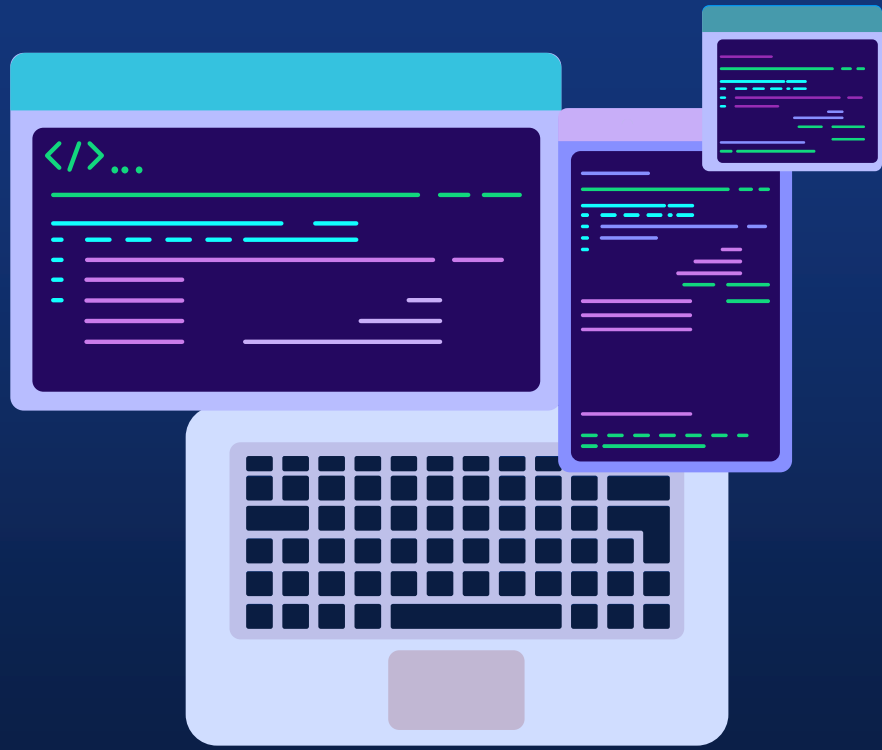
Présentation du labs



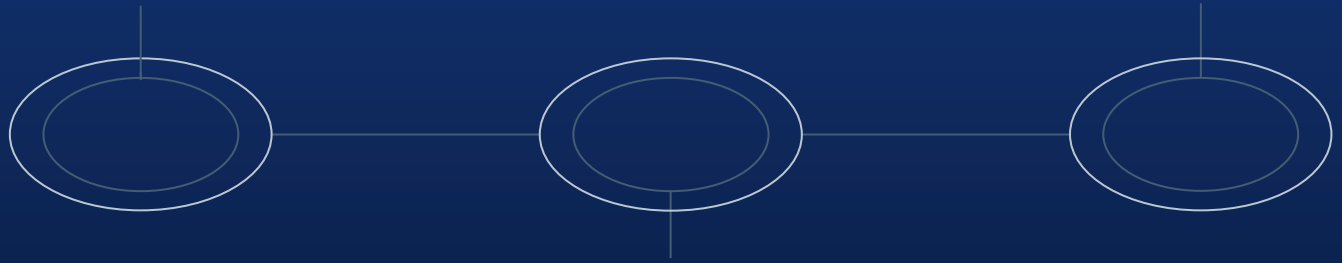
Réalisé par :
Nesrine Rouis
Rihab Mribah
3 DNI 1

PLAN

01
FACE DETECTION APPLICATION

02
SKIN CANCER DETECTION

03
CONCLUSION



01

FACE DETECTION APPLICATION

LES DIFFERENTES ETAPES DU MACHINE LEARNING :

PREPARATION DES DONNEES

Préparer et nettoyer les données brutes

LA MODELISATION

Choisir le bon algorithme d'apprentissage pour traiter le problème initial

L'EVALUATION

Entraîner et évaluer le modèle

LE DEPLOIEMENT

Tester et déployer le modèle

PARTIE 1 : FEATURES ENGINEERING

- Importation du data set.
- Extrait différents échantillons positifs dans les patches positifs.
- Obtenir un ensemble d'échantillons négatifs, puis prendre n'importe quel ensemble d'images d'entrée à partir de Scikit-Image et les patches ont été extraits à différentes échelles.
- Répétition de les mêmes étapes sur les échantillons négatifs.

PARTIE 2 : CLASSIFICATION BINAIRE A L ’AIDE DE SCIKIT-LEARN

- Divisons les données en ensembles d'apprentissage et de test.
- Construire le modèle avec différents types d'algorithmes tels que KNN, decision Tree, SVM et Random Forest.
- Calculer la précision de ces différents estimateurs pour découvrir le meilleur modèle.
- Tracer les performances de chaque modèle

PARTIE 3 : EVALUATION DE LA MEILLEUR DETECTEUR

- La création d’une fonction d'évaluation pour générer toutes les métriques des besoins .
- La prédictions à l'aide des données de test pour voir comment le meilleur modèle fonctionne.

LES DIFFERENTES METRIQUES :

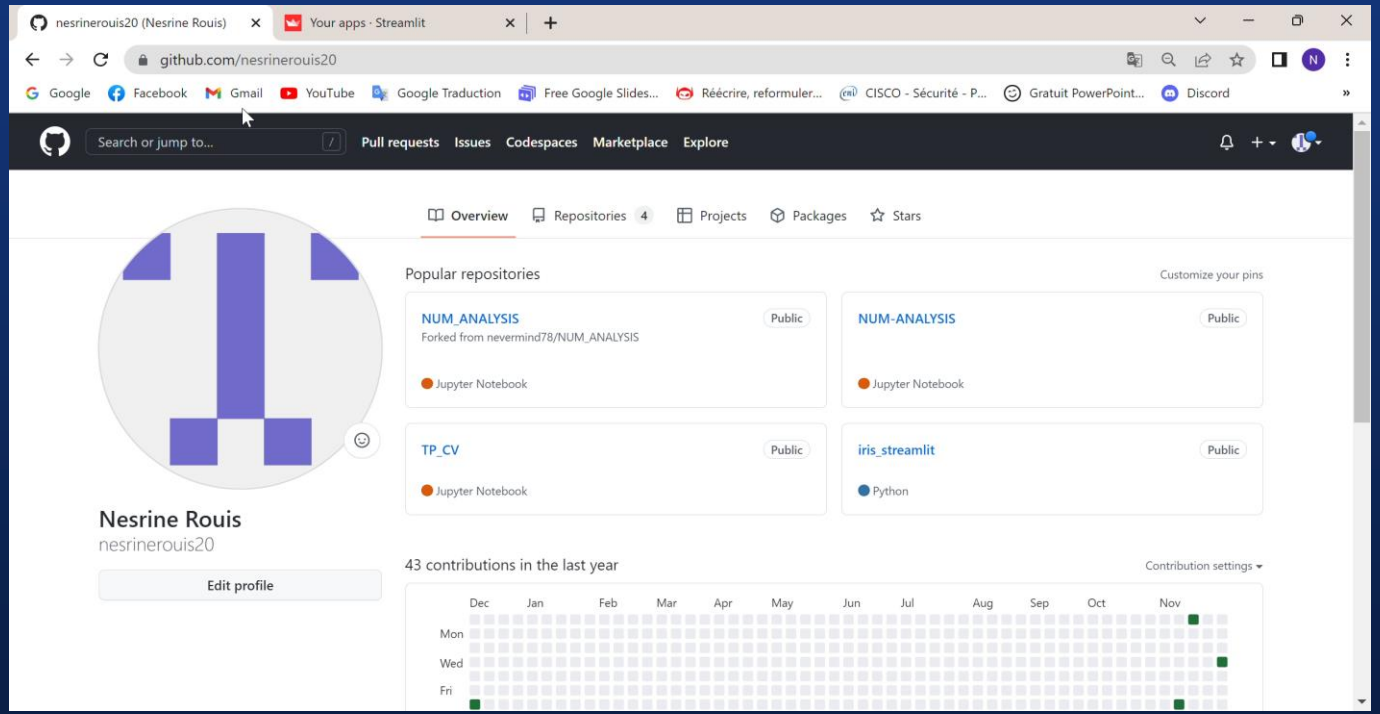
RECALL — Pourcentage d'exemples positifs qui sont correctement étiqueté.

PRECISION — Pourcentage d'étiquettes positives qui sont corrects.

ACCURACY — Pourcentage d'étiquettes correctes.

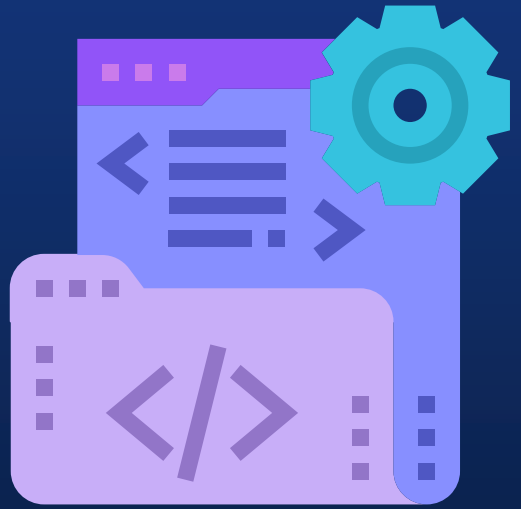
F1-MEASURE — Precision fois recall divisé par precision plus recall

Démo


nesrinerouis20 (Nesrine Rouis)
Your apps · Streamlit
github.com/nesrinerouis20
Google
Facebook
Gmail
YouTube
Google Traduction
Free Google Slides...
Réécrire, reformuler...
CISCO - Sécurité - P...
Gratuit PowerPoint...
Discord
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Overview
Repositories 4
Projects
Packages
Stars
Popular repositories
Customize your pins
NUM_ANALYSIS
Public
Forked from nevermind78/NUM_ANALYSIS
Jupyter Notebook
NUM-ANALYSIS
Public
Jupyter Notebook
TP_CV
Public
Jupyter Notebook
iris_streamlit
Public
Python
Nesrine Rouis
nesrinerouis20
Edit profile
43 contributions in the last year
Contribution settings
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Mon
Wed
Fri

02

SKIN CANCER DETECTION



PARTIE 1 : EFFICIENTNET_V2

```
[7]  module_selection = ("efficientnet_v2", 224, 1280)
     handle_base, pixels, FV_SIZE = module_selection
     MODULE_HANDLE ="https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b0/feature_vector/2".format(handle_base)
     IMAGE_SIZE = (pixels, pixels)
     BATCH_SIZE = 16
```

PARTIE 2 : DATA AUGMENTATION

```
[53] train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
          rescale = 1./255,
          rotation_range=0,
          horizontal_flip=True,
          width_shift_range=15,
          height_shift_range=15,
          shear_range=0.5,
          zoom_range=0.5,
          fill_mode='nearest',
          validation_split=0.4)
```

```
Found 24 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

PARTIE 3 : ARCHITECTURE DE CLASSIFIER

```
[56] print("Building model with", MODULE_HANDLE)
     model = tf.keras.Sequential([
         feature_extractor,
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(32, activation='relu'),
         tf.keras.layers.Dense(64, activation='relu'),
         tf.keras.layers.Dense(512, activation='relu'),
         tf.keras.layers.Dense(512, activation='relu'),
         tf.keras.layers.Dropout(rate=0.2),
         tf.keras.layers.Dense(train_generator.num_classes, activation='softmax',
                               kernel_regularizer=tf.keras.regularizers.l2(0.0001))
     ])
     #model.build((None,)+IMAGE_SIZE+(3,))

     model.summary()
```

```
Building model with https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b0/feature_vector/2
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              5919312

 flatten (Flatten)           (None, 1280)              0

 dense (Dense)               (None, 32)                40992

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 512)               33280

 dense_3 (Dense)             (None, 512)               262656

 dropout (Dropout)           (None, 512)               0

 dense_4 (Dense)             (None, 2)                 1026

=================================================================
Total params: 6,259,378
Trainable params: 340,066
Non-trainable params: 5,919,312
```

PARTIE 4 : LES TECHNIQUES DE REGULARISATION

Early Stopping

```
EPOCHS=150
BATCH_SIZE=20
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
history = model.fit(
        train_generator,
        steps_per_epoch=train_generator.samples//train_generator.batch_size,
        epochs=EPOCHS,
        validation_data=validation_generator,
        callbacks=[callback],
        validation_steps=validation_generator.samples//validation_generator.batch_size)
```

```
Epoch 1/150
1/1 [==============================] - 0s 457ms/step - loss: 0.2224 - accuracy: 1.0000
Epoch 2/150
1/1 [==============================] - 0s 111ms/step - loss: 0.3773 - accuracy: 0.7500
Epoch 3/150
1/1 [==============================] - 0s 341ms/step - loss: 0.2044 - accuracy: 0.9000
Epoch 4/150
1/1 [==============================] - 0s 326ms/step - loss: 0.2907 - accuracy: 0.9000
Epoch 5/150
1/1 [==============================] - 0s 336ms/step - loss: 0.3016 - accuracy: 0.9000
Epoch 6/150
1/1 [==============================] - 0s 109ms/step - loss: 0.0100 - accuracy: 1.0000
Epoch 7/150
1/1 [==============================] - 0s 111ms/step - loss: 0.0704 - accuracy: 1.0000
Epoch 8/150
1/1 [==============================] - 0s 332ms/step - loss: 0.1550 - accuracy: 0.9000
Epoch 9/150
1/1 [==============================] - 0s 101ms/step - loss: 0.2742 - accuracy: 0.7500
```

```
EPOCHS=50
BATCH_SIZE=20
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
history = model.fit(
        train_generator,
        steps_per_epoch=train_generator.samples//train_generator.batch_size,
        epochs=EPOCHS,
        validation_data=validation_generator,
        callbacks=[callback],
        validation_steps=validation_generator.samples//validation_generator.batch_size)
```

```
Epoch 1/50
1/1 [==============================] - 3s 3s/step - loss: 0.6930 - accuracy: 0.5000 - val_loss: 0.6957 - val_accuracy: 0.5000
Epoch 2/50
1/1 [==============================] - 1s 526ms/step - loss: 0.6945 - accuracy: 0.4375 - val_loss: 0.6978 - val_accuracy: 0.5625
Epoch 3/50
1/1 [==============================] - 1s 534ms/step - loss: 0.6646 - accuracy: 0.7500 - val_loss: 0.6845 - val_accuracy: 0.5625
Epoch 4/50
1/1 [==============================] - 1s 527ms/step - loss: 0.6418 - accuracy: 0.8125 - val_loss: 0.6736 - val_accuracy: 0.5000
Epoch 5/50
1/1 [==============================] - 0s 482ms/step - loss: 0.6262 - accuracy: 0.8750 - val_loss: 0.6294 - val_accuracy: 0.7500
Epoch 6/50
1/1 [==============================] - 0s 486ms/step - loss: 0.5534 - accuracy: 1.0000 - val_loss: 0.6229 - val_accuracy: 0.6875
Epoch 7/50
1/1 [==============================] - 0s 484ms/step - loss: 0.5592 - accuracy: 0.7500 - val_loss: 0.6012 - val_accuracy: 0.6875
Epoch 8/50
1/1 [==============================] - 0s 492ms/step - loss: 0.3751 - accuracy: 1.0000 - val_loss: 0.5923 - val_accuracy: 0.7500
Epoch 9/50
1/1 [==============================] - 1s 538ms/step - loss: 0.3147 - accuracy: 0.9375 - val_loss: 0.5568 - val_accuracy: 0.6250
Epoch 10/50
1/1 [==============================] - 0s 496ms/step - loss: 0.4290 - accuracy: 0.8750 - val_loss: 0.5418 - val_accuracy: 0.7500
Epoch 11/50
1/1 [==============================] - 1s 548ms/step - loss: 0.2684 - accuracy: 0.9375 - val_loss: 0.5346 - val_accuracy: 0.6250
```

MERCI POUR VOTRE ATTENTION !