Etude de l'invariance à la dimension des images lors de la stéganalyse par deep learning

Mémoire de fin d'étude

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours ICo

Auteur
Kévin Planolles

Superviseurs
Marc Chaumont
Mohamed Benkhettou
Frederic Comby

Lieu de stage
LIRMM UM5506 - CNRS, Université de Montpellier

version 1 soutenu publiquement le 9 juin 2022

Résumé

Dans le cadre de mon stage de deuxième année de master informatique, nous nous intéréssons à la stéganlyse en conditions réelles et plus particulièrement l'étude du cas où la dimension des images stéganographiées n'est pas connue. Nous nous focalisons sur des méthodes par deep learning et dans ce contexte nous voulons étudier la performance des modèles relativement à la dimension des images traitées. Notre approche essentiellement expérimentale vise à observer comment les réseaux de neurones se comportent sous ces contraintes, dans le but de dériver une loi de comportement (fonction de la dimension) ou un mécanisme permettant de rendre les réseaux moins sensibles en performance à la dimension.

A travers les différentes expériences, nous n'avons pas observé d'invariance de la part des réseaux (première constatation). Nous avons également observé une différence dans le comportement selon que la dimension testée est inférieure ou supérieure à la dimension apprise. Cette deuxième constatation expérimentale n'avait jamais été relevée auparavant. Enfin, la troisième constatation porte sur la non observation La loi « théorique » dite de la « racine carrée ».

Dans ce document, nous introduisons d'abord le sujet de la stéganalyse et ce que nous entendons par « invariance à la dimension des images ». Dans un second temps, nous présentons les réseaux de neurones profonds de manière générale, puis les architectures issus de la littérature que nous avons utilisées pour effectuer nos expériences. Nous précisons ensuite le protocole expérimental suivi et faisons la proposition d'une nouvelle architecture. Nous terminons enfin par les résultats obtenus.

Table des matières

1

Stéganographie et stéganalyse

Avant de parler de stéganalyse, il nous faut définir ce qu'est la stéganographie. Nous pouvons faire cela en imaginant un jeu à trois participants, comme illustré sur la figure 1.1 : Alice souhaite communiquer à Bob un message à travers un canal auquel Eve a accès (elle peut intercepter tout ce qui y transite). Alice ne souhaite pas qu'Eve sache que cette information a transité. Pour cela, elle va dissimuler le message au sein d'un support anodin (dans notre cas, ce sera toujours une image numérique) : c'est la *stéganographie*. L'objectif d'Eve est de détecter si l'image contient un message caché : c'est la *stéganalyse* [11].

1.1 Stéganographie

La stéganographie consiste donc à dissimuler au sein d'une image un message que l'on souhaite indétectable. L'image d'origine est appelée la *Cover* tandis que l'image qui comporte le message s'appelle une *Stego*. Pour insérer un message dans une image, on modifie la valeur des pixels selon la valeur des bits du message. Le nombre de bits insérés par pixels est appelé *payload relatif*.

Pour insérer le message, on utilise ce que l'on appelle des méthodes adaptatives, qui procèdent à une analyse préalable de la Cover pour opérer une meilleure sélection des pixels qui porteront le message. Empiriquement, il est plus sûr de modifier une partie très bruitée de l'image plutôt qu'une zone homogène (par exemple un ciel bleu) car il est alors moins facile d'utiliser le voisinage d'un pixel pour détecter un changement. On construit alors ce que l'on appelle une carte de coûts qui représente, pour chaque pixel de la Cover, son coût d'insertion (c'est à dire son impact sur la détéctabilité globale de l'image si le pixel est modifié lors de l'insertion). Plusieurs algorithmes utilisent cette carte de coût comme base, dont celui qui a été utilisé pour construire notre base d'image, **S-UNIWARD** [4]. La figure 1.3 présente un exemple d'insertion utilisant cet algorithme : nous l'avons obtenue en faisant la différence entre la Cover et la Stego. Les pixels blancs sont les endroits où il y a eu une modification.

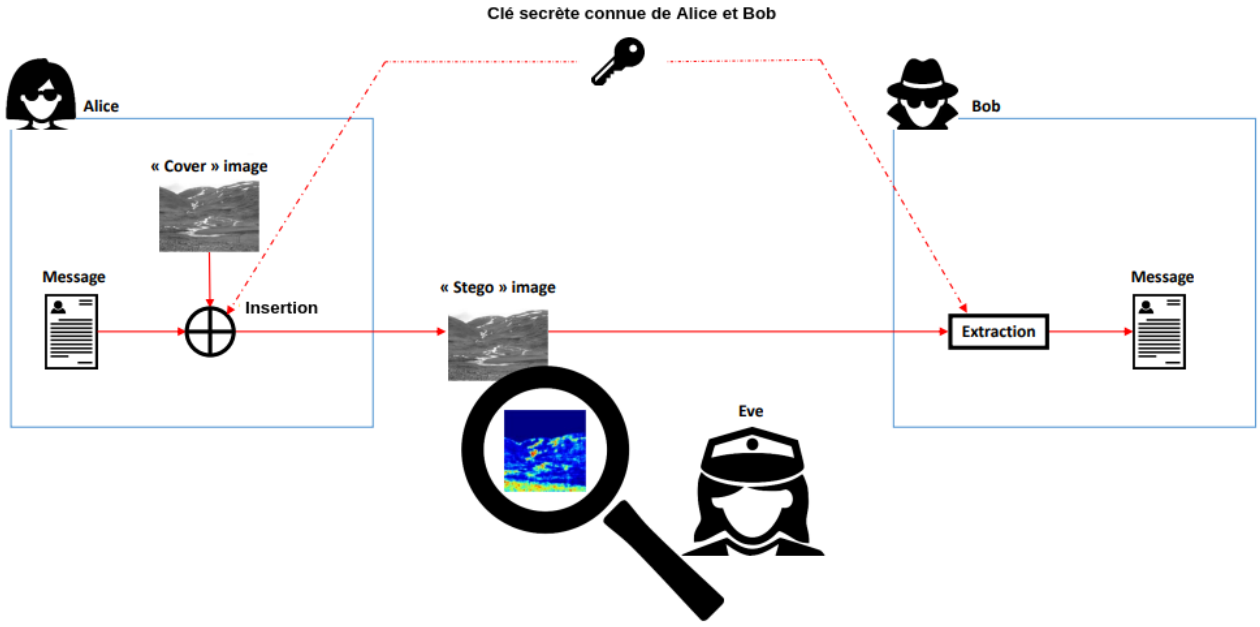

FIGURE 1.1 – Schéma d'illustration d'un échange impliquant Alice, Bob et Ève.

La loi de la racine carrée

La loi de la racine carrée [7] indique que, pour une image de dimensions $w \times h$, il existe une constante $k \in \mathbb{R}$, que l'on appelera *constante de détectabilité* telle que le payload relatif p est donné par la relation :

$$p = \frac{k}{wh} \times \sqrt{wh} \times \log(wh). \tag{1.1}$$

1.2 Stéganalyse

Détecter le signal montré dans la figure 1.3 relève de la *stéganalyse*. Au sens classique, il s'agit donc d'un problème de décision : étant donné une image, s'agit-il d'une Cover ou d'une Stego ?

Protocole clairvoyant

Dans les études théoriques dites « de laboratoire », ou « pire attaque », on suppose généralement qu'Eve a accès à priori à presque toutes les informations concernant les données en jeu, conformément aux principes de Kerckhoffs issus de la cryptographie [8]. Parmi ces données, on trouve :

- La dimension des images (hauteur et largeur),
- Eventuellement la résolution métrique (le nombre de pixels par pouce),
- Le payload,

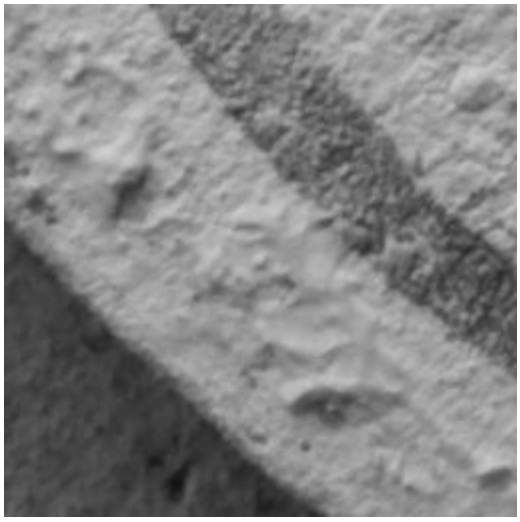

FIGURE 1.2 – Cover

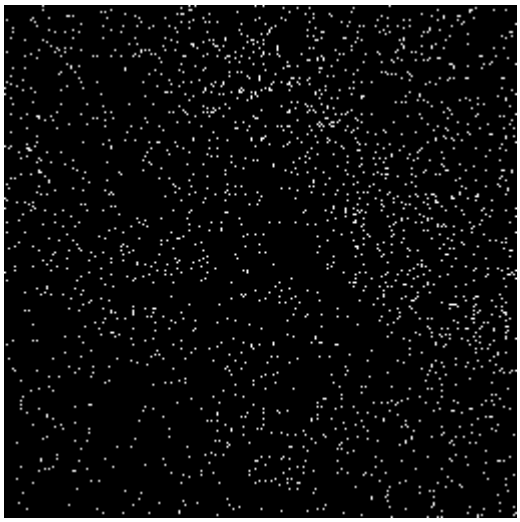

FIGURE 1.3 – Signal stéganographique

- Le type de l'image (couleurs ou niveaux de gris),
- Si l'image est compressé et le cas échéant par quel algorithme (JPG, PNG...),
- Le développement[1] de l'image,
- L'ISO (la quantité de lumière saisie par le capteur),
- Le type d'appareil photo,
- ...

[1] On entend ici par développement des opérations de traitement d'image comme la correction gamma, le flou, la balance des couleurs...

Les exceptions notables auxquelles Eve n'a pas accès sont : la présence ou non d'un message dans l'image, le cas échéant de la teneur du message et de la clé d'insertion utilisée (paramètre de l'algorithme d'insertion). On appelle cette hypothèse de travail un *protocole clairvoyant*.

Invariance à la dimension lors de la stéganlyse

Dans un cas concret, il est rare qu'Eve ait accès aux informations pré-citées. Notamment, Alice et Bob peuvent utiliser comme Cover des images de tailles différentes. De ce fait, une méthode conçue pour analyser une image de taille donnée pourrait être rendue inutile par un simple changement de dimension de la Cover. Il n'est en effet pas permis pour Eve de redimensionner les images pour se ramener à une taille qu'elle sait traiter : un ré-échantillonage pourrait détruire le message inséré. Pour cette raison, on souhaite disposer d'un algorithme de stéganalyse dont les performances ne varient pas avec la dimension des images d'entrée.

Formellement, soit $\mathbf{x}$ une image de taille $w \times h$. Un *stéganalyseur* est une application f définie par :

$$\begin{aligned} f\colon \{0,255\}^{w\times h} &\to [0,1]^2 \\ \mathbf{x} &\mapsto (p^{(C)}, p^{(S)}) \end{aligned}$$

tel que $p^{(C)} + p^{(S)} = 1$, avec $p^{(C)}$ (respectivement $p^{(S)}$) interprété comme la probabilité d'avoir une Cover (respectivement Stego). Dans ce document, nous disons qu'un stéganalyseur est *invariant en performance à la dimension* lorsqu'étant donné une constante $k \in \mathbb{R}$ (la constante de détectabilité), pour tout w, h et un message inséré avec un payload relatif $p = \frac{k}{wh} \times \sqrt{wh} \times \log(wh)$, $f(\mathbf{x})$ est une constante.

Ici, le stéganalyseur sera un réseau de neurones convolutif (voir section suivante), et l'objectif, en considérant la dimension d'image comme une inconnue, est d'analyser le comportement de ces réseaux lorsqu'il y a variation de dimension, et éventuellement de proposer un réseau qui soit invariant. Il s'agit d'un premier pas vers ce que l'on appelle la stéganalyse « real world » [6]. Nous nous focalisons dans ce document sur l'observation et la mesure de performances à travers différentes expérimentations.

2

Réseaux de neurones convolutifs

2.1 Présentation générale

Les réseaux de neurones sont des systèmes de calcul inspirés du fonctionnement des neurones biologiques. Ils font partie des méthodes d'apprentissage statistique, qui utilisent de grand jeux de données et un mécanisme d'inférence pour pouvoir faire des prédictions. Nous nous concentrons ici sur l'apprentissage *supervisé*, où les données sont réparties dans des classes bien définies (dans notre cas, deux). Les réseaux de neurones convolutifs (CNN) en sont des types particuliers, adaptés pour le traitement des images. Ils sont composés de deux parties principales : l'extracteur de caractéristiques, lui-même divisé en couches de convolution et en couches de pooling, et le bloc de classification qui consiste en un perceptron multi-couche. La figure 2.1 illustre l'architecture générale d'un réseau de neurones convolutif.

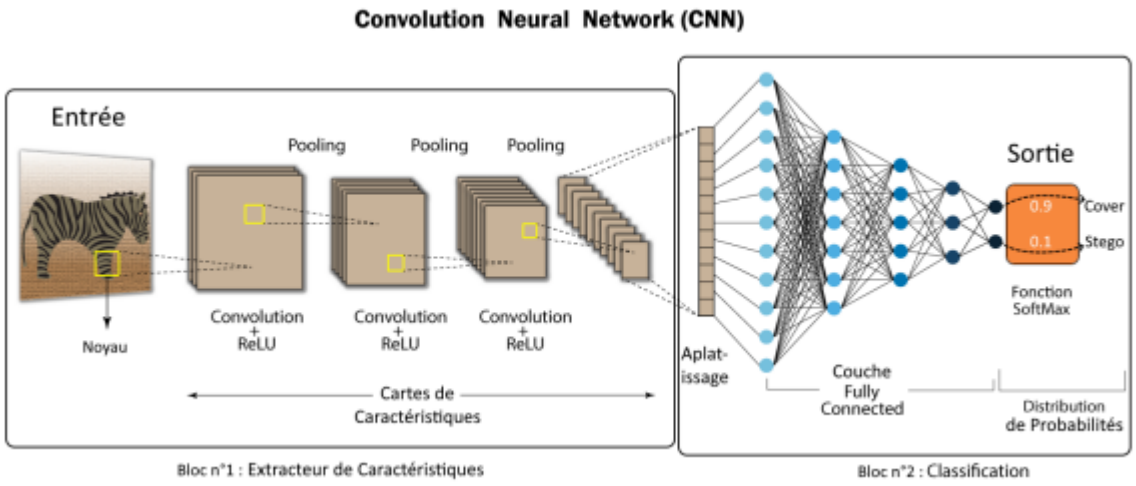

FIGURE 2.1 – Structure générale d'un CNN.

2.2 Architecture

Extracteur de caractéristiques

Couche de convolution

La convolution permet d'extraire les caractéristiques de l'image sous forme de matrices (appelées aussi *features maps* ou *cartes de caractéristiques*). Un autre avantage de ce procédé est qu'il permet de garder le nombre de paramètres petit au regard de la taille de l'entrée, car les calculs ne sont faits à chaque fois que sur une portion restreinte de l'image (typiquement un carré de taille 3 ou 5, voir la figure 2.2). Mathématiquement, pour une image I vue comme un élément de $\mathbb{R}^{n\times n}$ et un *noyau de convolution* K vu comme un élément de $\mathbb{R}^{k\times k}$, la convolution s'exprime pour tous entiers x, y compris entre 0 et $n-k$ par :

$$I * K(x,y) = \sum_{i=0}^{k}\sum_{j=0}^{k} K(i,j) \times I(x+i-\left\lfloor\frac{k}{2}\right\rfloor, y+j-\left\lfloor\frac{k}{2}\right\rfloor) \tag{2.1}$$

Le résultat de la convolution est ensuite *normalisé*. Une normalisation est une opération dont le but est de transformer un ensemble de données pour que sa moyenne soit 0 et son écart-type 1. Une normalisation classique est par exemple la *Batch-Normalization*. Pour une couche avec une entrée de dimension x $= (x^{(1)}, \cdots, x^{(d)})$, on calcule pour chaque dimension la quantité suivante [5] :

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathsf{E}[x^{(k)}]}{\sqrt{\mathsf{Var}[x^{(k)}]}} \tag{2.2}$$

L'intérêt de la normalisation est de stabiliser et d'accélérer l'apprentissage.

Pour introduire de la non-linéarité, cruciale pour l'apprentissage, on applique enfin une *fonction d'activation*. Une fonction d'activation très répandue est la fonction :

$$\mathbf{ReLU}(x) = \max(0, x). \tag{2.3}$$

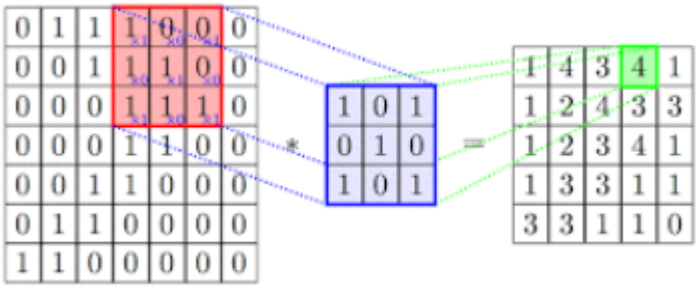

FIGURE 2.2 – Illustration de la convolution.

Pooling

Le pooling est une opération permettant d'agréger les valeurs d'une carte de caractéristiques, par exemple en prenant la moyenne ou le maximum (comme illustré sur la figure 2.3) d'une partie de cette carte, de manière à réduire la taille de celle-ci, donc l'espace occupé en mémoire et par conséquent le temps d'apprentissage.

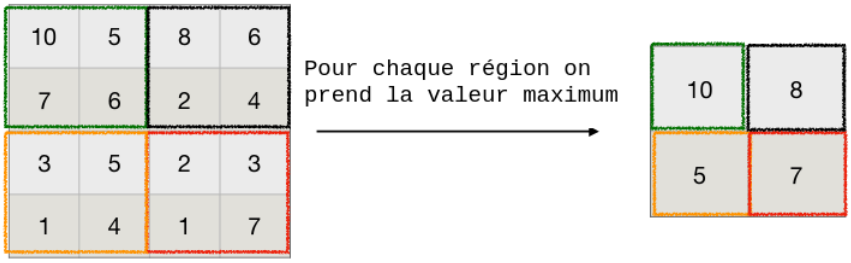

FIGURE 2.3 – Illustration d'un *max pooling*.

Classifieur

Les opérations de convolution et de pooling peuvent être répétés plusieurs fois. On obtient alors un ensemble de cartes de caractéristiques (c'est-à-dire un ensemble de matrices) qui vont être transformées en vecteur. Si on a n matrices de dimension $w \times h$, le résultat sera un vecteur composé de $n \times w \times h$ éléments. Ce vecteur constitue l'entrée d'un *perceptron multicouches*.

Ce dernier est constiué de plusieurs couches de neurones connectées en succession. Un neurone réalise la combinaison linéaire de plusieurs grandeurs numériques, et transmet le résultat, après application d'une fonction d'activation, à la couche suivante. Le signal transite ainsi de la couche d'entrée jusqu'à la sortie qui consiste en les probabilités d'appartenance aux classes (ici Cover ou Stego).

Formellement, un réseau de neurone est constitué des éléments suivants :

- x_i les entrées, i allant de 0 à $n \times w \times h - 1$.
- y_j les sorties, j allant de 0 à C le nombre de classes (ici 2).
- $a_{k,l}$ le neurone l de la couche cachée k, k allant de 1 à $P \in \mathbb{N}$.
- $b_{l,k}$ le biais du neurone l de la couche cachée k.
- $w_{k,p,q}$ le poids à appliquer lors du calcul de la couche k, pour le lien entre le neurone $a_{k-1,p}$ et le neurone $a_{k,q}$.

Par exemple pour la figure 2.4, avec

$$X = (x_0, x_1), Y = (y_0, y_1), A = (a_{1,1}, a_{1,2}), B = (b_{1,1}, b_{2,1}),$$
$$W_1 = \begin{pmatrix} w_{1,1,1} & w_{1,1,2} \\ w_{1,2,1} & w_{1,2,2} \end{pmatrix}, W_2 = \begin{pmatrix} w_{2,1,1} & w_{2,1,2} \\ w_{2,2,1} & w_{2,2,2} \end{pmatrix}$$

on a $A = \mathbf{ReLU}(XW_1)$, où la fonction **ReLU** est appliquée composante par composante. De plus, pour obtenir des probabilités en sortie, on utilise la fonction *softmax*, définie par :

$$\sigma(\mathbf{y})_j = \frac{e^{y_j}}{\sum_{j=0}^{C} e^{y_j}} \tag{2.4}$$

La sortie du réseau présenté en figure 2.4 serait donc $Y = \sigma(AW_2 + B)$.

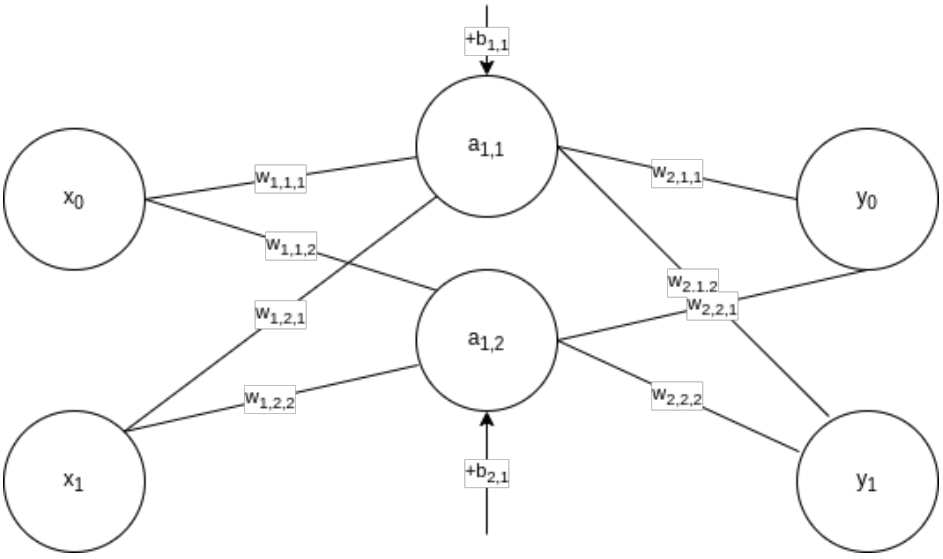

FIGURE 2.4 – Réseau de neurones avec une seule couche cachée.

2.3 Apprentissage

Pour déterminer les poids du classifieur ainsi que ceux du noyau de convolution, on effectue un *apprentissage*. Pour cela, on utilise une base d'images qui consiste en un grand nombre de couples (I, y) où I est une image et y est un encodage de la classe de l'image, que l'on connaît, sous forme de vecteur, par exemple $(1, 0)$ pour une Cover et $(0, 1)$ pour une Stego. Pour chaque image donnée en entrée au réseau, on obtient un vecteur de probabilités $\hat{y}$. Ceci nous permet de définir une fonction de coût que l'on peut voir comme la mesure de l'erreur commise par le réseau, par exemple l'entropie croisée :

$$L(y, \hat{y}) = -y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}) \tag{2.5}$$

L'apprentissage des poids se fait en minimisant cette fonction de coût avec l'algorithme de descente du gradient. Une *epoch* correspond à l'évaluation de l'ensemble des couples

(I, y). Après une epoch, on effectue une *validation* qui permet de mesurer la capacité de généralisation du réseau en utilisant d'autres couples (I', y'), mais sans mettre à jour les poids.

Remarque Il faut cependant donner des valeurs « par défaut » aux poids pour la première itération de l'algorithme. Ceci peut se faire soit avec un tirage aléatoire, soit avec des valeurs obtenues manuellement et qui permettent d'accélerer la convergence vers un minimum.

2.4 Évaluation

De la probabilité à la prédiction

Un réseau de neurones a pour but de faire des prédictions. La sortie d'un réseau de neurones étant un vecteur de probabilités, il faut transformer celles-ci en prédictions. La façon la plus intuitive de faire est de choisir la classe dont la probabilité est la plus élévée. Ce n'est pas la seule façon de faire, et selon la base d'images sur laquelle le modèle a appris, mais également selon le contexte d'application du modèle, on peut choisir de ne prédire une classe que si la probabilité qui lui a été associée est supérieure à un certain *seuil de discrimination* (dans le cas « intuitif », le seuil en question est égal à 50%).

Métriques

Une fois fixé le choix du mode de prédiction, on évalue le modèle grâce à une base de test. Il s'agit d'une base d'images $\mathcal{B}$ dont les images n'ont pas été « vues » lors de l'apprentissage. Pour chaque prédiction du réseau, il y a quatre possibilités :

- L'image est une Cover et a été prédite en Cover : on appelle cela un *vrai négatif*. On note VN le nombre de vrais négatifs.
- L'image est une Cover mais a été prédite en Stego : on appelle cela un *faux positif*. On note FP le nombre de faux positifs.
- L'image est une Stego mais a été prédite en Cover : on appelle cela un *faux négatif*. On note FN le nombre de faux négatifs.
- L'image est une Stego et a été prédite en Stego : on appelle cela un *vrai positif*. On note VP le nombre de vrais positifs.

Pour évaluer les performances d'un CNN, plusieurs métriques ont été définies. Celle que nous utilisons principalement est appelée *l'accuracy*. Elle est donnée par :

$$\text{Accuracy} = \frac{\text{VN} + \text{VP}}{\text{VN} + \text{FP} + \text{FN} + \text{VP}} = \frac{\text{VN} + \text{VP}}{|\mathcal{B}|} \tag{2.6}$$

VN	FP
FN	VP

FIGURE 2.5 – Schéma d'une matrice de confusion.

On trouve aussi dans la littérature la *probabilité d'erreur* P_e, définie par $P_e = 1 -$ Accuracy. Enfin, une matrice de confusion est un tableau se présentant sous la forme donnée par la figure 2.5 et permet de visualiser facilement ces informations.

3

Etat de l'art et proposition

Nous présentons ici les architectures qui permettent, en stéganalyse, de traiter des images de dimensions quelconques. Tous ces réseaux utilisent des noyaux de convolution initialisés avec des filtres *Spatial Rich Model* [3].

3.1 Etat de l'art

Réseau SID

SID [12] est basé sur une version modifiée du réseau YeNet [10], l'idée complémentaire de cette architecture est d'extraire quatre informations (*moments*) statistiques de l'ultime carte de caractéristiques fournie par le bloc de convolution. Ces moments sont le minimum, le maximum, la moyenne et la variance. D'après les auteurs de l'article, ces informations permettent au classifieur de s'adapter lui-même à la dimension et à la résolution de l'image d'entrée, au prix cependant d'un entraînement partiel (*transfert learning*) du classifieur sur la nouvelle dimension d'image. Les auteurs se basent sur l'hypothèse que ces moments décroissent continûment avec la taille des images pour justifier l'invariance de leur architecture. Cette hypothèse peut cependant être remise en question en l'absence de fondements théoriques ou expérimentaux. La figure 3.1 illustre l'architecture du réseau SID.

SiaSteg

L'architecture SiaSteg [15] reprend l'idée des moments statistiques présentée précédemment, mais se base sur un réseau *siamois*. Le principe de ces réseaux est qu'ils sont composés de sous-réseaux identiques (mêmes paramètres), et ils tendent à minimiser la distance entre deux entrées « similaires ». Dans le cas de l'architecture SiaSteg, l'image d'entrée est découpée en deux et les deux parties sont données en entrée au réseau siamois. L'idée est que si l'image d'entrée est une cover, alors la distance entre ses deux sous-parties doit être faible, tandis que s'il s'agit d'une stégo, les différences seront plus marquées, car les zones texturées (là où un algorithme de stéganographie aura le plus de chance d'effectuer une insertion) ne sont pas uniformément réparties. Le réseau apprend simultanément à minimiser l'écart des deux-sous parties d'une image cover et à

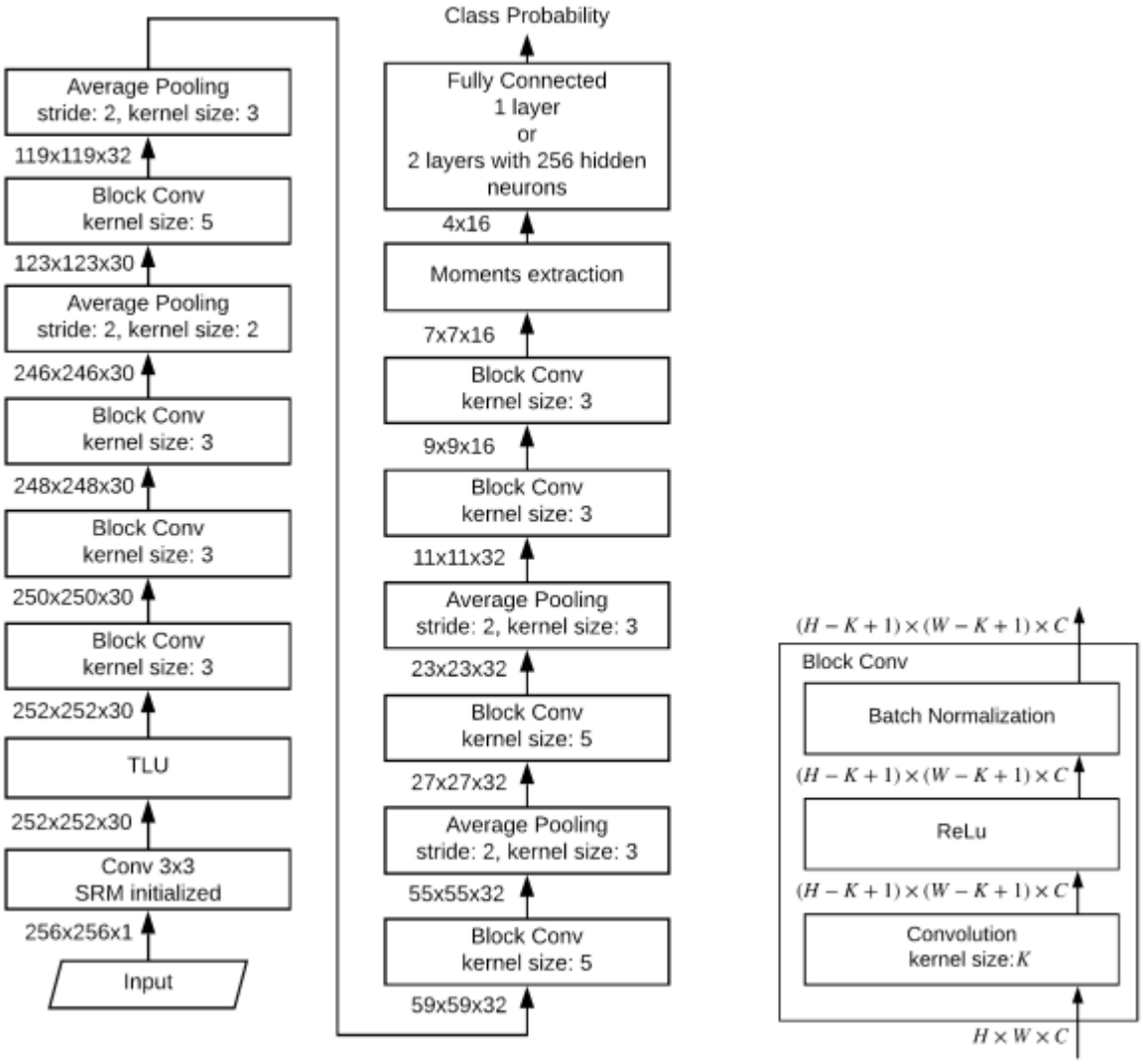

FIGURE 3.1 – Architecture du réseau SID, figure extraite de [12].

faire la distinction entre une cover et une stégo. Cependant, l'hypothèse selon laquelle les différences seront plus marquées entre deux parties d'une image stégo est sujette à doute. La figure 3.2 illustre l'architecture du réseau SiaSteg.

Réseau Zhu-Net

L'architecture Zhu-Net [16], inspirée du réseau Yedroudj-Net[14], est basée sur une opération de pooling différente des pooling classiques et qui s'effectue juste avant la couche de classification : le *pooling pyramidal*. Il consiste à enchaîner les GAP (*global average pooling*) sur des régions de plus en plus petites de la carte pour concaténer les résultats en un vecteur qui ne dépend pas des dimensions de l'image de départ. On observe cependant que plus l'image est grande, plus les sous-régions sont grandes et

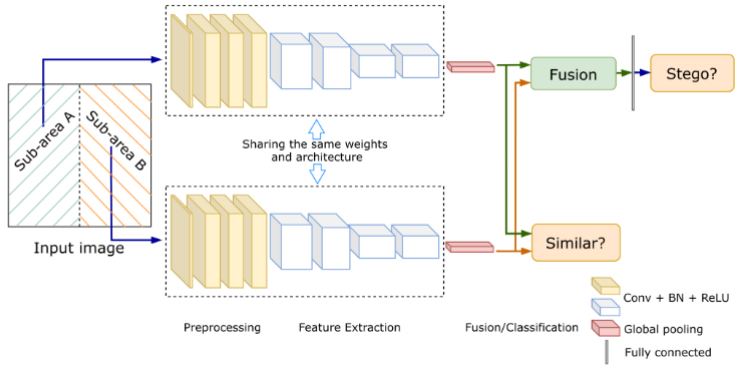

FIGURE 3.2 – Architecture du réseau SiaSteg, figure extraite de [15].

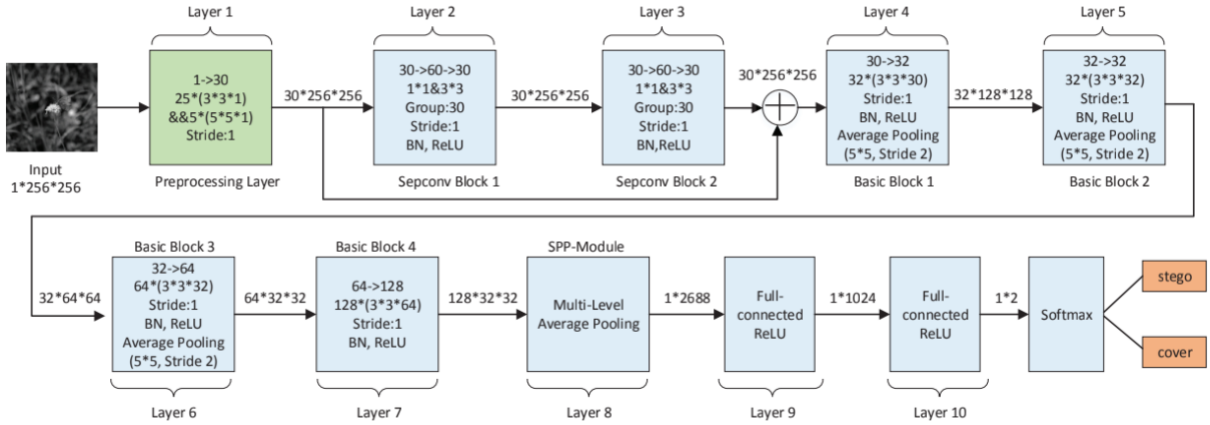

FIGURE 3.3 – Architecture du réseau Zhu-Net, figure extraite de [16].

moins le GAP est précis, et l'on peut se demander quel est l'impact sur l'invariance de ce modèle à la dimension des images. La figure 3.3 illustre l'architecture du réseau Zhu-Net.

Réseau Yedroudj-Net

L'architecture Yedroudj-Net [14] utilise cinq couches de convolution. Une des particularités de ce réseau est d'utiliser la fonction d'activation *truncature*[1] pour empêcher que des valeurs trop grandes ne se propagent dans le réseau. Juste avant la couche de classification, le réseau effectue un unique GAP pour chaque carte de caractéristiques en sortie de convolution. Yedroudj-Net est le réseau qui a donné les meilleurs résultats lors des expérimentations. La figure 3.4 illustre l'architecture du réseau Yedroudj-Net.

[1] La fonction *truncature* est donnée par $\begin{cases} -T, & x < -T \\ x, & -T \leq x \leq T \\ T, & x > T \end{cases}$, pour un certain $T \in \mathbb{N}$.

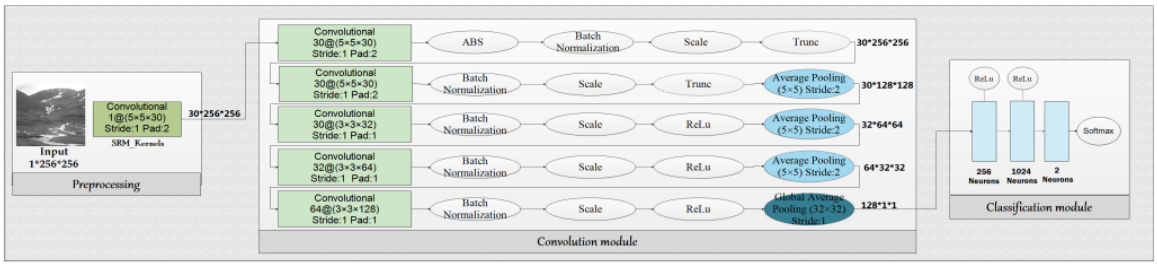

FIGURE 3.4 – Architecture du réseau Yedroudj-Net, figure extraite de [14].

3.2 Proposition d'architecture : Dilated-Yedroudj-Net

En complément des architectures issues de l'état de l'art, nous avons proposé une nouvelle architecture, *Dilated Yedroudj-Net*, appelée ainsi en raison de son utilisation de la convolution dilatée. Cette architecture a été conçue comme une extension de Yedroudj-Net choisi pour ses performances supérieures lors des expériences.

Principe

Cette architecture est inspirée de travaux effectués sur la détection d'objets à plusieurs échelles [9], dont l'idée de départ est d'effectuer plusieurs convolutions en parallèle sur des versions redimensionnées de l'image d'entrée. Cependant, comme nous l'avons dit, nous ne pouvons pas nous permettre d'effectuer des ré-échantillonages sans perdre le bruit stéganographique. Nous utilisons donc le principe de la convolution dilatée, qui consiste à espacer les éléments du noyau de convolution. Dans la figure 3.5, nous illustrons l'architecture Dilated Yedroudj-Net. Partant du schéma donné en figure 3.4, nous remplaçons la première convolution du *Convolution module* par trois convolutions parallèles, en faisant attention à ce que *le nombre de poids à apprendre reste le même*. Cette précaution nous permet de comparer les performances des architectures de manière équitable, et de s'assurer qu'un éventuel changement dans ces performances ne soit dû qu'à l'utilisation de ce nouveau type de convolution.

Au delà d'une invariance au changement de dimension, nous pensons que ce type de réseau serait particulièrement adapté à un *changement de résolution*, mais nous n'avons pas pu mener d'expérience en ce sens. En effet, la base d'images sur laquelle nous travaillons ne présente pas de changement de résolution ; les images de dimensions plus petites ont la même résolution métrique que les images plus grandes. Cependant, un des intérêts d'utiliser cette architecture sur la base dont nous disposons est de vérifier que les performances ne changeaient pas par rapport au réseau Yedroudj-Net original.

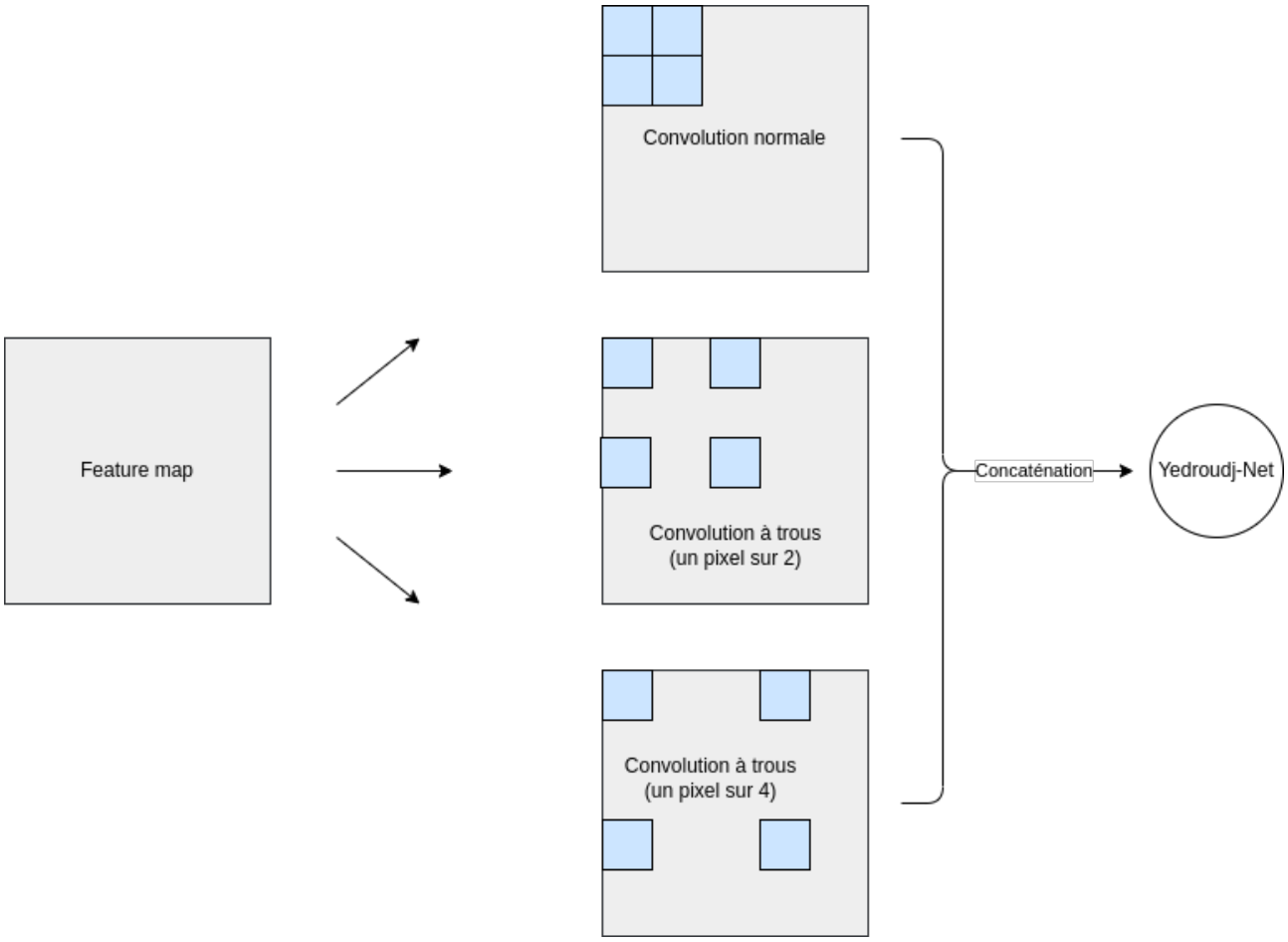

FIGURE 3.5 – Illustration des convolutions en parallèle.

4

Protocole expérimental

4.1 Base d'images

Le choix de la base d'image est primordial pour l'entraînement des différents réseaux. Ce choix est d'autant plus primordial dans notre cas que, à payload égal, la détéctabilité d'un signal stéganographique dépend de la dimension de la Cover (loi de la racine carrée [7]) et que nous cherchons justement à analyser la performance des modèles en fonction de la dimension des images d'entrée. Nous discutons ici de la mise au point de cette base d'images.

Set d'entraînement et set de test Nous disposons de deux ensembles d'images, l'un pour l'entraînement et l'autre pour le test. Ces deux ensembles sont constitués de la même façon, décrite ci-après. Il est important de noter que ces deux ensembles sont identiques pour tous les réseaux ; en particulier, **tous les réseaux sont testés sur le même ensemble d'images**. De plus, il y a toujours autant d'images Stegos que d'images Covers dans chacun des ensembles.

Base des Cover

Nous utilisons pour ce stage une base mise au point au cours de travaux antérieurs[1]. Cette base est constituée d'images de type « pseudo-gigognes ». À partir d'une grande image, l'on a découpé (*crop*) des images carrées de dimensions inférieures en respectant la carte de coût d'insertion (c'est à dire que le découpage doit avoir sensiblement la même distribution de coûts que l'image d'origine) de façon à obtenir des images aussi similaires que possible du point de vue de la sécurité empirique. Ce procédé que nous avons nommé **SmartCrop** a été répété plusieurs fois pour obtenir des images de dimensions 2048x2048, 1024x1024, 512x512, 256x256. Toutes les images sont en niveaux de gris car la couleur est un cas particulier encore aujourd'hui étudié ; voir l'article en cours de soumission [13] abordant les dernières propositions sur le traitement de la couleur.

[1] Réalisés par Douglas Benhamou lors d'un stage de M2 recherche en 2021.

Base des Stego

La base des Stego est constituée à partir de la base des Cover en leur appliquant l'algorithme d'insertion S-UNIWARD [4]. Chaque image Cover a son équivalent Stego. La question centrale est : à quelle valeur de payload faut-il insérer ?

Détermination des payload size

Nous avons fait le choix, pour des images 256×256, de fixer le payload à 0.4 bits par pixel (bpp). En utilisant l'équation 1.1, nous trouvons pour la constante de détectabilité. :

$$k = \frac{0.4 \times 256 \times 256}{\sqrt{256 \times 256} \times \log(256 \times 256)}.$$

Ceci nous a permis d'obtenir le payload à utiliser pour toutes les autres tailles. Ceux-ci sont résumés dans le tableau 4.1 :

$w \times h$	Valeur du payload en bpp
512×512	0.225
1024×1024	0.125
2048×2048	0.06875

TABLE 4.1 – Correspondance taille d'image/payload relatif en utilisant la loi de la racine carrée.

4.2 Protocole d'expérimentation

Réseau clairvoyant Nous appelons *réseau clairvoyant* un réseau entraîné sur une dimension d'image connue (par exemple, toutes les images vues par le réseau lors de l'entraînement sont de taille 256×256). Nous désignons ces réseaux de la façon suivante : `ARCHI-DIM-PAYLOAD`, où `ARCHI` désigne le nom de l'architecture, `DIM` désigne la dimension sur laquelle le réseau a été entraîné et `PAYLOAD` le payload associé. Par exemple, une architecture SID entraînée sur des images de taille 256×256 avec un payload de 0.4 sera désignée par `SID-256-0.4`.

Réseau multi Nous appelons *réseau multi* un réseau entraîné sur plusieurs dimensions d'images. Nous faisons attention à ce qu'un réseau multi reçoive lors de l'apprentissage autant d'images de chaque dimension et que la taille de l'ensemble de données soit la même pour un réseau multi et un réseau clairvoyant.

Nous souhaitons comparer entre eux les réseaux sur des dimensions vues mais aussi non vues lors de l'apprentissage.

Pipeline

Chaque expérience suit la même séquence d'étapes. On fixe d'abord l'architecture et le type de réseau (clairvoyant ou multi). Si le réseau est de type clairvoyant, on

fixe également la taille des images qui seront utilisées pour l'entraînement. Le set des images d'entraînement est ensuite séparé aléatoirement en train (représentant 80 % des images) et validation. A la fin du processus d'apprentissage, on enregistre les poids ayant permis la meilleure performance lors la validation. On teste le réseau ainsi entraîné sur l'ensemble des sets de test (i.e, on fait un test sur chaque taille d'image) et on calcule les Accuracy et les matrices de confusion.

5

Résultats

5.1 Discussion sur la loi de la racine carrée

Dès nos premières expériences, nous avons remarqué une déviation entre les résultats prévus par la loi de la racine carrée et les résultats expérimentaux. En effet, si la loi était vérifiée, alors pour une architecture fixée, chacun des réseaux clairvoyants devrait avoir la même accuracy lorsque testé sur sa propre dimension aux payload indiqués par la loi. Autrement dit, pour SID par exemple, en utilisant les données du tableau 4.1, les réseaux `SID-256-0.4` et `SID-512-0.225` devraient avoir la même accuracy lorsque testés respectivement sur des images 256×256 et des images 512×512. Or cela n'était pas le cas : nous avons obtenu pour `SID-256-0.4` une accuracy de 69.15% et pour `SID-512-0.225` une accuracy de 61,93%. Cet écart est probablement dû aux hypothèses que la loi de la racine carrée fait sur les images, que les auteurs de l'article [7] qualifient de fortes, et qui ne sont pas réalistes en pratique.

D'une part, le résultat du théorème est asymptotique (« for sufficiently large N »). La question se pose donc de savoir si les tailles d'images que l'on considère sont « suffisamment grandes » pour pouvoir appliquer le théorème. D'autre part, le théorème original porte sur la *batch steganography*, où le payload est distribué sur un ensemble de N images **supposées indépendantes**. Pour se ramener à notre cas, c'est à dire celui d'une image individuelle de taille variable, N devenant le nombre de pixels, les auteurs avancent que l'on peut considérer ces pixels comme un ensemble « d'images » parmi lesquels l'insertion peut se faire. Il est cependant très improbable que les pixels d'une même image soient indépendants, ce qui renforce la position selon laquelle cette hypothèse est pratiquement irréaliste.

En raison de l'impossibilité d'utiliser la loi de la racine carrée pour établir une même détectabilité, nous avons procédé à un ajustement manuel des payloads. Plus précisément, nous sommes parti de l'accuracy du réseau `Yedroudj-Net-256-0.4` sur les images 256 et avons cherché les payloads $\mathtt{p}_{512}$ et $\mathtt{p}_{1024}$ tels que les réseaux `Yedroudj-Net-512-`$\mathtt{p}_{512}$ et `Yedroudj-Net-256-`$\mathtt{p}_{1024}$ aient une accuracy comparable lorsque testés sur des images respectivement 512 et 1024. Nous obtenons alors le tableau 5.1.

Dimension `D`	Payload `P`	Accuracy `Yedroudj-Net-D-P`
256	0.4	76.97%
512	0.3204	76.38%
1024	0.28895	76.78%

TABLE 5.1 – Payload obtenus pour assurer une détéctabilité similaire sur toutes les dimensions.

Taille	Accuracy
256×256	71,6%
512×512	66,15%
1024×1024	62,67%

5.2 Etude de l'invariance

Réseaux SiaSteg, Zhu-Net

Nous n'avons pas pu aller au bout de notre démarche avec ces réseaux car nous avons rencontré l'impossibilité technique d'entraîner ces réseaux pour des dimensions supérieures à 256×256. Nous présentons cependant les résultats partiels obtenus, qui supportent malgré tout la constatation d'une absence d'invariance en performance à la dimension.

Architectures SID, Yedroudj-Net et Dilated-Yedroudj-Net

Nous avons effectué nos expérimentations principalement avec les architectures SID et Yedroudj-Net. Nous obtenons les tableaux 5.2 pour SID, 5.3 pour Yedroudj-Net[1] et 5.4 pour Dilated-Yedroudj-Net[2,3]. Nous constatons que lorsqu'un réseau clairvoyant est testé sur une autre dimension que celle sur laquelle il a été entraînée, ses performances baissent systématiquement. Par conséquent, l'invariance en performance ne semble pas assurée. Il est intéressant de constater que, bien que Yedroudj-Net soit globalement plus performant que SID, la distribution des scores est similaire. Notamment, la « diagonale » des résultats des SID clairvoyants varie peu, ce qui confirme notre critère de *même détectabilité*. Nous pouvons dire la même chose de l'architecture Dilated-Yedroudj-Net, bien que sa grande similarité avec l'architecture Yedroudj-Net constitue un biais important.

Taille	`SID-256-0.4`	`SID-512-0.3204`	`SID-1024-0.28895`
256×256	69.48%	67.05%	60,9%
512×512	69.30%	70.7%	66.93%
1024×1024	66.73%	66.93%	69.62%

TABLE 5.2 – Accuracy de l'architecture SID en fonction de la dimension des images testées.

[1] L'architecture Yedroudj-Net est abrégée par Y pour des raisons d'espace.
[2] L'architecture Dilated-Yedroudj-Net est pareillement abrégée par DYN.
[3] Nous n'avons pas eu le temps d'entraîner cette architecture sur des images 1024×1024.

Taille	Y-256-0.4	Y-512-0.3204	Y-1024-028895
256×256	76.97%	73.48%	71.76%
512×512	74.55%	76.38%	74.97%
1024×1024	72.83%	73.57%	76.78%

TABLE 5.3 – Accuracy de l'architecture Yedroudj-Net en fonction de la dimension des images testées.

Taille	DYN-256-0.4	DYN-512-0.3204
256×256	77,7%	76,25%
512×512	75,21%	77,3%
1024×1024	72,03%	76,88%

TABLE 5.4 – Accuracy de l'architecture Dilated-Yedroudj-Net en fonction de la dimension des images testées.

L'étude des matrices de confusion nous permet une analyse plus précise. On observe en effet que les erreurs ne sont pas du même type selon si un réseau clairvoyant est testé sur une dimension inférieure ou supérieure à la dimension sur laquelle il a été entraîné. Lorsque testé sur une dimension inférieure, le réseau a tendance à surclasser en Stego, tandis que lorsqu'il est testé sur une dimension supérieure, ce sont les Cover qui sont surestimées. Ceci se visualise bien dans le cas du réseau `Dilated-Yedroudj-Net-512-0.3204` dont les matrices de confusion sont données en table 5.5c, dont les scores lors des tests sur des images 256×256 et 1024×1024 sont très similaires, mais dont le comportement selon la dimension est très différent.

Autre exemple, si l'on compare les réseaux `Yedroudj-Net-256-0.4` et `Yedroudj-Net-1024-0.28895` qui présentent des Accuracy similaires lorsque testés sur des images 512×512. L'observation de leur matrices de confusion données par les tables 5.6c et 5.7c montre que `Yedroudj-Net-256-0.4` fait plus d'erreur en Cover (875 faux négatifs pour 652 faux positifs) tandis que `Yedroudj-Net-1024-0.28895` a le comportement opposé (546 faux négatifs contre 956 faux positifs). **Ainsi, même si leur nombre d'erreurs en valeur absolue est similaire, ils ne les font pas dans les même domaines**, ce qui, dans un cas réel, aurait un impact sur le choix de l'architecture à utiliser.

Ces résultats sont encore à compléter, notamment par la comparaison avec un réseau multi. A l'heure de la rédaction de ce rapport (Juin 2022), il n'a pas été possible d'entraîner un tel réseau, mais il est prévu de le faire au cours des mois de Juin et Juillet 2022.

Seuil de discrimination

La différence du comportement observé selon si nous testions sur une dimension supérieure ou inférieure à la dimension d'entraînement nous a amené à nous interroger sur l'existence d'un effet de seuil. Par défaut, lors d'une prédiction, nous choisissons la classe obtenant la probabilité la plus élevé, ce qui revient à fixer un seuil à 50%, comme

TABLE 5.5 – Matrices de confusion de `Dilated-Yedroudj-Net-512-0.3204` testé sur différentes tailles d'images. Sur des images 256, le réseau classe 54.92% d'entre elles en Stego, alors que sur des images 1024 il ne classe que 42.55% d'entre elles en Stego. Cela démontre un comportement très différent malgré une Accuracy similaire.

Vérité \ Prédiction	Cover	Stego
Cover	2140 (35.67%)	860 (14.33%)
Stego	565 (9.42%)	2435 (40.58%)

(a) Matrice de confusion de `Dilated-Yedroudj-Net-512-0.3204` testé sur images 256.

Vérité \ Prédiction	Cover	Stego
Cover	2410 (40.17%)	590 (9.83%)
Stego	772 (12.87%)	2228 (37.13%)

(b) Matrice de confusion de `Dilated-Yedroudj-Net-512-0.3204` testé sur images 512.

Vérité \ Prédiction	Cover	Stego
Cover	2530 (42.17%)	470 (7.83%)
Stego	917 (15.28%)	2083 (34.72%)

(c) Matrice de confusion de `Dilated-Yedroudj-Net-512-0.3204` testé sur images 1024.

TABLE 5.6 – Matrices de confusion de `Yedroudj-Net-256-0.4` testé sur différentes tailles d'images. Nous voyons une baisse importante du nombre de vrais positifs lorsque la dimension testée augmente, couplée à une hausse importante du nombre de faux négatifs. Le réseau prédit moins bien les Stegos sur des dimensions supérieures.

Vérité \ Prédiction	Cover	Stego
Cover	2279 (37.98%)	721 (12.02%)
Stego	661 (11.02%)	2339 (38.82%)

(a) Matrice de confusion de `Yedroudj-Net-256-0.4` testé sur images 256.

Vérité \ Prédiction	Cover	Stego
Cover	2348 (39.13%)	652 (10.87%)
Stego	875 (14.58%)	2125 (35.42%)

(b) Matrice de confusion de `Yedroudj-Net-256-0.4` testé sur images 512.

Vérité \ Prédiction	Cover	Stego
Cover	2363 (39.38%)	637 (10.62%)
Stego	993 (16.55%)	2007 (33.45%)

(c) Matrice de confusion de `Yedroudj-Net-256-0.4` testé sur images 1024.

TABLE 5.7 – Matrices de confusion de `Yedroudj-Net-1024-0.28895` testé sur différentes tailles d'images. Nous voyons une baisse importante du nombre de vrais négatifs lorsque la dimension testée diminue, couplée à une hausse importante du nombre de faux positifs. Le réseau prédit moins bien les Covers sur des dimensions inférieures.

Vérité \ Prédiction	Cover	Stego
Cover	1856 (30.93%)	1144 (19.07%)
Stego	550 (9.17%)	2450 (40.83%)

(a) Matrice de confusion de `Yedroudj-Net-1024-0.28895` testé sur images 256.

Vérité \ Prédiction	Cover	Stego
Cover	2044 (34.06%)	956 (15.93%)
Stego	546 (9.1%)	2454(40.9%)

(b) Matrice de confusion de `Yedroudj-Net-1024-0.28895` testé sur images 512.

Vérité \ Prédiction	Cover	Stego
Cover	2137 (35.62%)	863 (14.38%)
Stego	530 (8.83%)	2470 (41.17%)

(c) Matrice de confusion de `Yedroudj-Net-1024-0.28895` testé sur images 1024.

expliqué en section 2.4. Si un effet de seuil existe, alors par exemple nous pourrions corriger ce seuil lorsqu'un réseau fait une prédiction sur une dimension supérieure à celle sur laquelle il a été entraîné pour empêcher le surclassement en Cover.

Pour déterminer si un tel effet de seuil existe, nous cherchons à déterminer un seuil qui permette de maximiser l'Accuracy sur une dimension donnée. pour cela, nous disposons dans un tableau l'ensemble des prédictions réalisées par un stéganalyseur sur la base de train de cette dimension, dans l'ordre croissant des probabilités de Cover, comme illustré dans la figure 5.1. Pour maximiser l'Accuracy, d'après sa définition donnée par l'équation 2.6, nous cherchons dans ce tableau le seuil qui maximise la somme $\mathrm{VN} + \mathrm{FP}$, c'est à dire le nombre de Stego **à gauche** du seuil et le nombre de Cover **à droite** du seuil. Une fois ce seuil obtenu, nous l'utilisons pour tester le stéganalyseur sur la base de test.

Nous avons réalisé ce protocole avec les réseaux `Yedroudj-Net-512-0.3204` et `SID-512-0.3204`. Le tableau 5.8 résume les résultats obtenus, qui infirment l'hypothèse de l'existence d'un effet de seuil.

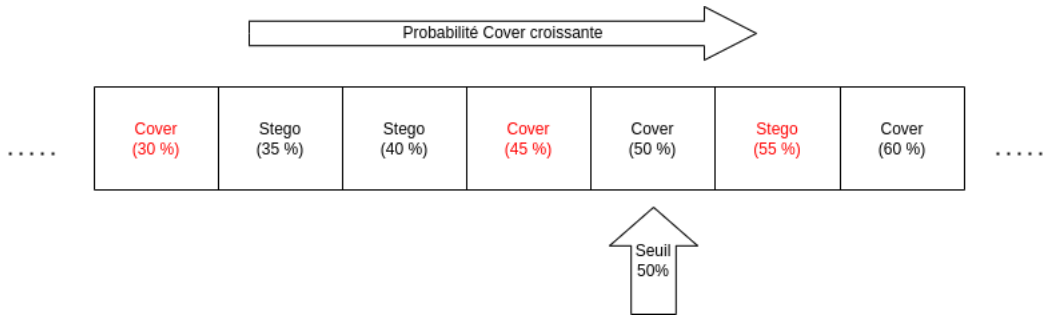

FIGURE 5.1 – Illustration du tableau trié des prédictions. Les prédictions en rouge sont les prédictions incorrectes, et les prédictions en noir sont les prédictions correctes.

	`Yedroudj-Net-512-0.3204`	`SID-512-0.3204`
Seuil	42.85%	48.28%
Accuracy	73.80%	66.85%

TABLE 5.8 – Résultats du seuillage sur les réseaux `Yedroudj-Net-512-0.3204` et `SID-512-0.3204` testés sur des images 256×256. On constate une amélioration marginale pour `Yedroudj-Net-512-0.3204` (73.80% d'Accuracy avec seuillage contre 73.48%), tandis que `SID-512-0.3204` voit ses performances légèrement baisser (66.85% d'Accuracy avec seuillage contre 67.03%). Il ne semble donc pas y avoir un effet de seuil.

Bibliographie

[1] Marc Chaumont. *Digital Media Steganography 1st Edition : Principles, Algorithms, and Advances*, chapter 14 « Deep Learning in steganography and steganalysis », pages 321–349. Elsevier, Juillet 2020.

[2] Marc Chaumont, Patrick Bas, and Rémi Cogranne. *Sécurité Multimédia - Partie 1 : Authentification et insertion de données cachées*, chapter 6 and 7. ISTE Editions, 2020.

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

[4] Vojtech Holub, Jessica J. Fridrich, and Tomás Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014 :1–13, 2014.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift, 2015.

[6] Andrew D. Ker, Patrick Bas, Rainer Böhme, Rémi Cogranne, Scott Craver, Tomáš Filler, Jessica Fridrich, and Tomáš Pevný. Moving Steganography and Steganalysis from the Laboratory into the Real World. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, IH & MMSec '13, page 45–58, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Andrew D. Ker, Tomáš Pevný, Jan Kodovský, and Jessica Fridrich. The Square Root Law of Steganographic Capacity. In *Proceedings of the 10th ACM Workshop on Multimedia and Security*, MM &;Sec '08, page 107–116, New York, NY, USA, 2008. Association for Computing Machinery.

[8] Auguste Kerckhoffs. La cryptographie militaire. In *Journal des sciences militaires*, volume IX, pages 5–83. 1883.

[9] Diego Marcos, Benjamin Kellenberger, Sylvain Lobry, and Devis Tuia. Scale equivariance in cnns with vector fields. *ArXiv*, abs/1807.11783, 2018.

[10] Jiangqun Ni, Jian Ye, and Yang YI. Deep Learning Hierarchical Representations for Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, PP :1–1, 06 2017.

[11] Gustavus J Simmons. The Prisoners' Problem and the Subliminal Channel. In David Chaum, editor, *Advances in Cryptology : Proceedings of Crypto 83*, pages 51–67. Springer US, Boston, MA, 1984.

[12] Clément Fuji Tsang and Jessica J. Fridrich. Steganalyzing Images of Arbitrary Size with CNNs. *electronic imaging*, 2018 :121–1–121–8, 2018.

[13] Kangkang Wei, Weiqi Luo, Shunquan Tan, and Jiwu Huang. Universal deep network for steganalysis of color image based on channel representation. *ArXiv*, abs/2111.12231, 2021.

[14] Mehdi Yedroudj, Frédéric Comby, and Marc Chaumont. Yedroudj-net : An efficient cnn for spatial steganalysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2092–2096, 2018.

[15] Weike You, Hong Zhang, and Xianfeng Zhao. A Siamese CNN for Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, 16 :291–306, 2021.

[16] Ru Zhang, Feng Zhu, Jianyi Liu, and Gongshen Liu. Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis. *IEEE Transactions on Information Forensics and Security*, 15 :1138–1150, 2020.