Génération d'Images et Insertion d'un Message Caché dans l'Espace Latent de *Glow*

Aurélien NOIRAULT[1] Jan BUTORA[1] Vincent ITIER[1,2] Patrick BAS[1]

[1]Centre de Recherche en Informatique, Signal et Automatique de Lille, Avenue Henri Poincaré, 59655 Villeneuve d'Ascq, France

[2]IMT Nord Europe, Institut Mines-Télécom, Centre for Digital Systems, F-59000 Lille, France

Résumé – En stéganographie générative, un message est inséré dans l'espace latent d'un générateur d'image. Le décodage de ce message suppose de pouvoir inverser la génération et de retrouver ce vecteur. Cependant, pour être sauvegardée, l'image doit être quantifiée, ce qui crée une erreur dans l'inversion et empêche le décodage du message. Nous proposons une méthode d'optimisation permettant de rendre robuste un algorithme d'insertion à cette distorsion. L'ajout d'un vecteur de biais appris par descente de gradient au vecteur latent permet de générer une image quantifiée et donc d'obtenir un taux d'erreur nul.

Abstract – In generative steganography, a message is embedded into the latent space of an image generator. Decoding this message means being able to reverse the generation and retrieve the vector. However, to be saved, the image must be quantized, which creates an error in the inversion and prevents the message from being decoded. We propose an optimization method to make an insertion algorithm robust to this distortion. By adding a bias vector learned by gradient descent to the latent vector, we can generate a quantized image and thus obtain a null bit error rate.

1 Introduction

La stéganographie est l'art de cacher un message dans un contenu anodin, par exemple une image. Contrairement à la cryptographie traditionnelle, le but n'est pas tant de rendre le contenu du message illisible que de rendre la transmission anodine. Le principe d'origine de la stéganographie est de partir d'une image de couverture et d'y insérer un message secret en introduisant des petites distorsions dans l'image (généralement des opérations +1 ou -1 sur les valeurs des pixels). Bien que les algorithmes modernes de stéganographie soient très efficaces pour préserver les propriétés statistiques de l'image, ils ne sont pas sans failles (*i.e.* la stéganographie est détectable) et ont tendance à avoir une faible capacité d'insertion (*i.e.* la taille du message secret est limitée). Ce compromis entre la capacité d'insertion et l'indétectabilité est une limite des algorithmes de stéganographie utilisant des médias de couverture.

Avec le développement des modèles génératifs ces dernières années, un nouveau paradigme de stéganographie s'appuyant sur ces générateurs est apparu : au lieu d'insérer un message directement dans une image, le but est de générer le média de couverture directement à partir du message. Comme le contenu sémantique de l'image de couverture n'est pas contraint, les méthodes de stéganographie générative se concentrent sur la génération d'une image contenant directement le message secret.

De nombreux travaux proposent d'utiliser des GANs (Générateur Adverses Neuronaux) pour atteindre ce but. Une des premières méthodes proposée [4] consiste à cacher le message dans le bruit qui sert d'entrée au générateur. La partie adversaire du GAN est, ici, constituée de deux réseaux différents, un décodeur qui apprend à décoder le message de l'image et un classifieur qui permet d'ajouter la contrainte de l'indétectabilité. Cependant, l'entraînement simultané des trois architectures (partie adversaire et générateur) complexifie l'entraînement adverse des GANs. De plus, les générateurs de type GANs restent limités dans la variété des images qu'ils peuvent produire.

Pour limiter ce problème et augmenter la qualité des images générées, les approches récentes utilisent des architectures telles que les modèles de diffusion (principalement les Modèles de Diffusion Latents, MDL) ou les modèles de générateurs inversibles tels que Glow [3]. Ces modèles génèrent une image à partir d'un vecteur latent échantillonné d'une distribution gaussienne multivariée. Ces générateurs sont inversibles ou, du moins, il existe une approximation de leur inversion, ce qui permet d'obtenir une correspondance entre l'espace latent et l'espace pixel. C'est sur ce principe que reposent les méthodes telles que S2IRT [7], PRC [2] ou encore celle proposée par Thang *et. al.* [6]. Dans ces papiers, les auteurs proposent d'insérer un message dans l'espace latent, de générer l'image et d'inverser la génération pour pouvoir décoder le message secret. Cependant, il existe deux problèmes liés à ce type d'approches

- là où le modèle Glow est conçu avec des couches inversibles, les MDL classiques nécessitent un solveur d'équations différentielles ordinaires ou un algorithme comme DDIM [5] pour retrouver le vecteur latent. Malheureusement, ce genre de méthodes ne sont que des approximations de l'inversion et ne permettent pas de retrouver exactement le vecteur latent d'origine, ce qui peut créer des erreurs de décodage du message secret inséré ;
- les générateurs produisent des images dans un format continu. Pour pouvoir les sauvegarder, il faut les quantifier et bien souvent les restreindre à une plage de valeurs donnée, *e.g.* $[0, \ldots, 255]$ pour un codage 8 bits. Ces opérations anodines provoquent une erreur considérable lors de l'inversion pouvant faire s'effondrer le taux de récupération du message secret. De plus, les messages insérés sont très souvent chiffrés et les bits erronés, dans

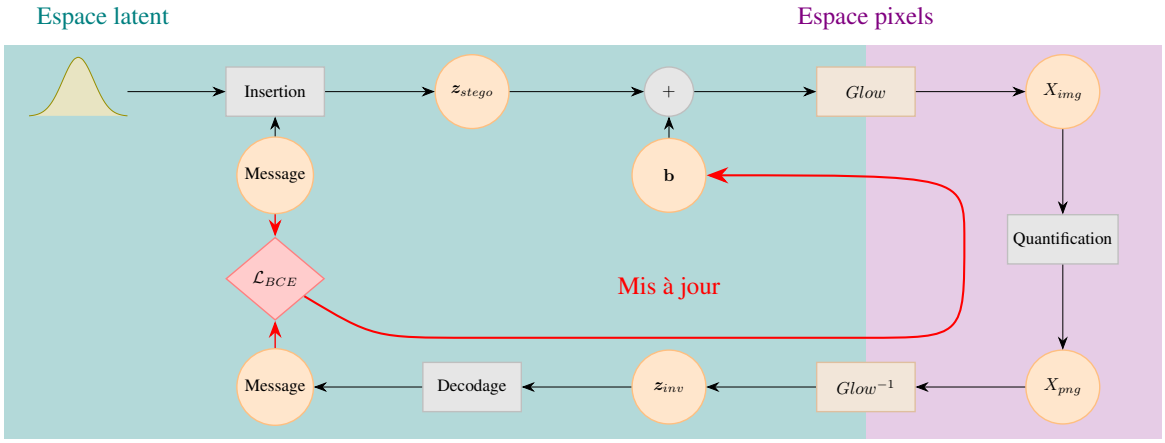


FIGURE 1 : Principe de notre méthode : lors de l'insertion, le biais $\mathbf{b}$ permettant de compenser l'erreur de décodage due à la quantification de l'image est appris via une descente de gradient.

le message décodé, peuvent compromettre le déchiffrement. Notons que le problème de la quantification de l'image générée est très peu pris en compte dans la littérature actuelle. Ce point est pourtant crucial dans la mise en place de systèmes opérationnels.

Les deux méthodes principales pour limiter les erreurs au décodage sont l'utilisation de codes correcteurs d'une part et de *zones mortes* d'autre part. Bien que les codes correcteurs permettent de corriger le message en fonction de leurs capacités, ils nécessitent l'ajout d'informations redondantes ce qui décroît la capacité d'insertion effective du message. L'approche la plus courante pour insérer un message dans le vecteur latent est d'attribuer un label à une partie de la distribution, la valeur de chaque échantillon porte alors une partie du message et des zones mortes sont alors utilisées comme séparations supplémentaires entre les zones de codage. Les zones mortes permettent de décoder le message tant que l'erreur reste inférieure à la largeur de ces zones. Cette technique est très efficace pour réduire le taux d'erreurs, mais réintroduit un compromis entre la capacité d'insertion et la robustesse à la quantification. En outre, l'utilisation de zones mortes trop larges affecte la qualité de la génération tant du point de vue de la détectabilité que de la qualité visuelle de l'image produite.

Nous proposons d'utiliser une méthode de tatouage numérique, faisant usage de zones mortes, pour insérer un message dans l'espace latent du générateur d'image Glow. Nous proposons également d'utiliser une descente de gradient pour améliorer les performances de l'insertion et atteindre un taux d'erreur binaire nul.

Dans ce travail, nous détaillons en section 2 la méthode proposée, puis, en section 3, nous comparons plusieurs algorithmes d'insertion robustes et montrons le gain offert par notre méthode sur le taux d'erreur de décodage. Nous ferons une rapide synthèse en section 4.

2 Insertion dans l'espace latent

Grâce à la nature parfaitement inversible de Glow, il existe des points qui génèrent des images avec une erreur de quantification nulle que nous choisissons d'appeler des points fixes. Ces points de l'espace latent ne subissent donc pas de distorsions liées à la quantification de l'image et ne changent pas de position après inversion de la fonction générative. En attribuant un message à différentes parties de l'espace latent, insérer un message revient à chercher un point fixe appartenant à une partie de l'espace latent qui contient le message souhaité.

Dans la littérature actuelle, peu d'algorithmes tirent parti de la connaissance du générateur. Cependant, il est possible d'utiliser le générateur comme un oracle afin de prévoir les dégradations que produira la quantification de l'image. Dans cette optique, nous proposons une méthode d'optimisation de ce vecteur latent permettant de décoder sans perte un message malgré la quantification de l'image. Le cadre de cette optimisation est illustré en Figure 1.

Cette démarche itérative est gourmande en termes de calcul, pour limiter le surcoût de calcul, nous utiliserons un modèle Glow déjà entraîné sur une base de visages pour générer des images RGB de 64x64 pixels à partir d'un vecteur de taille 3072.[1]

2.1 Insertion

L'algorithme d'insertion très populaire en tatouage numérique, appelé *Quantization Index Modulation* (QIM) [1] fait usage de la quantification pour insérer un message dans un signal. Cet algorithme transforme l'échantillon original x en un échantillon quantifié avec une grille de quantification dépendant du mot inséré m, tel que :

$$Q_\Delta(x, m) = \text{Round}\left(\frac{x}{\Delta}\right) \times \Delta + m \times \frac{\Delta}{N}, \qquad (1)$$

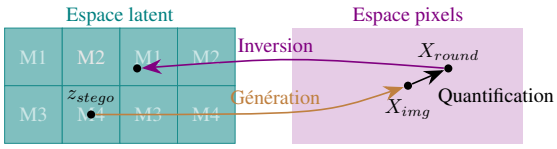
avec Δ le pas de quantification, N la taille du dictionnaire et $\text{Round}(\cdot)$ une fonction d'arrondi. Le décodage se fait en insérant les différents mots du dictionnaire dans le signal reçu y pour trouver l'échantillon d'origine (associé au mot $\hat{m}$) le plus proche de l'échantillon reçu :

$$\hat{m} = \underset{m}{\text{argmin}} \, ||y - Q_\Delta(y, m)||. \qquad (2)$$

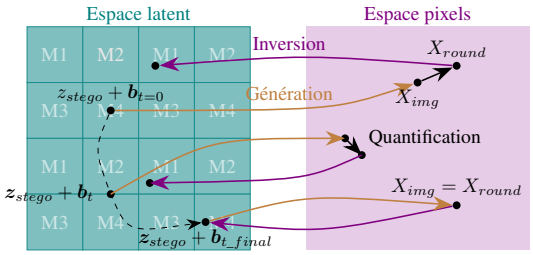
L'algorithme QIM revient à se placer dans un cas extrême de zone morte où tous les échantillons sont concentrés au centre de la cellule de quantification. Dans ce cas, la largeur de la zone morte est de $\Delta/2$, QIM peut alors décoder un message tant que l'erreur reste inférieure à $\Delta/4$.

[1] github.com/chaiyujin/glow-pytorch

2.2 Exploration de l'espace latent



(a) Nécessité d'avoir une méthode d'insertion robuste : après insertion avec QIM du mot M4, le vecteur latent se situe dans un hypercube de l'espace latent, la quantification introduit une perturbation qui fait sortir le vecteur de ce cube.



(b) Principe de l'insertion : à l'instant initial $t = 0$, un mot M4 est inséré dans le vecteur latent ce qui le place au centre de l'hypercube. À chaque itération t, un vecteur de biais $\boldsymbol{b}$ est ajouté au vecteur latent. Ce vecteur $\boldsymbol{b}$ est mis à jour avec une descente de gradient à chaque itération afin de déplacer le vecteur latent vers un point fixe.

FIGURE 2 : Illustration géométrique de l'insertion d'un mot dans un espace latent sans (2a) et avec (2b) exploration de l'espace latent.

Insérer un message dans un vecteur latent avec la méthode QIM revient à découper l'espace latent en hypercubes de côté $\Delta/2$ et à placer ce vecteur au centre de cet hypercube. La quantification de l'espace image va potentiellement déplacer le point dans un autre hypercube ne correspondant plus au même message, à une distance et dans une direction inconnue (voir la Figure 2a). Il est très compliqué de prévoir ce déplacement, car il changera en fonction du point.

Il faut donc explorer l'espace latent afin de trouver un point fixe dans un hypercube qui porte le message.

Ici, l'exploration se fait via l'apprentissage d'un vecteur de biais $\mathbf{b}$. L'ajout de ce vecteur à chaque étape de l'apprentissage permet de tester de nouveaux points et donc d'explorer l'espace latent, cette démarche est illustrée en Figure 2b.

L'apprentissage du biais nécessite que toutes les fonctions utilisées lors de la génération soient différentiables.[2] Nous utilisons une approximation lisse de la fonction d'arrondi pour la quantification de l'image et pour la fonction de décodage.

La fonction de décodage binaire (DB), qui dans sa version primaire retourne la parité des coefficients quantifiés, doit ici être une fonction différentiable périodique de demi-période $\Delta/2$, c'est pourquoi nous utilisons la fonction suivante :

$$\mathrm{DB}(z, \Delta, d) = A \arctan\left(\sin 2\pi \frac{z}{\Delta}\right) + \frac{1}{2}. \tag{3}$$

avec A déterminé expérimentalement pour maintenir les valeurs maximales et minimales dans l'intervalle $[0, 1]$, *i.e.* $A = \frac{1}{2.1}$, plus d diminue, plus l'approximation sera faible, ce qui est illustré en Figure 4 :

[2]Nous n'utilisons pas d'approximations pour le *clip* car il reste en grande partie différentiable ce qui suffit pour notre optimisation.

Pour écrire la fonction Smooth Round (SR) différentiable, nous utilisons une approximation trigonométrique qui évite les phénomènes de Gibbs, elle s'appuie sur les approximations différentiables du signal triangle et du signal carré respectivement :

$$\mathrm{T}_d(x) = 1 - \frac{2}{\pi} \arccos\left((1-d)\sin(2\pi x)\right), \tag{4}$$

$$\mathrm{S}_d(x) = \frac{2}{\pi} \arctan\left(\frac{1}{d}\sin(2\pi x)\right). \tag{5}$$

On peut alors écrire la fonction SR comme :

$$\mathrm{SR}(x, d) = -\frac{1}{2}\left(\mathrm{T}_d\left(\frac{2y-1}{4}\right) \times \mathrm{S}_d\left(\frac{y}{2}\right)\right) + y - c, \tag{6}$$

où y et x sont fixés expérimentalement, respectivement, à $y = x - \frac{\pi}{2} + 0.07$ et $c = 1.5$.

L'exploration de l'espace latent est une descente de gradient dont l'objectif pour chaque échantillon est de faire correspondre des *logits* x avec un label binaire y. Cette optimisation revient à faire une classification binaire, c'est pourquoi nous utilisons une entropie croisée binaire (BCE) comme fonction de coût à minimiser :

$$\mathcal{L}(x, y) = \frac{1}{N_{dim}} \sum_{i=0}^{N_{dim}} [y_i \log(x_i) + (1-y_i)\log(1-x_i)]. \tag{7}$$

3 Expériences et résultats associés

Dans cette section, nous montrons à la fois l'intérêt d'utiliser des zones mortes pour augmenter la robustesse d'un schéma d'insertion et nous soulignons le gain quant à la robustesse associé à la méthode proposée dans la section précédente. Les résultats obtenus sont résumés en Table 1. Les zones mortes utilisées sont volontairement importantes (1.25) pour mettre en exergue les limites de cette méthode. De plus, nous testons un taux d'insertion de 1 bps[3] et de 0.5 bps, *i.e.* un échantillon sur deux est modifié par l'algorithme. Les résultats présentés sont moyennés sur 50 réalisations d'insertions avec des vecteurs initiaux tirés aléatoirement.

Afin d'accélérer l'apprentissage, nous appliquons un planificateur sur le niveau d'approximation d de la fonction SR. Lors des premières itérations, l'approximation est forte ($d \geq 0.5$) et est réduite d'un facteur γ après un nombre d'itérations $\tau_{schedule}$. L'apprentissage est plus complexe avec une valeur de d faible, à chaque diminution de d, la valeur de $\tau_{schedule}$ est augmentée. L'apprentissage s'arrête lorsque le taux d'erreur est nul et que l'approximation est suffisamment faible pour produire les mêmes résultats que la fonction $\mathrm{Round}(\cdot)$.

Nous observons que l'ajout d'une zone morte sur un algorithme d'insertion, même naïf comme l'insertion de signe, peut en diminuer drastiquement le taux d'erreur le faisant passer de 10% à 1%. Cependant, même avec l'utilisation d'une grande zone morte (1.25), il est peu probable d'obtenir un décodage sans erreurs. De plus, même un schéma d'insertion robuste comme QIM avec un très grand pas de quantification $\Delta = 2.5$ ne suffit pas, il faut passer par la phase d'optimisation proposée pour atteindre un décodage du message sans erreur.

[3]*bits per sample* soit le nombre de bits insérés dans chaque échantillon du vecteur latent.

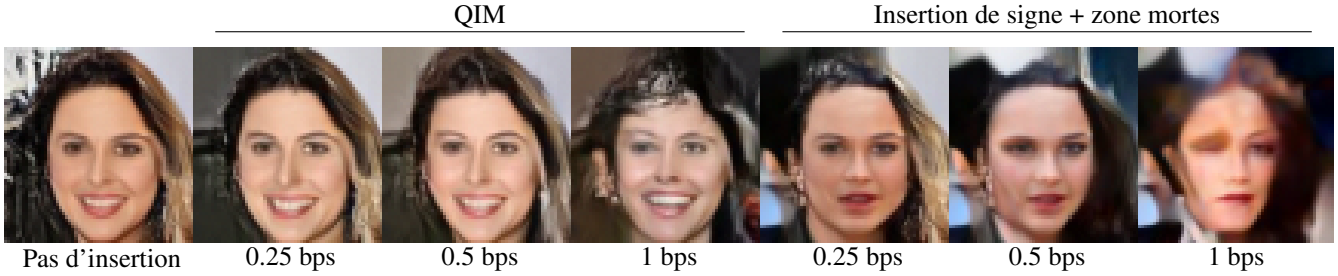


FIGURE 3 : Images générées avec notre méthode et une insertion de signe + zone mortes pour différents taux d'insertion en utilisant le même vecteur initial.

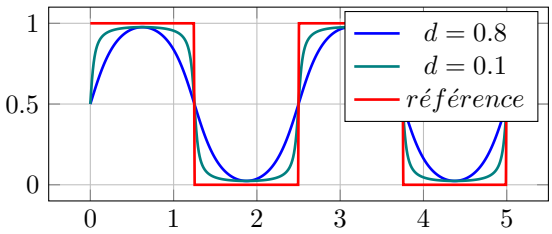


FIGURE 4 : Approximation différentiable de la fonction de décodage pour différents paramètres de lissage d.

L'apprentissage du biais est un processus relativement long par rapport à une insertion, bien qu'il soit théoriquement possible de compenser la distorsion pour n'importe quelle valeur de Δ, partir d'une solution avec un faible taux d'erreur réduit considérablement le coût de calcul et la durée de l'apprentissage. Ce problème peut être limité en diminuant le taux d'insertion, cela a un double avantage. D'une part, réduire le nombre de dimensions à optimiser, donc le temps de calcul. D'autre part, mieux préserver la distribution d'origine et donc limiter la détectabilité et améliorer la qualité visuelle de l'image générée.

TABLE 1 : Taux d'erreur d'extraction pour plusieurs algorithmes d'insertions. Les zones mortes utilisées sont de 1.25 ($\Delta = 2.5$).

	$BER(\%)$	
	$bps = 0.5$	$bps = 1$
Insertion de signe	9.79%	9.75%
Insertion de signe + zone mortes	1.35%	1.3%
QIM	0.73%	0.81%
QIM + exploration	0.0%	0.0%

4 Conclusions et perspectives

La stéganographie générative a le potentiel de surpasser la stéganographie traditionnelle tant en termes de capacité d'insertion qu'en termes d'indétectabilité. Cependant, pour atteindre ce but, les algorithmes d'insertion se doivent d'être robustes à la quantification appliquée aux images générées. Dans cette optique, nous avons présenté une approche permettant de garantir un décodage sans erreur. Cette méthode a le potentiel de s'adapter à n'importe quel schéma d'insertion tant que son décodage peut être approximé de façon différentiable.

Nos travaux futurs se focaliseront sur la minimisation de l'indétectabilité de la méthode d'insertion. L'utilisation d'un vecteur de *dither* lors de l'insertion QIM, mais également le choix de pas de quantification appropriés permettront de prendre en compte le compromis robustesse/indétectabilité.

5 Remerciement

Ce travail a été financé par le gouvernement Français via l'Agence Nationale de la Recherche pour le programme France 2030, référence ANR-22-PECY0011 et le projet ANR-23-IAS4-0004. Nous aimerions également remercier Tomas Pevny de l'université technologique tchèque de Prague pour ses conseils sur la mise en place de l'algorithme d'optimisation.

Références

[1] B. CHEN et G.W. WORNELL : Quantization index modulation : a class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4):1423–1443, 2001.

[2] Sam GUNN, Xuandong ZHAO et Dawn SONG : An undetectable watermark for generative image models. *arXiv preprint arXiv :2410.07369*, 2024.

[3] Durk P KINGMA et Prafulla DHARIWAL : Glow : Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[4] Ming-ming LIU, Min-qing ZHANG, Jia LIU, Ying-nan ZHANG et Yan KE : Coverless information hiding based on generative adversarial networks. *arXiv preprint arXiv :1712.06951*, 2017.

[5] Jiaming SONG, Chenlin MENG et Stefano ERMON : Denoising diffusion implicit models. *arXiv preprint arXiv :2010.02502*, 2020.

[6] Weixuan TANG, Yuan RAO, Zuopeng YANG, Fei PENG, Xutong CUI, Junhao HUANG et Peijun ZHU : Reversible generative steganography with distribution-preserving. *Cybersecurity*, 8(1): 18, 2025.

[7] Zhili ZHOU, Yuecheng SU, Jin LI, Keping YU, Q. M. Jonathan WU, Zhangjie FU et Yunqing SHI : Secret-to-image reversible transformation for generative steganography. *IEEE Transactions on Dependable and Secure Computing*, 20(5):4118–4134, 2023.