Étude des réseaux de neurones sur la stéganalyse

L. Pibre[1,3] M. Chaumont[1,2,3] D. Ienco[1,4] J. Pasquet[1,3]

[1] UNIVERSITE MONTPELLIER, UMR5506-LIRMM, F-34095 Montpellier Cedex 5, France

[2] UNIVERSITE DE NIMES, F-30021 Nîmes Cedex 1, France

[3] CNRS, UMR5506-LIRMM, F-34392 Montpellier Cedex 5, France

[4] IRSTEA, UMR TETIS, F-34093 Montpellier, France

{pibre, chaumont, pasquet, ienco}@lirmm.fr

Résumé

Des travaux récents ont montré que les réseaux de neurones ont un fort potentiel dans le domaine de la stéganalyse. L'avantage d'utiliser ce type d'architecture, en plus d'être robuste, est que le réseau apprend les vecteurs caractéristiques de manière automatique grâce aux couches de convolution. On peut dire qu'il crée des filtres intelligents. Dans cet article nous étudions le deep learning dans le domaine de la stéganalyse afin d'avoir une meilleure compréhension de son fonctionnement. Dans ce document nous présentons les travaux que nous avons effectués sur les réseaux de neurones convolutionnels. Tout d'abord, nous expliquons les aspects théoriques des réseaux de neurones, puis nous présentons nos protocoles expérimentaux et nous commentons les résultats obtenus.

Mots clefs

Stéganalyse, Deep Learning, réseaux de neurones, cover source mismatch.

1 Introduction

La stéganographie est l'art de cacher un message qui doit être indétectable à l'oeil nu dans un contenu. La stéganalyse, quant à elle a pour but de déceler la présence d'un message caché dans un document. Dans notre cas, nous nous sommes intéressé au scénario où le stéganographe utilise toujours la même clé d'insertion [1].

Jusqu'à présent, l'état de l'art de la stéganalyse utilise l'apprentissage automatique en appliquant deux étapes.

La première étape consiste à extraire les vecteurs de caractéristiques à partir des images. Cette étape permet de récupérer un maximum d'informations à partir des images afin de les modéliser et de différencier les images *stégos* [2] des images *covers* [3].

Durant la seconde étape, le stéganalyste utilise un classifieur pouvant être un Ensemble Classifieur [1], un SVM [2] ou bien un Perceptron [3] afin d'apprendre un modèle qui distingue les images *covers* des images *stégos*.

De nos jours, avec l'amélioration de la puissance de calcul des processeurs et des processeurs graphiques, le *Deep Learning* [4] est devenu une méthode beaucoup plus abordable. L'utilisation des réseaux de neurones a été un succès dans de nombreuses disciplines au vu des résultats obtenus, notamment celui de Krizhevsky utilisé sur les images des bases de données ImageNet [4] et Cifar [5] [5, 6].

Récemment, les premières architectures *Deep* ont obtenu des résultats très prometteurs dans le domaine de la stéganalyse [7]. En effet les auteurs de [7] ont montré des résultats proches d'une des méthodes les plus efficace du domaine de la stéganalyse, la classification par Ensemble Classifieur avec le Rich Models [8]. Afin de comparer les deux méthodes, ils ont utilisé les algorithmes d'insertion HUGO [9], WOW [10] et S-UNIWARD [11] sur la base de données BOSSBase [12]. Dans cet article, nous proposons une étude du fonctionnement des réseaux de neurones afin d'expliquer les résultats obtenus par Qian et al. [7].

Les avantages d'utiliser des réseaux de neurones convolutionnels pour la stéganalyse sont multiples. En plus d'être robustes, l'apprentissage est guidé par l'objectif de détection d'un algorithme d'insertion. La classification et le calcul des vecteurs caractéristiques sont faits en même temps, ce qui n'est pas le cas pour le Rich Models avec un Ensemble Classifieur [8] où les deux tâches sont dissociées.

Au cours de notre étude, nous avons voulu comparer une architecture *Deep* avec la méthode classique qui est l'utilisation du Rich Models avec un Ensemble Classifieur [8] qui jusqu'à présent a obtenu les meilleurs résultats sur le scénario de la clairvoyance [13]. De plus, nous avons analysé les résultats obtenus en utilisant le réseau de neurones convolutionnels afin de mieux comprendre son fonctionnement.

1. Ce scénario n'est pas recommandé puisqu'il affaibli la sécurité de l'algorithme d'insertion.
2. Une image *stégo* est une image qui contient un message caché.
3. Une image *cover* est une image qui ne contient pas de message caché.
4. http ://www.image-net.org/
5. http ://www.cs.toronto.edu/∼kriz/cifar.html

Nous présenterons en section 2 des rappels sur les concepts des réseaux de neurones convolutionnels. En section 3, nous verrons la structure de notre réseau. Puis, dans la section 4 nous expliquerons dans un premier temps la méthodologie utilisée pour réaliser nos expérimentations, et dans un second temps nous présenterons les résultats obtenus. Enfin, en section 5 et 6, nous discuterons des liens qui existent entre la construction de notre réseau et les travaux déjà effectués sur le sujet et nous conclurons.

2 Les réseaux de neurones convolutionnels

Un réseau de neurones est un modèle mathématique dont la conception est inspirée des neurones biologiques. Les représentations de ces réseaux sont fortement inspirées de l'article [14].

Ces représentations sont composées de trois parties que l'on appelle des couches qui sont elles même composées de neurones. Les neurones ont une fonction de propagation appelée fonction d'activation, un poids et un biais sur chacune de ses entrées. La première couche est la couche d'entrée dans laquelle on injecte les données que l'on souhaite analyser. La dernière couche est la couche de sortie, dans le cas d'une classification, elle retourne un numéro de classe. La partie intermédiaire est un ensemble de couches dites cachées.

L'opération réalisée par un neurone consiste à additionner le biais avec la somme de ses variables d'entrées pondérées par les poids de connexion. En pratique on réalise le produit scalaire entre le vecteur d'entrée et le vecteur poids (voir Eq. 1.a).

Dans un réseau de neurones convolutionnels, une couche est composée de trois étapes : la convolution, l'application d'une fonction d'activation et enfin le pooling. Le résultat de ces trois étapes est appelé une *feature map*.

Pour détailler les différentes étapes d'une couche, nous allons utiliser le réseau de Qian et al. [7] (voir figure 1). Pour ce réseau, on commence par filtrer une image de taille 256×256 avec un filtre passe-haut $F^{(0)}$. Ce pré-traitement est une spécificité de la stéganalyse, nous avons pu constater que sans ce filtre passe-haut, le réseau mettait beaucoup plus de temps pour converger. Les images filtrées de taille 252×252 sont ensuite données en entrée du réseau.

2.1 La convolution

La convolution de la première couche est une convolution classique. On applique la convolution entre l'image d'entrée et les filtres de la première couche.

Sur la deuxième couche du réseau, une convolution moins classique est utilisée. En effet lors de cette convolution, l'image résultante de cette convolution est la somme de $K^{(l-1)}$ convolutions, avec $K^{(l-1)}$ le nombre de sorties de la couche $l-1$ (voir Eq. 1.b).

$$\sigma_k^{(l)} = \begin{cases} \mathbf{x}_k^{(l)} \cdot \mathbf{w}_k^{(l)} + b_k^{(l)} & \text{si neurone “simple”(1.a)} \\ \sum_{i=1}^{i=K^{(l-1)}} \mathbf{x}_{k,i}^{(l)} \star w_{k,i}^{(l)} + \mathbf{b}_k^{(l)} & \text{si convolution(1.b)} \end{cases}$$

Avec $\sigma_k^{(l)}$ la valeur du neurone k de la couche l, $\mathbf{x}_k^{(l)}$ le vecteur d'entrée du neurone k, $\mathbf{w}_k^{(l)}$ le vecteur poids et $b_k^{(l)}$ le bias. Pour l'équation (2.b), les $\mathbf{x}_{k,i}^{(l)}$ sont les entrées du neurone k et $w_{k,i}^{(l)}$ les noyaux de convolution.

2.2 La fonction d'activation

Une fois la convolution effectuée, une fonction d'activation est appliquée sur toutes les valeurs de l'image filtrée.
Il existe beaucoup de fonctions d'activation, par exemple on peut utiliser le ReLU [15, 16] qui se définit comme $f(\sigma) = max(0, \sigma)$, la fonction $tanh()$ [17] ou bien la fonction sigmoïde [18]. La valeur de sortie $s_k^{(l)}$ d'un neurone k de la couche l dépend donc de sa fonction d'activation et est définie comme :

$$s_k^{(l)} = f(\sigma_k^{(l)}), \tag{2}$$

avec $s_k^{(l)}$ la valeur de sortie du neurone k de la couche l, f la fonction d'activation et $\sigma_k^{(l)}$ la valeur du neurone k.
Le choix de la fonction d'activation peut dépendre du problème de classification. Par exemple, Qian et al. [7] ont utilisé une fonction Gaussienne, ce qui est inhabituel dans les réseaux de neurones.

2.3 Le pooling

Le pooling est une spécificité des réseaux de neurones convolutionnels. Les deux méthodes les plus utilisées pour appliquer cette opération sont les suivantes, soit on fait la moyenne des valeurs de la zone (*pooling average*), soit on extrait uniquement la valeur la plus élevée (*pooling max*). Cette étape permet de faire une réduction de la dimension. L'opération de pooling est une étape de sous-échantillonnage. En pratique, le pooling permet de gagner en temps de calcul.
Le principal avantage du pooling average est qu'il est efficace lorsque l'on souhaite détecter des signaux faibles comme pour le cas de la stéganalyse. Le pooling max est quant à lui efficace lorsque l'on veut détecter des signaux forts, comme des objets par exemple, de plus il permet au modèle d'être invariant aux translations.
Après cette étape de sous-échantillonnage nous obtenons une *feature map* qui est définie comme :

$$I_k^{(l)} = pool(s_k^{(l)}), \tag{3}$$

avec $I_k^{(l)}$ une *feature map* de la couche l, $pool()$ l'opération de pooling et $s_k^{(l)}$ la valeur de sortie du neurone k de la couche l.
La succession de couches de convolutions se termine par un réseau neuronal *fully connected*. Celui-ci est composé

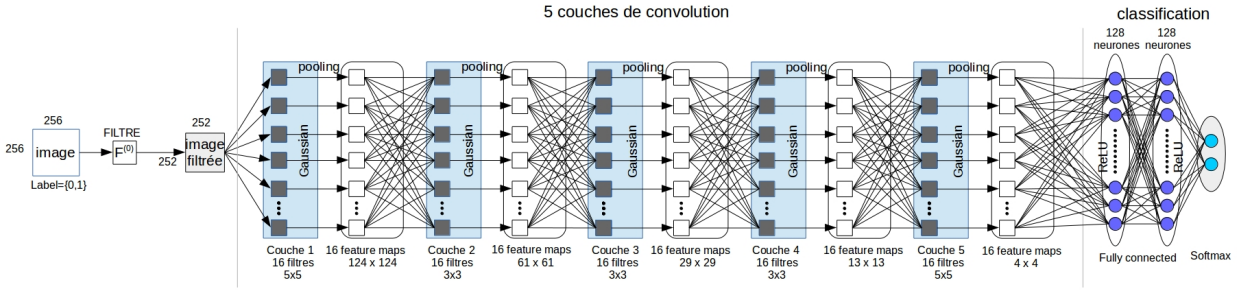

Figure 1 – *Schéma du réseau utilisé par Qian et al. [7].*

de deux parties, la première partie est constituée de couches que l'on appelle des couches *fully connected*. La particularité de cette couche est que tous ses neurones sont connectés avec tous les neurones de la couche précédente mais aussi avec tous les neurones de la couche suivante. La deuxième partie normalise les résultats, on connecte la dernière couche *fully connected* à un *softmax*. Cette fonction est utilisée pour produire une distribution de probabilité entre les différentes classes (chaque classe aura une valeur réelle comprise dans l'intervalle [0, 1]).

3 Configuration de notre réseau

Au cours de nos expérimentations, nous avons testé une quarantaine de réseaux différents. Pour construire notre réseau, nous avons choisi de suivre la même idée que le Rich Models, c'est-à-dire avoir une grande diversité. Contrairement au réseau de Qian et al. [7], nous avons utilisé peu de couches de convolutions, cependant nous avons mis un nombre important de filtres sur nos couches de convolution.

Le premier réseau que nous présentons figure 2 est composé de deux couches de convolution et de trois couches de *fully connected*.

L'image en entrée de taille 256×256 est d'abord filtrée par un filtre passe-haut $F^{(0)}$ de taille 5×5. La taille de l'image devient donc 252×252.

L'image filtrée passe ensuite à la première couche de convolution. Cette couche est composée de 64 filtres de taille 7×7. Notons que pour des contraintes matérielles, notamment à cause de la mémoire, nous appliquons la convolution avec un pas de deux pixels, c'est-à-dire que un pixel sur deux va être traité. Le fait d'appliquer la convolution un pixel sur deux réduit la taille de l'image qui devient 127×127.

Nous pensons que le fait d'utiliser un réseau en hauteur est une des raisons qui nous permis d'obtenir de meilleurs résultats. Augmenter le nombre de couches fait perdre de l'information, cela vient du sous-échantillonnage à l'étape du pooling qui a un effet néfaste sur les résultats du réseau. Nous avons donc décidé de supprimer l'étape de pooling de notre réseau.

Après les 64 convolutions, la fonction d'activation ReLU [15, 16] est appliquée, cette fonction d'activation force les neurones à retourner des valeurs positives. Après chaque couche de convolution, on applique une normalisation locale entre les neurones d'une même couche.

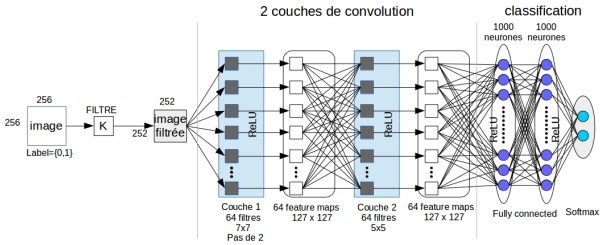

Figure 2 – *Schéma du réseau de neurones convolutionnels (CNN) utilisé.*

Ce type de normalisation se trouve être très efficace lorsque l'on utilise une fonction d'activation non bornée comme le ReLU [15, 16] car elle permet la détection de caractéristiques de hautes fréquences avec de grandes valeurs sur les sorties des neurones. Cette normalisation encourage la "concurrence" pour les grandes activités entre les différentes *feature maps*.

Ensuite, les 64 *feature maps* sont données en entrée de la deuxième couche de convolution qui est composée de 16 filtres. Sur cette couche, la convolution de l'Eq. 1.a est appliquée.

Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU [15, 16] et d'une normalisation. À la sortie de cette couche, nous avons 16 *feature maps* de taille 127×127. Le vecteur de caractéristiques issu des convolutions a une dimension de 258 064, ce qui est 7 fois plus que le Rich Models qui lui obtient des vecteurs de caractéristiques de taille 34 671.

Après ces deux couches de convolution, nous utilisons un réseau de neurones composé de trois couches *fully connected*. Les deux premières couches ont chacune 1 000 neurones où la fonction d'activation utilisée est le ReLU [15, 16], et la troisième couche est un *softmax* qui permet de calculer la distribution de probabilité des deux classes (*cover* et *stégo*).

4 Expériences

4.1 Protocole expérimental général

Afin de réaliser nos expérimentations, nous avons utilisé la base de données BOSSBase v1.0. Cette base de données est composée de 10 000 images en niveau de gris de taille 512×512 provenant de 7 appareils photo différents. Nous avons découpé les images en quatre afin d'obtenir 40 000 images de taille 256×256. Nous avons dû découper nos

images à cause des contraintes de mémoire GPU, cependant cela nous a permis d'utiliser une base d'apprentissage de grande taille. Qian et al. [7] lui aussi a dû réduire la dimension des images pour des contraintes de mémoire GPU, cependant il est à noté que Qian et al. [7] ont utilisé un redimensionnement et non pas un découpage.

Nous avons créé une deuxième base de données que nous avons appelé LIRMMBase [6]. Cette base de données est composée de 1 008 images en niveau de gris de taille 256×256. Pour créer cette base de données, nous avons utilisé six appareils photo, tous différents de ceux de BOSSBase, et 168 images par appareil photo. Cette base de données nous a permis d'évaluer la sensibilité au *cover source mismatch* de notre modèle.

Pour nos expériences, nous avons avons insérer les message en utilisant le simulateur de l'algorithme d'insertion S-UNIWARD [11] avec une taille de message de 0.4 bit par pixel en utilisant toujours la même clé d'insertion. Après l'insertion, nous avions une base de données de 80 000 images (40 000 images *covers* et 40 000 images *stégos*) pour BOSSBase, et 2 016 images (1 008 images *covers* et 1 008 images *stégos*) pour LIRMMBase.

Les résultats présentés dans les parties qui suivent sont, sauf mention du contraire, la moyenne de 10 exécutions sur des bases d'apprentissages et de tests tirées aléatoirement.

Les résultats donnés dans cette partie sont la probabilité d'erreur P_E qui est définie comme :

$$P_E = \min_{P_{FA}} \frac{1}{2}(P_{FA} + P_{MD}(P_{FA})) \tag{4}$$

avec P_{FA} la probabilité de fausse alarme et P_{MD} la probabilité d'erreur de détection [8].

Dans nos expérimentations, nous comparons trois approches de stéganalyse différentes, un réseau de neurones convolutionnels (CNN), un réseau de neurones entièrement connectés (FNN) représenté figure 3, et le Rich Models avec un Ensemble Classifieur (RM+EC). L'Ensemble Classifieur classe les vecteurs de caractéristiques des images que nous avons extraits avec l'algorithme SRM [8]. Le vecteur de caractéristiques d'une image a une dimension de 34 671.

4.2 Expérimentations sur le scénario de la *clairvoyance*

Dans ce premier test nous nous intéressons au scénario de la *clairvoyance* [13] où le stéganalyste connaît tous les paramètres publics, c'est-à-dire l'algorithme d'insertion, la taille du message inséré et possède une bonne connaissance de la distribution statistique des images du stéganographe. Nous faisons l'hypothèse que le stéganographe a toujours utilisé la même clé d'insertion et que le stéganalyste a accès à des couples d'images *cover/stégo* où les images *stégos* ont été générées avec la même clé.

6. www.lirmm.fr/∼chaumont/LIRMMBase.html

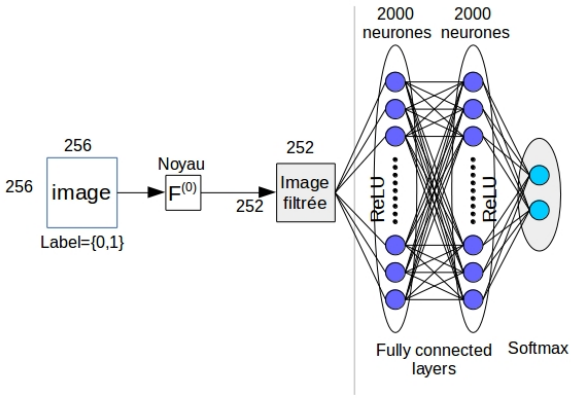

Figure 3 – *Schéma du réseau de neurones entièrement connectés (FNN) utilisé.*

Résultats. Les résultats des trois stéganalyses, par réseau convolutionnel (CNN), par réseau entièrement connecté (FNN) et par Rich Models avec Ensemble Classifieur (RM+EC) sont donnés dans la table 1.

Le Rich Models avec l'Ensemble Classifieur donne une probabilité d'erreur de 24.67% alors que le CNN donne une probabilité d'erreur de 7.4% et le FNN une probabilité d'erreur de 8.66%. Le CNN permet d'améliorer les résultats de plus de 17% et le FNN de plus de 15%.

Le CNN obtient de meilleurs résultats que le Rich Models avec l'Ensemble Classifieur, car les étapes d'extraction de caractéristiques et de classification sont faites en même temps. Avec le Rich Models et l'Ensemble Classifieur, ces deux tâches sont totalement dissociées.

Le nombre important de filtres permet d'avoir une grande diversité dans les vecteurs de caractéristiques. La deuxième couche de convolution semble rechercher la présence de signaux dans les bandes spatio-fréquentielles que nous avons obtenues grâce à la première couche de convolution.

De plus, le fait d'avoir supprimé l'étape de *pooling*, qui agit comme un sous-échantillonnage, nous permet de ne pas avoir une perte d'informations contraitrement au réseau utilisé par Qian et al. [7]. Notons aussi les résultats impressionnants que nous avons obtenus avec le FNN. Il est à noter que le nombre de paramètres inconnus du CNN est d'environ 259 millions alors que celui du FNN est de 131 millions.

Le fait que le stéganographe utilise toujours la même clé d'insertion, le chemin d'insertion est toujours le même, et le simulateur utilise toujours la même séquence de nombres pseudo-aléatoire pour générer la probabilité de modification de la valeur d'un pixel. Dans ce scénario le CNN et le FNN sont plus efficaces, car ils sont sensibles au contenu spatial alors que le Rich Models avec l'Ensemble Classifier est quant à lui sensible aux statistiques de l'image.

Notons que lorsque le stéganographe utilise une clé d'insertion différente pour chaque insertion les résultats chutes à une probabilité d'erreur moyenne de 45.31%. Lorsque la clé d'insertion change, le CNN n'est plus capable de trou-

ver des motifs *stégos*.

	CNN	FNN	RM+EC
Max	7.94%	8.92%	24.93%
Min	7.01%	8.44%	24.21%
Variance	0.12	0.16	0.14
Moyenne	**7.4%**	**8.66%**	24.67%

Tableau 1 – *Résultats des tests sur le scénario de la clairvoyance*

4.3 Expérimentations sur le scénario avec *cover source mismatch*

Le phénomène du *cover-source mismatch* apparaît lorsque l'on effectue l'apprentissage sur des images d'une base de données et que l'on fait les tests sur une autre base de données. La provenance des images apprises par le modèle étant différente de celle utilisée par le stéganographe, la distribution de probabilité des images des deux sources sont différentes et cela crée une difficulté supplémentaire pour le classifieur. Ce problème a été mis en avant lors du challenge BOSS [12]. Notons que [19, 3, 20] ont donné des pistes intéressantes pour lutter contre ce phénomène.
Dans notre cas, puisque les messages ont tous été insérés avec la même clé secrète, les expérimentations suivantes nous ont permis de vérifier que les réseaux apprennent bien la localisation des pixels modifiés.

Résultats. Les résultats des trois stéganalyses, par réseau convolutionnel (CNN), par réseau entièrement connecté (FNN) et par Rich Models avec Ensemble Classifieur (RM+EC) sont donnés dans la table 2.
Malgré la présence de *cover-source mismatch*, on peut constater que les réseaux CNN et FNN ne sont pas du tout impactés par ce phénomène puisqu'ils ont une probabilité d'erreur respectivement de 5.16% et 5.89%. En ce qui concerne le Rich Models avec l'Ensemble Classifieur, on peut voir que le *cover-source mismatch* a un effet très néfaste puisque le résultat est presque aléatoire.
Comme on peut le constater, nous avons obtenu de meilleurs résultats en effectuant nos tests sur LIRMM-Base que sur BOSSBase. Cela vient du fait que les images de BOSSBase sont plus texturées que celles de LIRMM-Base. LIRMMBase est donc plus facile à stéganalyser. Les réseaux de neurones étant invariant au *cover-source mismatch*, les résultats dépendent donc de la complexité des images de la base de données.
La robustesse du CNN peut s'expliquer par le fait que la première couche de convolution décompose les signaux *stégos* en une décomposition spatio-fréquentielle, et qu'ensuite la deuxième couche recherche des motifs particuliers dans les bandes spatio-fréquentielles. De plus, le CNN et le FNN sont spécialisés dans la recherche de motifs spatiaux où la probabilité de modification de la valeur d'un pixel est forte, ce qui est le cas lorsque l'on insère avec la même clé. Notons que dans le cas où la clé d'insertion est différente à chaque insertion, les résultats du CNN chutent à 42.07%.

	CNN	FNN	RM+EC
Max	5.90%	6.60%	49.85%
Min	4.00%	5.40%	47.20%
Variance	0.45	0.31	0.35
Moyenne	**5.16%**	**5.89%**	48.29%

Tableau 2 – *Résultats des tests sur le scnéario avec cover source mismatch*

5 Analyse et discussion

Dans cette section, nous présentons les liens entre le fonctionnement du réseau de neurones les recherches dans la domaine de la stéganalyse.
On peut retrouver dans des articles récents des éléments qui sont similaires au réseau, comme par exemple la projection d'un résidu sur une base de filtre du Rich Models par des projections [21] qui ressemble à ce qui est fait dès la première couche du réseau, ou bien par décomposition spatio-fréquentielle en utilisant des filtres de Gabor [22]. Ces filtres sont utilisés pour définir les projections qui seront utilisées pour calculer un histogramme qui va permettre d'obtenir les vecteurs de caractéristiques.
Sur la deuxième couche de convolution, on constate que l'opération qu'effectue cette couche est très inhabituelle. Nous pensons que c'est cette étape qui permet de rendre les vecteurs de caractéristiques invariants au *cover-source mismatch*. La somme de ces convolutions (voir Eq. 1.b) recherche la présence de motifs dans toutes les bandes issues de la première couche de convolution. La sortie de la deuxième couche de convolution permettrait de donner un indice sur la présence d'un signal *stégo*.
Il est aussi à noter que la normalisation que l'on effectue à la fin de chaque couche de convolution est une opération que l'on peut retrouver dans les articles [23, 24]. Cette normalisation permet à chaque neurone d'avoir des valeurs de sorties du même ordre. La fonction d'activation a elle aussi un impact important sur les performances du réseau, probablement car elle introduit de la non linéarité. Pour le moment l'impact de cette fonction n'est pas encore bien connu. On peut retrouver des opérations non linéaires dans l'Ensemble Classieur [1] avec le vote majoritaire, ou bien dans le Rich Models [8] avec les caractéristiques Min-Max.
Des travaux où la clé d'insertion est réutilisée sur plusieurs images par le stégnographe ont déjà été menés, notamment dans l'article de Ker [25]. Dans cet article Ker détermine les pixels qui ont été modifiés lors de l'insertion en utilisant la méthode de stéganalyse *Weighted Stego-image* (WS).

6 Conclusion

Dans cet article, nous avons poursuivi les travaux sur les réseaux de neurones convolutionnels dans le domaine de la stéganalyse réalisés par Qian et al. [7].
Nous avons évalué notre réseau sur deux scénarios différents. Les premiers tests ont été réalisés avec le scénario

de la *clairvoyance*. Nous avons utilisé la base de données BOSSBase, et inséré avec S-UNIWARD avec une taille de message de 0.4 bit par pixel en utilisant toujours la même clé d'insertion. Les résultats que nous avons obtenus avec le CNN surpassent de loin l'état de l'art puisque l'on a un gain de plus de 17%.
Les derniers tests que nous avons faits portent sur le scénario avec *cover-source mismatch*. Nous avons utilisé la base de données BOSSBase et S-UNIWARD en utilisant toujours la même clé d'insertion et avec une taille de message de 0.4 bit par pixel pour l'apprentissage du CNN. Nous avons ensuite fait nos tests sur la base de données LIRMM-Base. Alors que le Rich Models avec un Ensemble Classifieur obtient des résultats proches de 50%, le CNN est quant à lui insensible au phénomène du *cover-source mismatch* puisqu'il permet d'obtenir une probabilité d'erreur de classification de 5.16%.

Références

[1] J. Kodovský, J. Fridrich, et V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2) :432–444, April 2012.

[2] C. Chang et C. Lin. Libsvm : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3) :27, 2011.

[3] I. Lubenko et A. Ker. Steganalysis with mismatched covers : Do simple classifiers help ? Dans *Proceedings of the on Multimedia and Security*, MM&Sec2012, pages 11–18, New York, NY, USA, 2012. ACM.

[4] Y. Bengio, I. Goodfellow, et A. Courville. Deep learning. Book in preparation for MIT Press, 2015.

[5] A. Krizhevský, I. Sutskever, et G. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[6] A. Krizhevský et G. Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.

[7] Y. Qian, J. Dong, W. Wang, et T. Tan. Deep learning for steganalysis via convolutional neural networks. Dans *Media Watermarking, Security, and Forensics 2015*, volume 9409 de *Proceedings of SPIE*, pages 94090J–94090J–10, San Francisco, CA, March 2015.

[8] J. Fridrich et J. Kodovský. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3) :868–882, June 2012.

[9] T. Pevný, T. Filler, et P. Bas. Using high-dimensional image models to perform highly undetectable steganography. Dans Rainer Böhme, Philip W.L. Fong, et Reihaneh Safavi-Naini, éditeurs, *Information Hiding*, volume 6387 de *Lecture Notes in Computer Science*, pages 161–177. Springer Berlin Heidelberg, 2010.

[10] V. Holub et J. Fridrich. Designing steganographic distortion using directional filters. Dans *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 234–239, Dec 2012.

[11] V. Holub, J. Fridrich, et T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1) :1–13, 2014.

[12] P. Bas, T. Filler, et T. Pevný. "Break Our Steganographic System" : The Ins and Outs of Organizing BOSS. Dans *Information Hiding*, volume 6958 de *Lecture Notes in Computer Science*, pages 59–70, Czech Republic, Mai 2011.

[13] T. Pevný. Detecting messages of unknown length. Dans *Media Watermarking, Security, and Forensics III*, volume 7880 de *Proceedings of SPIE*, pages 78800T–78800T–12, Février 2011.

[14] J. Lettvin, H. Maturana, W. Mcculloch, et W. Pitts. What the frog's eye tells the frog's brain. *Proceedings of the Institute of Radio Engineers*, 47(11) :1940–1951, Nov 1959.

[15] K. Jarrett, K. Kavukcuoglu, M. Ranzato, et Y. LeCun. What is the best multi-stage architecture for object recognition ? Dans *Proceedings of the IEEE International Conference on Computer Vision*, pages 2146–2153, September 2009.

[16] V. Nair et G. Hinton. Rectified linear units improve restricted boltzmann machines. Dans *Proceedings of International Conference on Machine Learning*, pages 807–814, Haifa, Israel, June 2010.

[17] D. Nguyen et B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Dans *International Joint Conference on Neural Networks*, pages 21–26 vol.3, June 1990.

[18] M. Norouzi, M. Ranjbar, et G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. Dans *IEEE Transactions on Conference on Computer Vision and Pattern Recognition*, pages 2735–2742, June 2009.

[19] J. Pasquet, S. Bringay, et M. Chaumont. Steganalysis with cover-source mismatch and a small learning database. Dans *Proceedings of the European Signal Processing Conference*, pages 2425–2429. IEEE, September 2014.

[20] J. Pasquet, S. Bringay, et M. Chaumont. Des millions d'images pour la stéganalyse : inutiles. *COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, page 53, 2013.

[21] V. Holub, J. Fridrich, et T. Denemark. Random projections of residuals as an alternative to co-occurrences in steganalysis. volume 8665, pages 86650L–86650L–11, 2013.

[22] X. Song, F. Liu, C. Yang, X. Luo, et Y. Zhang. Steganalysis of adaptive jpeg steganography using 2d gabor filters. Dans *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, pages 15–23, New York, NY, USA, 2015. ACM.

[23] S. Kouider, M. Chaumont, et W. Puech. Adaptive steganography by oracle (aso). Dans *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6, July 2013.

[24] R. Cogranne, T. Denemark, et J. Fridrich. Theoretical model of the fld ensemble classifier based on hypothesis testing theory. Dans *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*, pages 167–172, Dec 2014.

[25] A. Ker. Locating steganographic payload via ws residuals. Dans *Proceedings of the 10th ACM Workshop on Multimedia and Security*, MM&Sec2008, pages 27–32, New York, NY, USA, 2008. ACM.