# Projet Image - Compte-rendu 5

**Sécurité Visuelle - Obscuration d'image**

Loïc Kerbaul - Valentin Noyé

18 novembre 2024

## 1 Introduction

Dans ce cinquième compte-rendu, nous mettons en place quatre nouvelles méthodes d'obscuration, 3 classiques et 1 par réseau de neurones. Nous évaluons ensuite ces méthodes, et performons enfin une classification des différents types de méthodes utilisées afin de reconnaître si l'image a été modifiée, et avec quelle méthode parmi celles que nous avons implémentées.

## 2 Nouvelles méthodes d'obscuration

### 2.1 Chiffrement des bits

Comme précédemment, cette méthode utilise le chiffrement AES (grâce cette librairie) pour chiffrer une région sélectionnée, mais cette fois-ci uniquement sur un certain nombre de bits de chaque pixel, entre 1 (le bit de poids faible) et 8 (le bit de poids fort). Les exemples ci-dessous ont été chiffrés avec la clé `abcdef0123456798`.

**Utilisation :**

```
./aes_bits <entrée> <sortie> <x1> <y1> <x2> <y2> <clé de chiffrement> <nombre de bits chiffrés>
```

(a) Image originale (b) Uniquement le LSB (c) Les 5 premiers bits (d) Tous sauf le MSB

FIGURE 1 – Chiffrement par AES (sur certains bits uniquement) en mode CBC de la région

### 2.2 Distorsion géométrique

La méthode de distorsion géométrique consiste à remplacer chaque pixel de la région par un pixel avoisinant. Dans notre cas, nous avons utilisé les formules ci-dessous, qui se basent sur les caractéristiques d'un signal sinusoïdal ($X'$, $Y'$ est la position du pixel à placer en $X$, $Y$ sur l'image produite).

— Pour une distorsion horizontale :

$$X' = X + amplitude \times sin(2 \times \pi \times Y \times frequence)$$

$$Y' = Y$$

— Pour une distorsion verticale :

$$X' = X$$

$$Y' = Y + amplitude \times sin(2 \times \pi \times X \times frequence)$$

**Utilisation :**

```
./distorsion_sinus <entrée> <sortie> <x1> <y1> <x2> <y2> <amplitude> <fréquence>
<sens de la distorsion (0/1)>
```

(a) Image originale (b) Amplitude = 5 Fréquence = 0.99 (c) Amplitude = 10 Fréquence = 0.8 (d) Amplitude = 10 Fréquence = 0.98

FIGURE 2 – Distorsion géométrique selon sinusoïdale (avec Amplitude et Fréquence)

## 2.3 Distorsion colorimétrique

Nous avons implémenté une seconde technique de distorsion, dont le principe repose sur le mélange des trois canaux RGB : dans une région obscurcie, chaque pixel utilise les composantes R, G et B de trois pixels différents (cela peut aussi ne pas être le cas) dans son voisinage. Ainsi, plus le voisinage utilisé est étendu, plus l'image sera obscurcie. Dans notre cas, les composantes rouge et bleue sont récupérées dans le voisinage horizontal, tandis que la composante verte est récupérée dans le voisinage vertical.

**Utilisation :**

```
./distorsion_RGB <entrée> <sortie> <x1> <y1> <x2> <y2> <dxR> <dyG> <dxB>
```

(a) Image originale (b) dxR = -2, dyG = 2, dxB = 2 (c) dxR = -5, dyG = -5, dxB = 5 (d) dxR = 10, dyG = -1, dxB = 6

FIGURE 3 – Distorsion colorimétrique (décalage horizontal pour R et B, vertical pour G)

### 2.4 Obscuration par auto-encodeur

L'obscuration par auto-encodeur consiste en la modification des caractéristiques. Il fait suite à l'entraînement de notre modèle qui encode l'image d'entrée vers un domaine de caractéristiques, qu'il s'agit alors de décoder en appliquant des opérations de déconvolution et de sur-échantillonnage.

Le modèle utilisé est le même modèle que celui qui est introduit dans le précédent compte-rendu, mais est étendu par l'ajout d'une phase de décodage. D'abord, l'entraînement est effectué sur la librairie CIFAR-10, et le modèle peut être ensuite utilisé avec une image d'entrée aux dimensions $32 \times 32$.

Le script permet de choisir un nombre de caractéristiques $n$ (sélectionnées aléatoirement) et leur attribue des valeurs aléatoires comprises entre 0 et 1.

**Utilisation :**

```
python cifar-autoencoder-test.py <entrée> <sortie> <x1> <y1> <x2> <y2> <nombre de caractéristiques à modifier>
```

(a) Originale $32 \times 32$ (b) Auto-encodage (c) Obscuration à 25%

(d) Obscuration à 50% (e) Obscuration à 75% (f) Obscuration à 100%

FIGURE 4 – Obscuration par modification de l'espace latent

## 3 Évaluation des nouvelles méthodes d'obscuration

### 3.1 Évaluation classique

Pour évaluer de manière classique les résultats obtenus avec nos nouvelles méthodes d'obscuration, nous utilisons, comme précédemment, les mesures du PSNR (en dB), du SSIM et de l'entropie (en bits/canal/pixel).

| | **1 bit chiffré** | **5 bits chiffrés** | **7 bits chiffrés** |
|---|---|---|---|
| **PSNR** | 53.394 | 27.922 | 15.331 |
| **SSIM** | 0.999 | 0.981 | 0.770 |
| **Entropie** | 7.215 | 7.367 | 7.496 |

TABLE 1 – Évaluation de la méthode de chiffrement des bits

| | **A = 5 et F = 0.99** | **A = 10 et F = 0.8** | **A = 10 et F = 0.98** |
|---|---|---|---|
| **PSNR** | 19.165 | 16.428 | 16.505 |
| **SSIM** | 0.900 | 0.816 | 0.834 |
| **Entropie** | 7.189 | 7.199 | 7.189 |

TABLE 2 – Évaluation de la méthode de distorsion géométrique

| **(dxR, dyG, dxB)** | **(2, 2, 2)** | **(-5, -5, 5)** | **(10, -1, 6)** |
|---|---|---|---|
| **PSNR** | 21.145 | 17.166 | 16.796 |
| **SSIM** | 0.935 | 0.845 | 0.836 |
| **Entropie** | 7.190 | 7.197 | 7.188 |

TABLE 3 – Évaluation de la méthode de distorsion colorimétrique

| | **Obscuration à 25%** | **Obscuration à 75%** | **Obscuration à 100%** |
|---|---|---|---|
| **PSNR** | 16.400 | 13.627 | 12.048 |
| **SSIM** | 0.575 | 0.418 | 0.318 |
| **Entropie** | 7.320 | 7.266 | 7.260 |

TABLE 4 – Évaluation de l'obscuration par auto-encodeur

## 3.2 Classification par réseau de neurones

Nous nous attaquons à présent à l'évaluation des méthodes d'obscuration par réseau de neurones, en utilisant le même classifieur que dans le précédent compte-rendu, afin de faire sens avec les résultats obtenus. Nous notons que la méthode par auto-encodeur est *très peu performante* ; nous réduisons la taille du dataset entre 5% et 10% pour la plupart des classifications sur cette dernière. Afin de simplifier la lecture des données, la précision d'entraînement a été omise.

Les paramètres pour ces différentes méthodes varient de la sorte :

1. **Chiffrement de bits** : Une clé alphanumérique de 256 bits générée automatiquement ainsi que le nombre de bits de poids les plus faibles à modifier, $b \in [3, 8]$. La raison pour laquelle nous ne souhaitons pas utiliser une valeur entre 1 et 2 est car le résultat est plutôt typique d'un bruit naturellement généré qu'un bruit artificiel, ce que nous ne percevons pas forcément, et de même pour un réseau de neurones.
2. **Distorsion géométrique** : L'amplitude $A \in [1, 20]$, la frequence $f \in [0.1, 10]$ et le sens, soit horizontal, soit vertical.
3. **Distorsion colorimétrique** : La divergence colorimétrique $(dR, dY, dZ) \in ([-10, -5] \cup [5, 10])^3$.
4. **Modification de l'espace latent** : Le nombre de caractéristiques $n \in [0, 4]$ à modifier. 4 contrôle tout l'espace de caractéristiques.

Encore une fois, nous faisons également varier la région à offusquer lors d'évaluations contextuelles.

### 3.2.1 Sans entraînement sur des donneés obscurcies

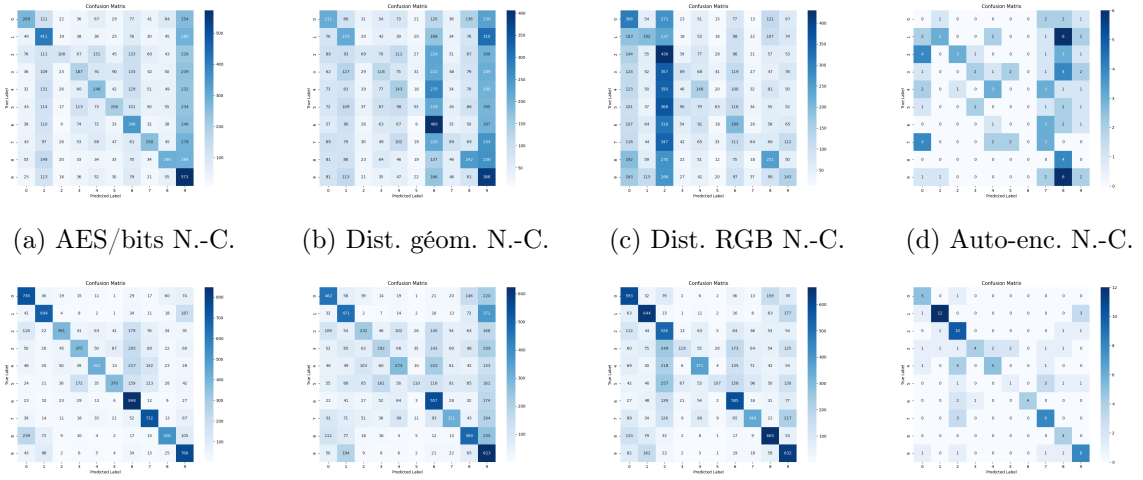
La première étape est de tester nos données sur un modèle entraîné à reconnaître des images naturelles. Le tableau 5 montre que l'ajout de contexte est important dans la majorité des

obscurations qui ont été menées sur CIFAR-10, doublant voire triplant la précision du modèle. Nous notons d'ailleurs que malgré la capacité de la distorsion à maintenir une image légèrement facile à interpréter à l'oeil nu, le modèle en est beaucoup moins capable.

| | **Chiffrement de bits** | **Dist. géom.** | **Dist. RGB** | **Auto-encodeur** |
|---|---|---|---|---|
| **Non-contextuel** | 28,84% | 19,34% | 18,87% | 20,00% |
| **Contextuelle** | 58,62% | 37,22% | 45,91% | 61,00% |

TABLE 5 – Précision de la classification obtenue par chaque méthode

La figure 5 montre ce que nous attendions d'un tel modèle au vu des précisions obtenues. Nous notons qu'avec un faible contexte, nous avons une forte tendance du modèle à classifier de manière biaisée. En outre, l'observation que nous pouvons mener sur la structure de ces matrices se rapproche de celle qui est discutée dans le précédent compte-rendu.

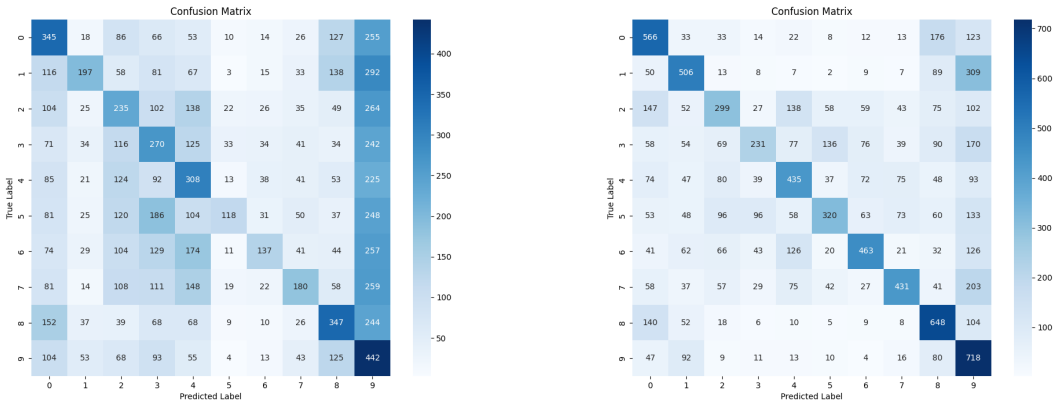


(a) AES/bits N.-C. (b) Dist. géom. N.-C. (c) Dist. RGB N.-C. (d) Auto-enc. N.-C.

(e) AES/bits C. (f) Dist. géom. C. (g) Dist. RGB C. (h) Auto-enc. C.

FIGURE 5 – Matrices de confusion obtenues lors de la classification

À présent, en faisant l'évaluation de la classification sur toutes les méthodes combinées, nous obtenons une précision de 25,79% sans contexte et 46,17% avec, ce qui signifie que notre modèle, sans entraînement sur des données offusquées, est capable de reconnaître le quart jusqu'à la moitié des images du dataset même après leur obscuration. Cela s'avère donner des résultats légèrement meilleurs qu'auparavant.




(a) Obscuration non contextuelle (b) Obscuration contextuelle

FIGURE 6 – Matrices de confusion obtenues lors de la classification avec toutes les méthodes

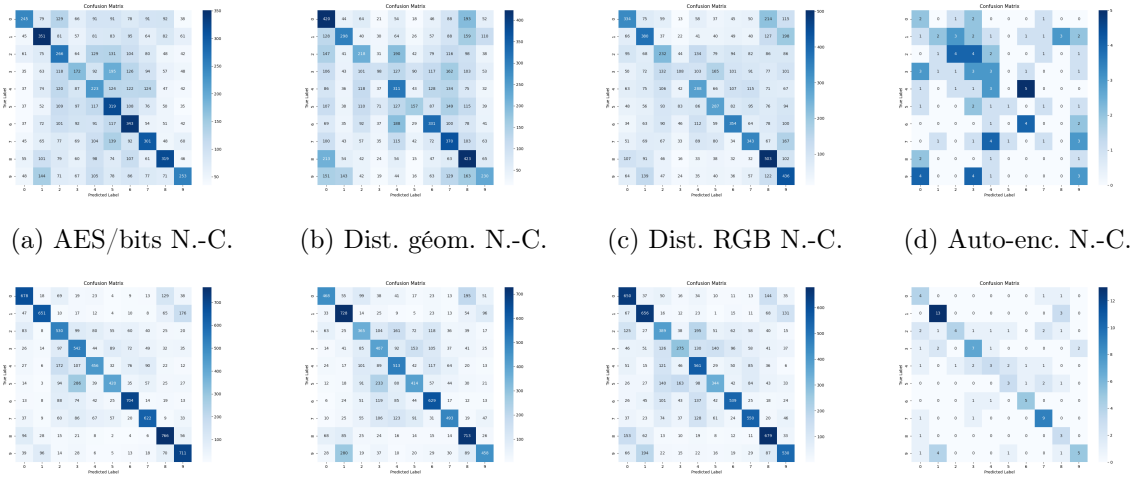
### 3.2.2 Avec entraînement sur des donneés obscurcies

En entraînant à présent le modèle, les précisions obtenues dans le tableau 6 mettent en évidence une amélioration de la classification, contextuelle ou non, pour les deux méthodes de distorsion. Or, les deux autres méthodes ne voient pas de fortes améliorations car leur obscuration dépend moins du contenu visuel de l'image.

| | **Chiffrement de bits** | **Dist. géom.** | **Dist. RGB** | **Auto-encodeur** |
|---|---|---|---|---|
| **Non-contextuel** | 27,92% | 28,56% | 32,65% | 23,00% |
| **Contextuelle** | 60,80% | 51,88% | 51,73% | 56,00% |

TABLE 6 – Précision de la classification obtenue par chaque méthode

La figure 7 constitue l'ensemble des matrices de confusion obtenues durant nos tests pour ces méthodes. Sauf pour une évaluation de l'obscuration par auto-encodeur non-contextuelle, nous avons des diagonales très marquées, ce qui signifie une bonne classification.

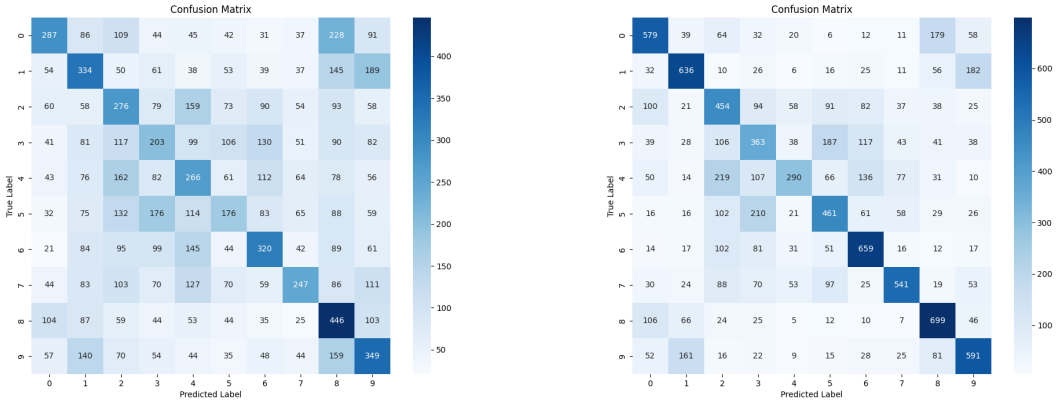


(a) AES/bits N.-C. (b) Dist. géom. N.-C. (c) Dist. RGB N.-C. (d) Auto-enc. N.-C.

(e) AES/bits C. (f) Dist. géom. C. (g) Dist. RGB C. (h) Auto-enc. C.

FIGURE 7 – Matrices de confusion obtenues lors de la classification

En employant toute notre panoplie de méthodes d'obscurcissement, l'entraînement préalable de notre modèle augmente les capacités de notre modèle à classifier malgré l'obscuration présente sur les diverses images du dataset (voir figure 8). Sans contexte, nous avons une précision de 29,04% et 52,73% avec contexte, ce qui est très bon, en sachant que notre modèle avait une précision de 70% à 80% sur CIFAR-10 inchangé.




(a) Obscuration non contextuelle

(b) Obscuration contextuelle

FIGURE 8 – Matrices de confusion obtenues lors de la classification avec toutes les méthodes

### 3.2.3 Détection de l'offuscation

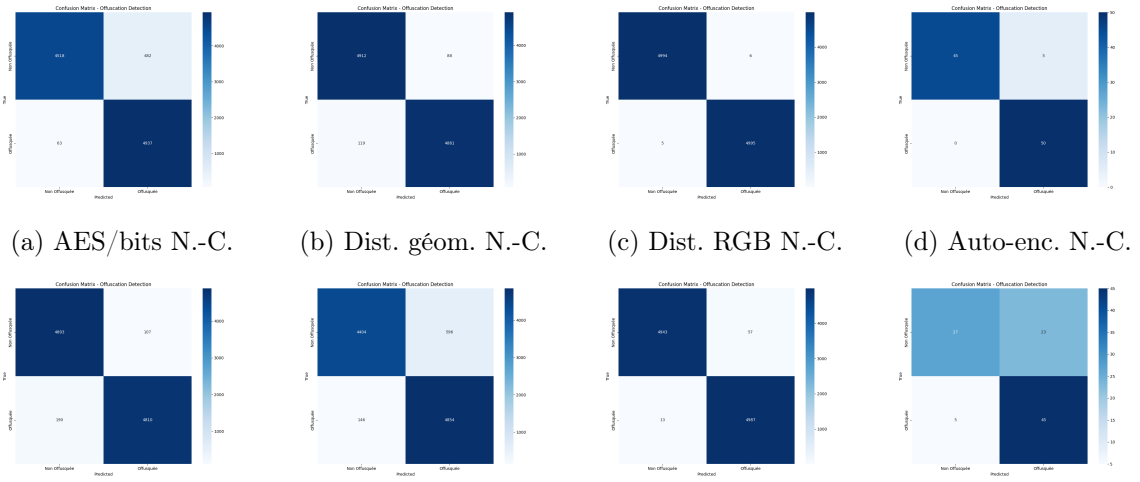
Nous effectuons la détection de l'offuscation sur l'image après avoir entraîné notre modèle. La figure 7 contient les résultats obtenus, qui s'avèrent notamment intéréssants pour l'obscuration par auto-encodeur qui considère plus facilement une image modifiée comme plus naturelle face aux autres méthodes d'obscuration. En revanche, nous pouvons également attribuer cet effet au nombre réduit de données d'entraînement et de test pour cette méthode-ci.

Par ailleurs, le chiffrement par bits et les deux méthodes de distorsion sont très précisémment détectées par notre modèle lorsqu'il y a peu voire aucune information non-offusquée dans l'image. Nous considérons toujours les 6 bits de poids les plus forts pour le chiffrement sur les bits.

| | **Chiffrement de bits** | **Dist. géom.** | **Dist. RGB** | **Auto-encodeur** |
|---|---|---|---|---|
| **Non-contextuel** | 95,30% | 97,93% | 100,00% | 95,00% |
| **Contextuelle** | 97,03% | 92,58% | 99,30% | 71,00% |

TABLE 7 – Précision de la reconnaissance obtenue par chaque méthode

La figure 9 met en évidence les performances de ce modèle, et notamment le fort taux de faux positifs lorsque le modèle tente de reconnaître une image manipulée par auto-encodeur ou non.



(a) AES/bits N.-C. (b) Dist. géom. N.-C. (c) Dist. RGB N.-C. (d) Auto-enc. N.-C.

(e) AES/bits C. (f) Dist. géom. C. (g) Dist. RGB C. (h) Auto-enc. C.

FIGURE 9 – Matrices de confusion obtenues lors de la reconnaissance

## 4 Reconnaissance de la méthode d'obscuration

Nous avons crée un nouveau réseau de neurones basé sur le précédent, capable de déterminer quelle méthode d'obscuration parmi celles que nous avons proposé est utilisée sur une image. Nous avons alors 9 classes, la première étant **aucune méthode** d'utilisée, et les 8 autres sont celles que nous avions discuté tout au long de ce projet Image.

La reconnaissance s'est avérée très correcte, avec une précision de 74,70% sur l'ensemble des images sur lesquelles nous avons fait une obscuration sans introduction de contexte. En revanche, nous obtenons une précision de 77,10% avec contexte, ce qui n'est qu'une légère amélioration, mais qui reste fortement intéressante puisque nous reconnaissons maintenant avec certitude si plus de trois quarts des images ont été obscurées et la méthode d'obscuration utilisée.

Ces divers aléas peuvent être constatés dans la figure 10 et être expliqués par la présence de similarités entre certaines méthodes. Tout d'abord, il est clair que les deux méthodes utilisant un chiffrement AES allaient engendrer des similarités entre elles et donc influencer cette classification. Par ailleurs, cette la méthode AES sur les bits fait remarquer que le modèle a tendance à confondre l'image légèrement chiffrée avec du bruit naturel, et donc classifier en conséquence en tant qu'image non-modifiée. Parfois encore, l'obscuration non-contextuelle par auto-encodeur laisse penser que l'image est floutée, qui est un défaut de notre implémentation, mais dès que ce contexte est donné, l'image est alors plus aisément considérée comme inchangée.


(a) Obscuration non contextuelle

(b) Obscuration contextuelle

FIGURE 10 – Matrices de confusion pour la reconnaissance des méthodes d'obscuration