

FACEX² - RECONNAISSANCE FACIALE

Traitement des signaux - Projet d'intégration

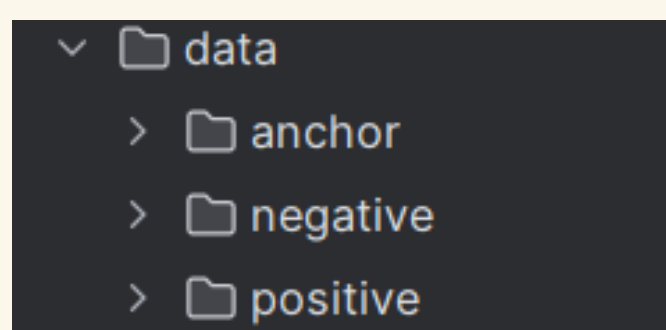
DELCORTE Maxime
DELCROIX Brice
ISTAS Loan

01. JEU DE DONNÉES

Les photos de Maxime sont Vrai
Les autres photos sont Fausse

Trois type de données :

- Anchor : Input / Vrai
- Positive : Images Vrai
- Negative : Images fausse



02. TRAITEMENT

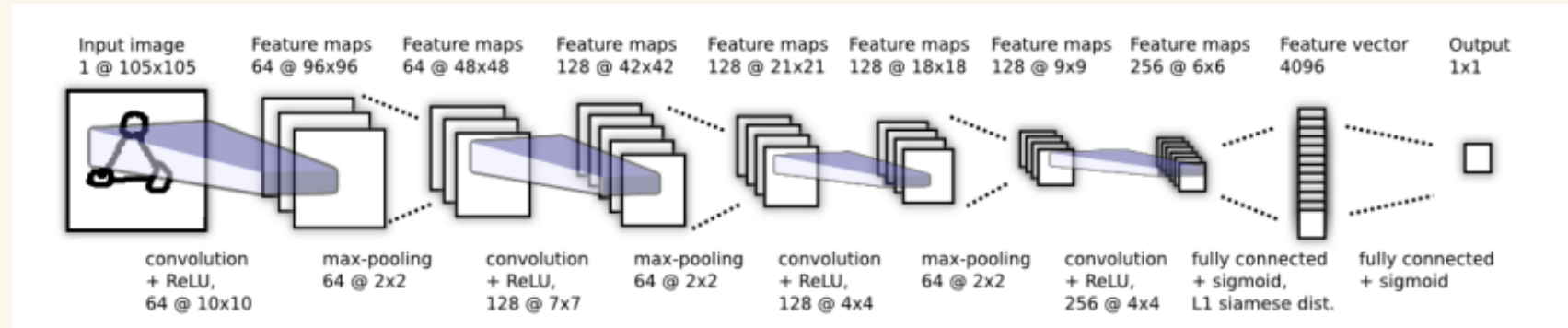
On effectue une série de traitements sur les images

- Format jpeg
- Resize 100x100 pixels
- Normalisation -> Renvoi un range [0,1]

```
<tf.Tensor: shape=(100, 100, 3), dtype=float32, numpy=
array([[0.18235295, 0.11176471, 0.06470589],
       [0.17720588, 0.10759804, 0.06838235],
       [0.16495098, 0.09828431, 0.05906863],
       ...,
       [0.2887255 , 0.19264705, 0.09901961],
       [0.2877451 , 0.19975491, 0.09436274],
       [0.28995097, 0.2017157 , 0.10563725]],
      dtype=float32)>
```

04. CONSTRUCTION DU MODÈLE

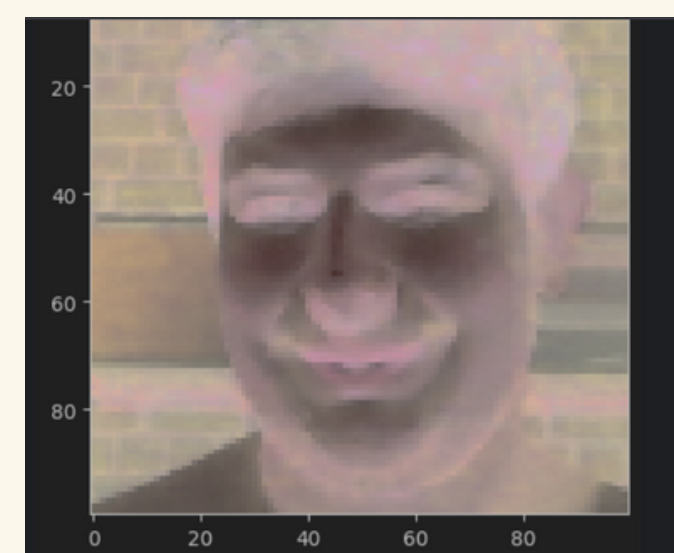
- Applications de filtres
- Rendre le jeu interprétable (entre 0 et 1) par le modèle
- Siamese Neural Networks for One-shot Image Recognition
- On gère la distance entre deux embedding



03. LABEL

Chaque image va prendre un label :

- Un label peut être 0 ou 1
- 1 : C'est maxime
- 0 : Ce n'est pas Maxime



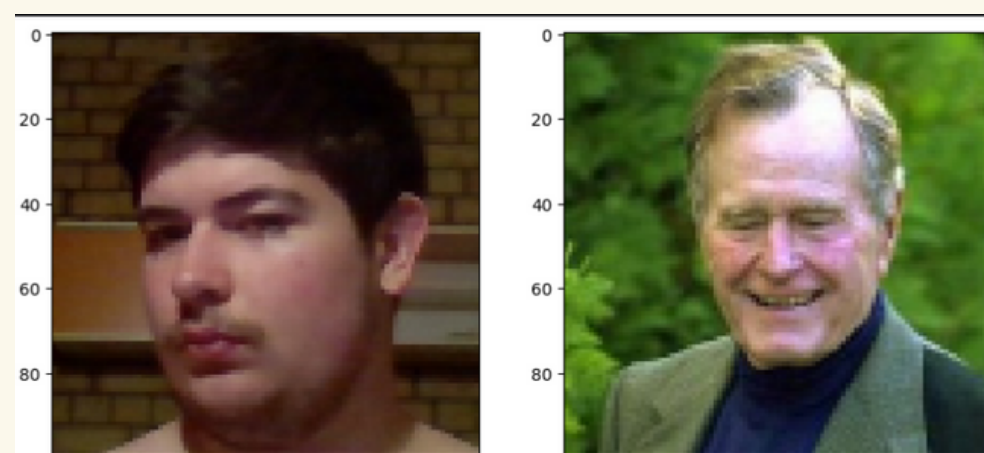
Label

1.0

05. CRÉATION DE LA COMPARAISON

Une image d'entrée et de vérification :

- On applique des filtres
- On attribue un score en comparant la distances des deux images.
- On renvoie une probabilité de 0 ou 1, positif ou négatif.



0.

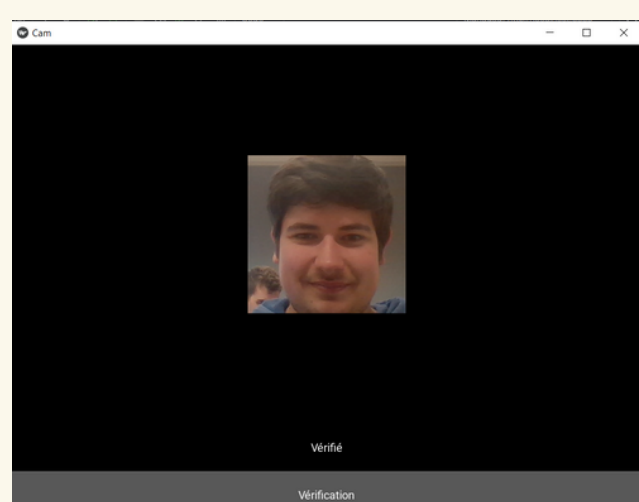
06. ENTRAÎNEMENT

- Il reçoit des images (ancres, positives, négatives) avec un label.
- Le modèle essaye de deviner et s'autocorrige avec le label.
- Il fait ça 100 fois avec 6 jeux de données.

```
>Epoch 97/100
6/6 11s 2s/step
>Epoch 98/100
6/6 11s 2s/step
>Epoch 99/100
6/6 11s 2s/step
>Epoch 100/100
6/6 11s 2s/step
```

08. INTERFACE

- GUI qui capture la webcam
- Compare l'image avec les images positives
- Répond positivement si maxime est sur l'image d'entrée sinon il répond négativement.



07. UTILISATION

- Sauvegarde du modèle pour utilisation.
- Nos critères de validation étaient 90% de résultat correct à l'entraînement.

```
[1 if prediction > 0.5 else 0 for prediction in y_hat]
[0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1]

y_true
array([0., 0., 0., 0., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1.],
      dtype=float32)

m = Recall()
m.update_state(y_true, y_hat)
m.result().numpy()
1.0
```