Projet Image M2 : Evaluation de la sécurité visuelle d'images obscures par CNN
HAI918I : Image, sécurité et deep learning

Université de Montpellier - FDS
2ième année Master IMAGINE
Oren AMSALHEM - Thomas CARO

10 Novembre 2024

1 Introduction

Cette semaine nous allons étudier la potentielle réversibilité de la méthode par distorsion et commencer à faire notre CNN.

2 Reversibilité de la distorsion

Nous allons tout d'abord rappeler les formules utilisées pour la distorsion :
Soit $I(x, y)$ une image d'entrée où (x, y) représente les coordonnées d'un pixel. Les transformations des coordonnées de chaque pixel sont définies comme suit :

2.1 Paramètres

- A : Amplitude de la distorsion. - f : Fréquence de la distorsion.

2.2 Calcul des Nouvelles Coordonnées

Pour chaque pixel à la position (x, y), les nouvelles coordonnées $(newX, newY)$ sont calculées en utilisant les formules suivantes :

$$newX = x + A \cdot \sin\left(2\pi \frac{y}{f}\right) \tag{1}$$

$$newY = y + A \cdot \sin\left(2\pi \frac{x}{f}\right) \tag{2}$$

Un premier problème que l'on peut avoir serait le problème des bords car en effet le calcul des nouvelles coordonnées peut sortir de l'image. Cela dépend de l'amplitude utilisée, et si on fait une distorsion de zone de la distance aux bords de cette zone.

Il y a 2 méthodes pour gérer ces bords. D'abord, on pourrait choisir de simplement prendre la valeur de la bordure lorsque les coordonnées dépassent. La seconde option serait de boucler sur l'image et de revenir de l'autre côté pour récupérer des valeurs. Voici en exemple avec les mêmes paramètres les différents résultats :

FIGURE 1 – Image de base

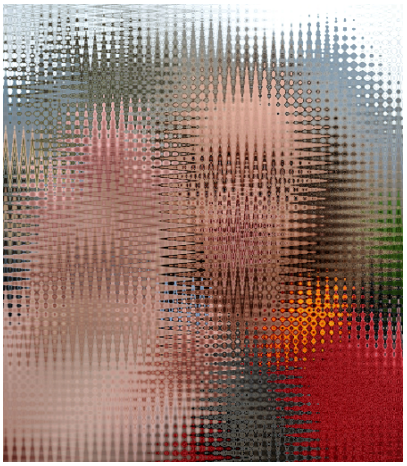

FIGURE 2 – Arrêt sur bordure

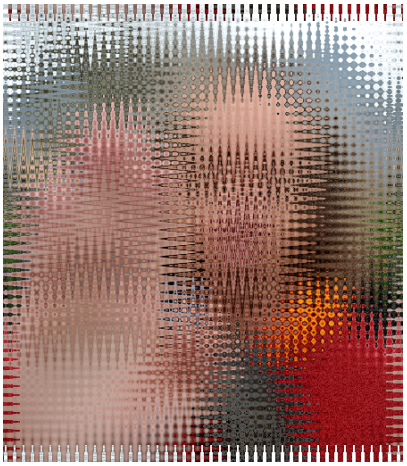

FIGURE 3 – Contour

FIGURE 4 – Différentes gestion de bord avec distorsion

Nous pourrions aussi calculer pour chaque image à l'aide la taille et de l'amplitude de la distorsion une zone morte pour laquelle sur les bords nous ne ferions rien.

Quant à la réversibilité totale de la distorsion nous nous heurtons à un gros problème. En effet nous n'avons aucune garantie de ne pas perdre l'information de certains pixels, nous pourrions très bien avoir 2 newX et newY identiques selon la position des pixels en train d'être remplacés. Une fréquence elevées augmenteraient encore plus les chances d'avoir 2 pixels associés au même.

Au final pour s'assurer qu'il n'y a aucune perte d'information il faudrait stocker les pixels perdus, valeur et position, lors de la distorsion, et à la fin de la réversion, lorsque l'on a des pixels manquant les remplir à l'aide de cette information.

Si l'on essaye tout de même de faire une réversion de l'image, par exemple avec la version qui s'arrête sur les bords on obtient le résultat suivant, le calcul se résume à retirer le terme avec amplitude au lieu de le rajouter :

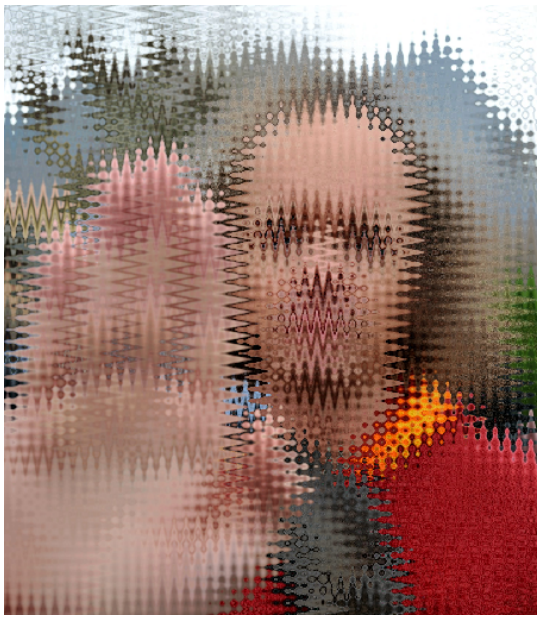

FIGURE 5 – Distorsion

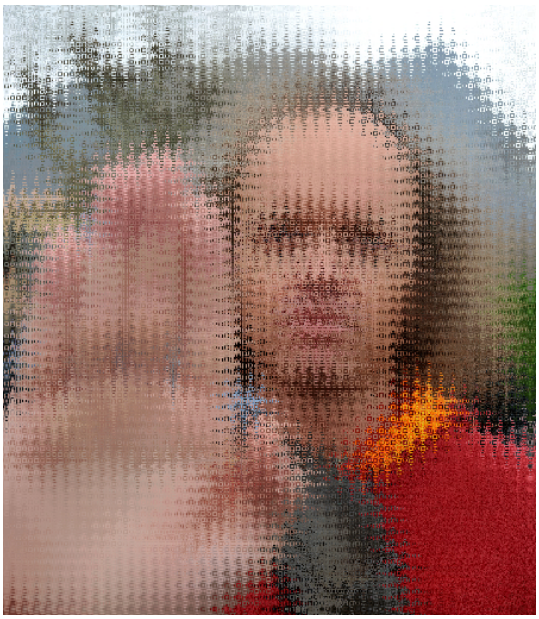

FIGURE 6 – Image inverse

FIGURE 7 – Réversibilité

Nous n'avons pas pu revenir à l'image de départ car comme prévu, nous avons en effet perdu des pixels lors de la distorsion.

3 Mise en place de CNN

Pour notre CNN nous avons décidé d'utiliser keras de TensorFlow en python étant donné que lors d'un TP de l'année nous l'avons déja utilisé. Notre première étape est de trouver un jeu de données satisfaisant. Pour que ce soit pratique d'obscurcir les données à l'aide d'un script python pour utiliser nos programmes C++, il faudrait que la partie de l'image a obscurcir soit la même pour toute. Pour faire cela nous avons décidé de prendre des images de visage bien centrés sur la caméra.

Tout d'abord nous avions trouvé une base de donnés de personne regardant tout droit, le problème était que celle-ci ne contenant que 72 images :

FIGURE 8 – Image type de la première base

Pour pallier à la faible quantité de donnés nous avons décidé d'augmenter la base de données et de créer 10 images augmentées pour chaque image du set avec des rotations, de zooms et de flips différents. Nous l'avons fait à l'aide d'ImageDataGenerator de keras. Voici une image de ce type :

FIGURE 9 – Image type de la première base augmentée

Ave ça nous avions environ 800 images, ce qui n'était toujours pas assez. Nous avons donc chercher une autre base de données semblable, et nous avons trouvé notre bonheur avec le set suivant : https ://www.kaggle.com/datasets/mahmoodulhaq/face-dataset ?resource=download

Voici le genre d'image proposée :

FIGURE 10 – Image du second set

Ce set comprend les visages de différentes personnes faites pour chaque personnes avec des angles différents, mais le visage est toujours centré ce qui nous a permis d'appliquer nos programmes C++ aux dossiers entiers. Nous avons donc combiné les 2 bases.

Pour le CNN de nous inspirer de VGG-19 et nous avons donc redimensionné nos images à la taille 224*224 :

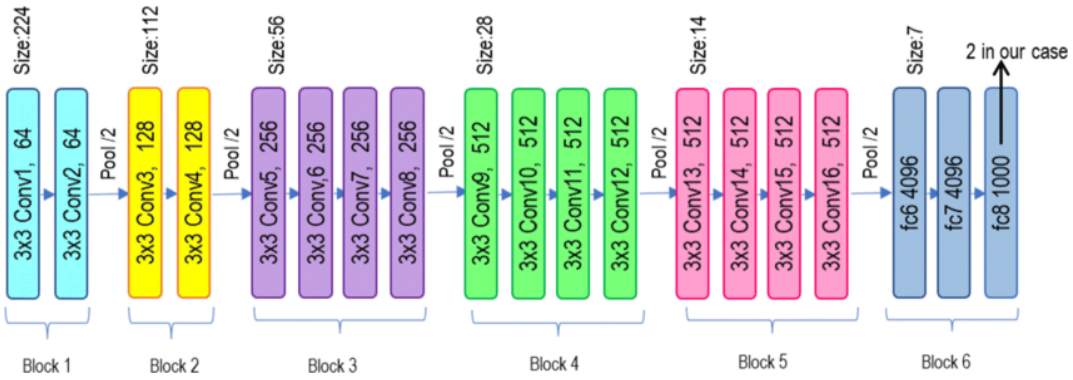

FIGURE 11 – Architecture VGG-19

Nous n'avons pas intégré autant de couches pour l'instant nous verrons plus tard si cela est intéressant.

Pour les classes que nous avons décidé de tester au départ nous avions prévu de faire 3 niveaux d'obscuration pour chaque type d'obscuration, en faisant donc varier les paramètres avec comme idée générale que le niveau 1 permet de toujours reconnaitre la personne , le niveau 2 permet de toujours distinguer le visage mais pas de reconnaitre la personne et le niveau 3 empêche de reconnaitre le visage.

Initialement nous avions décidé de générer notre modèle avec donc les 13 classes différentes mais après quelque tests nous nous sommes rendus compte que ce serait difficile. Au final, afin de bien comparer chaque méthode et chaque niveau d'obscuration nous avons décidé de faire un modèle pour chaque niveau d'obscuration, en paire avec la classe des images de base. Cela nous donnera au final 12 modèles dont nous testerons la précision et autres mesures afin de comparer nos différentes méthodes et niveaux d'obscuration.