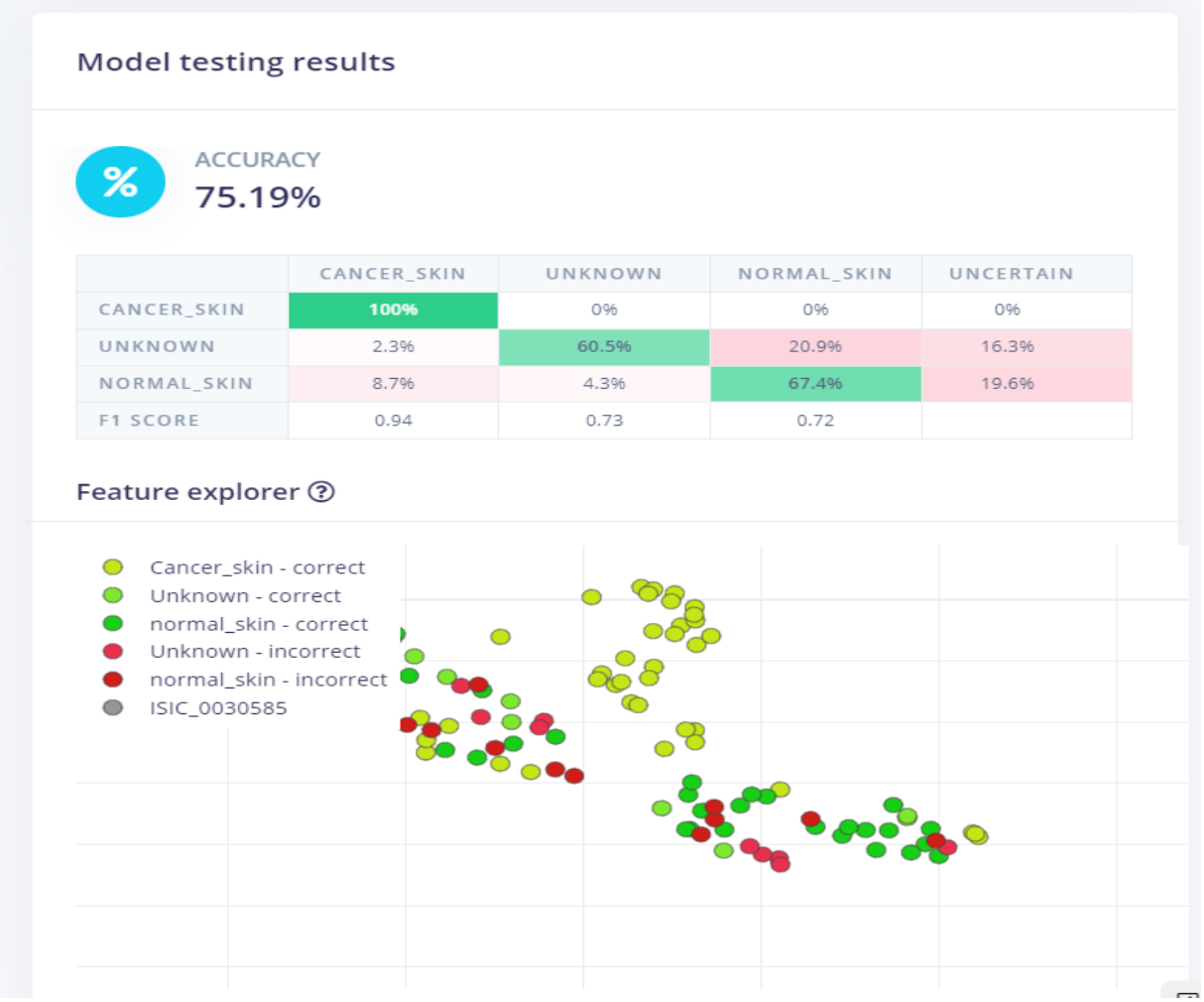
Enregistrer les nouveaux paramètres et relancer l'entrainement : on a



- **Model testing**

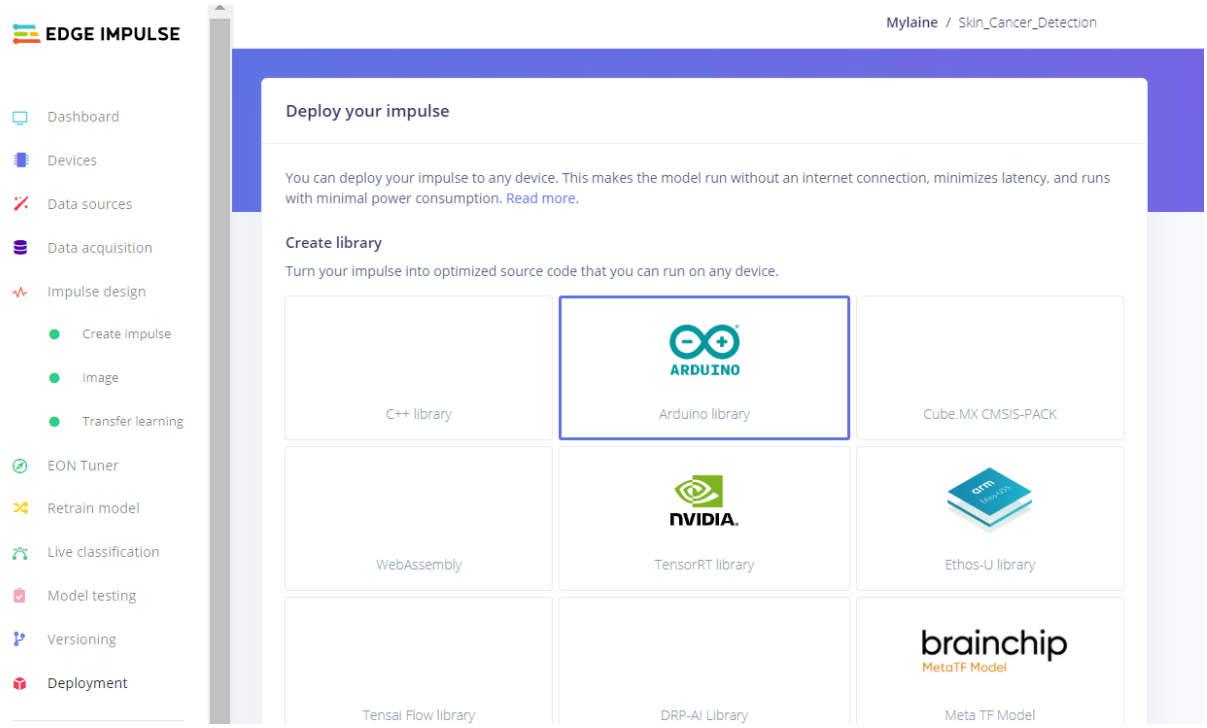


Model testing results :

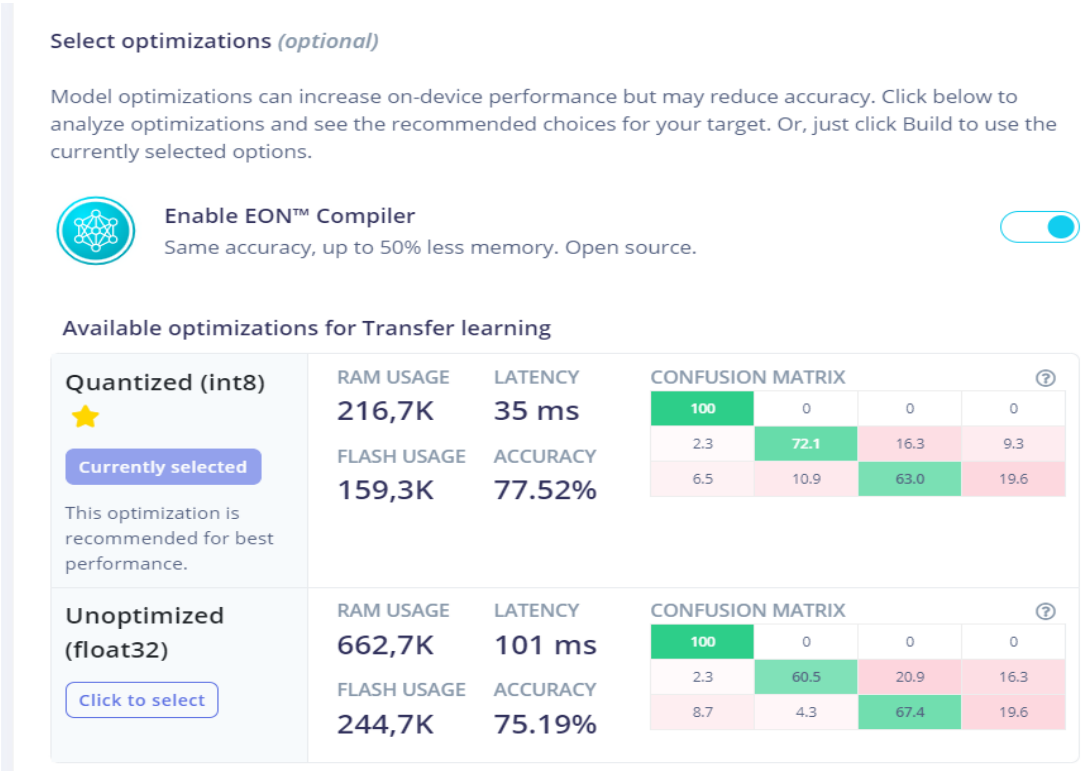


- **Deployement**

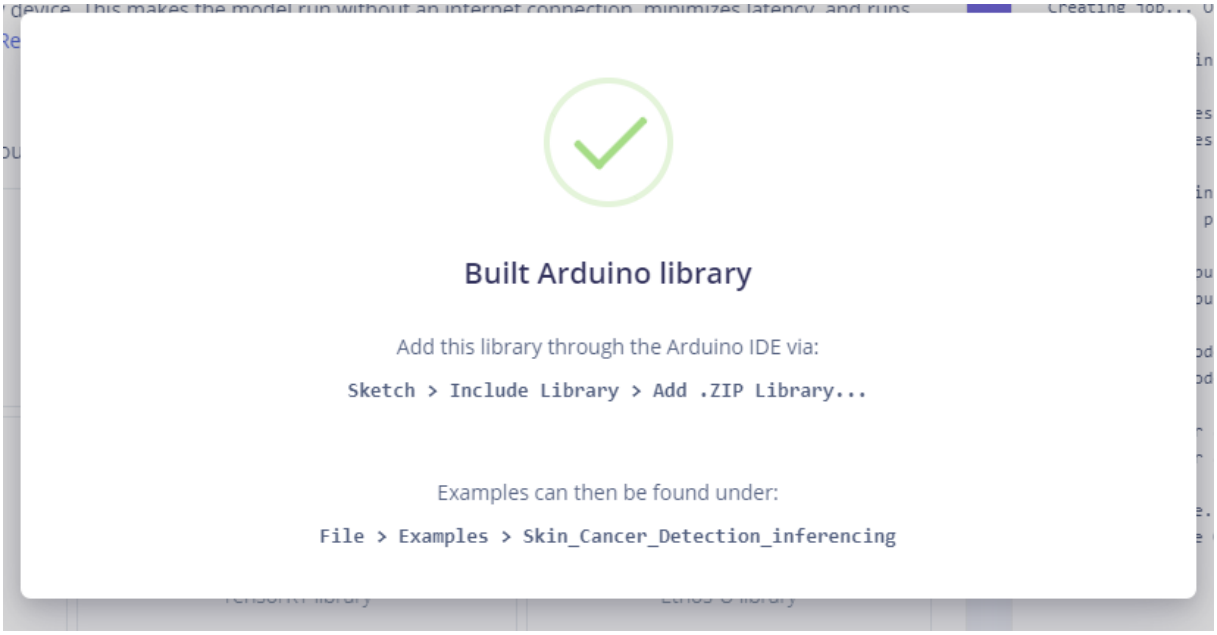
Aller dans deployement et sélectionner arduino (arduino Library)



Ensuite analyse optimizations



Et enfin build



B : Arduino IDE

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

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'entrainement

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-2SEM1xBbVAaUPEr7-xvO

L'entrainement 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_OdL8mj1Is-C-Tg_0StuMAeaZwjnNq3?usp=sharing

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