

# Rapport du projet S7 : Robust Segmentation Networks

Yazid Charef, Julius Graf, Sarra Nagaz, Ayoub Sboui  
*Équipe P06, Pôle Projet Data Science P04, CentraleSupélec, Université Paris-Saclay*

Encadré par Jean-Christophe Pesquet et Matthieu Terris  
*Center for Visual Computing, OPIS Inria Group, CentraleSupélec, Université Paris-Saclay*

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>	6.3.1	Apprentissage correct du modèle . . . . .	8
<b>2</b>	<b>Problématique</b>	<b>2</b>	6.3.2	Quel poids de régularisation lipschitzienne $\lambda$ permet d'optimiser le trade-off entre précision du modèle et constante de Lipschitz $L(A_\theta^R)$ faible? . . . . .	8
<b>3</b>	<b>Contexte scientifique et contribution</b>	<b>2</b>	6.3.3	Instabilités numériques . . . . .	9
<b>4</b>	<b>Concepts mathématiques</b>	<b>3</b>	6.3.4	Robustesse aux perturbations adverses du modèle régularisé . . . . .	10
	4.1 Réseaux de neurones comme opérateurs . . . . .	3			
	4.2 Régularisation lipschitzienne . . . . .	3	7	<b>Comparaison des deux approches</b>	<b>10</b>
<b>5</b>	<b>Modèle supervisé</b>	<b>4</b>	8	<b>Conclusion</b>	<b>10</b>
	5.1 Architecture du réseau et formalisation mathématique . . . . .	4	8.1	Conclusion . . . . .	10
	5.1.1 Architecture du réseau . . . . .	4	8.2	Perspectives . . . . .	10
	5.1.2 Fonction de perte . . . . .	4	9	<b>Annexe</b>	<b>12</b>
	5.1.3 Optimiseur . . . . .	5	9.1	Simulations sur le modèle auto-supervisé . . . . .	12
	5.1.4 Évaluation de la précision du modèle . . . . .	5	9.2	Simulations sur instabilités numériques du modèle auto-supervisé . . . . .	13
	5.1.5 Base de données COCO . . . . .	5	9.2.1	Masques . . . . .	13
	5.2 Simulations numériques . . . . .	5	9.2.2	Graphiques . . . . .	14
	5.2.1 Résultats sans régularisation de la constante de Lipschitz . . . . .	5	9.3	Simulations sur la robustesse du modèle auto-supervisé . . . . .	14
	5.2.2 Résultats avec régularisation de la constante de Lipschitz . . . . .	5	9.3.1	Masques . . . . .	14
			9.3.2	Graphiques . . . . .	15
<b>6</b>	<b>Modèle auto-supervisé</b>	<b>6</b>	9.4	Simulations pour le modèle supervisé . . . . .	15
	6.1 Formalisation mathématique . . . . .	6			
	6.1.1 Notations . . . . .	6			
	6.1.2 Traitement en entrée et en sortie . . . . .	6			
	6.1.3 Fonction de perte . . . . .	7			
	6.1.4 Visualisation des masques . . . . .	7			
	6.1.5 Réseau de neurones . . . . .	7			
	6.1.6 Boucle <i>Backward</i> . . . . .	7			
	6.2 Paramètres des simulations numériques . . . . .	8			
	6.2.1 Base de données CIFAR-10 . . . . .	8			
	6.3 Simulations numériques . . . . .	8			

## Abstract

Les réseaux de neurones sont de plus en plus utilisés pour la segmentation d'images dans divers domaines, par exemple l'imagerie médicale. Cependant, ils peuvent être sensibles à des perturbations adverses petites et produire des résultats incorrects. Nous proposons deux approches pour construire des réseaux de neurones de segmentation d'images robustes. Nous réduisons la sensibilité des réseaux aux perturbations

indésirables, améliorant ainsi leur robustesse et leur précision dans la segmentation des images.

La première approche consiste en un modèle d'apprentissage auto-supervisé découlant d'un modèle de classification. En utilisant PyTorch, nous avons construit un réseau de neurones basique mais très répandu, le ResNet-18, adapté à la classification. De fait, en implémentant une méthode de visualisation ainsi que la régularisation lipschitzienne, nous avons construit le modèle de segmentation auto-supervisé.

Pour mettre en œuvre notre méthode de contrôle de la constante de Lipschitz, nous nous sommes basés sur le travail accompli dans [12]. L'approche proposée dans cette publication consiste à ajouter une régularisation basée sur la norme spectrale de la réflexion du réseau dans la fonction de perte du réseau. Nous avons étudié son code source et l'avons adapté à notre projet. Pour évaluer les performances de notre approche, nous avons utilisé la base de données CIFAR-10. Cette base de données contient un ensemble diversifié d'images, ce qui nous permet de tester notre réseau sur une grande variété de cas. Nous avons analysé les améliorations potentielles apportées par notre méthode de contrôle de la constante de Lipschitz.

La deuxième approche consiste en un modèle d'apprentissage supervisé. Précisément, nous avons implanté à la main une version du U-Net afin de réaliser de la segmentation d'images multi-classes. Les performances de cette approche ont été testé sur la base de données COCO, qui est parfaitement adapté à des tâches de classification multi-classes.

## 1 Introduction

Les réseaux de neurones sont utilisés dans plusieurs domaines, notamment dans l'imagerie médicale. Ils servent à détecter et diagnostiquer des tumeurs cancéreuses. Aujourd'hui, il est possible grâce à l'intelligence artificielle d'identifier les caractéristiques spécifiques associées aux tumeurs.

Cependant, ces réseaux de neurones peuvent être sensibles à des perturbations adverses et invisibles à l'œil humain, ce qui remet en question leur fiabilité et leur utilité. Ces réseaux non robustes face aux perturbations peuvent générer des résultats incorrects, comme illustré par la figure 1 où on associe à tort une image perturbée d'un cochon à un avion.

L'objectif de notre travail est de construire un réseau de neurones pour la segmentation d'images qui soit fermement nonexpansif. Cette contrainte de régularité devrait considérablement améliorer la robustesse de réseau de neurons. Le problème consiste en la manière on peut contrôler la constante de Lipschitz de toutes les couches de notre réseau pour assurer la robustesse de la segmentation.

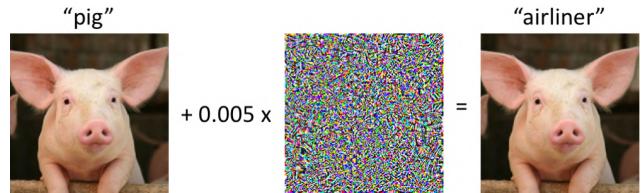


FIGURE 1 – Classification erronée par perturbation adverse dans [11]

Afin de résoudre ce problème, nous l'avons formalisé puis implanté sur deux approches différentes une régularisation lipschitzienne, visant à réduire la constante de Lipschitz du réseau. Pour le modèle auto-supervisé, nous testons la robustesse sur la base de données CIFAR-10, puis pour le modèle supervisé sur la base de données COCO.

## 2 Problématique

Comment, en utilisant des approches d'apprentissage auto-supervisé et supervisé, peut-on développer des méthodes efficaces pour renforcer la robustesse et la précision des réseaux de neurones dans la segmentation d'images, en particulier face à des perturbations adverses ?

## 3 Contexte scientifique et contribution

La segmentation robuste des images représente l'un des défis majeurs en machine learning, comme exposé dans l'introduction. L'enjeu consiste à développer un réseau neuronal capable de segmenter des images tout en étant insensible aux perturbations.

De nombreux travaux se sont penchés sur la résolution de ce problème complexe. La segmentation trouve des applications diverses dans différents domaines, chacune présentant des problématiques spécifiques.

Les réseaux de neurones dédiés à la segmentation d'images ont été appliqués avec succès à l'imagerie médicale pour l'identification des zones tumorales cancéreuses [4]. Les résultats obtenus ont démontré une précision remarquable, atteignant 97,50% après un entraînement de 37 époques. Cependant, l'optimisation des paramètres du réseau demeure un défi crucial pour assurer une segmentation aussi précise que souhaitée.

Dans le domaine de l'imagerie médicale, ces réseaux ont également été utilisés pour la segmentation des vaisseaux sanguins rétiniens [1]. Les applications en imagerie biomédicale ont souvent recours au réseau UNet en tant que réseau neuronal convolutif, exploitant son architecture Encoder-Decoder avantageuse. Ce modèle a été employé avec succès pour la segmentation d'images du pancréas, atteignant une précision de 84,8% [7].

En raison des limitations des bases de données, telles que leur petite taille ou l'absence de masques, certains travaux se sont tournés vers l'apprentissage auto-supervisé,

notamment pour la segmentation en télédétection à partir de grandes données satellites [2]. Cependant, ces approches peuvent être sensibles aux perturbations infinitésimales.

Une alternative pour atteindre l'objectif de segmentation robuste consiste en la combinaison de réseaux de neurones, comme SFNet - ResNet18 [6]. Cette approche permet d'obtenir une segmentation rapide, jusqu'à 0,211s par image de test, avec une précision de 92,58%.

En ce qui concerne la robustesse du modèle, nous avons fait référence au travail [12], qui a résolu ce problème dans le contexte du débruitage en contrôlant la constante de Lipschitz du réseau. Nous avons adapté cette démarche pour l'appliquer à la segmentation d'images.

Par ailleurs, nous avons utilisé un modèle supervisé en exploitant des masques prédefinis sur une base de données importante, tout en optimisant les hyperparamètres du réseau et en contrôlant sa constante de Lipschitz afin d'obtenir un modèle robuste.

## 4 Concepts mathématiques

### 4.1 Réseaux de neurones comme opérateurs

La sensibilité des réseaux de neurones vis-à-vis des perturbations adverses (voir figure 1 ainsi que la sous-partie 6.2 de la partie 6) rend nécessaire l'ajout d'une régularisation lipschitzienne. Pour cela, définissons les termes nécessaires à la compréhension du problème : qu'est-ce qu'un réseau de neurones ? Qu'est-ce qu'un opérateur fermement non expansif ? Comment imposer la contrainte de régularité ?

Pour définir les réseaux de neurones, nous nous appuyons sur la définition générale proposée dans [12].

**Définition 1.** Soient  $(H_j)_{0 \leq j \leq m}$  des espaces de Hilbert réels non triviaux. On appelle réseau de neurones à  $M + 1$  couches tout opérateur  $A: H_1 \rightarrow H_m$  sous la forme

$$A = T_m \circ \cdots \circ T_1$$

où  $(T_j)_{1 \leq j \leq M}$  sont les couches du réseau, définies pour tout  $j \in \llbracket 1, M \rrbracket$  par

$$\begin{aligned} T_j: H_{j-1} &\rightarrow H_j \\ x &\mapsto \varphi_j(W_j x + b_j). \end{aligned}$$

Dans ce modèle, les  $(\varphi_j)_{1 \leq j \leq M}$  sont des opérateurs d'activations non linéaires,  $(\beta_j)_{1 \leq j \leq M} \in \mathcal{H}_j^M$  et les  $(T_j)_{1 \leq j \leq M}$  sont des opérateurs linéaires.

Dans toute la suite, nous notons pour simplifier  $A: C \rightarrow H$  l'opérateur de réseau de neurones.

**Définition 2.** Une fonction  $A: C \rightarrow H$  est dite lipschitzienne si il existe une constante  $L$  telle que

$$\forall (x, y) \in C^2, \|A(x) - A(y)\| \leq L(A) \|x - y\|$$

Dans notre travail, on recherche la robustesse vis-à-vis des perturbations adverses. Pour une application lipschitzienne, le comportement est décrit par la proposition ci-dessous.

**Proposition 1.** Soit  $A$  un réseau de neurones, de constante de Lipschitz  $L(A)$ . Pour une image  $x \in C$  et une perturbation  $\varepsilon \in C$ , nous avons

$$\|A(x + \varepsilon) - A(x)\| \leq L(A) \|\varepsilon\|.$$

On observe qu'imposer une condition sur  $L(A)$  permettrait de résoudre le problème. De fait, la condition de non-expansivité ferme permettrait de résoudre le problème. Définissons, en utilisant [10], ce qu'est un opérateur fermement nonexpansif.

**Définition 3.** Soit  $H$  un espace de Hilbert et  $C$  un sous-espace non vide de  $H$ . Un opérateur  $A: C \rightarrow H$  est dit fermement nonexpansif lorsque pour tous  $x$  et  $y$  de  $C$ , on a

$$\|A(x) - A(y)\|^2 \leq \langle A(x) - A(y), x - y \rangle.$$

En théorie, la nonexpansivité ferme est une contrainte de régularité très contraignante. Cela étant, comme le montre la proposition 2, elle revient à la nonexpansivité de la réflexion  $A^R := 2A - \text{Id}$  de l'opérateur.

**Proposition 2.** Un opérateur  $A$  est fermement nonexpansif si, et seulement si sa réflexion  $A^R := 2A - \text{Id}$  est nonexpansive.

*Démonstration.* Notons  $A^R := 2A - \text{Id}$ . Alors pour tous  $(x, y) \in C^2$ , on a

$$\|A^R(x) - A^R(y)\|^2 \leq \|x - y\|^2$$

si, et seulement si, en développant le carré

$$4\|A(x) - A(y)\|^2 - 4\langle A(x) - A(y), x - y \rangle \leq 0$$

ce qui se réécrit en l'inégalité

$$\|A(x) - A(y)\|^2 \leq \langle A(x) - A(y), x - y \rangle$$

d'où le résultat.  $\square$

### 4.2 Régularisation lipschitzienne

Nous allons alors appliquer cette propriété au réseau de neurones. En notant  $A_\theta$  le réseau de neurones, où  $\theta$  est le vecteur de paramètres à optimiser, on cherche alors à imposer que  $L(A_\theta^R) \leq 1$ . Pour cela, nous allons exploiter le théorème suivant.

**Théorème 1.** Soit  $A: C \rightarrow H$  une application lipschitzienne. Alors

$$L(A) = \sup_{x \in C} \|J_A(x)\|_S$$

où  $\|\cdot\|_S$  désigne la norme spectrale définie plus bas. On en déduit que si pour tout  $x \in C$ , on a  $\|J_A(x)\|_S \leq 1$ , alors  $L(A) \leq 1$ .

En conclusion, il s'agit de construire un modèle tel que  $A_\theta^R$  soit non-expansif, c'est-à-dire que  $L(A_\theta^R) \leq 1$ , ce qui, par le théorème précédent, revient à obtenir que pour tout

$x \in C$ , on ait  $\|J_{A_\theta^R(x)}\|_S \leq 1$ . On peut alors se poser la question de comment calculer en pratique la quantité  $\|J_{A_\theta^R(x)}\|_S$ . Pour cela, on peut revenir à la définition de norme spectrale. Celle-ci invoque la décomposition en valeurs singulières, rappelée ci-dessous.

**Théorème 2.** Soit  $M \in \mathcal{M}_{p,q}(\mathbb{R})$ . Il existe  $U \in O_p(\mathbb{R})$ ,  $V \in O_q(\mathbb{R})$  et  $\Sigma = (d_i \delta_{i,j})_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}}$  telle que  $M = U\Sigma V^\top$ .

On l'appelle **décomposition en valeurs singulières de  $M$** . Plus précisément,

$$\text{Sp}(\bar{A}^\top A) = \{d_i^2, i \in [1, q]\}.$$

On dit que  $\{d_i, i \in [1, q]\}$  est l'ensemble des valeurs singulières de  $M$ .

Par souci de rigueur,  $O_p(\mathbb{R})$  (pour  $p \in \mathbb{N}$ ) désigne l'ensemble des matrices orthogonales de  $\mathcal{M}_p(\mathbb{R})$ , c'est-à-dire que  $M \in O_p(\mathbb{R})$  lorsque  $M^\top M = MM^\top = I_p$ .

**Définition 4.** Soit  $M \in \mathcal{M}_{p,q}(\mathbb{R})$  avec  $p, q \in \mathbb{N}^*$ . On appelle **norme spectrale de  $M$**  l'objet

$$\|M\|_S = \max_{x \in \mathbb{R}^q \setminus \{0\}} \frac{\|Mx\|_{\ell^{p,2}}}{\|x\|_{\ell^{p,2}}}.$$

La norme spectrale correspond à la plus grande valeur singulière de  $M$ . Dans le cas d'une matrice carrée, on l'appelle **rayon spectral, noté  $\rho(M)$** .

Pour les simulations numériques, la plus grande valeur singulière de  $J_{A_\theta^R(x)}$  sera calculée par la méthode de la puissance.

**Théorème 3.** Soient  $M \in \mathcal{M}_q(\mathbb{R})$ ,  $\|\cdot\|$  une norme vectorielle sur  $\mathbb{R}^q$ ,  $\lambda_1, \dots, \lambda_q$  les valeurs propres de  $M$ . On suppose  $|\lambda_1| \geq \dots \geq |\lambda_q|$ . On note  $e_1$  un vecteur propre associé à  $\lambda_1$  et  $F = \text{Im}(M - \lambda_1 I_q)$ . Soient  $x_0 = \mu e_1 + f$  avec  $\mu \neq 0$  tiré aléatoirement et  $f \in F$ . Alors la suite  $(x_n)_{n \in \mathbb{N}}$  définie par

$$\forall n \in \mathbb{N}, x_{n+1} = \frac{Mx_n}{\|Mx_n\|}$$

est telle que  $\lim_{n \rightarrow +\infty} \|Mx_n\| = \rho(M)$ .

Ce théorème s'étend bien sûr au cas des matrices rectangulaires.

Pour conclure sur cette partie théorique, on s'intéresse à la mise en oeuvre formelle de la régularisation. Lors de l'entraînement, l'optimisation de  $\theta$  se fait par minimisation de la quantité de *loss*, qu'on notera ici simplement  $\mathcal{L}_{x,y}(\theta)$  ( $x$  et  $y$  désignent respectivement l'entrée et le label associé), mais qui sera expliquée plus en détail dans les parties qui suivent. On ajoute alors à cette quantité un terme qui, par processus de minimisation, permettra de réduire la quantité  $\|J_{A_\theta^R(x)}\|_S$  afin de la rendre plus petite que 1. Mathématiquement, le problème d'optimisation à résoudre s'écrit alors ainsi

$$\min_{\theta} \mathcal{L}_{x,y}(\theta) + \lambda \max \left( \|J_{A_\theta^R(X)}\|_S, 1 - \varepsilon \right)$$

où  $\lambda$  désigne le poids accordé à cette régularisation, et  $\varepsilon$  le niveau de précision.

## 5 Modèle supervisé

### 5.1 Architecture du réseau et formalisation mathématique

#### 5.1.1 Architecture du réseau

Nous avons implémenté le réseau U-net en nous inspirant du travail de [8]. La structure du réseau est représentée dans la Figure 2, comprenant un chemin contractant à gauche et un chemin expansif à droite. Le chemin contractant suit l'architecture classique d'un réseau convolutionnel, avec une séquence de deux convolutions  $3 \times 3$  (non rembourrées), suivies d'une activation par unité linéaire redressée (ReLU) et d'une opération de max pooling  $2 \times 2$  avec un pas de 2 pour le sous-échantillonnage.

À chaque étape de sous-échantillonnage, le nombre de canaux de caractéristiques est doublé. Le chemin expansif consiste en un upsampling de la carte de caractéristiques, suivi d'une convolution  $2 \times 2$  ("up-convolution") qui divise par deux le nombre de canaux de caractéristiques. Ensuite, il y a une concaténation avec la carte de caractéristiques correspondante du chemin contractant (préalablement rognée), suivie de deux convolutions  $3 \times 3$ , chacune suivie d'une activation ReLU.

Le rognage est nécessaire en raison de la perte de pixels de bord lors de chaque convolution. À la couche finale, une convolution  $1 \times 1$  est utilisée pour mapper chaque vecteur de caractéristiques à 64 composants, correspondant au nombre souhaité de classes. Dans l'ensemble, le réseau est composé de 23 couches de convolution.

Pour notre application de segmentation binaire, nous avons configuré le réseau avec 3 canaux en entrée, correspondant au RGB de l'image d'entrée, et un canal de sortie pour la segmentation binaire.

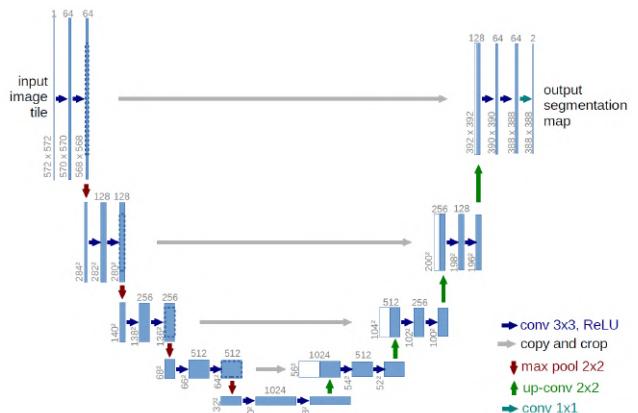


FIGURE 2 – Architecture U-Net

#### 5.1.2 Fonction de perte

Dans le contexte de la segmentation binaire, nous avons choisi d'utiliser la perte de log vraisemblance binaire avec logits comme fonction de perte pour notre modèle. Cette

fonction de perte peut être exprimée de la manière suivante :

$$\begin{aligned}\mathcal{L}(A_\theta(X), y) &= -y \cdot \log(\sigma(A_\theta(X))) \\ &\quad - (1 - y) \cdot \log(1 - \sigma(A_\theta(X)))\end{aligned}$$

Notons  $A_\theta$  notre modèle et  $X \in \mathbb{R}^{m \times n \times 3}$  une image d'entrée, où  $m$  et  $n$  désignent le format de l'image, et 3 est le nombre de canaux d'entrée d'une image RGB. Soient  $Y$  le vrai label et  $A_\theta(X)$  la prédiction.

Ainsi, nous cherchons à minimiser, par entraînement, pour une entrée  $X$  et le label associé  $y$  la quantité  $\mathcal{L}(A_\theta(X), y)$ .

### 5.1.3 Optimiseur

Nous avons choisi d'utiliser l'optimiseur de descente de gradient stochastique détaillé dans la sous-sous-section 6.1.6. Notre objectif est de minimiser l'expression

$$\mathcal{L}(A_\theta(X), y) + \lambda \max \left( \|J_{A_\theta^R(X)}\|_S, 1 - \varepsilon \right)$$

par rapport au vecteur de paramètres  $\theta$ . Où  $\lambda$  désigne le poids accordé à cette régularisation, et  $\varepsilon$  le niveau de précision,  $A_\theta(X)$  représente la prédiction du modèle pour le point de la base de données  $X$ ,  $y$  est le vrai masque binaire,  $\mathcal{L}$  est la fonction de perte définie précédemment, et  $J_{A_\theta^R(X)}$  représente le Jacobien de la sortie du modèle par rapport aux paramètres  $\theta$ .

### 5.1.4 Évaluation de la précision du modèle

Pour évaluer la précision du réseau, on utilise l'indice de Dice défini par :

$$D(\sigma(\hat{Y}), Y) = \frac{2 \cdot \#(\sigma(\hat{Y}) \cap Y)}{\#Y + \#\sigma(\hat{Y})}$$

avec  $\#\sigma(\hat{Y})$  représentant le nombre d'éléments de  $\sigma(\hat{Y})$  qui sont supérieurs à  $1/2$ .

### 5.1.5 Base de données COCO

Nous avons appliqué le modèle à la base de données COCO (Common Objects in Context). Il s'agit d'une collection d'images annotées utilisée pour la reconnaissance d'objets dans le domaine de la vision par ordinateur. Elle comprend plus de 200 000 images réparties sur 80 catégories d'objets, avec des annotations détaillées telles que des boîtes englobantes et des descriptions contextuelles. COCO est largement utilisée pour l'évaluation des algorithmes de détection d'objets, segmentation en raison de sa diversité et de sa qualité d'annotation. Nous avons commencé par une segmentation binaire de la base de données, c'est-à-dire avec deux classes : la Classe 0 correspond au background, et la Classe 1 correspond aux objets segmentés.



FIGURE 3 –  $B = 24$  images de COCO et masques associés

## 5.2 Simulations numériques

### 5.2.1 Résultats sans régularisation de la constante de Lipschitz

Nous avons commencé par réaliser des simulations sans régularisation de la constante de Lipschitz.

Nous avons choisi les hyperparamètres suivants :

- **Taille de batch** : 8
- **Nombre d'époques** : 10
- **Pas d'apprentissage initial** :  $\gamma = 10^{-3}$
- **Moment** :  $\mu = 0.9$
- **Weight decay** :  $\tau = 10^{-4}$ .

Nous avons obtenu les résultats suivants :

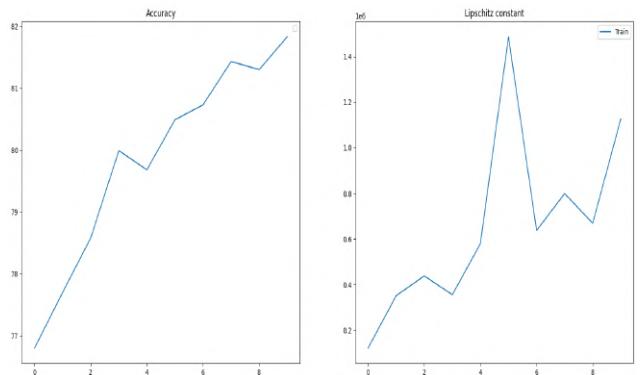


FIGURE 4 – Résultats de simulation du modèle pré-entraîné non régularisé

### 5.2.2 Résultats avec régularisation de la constante de Lipschitz

Dans cette section, les hyperparamètres du modèle demeurent inchangés comme mentionné précédemment. Cependant, nous avons varié les valeurs du coefficient de régularisation lipschitzienne  $\lambda$  afin d'explorer le compromis entre la précision du modèle et la constante de Lipschitz.

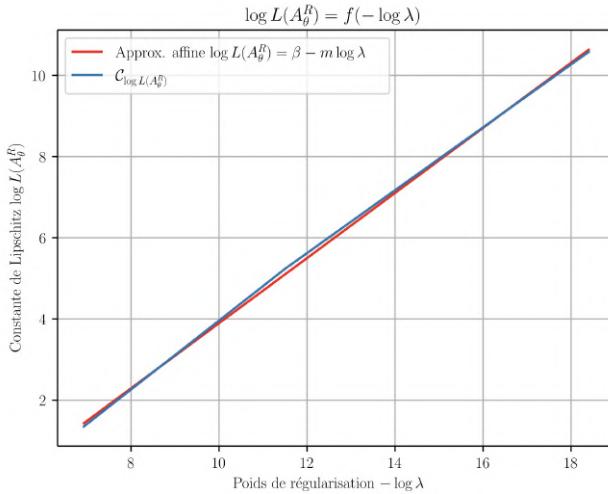


FIGURE 6 – Tracé du graphe approximatif de  $L(A_\theta^R)$  en fonction du poids de régularisation  $\lambda$  en échelle logarithmique. Le coefficient directeur de l’approximation affine est estimé à  $m = 0.802$ , l’ordonnée à l’origine à  $\beta = -4.12$ .

Par la suite de notre étude, nous avons révélé les résultats de la segmentation pour un ensemble de 8 images.

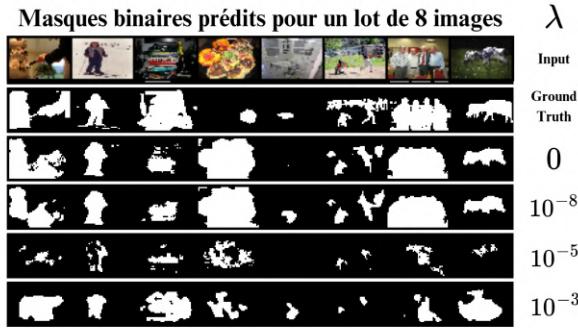


FIGURE 5 – Images d’entrée, masques binaires réels d’un lot de 8 images et en-dessous les masques binaires prédits pour un lot de 8 images avec différents poids de régularisation

Les résultats complets se trouvent dans l’annexe, i.e. section 9. Nous avons, comme pour le modèle auto-supervisé, étudié l’évolution de la constante de Lipschitz en fonction du poids de régularisation, ce qui mène au graphique et à la proposition suivante.

**Proposition 3.** Pour  $\lambda \in ]0, 1]$ , la constante de Lipschitz  $L(A_\theta^R)$  vérifie la loi empirique  $\log L(A_\theta^R) = \beta - m \log(\lambda)$  ce qui équivaut à

$$L(A_\theta^R) = \frac{C}{\lambda^m}$$

avec  $m \approx 0.802$  et  $\beta \approx -4.12$  soit  $C := e^\beta \approx 1.62 \cdot 10^{-2}$ .

Notamment, on observe sur la figure 6 que pour  $\lambda = 10^{-3}$ , la constante de Lipschitz se rapproche de la contrainte (mieux visible dans l’annexe sur la figure 40), atteignant une valeur d’environ 1. Cela se traduit par une pré-

cision qui rivalise avec l’état de l’art. Cependant, l’application d’une régularisation à des poids plus importants montre que cela affaiblit la précision du réseau. Les masques obtenus dans ce cas sont plus clairsemés, confirmant ainsi l’observation du modèle auto-supervisé, ce qui sera vu dans la partie suivante.

En parallèle, notre exploration a révélé qu’augmenter le poids de la régularisation conduit à des valeurs faibles pour la constante de Lipschitz, atteignant environ 2 pour  $\lambda = 10^{-3}$ . Bien que la précision diminue légèrement, elle reste au-dessus de 70%. Ce compromis entre la précision et la régularisation de la constante de Lipschitz est exprimé par la formule suivante :

$$\min_{\theta} \mathcal{L}(A_\theta(X), y) + \lambda \max \left( \left\| J_{A_\theta^R(X)} \right\|_S, 1 - \varepsilon \right)$$

À la lumière de ces observations, nous avons choisi de maintenir la valeur de  $\lambda$  à  $10^{-3}$ , ce qui résulte en une précision de 71.5% après 10 époques d’entraînement. Cette décision est basée sur la recherche d’un équilibre entre la précision du modèle et la régularisation de la constante de Lipschitz.

## 6 Modèle auto-supervisé

### 6.1 Formalisation mathématique

#### 6.1.1 Notations

Notons  $X \in \mathcal{M}_{n,p}(\mathbb{R}^3)$  une image d’entrée, où  $n$  et  $p$  désignent le format de l’image, et 3 est le nombre de canaux d’entrée d’une image RGB. De fait,  $X$  est en fait même à valeurs dans  $[0, 255]^3$ . Le réseau de neurones est noté  $A_\theta$  avec  $\theta$  le vecteur de paramètres à optimiser par entraînement. Dans notre modèle, nous faisons de la classification multi-classes, le nombre de classes étant noté  $c$ . La sortie du réseau est notée  $Y = A_\theta(X)$  et est dans  $\mathcal{M}_{n,p}(\mathbb{R}^c)$ . La structure du réseau sera plus expliquée en détail dans le paragraphe 6.1.5.

#### 6.1.2 Traitement en entrée et en sortie

Là, il s’agit de normaliser l’entrée du réseau  $X$  et la sortie. Par la fonction `ToTensor` de PyTorch, on normalise  $X$  dans  $[0, 1]^3$ . Dans la littérature, on peut trouver que la moyenne est  $(0.4914, 0.4822, 0.4465)^\top$  et l’écart-type  $(0.2023, 0.1994, 0.2010)^\top$ . En sortie, on réalise d’abord l’agrégation spatiale du modèle par l’application  $\varphi: \mathcal{M}_{n,p}(\mathbb{R}^c) \rightarrow \mathbb{R}^c$  définie par

$$\varphi(Y) = \left( 2 \log \left( \frac{1}{np} \sum_{i=1}^n \sum_{j=1}^p \exp \left( \frac{Y_{i,j}^{(k)}}{2} \right) \right) \right)_{k \in [1, c]}$$

pour  $Y \in \mathcal{M}_{n,p}(\mathbb{R}^c)$ . Pour clarifier ce procédé, on prend le LogSumExp sur les pixels afin que pour chaque classe i.e. chaque canal, on a une approximation lisse du logits maximal prédit sur l’image pour cette classe en sortie de la fonction  $\varphi$ . Ainsi, on a donc un détecteur pour chaque classe par agrégation spatiale.

### 6.1.3 Fonction de perte

Ensuite, on peut simplement calculer l'entropie croisée, comprenant la projection dans  $[0, 1]$  de la sortie agrégée, donnée par

$$\mathcal{L}(\varphi(Y), y) = - \sum_{k=1}^c y_k \frac{e^{\varphi(Y)_k}}{\sum_{\ell=1}^c e^{\varphi(Y)_\ell}}.$$

Ainsi, on cherche alors à résoudre par entraînement le problème d'optimisation pour une entrée  $X$  et le label  $y$  associé

$$\min_{\theta} \mathcal{L}(\varphi(A_{\theta}(X)), y).$$

La sensibilité des réseaux de neurones vis-à-vis des perturbations adverses (ajouter références de la littérature et résultats numériques) rend nécessaire l'ajout d'une régularisation lipschitzienne. On parvient alors à la proposition suivante, résumant l'approche auto-supervisée

**Proposition 4.** Notons  $X \in \mathcal{M}_{n,p}(\mathbb{R}^3)$  une image d'entrée et  $y \in \mathbb{R}^c$  le label de classification associé. En définissant  $\varphi$  et  $L$  comme ci-dessus, et en notant  $\lambda \in \mathbb{R}_+$  le poids accordé à cette régularisation, et  $\varepsilon > 0$  le niveau de précision, on cherche à minimiser la quantité

$$\mathcal{L}(\varphi(A_{\theta}(X)), y) + \lambda \max \left( \left\| J_{A_{\theta}^R(X)} \right\|_S, 1 - \varepsilon \right)$$

vis-à-vis du vecteur de paramètres  $\theta$ .

### 6.1.4 Visualisation des masques

Enfin, on souhaite visualiser les masques produits par le réseau. Pour ce faire, on utilise un procédé d'attention, qui permettra de représenter les masques de sortie sous forme d'image HSV. L'attention est notée  $A \in \mathcal{M}_{n,p}([0, 1])$  et est définie pour tout couple  $(i, j)$  de  $\llbracket 1, n \rrbracket \times \llbracket 1, p \rrbracket$  par

$$A_{i,j} = \sigma \left( \log \left( \sum_{k=1}^c \exp(Y_{i,j}^{(k)}) \right) \right) \in [0, 1]$$

où  $\sigma$  désigne la fonction sigmoïde définie sur  $\mathbb{R}$  par

$$\forall x \in \mathbb{R}, \sigma(x) = \frac{1}{1 + e^{-x}}.$$

Le masque est noté  $S \in \mathcal{M}_{n,p}(\llbracket 1, c \rrbracket)$  et est défini pour tout couple  $(i, j)$  de  $\llbracket 1, n \rrbracket \times \llbracket 1, p \rrbracket$  par

$$S_{i,j} = \arg \max_{k \in \llbracket 1, c \rrbracket} \sigma(Y_{i,j}^{(k)}).$$

L'attention  $A$  et le masque  $S$  représentent respectivement la saturation et le hue de l'image de prédiction.

Il est important de noter que cette méthode d'apprentissage est une méthode supervisée de classification. Les masques, eux, ne sont pas supervisés et sont en réalité qu'une méthode de visualisation tirée de la classification.

### 6.1.5 Réseau de neurones

Intéressons-nous alors à la résolution numérique de ce problème d'optimisation. Pour le modèle, nous avons

construit un ResNet-18 légèrement modifié en début et sortie. L'architecture est présentée en détail sur la figure 7. L'objectif de ces modifications est d'empêcher un *downsampling* trop rapide des données spatiales. De plus, on arrête le réseau avant les couches de projection habituelles afin d'obtenir en sortie une image de taille  $n \times p$  à  $c$  canaux (1 pour chaque classe) pour appliquer la méthode de visualisation développée ci-dessus. L'agrégation spatiale est alors réalisée "à la main" grâce à la fonction  $\varphi$ . En somme, on obtient le réseau suivant.

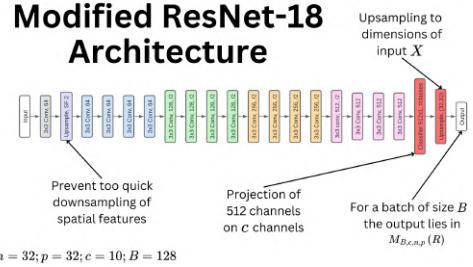


FIGURE 7 – Architecture du ResNet-18 modifié utilisé

À partir du réseau, noté  $A_{\theta}$ , on peut réaliser des prédictions  $A_{\theta}(X)$  pour  $X$  une image d'entrée. Le paragraphe suivant explique alors comment fonctionne le processus de *feedback* ou *backward* pour ajuster les paramètres  $\theta$  du réseau.

### 6.1.6 Boucle Backward

L'optimisation du vecteur de paramètres est obtenu par descente de gradient stochastique donné ci-dessous.

---

**Algorithm 1** Stochastic Gradient Descent : find  $\theta^*$  minimizing  $\mathcal{L}(\theta)$

---

**Require:**  $\gamma, \mu, \tau, \theta_0, \mathcal{L}$

```

 $t \leftarrow 0$ 
while  $\theta_t$  n'a pas convergé do
     $t \leftarrow t + 1$ 
     $\gamma_t \leftarrow \text{Scheduler}(\gamma_{t-1}, T_{\max}, \gamma_{\min})$ 
     $g_t \leftarrow \nabla_{\theta} \mathcal{L}(\theta_{t-1}) + \tau \theta_{t-1}$ 
    if  $t > 1$  then
         $b_t \leftarrow \mu b_{t-1}$ 
    else
         $b_t \leftarrow g_t$ 
    end if
     $g_t \leftarrow b_t$ 
     $\theta_t \leftarrow \theta_{t-1} - \gamma_t g_t$ 
end while
return  $\theta_t$ 

```

---

Pour la mise à jour du pas d'apprentissage, ce qui permet d'augmenter la vitesse de convergence, nous utilisons ici le scheduler CosineAnnealingLR.

## 6.2 Paramètres des simulations numériques

### 6.2.1 Base de données CIFAR-10

Nous avons appliqué le modèle à la base de données CIFAR-10, ce qui explique les valeurs du nombre de classes  $c = 10$  ou de la taille d'une entrée de  $32 \times 32 \times 3$  par exemple. Il s'agit d'une base de données simple d'images de résolution faibles, subdivisées en 10 classes : avions, voitures, chats, cerfs, chiens, grenouilles, bateaux et camions, oiseaux et chevaux.

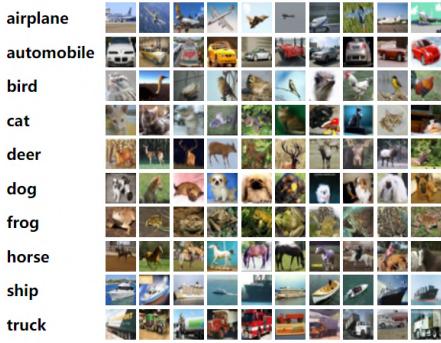


FIGURE 8 – Extrait de la base de données CIFAR-10 à partir de [5]

**Hyperparamètres.** Dans notre modèle, on a adopté les paramètres suivants

- **Pas d'apprentissage initial :**  $\gamma = 10^{-4}$
- **Moment :**  $\mu = 0.9$
- **Weight decay :**  $\tau = 10^{-4}$ .
- **Niveau de précision :**  $\varepsilon = 10^{-2}$
- $\gamma_{\min} = 10^{-3}$
- $T_{\max} = 1$

## 6.3 Simulations numériques

Nous avons réalisé plusieurs simulations numériques. Pour cela, nous avons téléchargé la base de données CIFAR-10 depuis le [site d'Alex Krizhevsky](#). Les simulations numériques ont pour objectif de

1. Vérifier l'apprentissage correct du modèle
2. Vérifier la réduction de  $L(A_\theta^R)$  par régularisation lipschitzienne
3. Mettre en évidence la sensibilité aux perturbations du modèle non régularisé
4. Mettre en évidence la robustesse aux perturbations adverses du modèle régularisé
5. Identifier quel poids de régularisation lipschitzienne  $\lambda$  permet d'optimiser le trade-off entre précision du modèle et constante de Lipschitz faible

### 6.3.1 Apprentissage correct du modèle

Nous avons tout d'abord réalisé des simulations sans régularisation ( $\lambda = 0$ ) et sans bruit sur les images de test, afin de s'assurer que le modèle construit fonctionne bien.

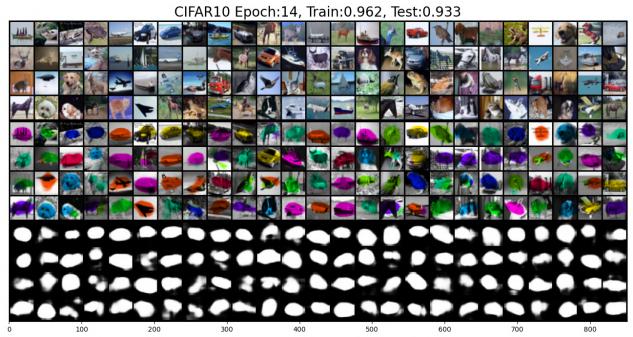


FIGURE 9 – Résultats de simulation du modèle non régularisé ( $\lambda = 0$ ) sans bruit sur 15 époques

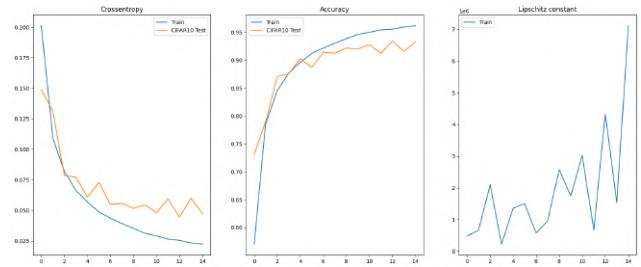


FIGURE 10 – Masques en sortie de simulation du modèle non régularisé ( $\lambda = 0$ ) sans bruit sur 15 époques

Les résultats montrent que le modèle fonctionne de manière satisfaisante, bien que les résultats soient plus faibles que les benchmark sur la classification de CIFAR-10. Celles-ci sont de l'ordre de 99,50%, comme obtenu par exemple dans [3] ou [9]. Pour notre étude de régularisation, la précision de 93,3% est jugée suffisante.

### 6.3.2 Quel poids de régularisation lipschitzienne $\lambda$ permet d'optimiser le trade-off entre précision du modèle et constante de Lipschitz $L(A_\theta^R)$ faible ?

Ensuite, nous avons testé si l'ajout de la régularisation lipschitzienne réduit effectivement la constante de Lipschitz. En même temps, nous avons réalisé les simulations pour plusieurs valeurs de poids de régularisation, en général sur 10 à 13 époques. Le code utilisé a été réalisé sur Google Colab et se trouve [ici](#). Contrairement à la partie 5, nous avons pas utilisé l'accès au DCE Metz, étant donné qu'il y a eu des problèmes dans la transcription du code de Google Colab vers un format adapté au DCE Metz. Le code, non abouti donc, se trouve sur [GitHub](#).

Nous avons résumé les résultats de simulations (qui se trouvent en annexe i.e. partie 9) sur le tableau 6.3.2. On constate notamment que l'augmentation du poids de régularisation entraîne bien une diminution de la constante de Lipschitz considérée.

Sur la figure 11 sont résumés les variations de la constante de Lipschitz avec le poids de régularisation  $\lambda$ . Le graphique nous a alors conduit à établir une loi empirique, donnée par la proposition 5 ci-dessous.

**Proposition 5.** Pour  $\lambda \in ]0, 1]$ , la constante de Lipschitz  $L(A_\theta^R)$  vérifie la loi empirique  $\log L(A_\theta^R) = \beta - m \log(\lambda)$

Poids de régularisation $\lambda$	$0$	$10^{-10}$	$10^{-9}$	$10^{-8}$	$10^{-7}$	$10^{-6}$	$10^{-5}$	$10^{-4}$	$10^{-3}$	$10^{-2}$
Accuracy	93, 3%	91, 3%	89, 4%	87, 4%	90, 5%	88, 0%	79, 8%	Pas de valeur stable	Pas de valeur stable	Pas de valeur stable
Constante de Lipschitz	$1, 24 \cdot 10^8$	$1, 12 \cdot 10^7$	$7, 24 \cdot 10^5$	$2, 72 \cdot 10^5$	$6, 76 \cdot 10^4$	$2, 06 \cdot 10^3$	$1, 80 \cdot 10^3$	$7, 73 \cdot 10^2$	$3, 43 \cdot 10^1$	$4, 84$

TABLE 1 – Performances du modèle ResNet-18 modifié sur la base de données CIFAR-10 sur 13 époques (pour  $\lambda \in [0, 10^{-5}]$ ) et 5 époques (pour  $\lambda \in [10^{-5}, 10^{-2}]$ ) en fonction du poids de régularisation lipschitzienne  $\lambda$  et performance de la réduction de la constante de Lipschitz du réseau. Pour  $\lambda = 10^{-5}$ , on a pris la valeur obtenue sur la figure 27.

ce qui équivaut à

$$L(A_\theta^R) = \frac{C}{\lambda^m}$$

avec  $m \approx 0.752$  et  $\beta \approx -1.47$  soit  $C := e^\beta \approx 0.230$ .

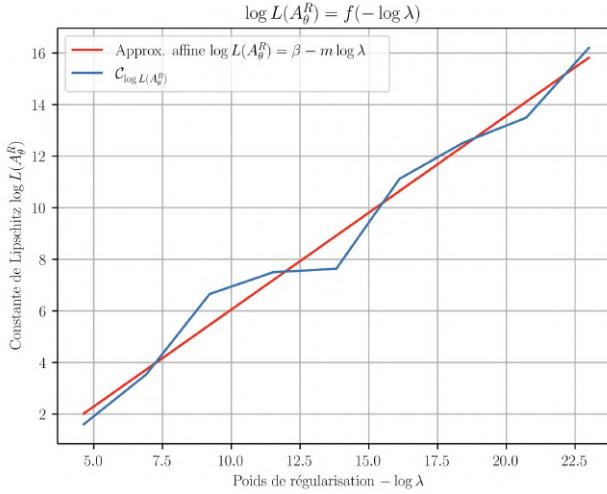


FIGURE 11 – Tracé du graphe approximatif de  $L(A_\theta^R)$  en fonction du poids de régularisation  $\lambda$  en échelle logarithmique. Le coefficient directeur de l’approximation affine est estimé à  $m = 0.752$ , l’ordonnée à l’origine à  $\beta = -1.47$ .

Cela étant, comme observé sur le tableau, nous avons pas pu réaliser des simulations complètes dès que  $\lambda > 10^{-5}$ , comme on observe des instabilités numériques. La partie suivante 6.3.3 est donc consacrée à ces instabilités, et en particulier ce que nous avons considéré comme méthode pour pouvoir évaluer la constante de Lipschitz pour des poids de régularisation plus élevés.

### 6.3.3 Instabilités numériques

De fait, on observe des instabilités numériques lorsque le poids de régularisation devient trop important. Numériquement, l’apprentissage n’a alors pas lieu : la précision reste très faible (<10%) et la constante de Lipschitz calculée explose (déclarée nan).

Cela est dû au fait que les *weights* (à ne pas confondre avec le poids de régularisation) du réseau de neurones sont initialement mal réglés donc il y a besoin de beaucoup de "travail" pour les ajuster, ce qui rentre en conflit avec la régularisation lipschitzienne qui pèse trop lourd dans la perte  $\mathcal{L}_{x,y}(\theta)$  à minimiser. On a donc d’abord entraîné le réseau pour des poids de régularisation plus faibles, ici  $\lambda_0 = 10^{-5}$ ,

pendant 5 époques, pour ensuite entraîner successivement le réseau à des poids plus élevés, donnés par  $\lambda_{n+1} = 10\lambda_n$ . Ainsi, on a obtenu les résultats suivants, allant jusqu’au calcul pour  $\lambda_4 = 10^{-2}$ .

Nous avons constaté le fait que la précision et la constante de Lipschitz diminuent en augmentant le poids de régularisation, ce qui est représenté sur la figure 14. Noter que  $\log L(A_\theta^R)^\nu$  y signifie qu’on a normalisé les constantes de Lipschitz pour représenter accuracy et constante de Lipschitz sur un même graphique de manière plus lisible. Concrètement, on a divisé par la valeur maximale, i.e. celle pour  $\lambda = 10^{-5}$  pour normaliser.

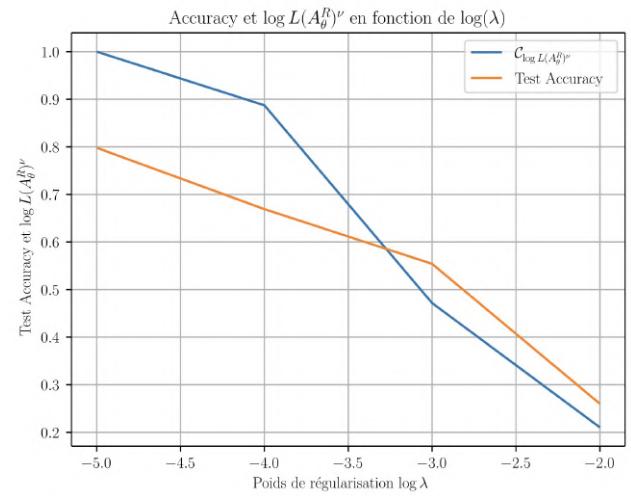


FIGURE 12 – Tracé du graphe approximatif de  $\log L(A_\theta^R)^\nu$  et de la précision de test en fonction du poids de régularisation  $\lambda$  en échelle logarithmique

En conclusion, nous avons donc bien réussi à réduire la constante de Lipschitz. En particulier la dernière simulation pour  $\lambda = 10^{-2}$  montre qu’on peut atteindre la non-expansivité ferme. Cela étant, les simulations ont aussi soulevé un défi majeur : lorsque le poids de régularisation devient trop important, on obtient des instabilités numériques, ce qui fait diverger le modèle.

**Dans la suite du projet, il faudra s’intéresser à des techniques de régularisation numériquement plus stables.** Bien que pour  $\lambda = 10^{-2}$  la constante de Lipschitz soit réduite, la précision du réseau n’est pas du tout satisfaisante : ce défi est donc à relever.

### 6.3.4 Robustesse aux perturbations adverses du modèle régularisé

Enfin, nous étudions l'apport de la régularisation lipschitzienne. Bien qu'en théorie, la réduction de la constante de Lipschitz définit le gain en robustesse, nous voulons le tester. Pour cela, on ajoute une perturbation adverse. Nous la modélisons par un bruit  $\epsilon \sim \mathcal{N}(0, \alpha^2 \text{Id})$  pour  $\alpha = 0.05$  dans les simulations ci-dessous. Les images de test sont alors transformées en leur ajoutant ce bruit  $\epsilon$ . Le critère testé est alors la précision par rapport à la simulations sans bruit gaussien.

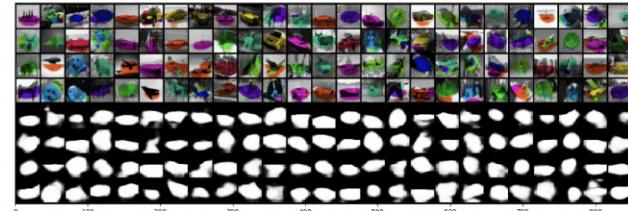
Les résultats de simulations sont résumées sur le tableau 6.3.4. Les masques obtenus dans chacun des cas bruitées sont montrées elles sur la figure 13.

Poids de régularisation $\lambda$	<b>0</b>	<b>0</b>	<b><math>10^{-6}</math></b>	<b><math>10^{-6}</math></b>
<b>Accuracy</b>	90,5%	93,3%	86,1%	88,0%
<b>Constante de Lipschitz</b>	$1,08 \cdot 10^7$	$1,24 \cdot 10^8$	$9,87 \cdot 10^3$	$2,06 \cdot 10^3$
<b>Bruit d'entrée</b>	$\epsilon$	0	$\epsilon$	0

TABLE 2 – Étude de robustesse du modèle auto-supervisé par ajout d'un bruit gaussien sur les images de test.



Pour un poids de régularisation nul avec bruit sur les images d'entrée en test



Pour un poids de régularisation à  $10^{-6}$  avec bruit sur les images d'entrée en test

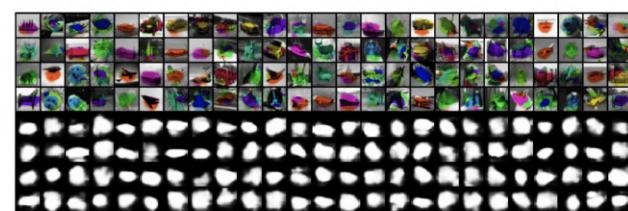


FIGURE 13 – Comparaison des masques en sortie de simulation sur 13 époques du modèle régularisé à  $\lambda = 10^{-6}$  et du modèle non-régularisé, avec un bruit d'entrée  $\epsilon$  sur les images de test

On constate alors que pour le modèle régularisé à  $\lambda = 10^{-6}$ , la précision baisse de 2.15%. Dans le modèle non régularisé, la baisse est de 3.22%, donc le modèle est moins robuste face à la perturbation. Toutefois, le modèle reste plus performant qu'avec la régularisation. Cependant, ce résultat est un indice pour que si on arrive à implémenter une régularisation numériquement stable et pondérée plus forte-

ment, alors le modèle sera plus robuste que sans régularisation. Ce point pourra être abordé dans la suite du projet.

## 7 Comparaison des deux approches

On constate que pour les deux modèles, l'ajoute de la régularisation lipschitzienne permet de diminuer la constante de Lipschitz du réseau et donc de rendre le réseau plus robuste.

Cependant, on voit bien que le modèle supervisé est plus stable numériquement et permet même pour un poids de régularisation de  $\lambda = 10^{-3}$  d'atteindre une précision de 73%. Cela est sûrement dû au fait que la base de données COCO est plus complexe, le modèle U-Net est plus profond et bien sûr que l'approche est une approche de segmentation supervisée, non de classification à laquelle on introduit une méthode de visualisation.

En conclusion, l'approche auto-supervisée est très intéressante et permet la régularisation du réseau, mais révèle des instabilités numériques que nous n'avons pas rencontré dans le modèle supervisé, sans doute plus classique.

## 8 Conclusion

### 8.1 Conclusion

En résumé, notre étude explore l'utilisation croissante des réseaux de neurones dans la segmentation d'images, en mettant en évidence les défis liés à leur sensibilité aux perturbations adverses. Pour renforcer la robustesse des réseaux, nous avons développé deux approches innovantes. La première, basée sur un modèle auto-supervisé dérivé du ResNet-18, intègre des méthodes de visualisation et une régularisation lipschitzienne. La seconde, utilisant un modèle supervisé U-Net, a été évaluée sur la base de données COCO pour la segmentation d'images multi-classes. Les résultats indiquent que l'ajout de la régularisation lipschitzienne améliore la robustesse des deux modèles, bien que le modèle supervisé se révèle numériquement plus stable, atteignant une précision de 73% pour un poids de régularisation de  $\lambda = 10^{-3}$ . En conclusion, bien que l'approche auto-supervisée montre un potentiel prometteur pour la régularisation, notre étude souligne la stabilité numérique supérieure du modèle supervisé, en particulier dans des contextes complexes tels que la base de données COCO.

### 8.2 Perspectives

En ce qui concerne le modèle auto-supervisé, il faudra essayer de trouver une manière plus stable numériquement d'implémenter la régularisation lipschitzienne, car c'est ce qui nous a empêché de correctement contrôler la constante de Lipschitz pour des poids  $\geq 10^{-4}$  sans pré-entraîner à des poids plus faibles. Cette nouvelle approche devrait permettre d'atteindre des précisions proches de celles des poids plus faibles, c'est-à-dire  $\geq 88\%$ .

Pour le modèle supervisé, nous pouvons envisager de

faire la segmentation en multi-classes de la base de données COCO en utilisant la même architecture du U-Net tout en changeant les entrées, sorties et paramètres du réseau. Cela pourrait amener à une augmentation de la précision de la segmentation en évitant les 'overlapping' des segments d'images. Ceci est un exemple de segmentation en multi-classes sans régularisation qu'on a obtenu :

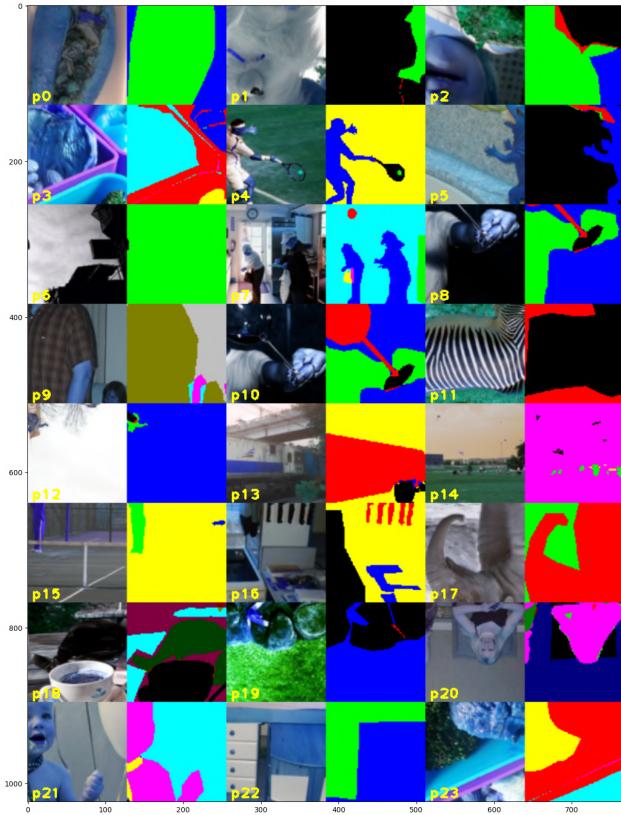


FIGURE 14 – Résultat d

## Références

- [1] Taibou BIRGUI-SEKOU et al. « Réseaux de neurones à convolution pour la segmentation de vaisseaux sanguins rétiniens – Des patchs aux images de taille réelle ». In : *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP)*. Marne-la-Vallée, France, juin 2018. URL : <https://hal.science/hal-01895868>.
- [2] Javiera CASTILLO-NAVARRO et al. « Réseaux de neurones semi-supervisés pour la segmentation sémantique en télédétection ». In : *Colloque GRETSI sur le Traitement du Signal et des Images*. Lille, France, 2019. URL : <https://hal.science/hal-02343961>.
- [3] Alexey DOSOVITSKIY et al. *An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale*. 2021. arXiv : [2010.11929 \[cs.CV\]](2010.11929).
- [4] Fereshteh KHODADADI SHOUSHTARI, Sedigheh SINA et Azimeh N.V. DEHKORDI. « Automatic segmentation of glioblastoma multiform brain tumor in MRI images : Using Deeplabv3+ with pre-trained

Resnet18 weights ». In : *Physica Medica* 100 (2022), p. 51-63. ISSN : 1120-1797. DOI : <https://doi.org/10.1016/j.ejmp.2022.06.007>. URL : <https://www.sciencedirect.com/science/article/pii/S1120179722020002>.

- [5] Alex KRIZHEVSKY. *The CIFAR-10 dataset*. <https://www.cs.toronto.edu/~kriz/cifar.html>. [Online ; accessed 18-April-2023]. 2009.
- [6] Chang-Mei LIANG et al. « Segmentation and weight prediction of grape ear based on SFNet-ResNet18 ». In : *Systems Science and Control Engineering* 10 (déc. 2022), p. 722-732. DOI : <10.1080/21642583.2022.2110541>.
- [7] Ozan OKTAY et al. *Attention U-Net : Learning Where to Look for the Pancreas*. 2018. arXiv : [1804.03999 \[cs.CV\]](1804.03999).
- [8] Philipp Fischer OLAF RONNEBERGER, Thomas Brox Computer Science DEPARTMENT et Germany BIOSS CENTRE FOR BIOLOGICAL SIGNALLING STUDIES University of Freiburg. « U-Net : Convolutional Networks for Biomedical Image Segmentation ». In : 10 (nov. 2015), p. 2. URL : [https://link.springer.com/chapter/10.1007/978-3-319-24574-4\\_28](https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28).
- [9] Maxime OQUAB et al. *DINOv2 : Learning Robust Visual Features without Supervision*. 2023. arXiv : [2304.07193 \[cs.CV\]](2304.07193).
- [10] Jean-Christophe PESQUET. *Université Paris-Saclay – M2 Optimisation Convex analysis and optimization Part II.2 : Nonexpansive operators*.
- [11] Christian SZEGEDY et al. *Intriguing properties of neural networks*. 2014. arXiv : [1312.6199 \[cs.CV\]](1312.6199).
- [12] Matthieu TERRIS et al. « Building Firmly Nonexpansive Convolutional Neural Networks ». In : *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, p. 8658-8662. DOI : <10.1109/ICASSP40776.2020.9054731>.

## 9 Annexe

### 9.1 Simulations sur le modèle auto-supervisé

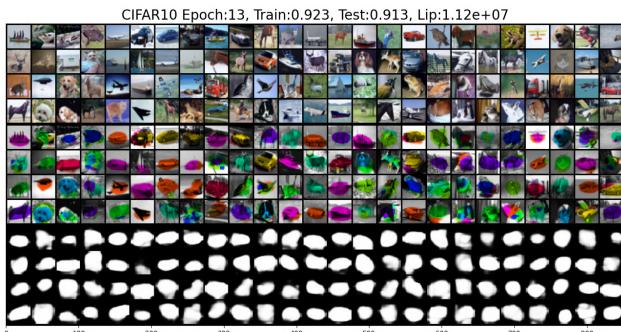


FIGURE 15 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-10}$  sans bruit sur 13 époques

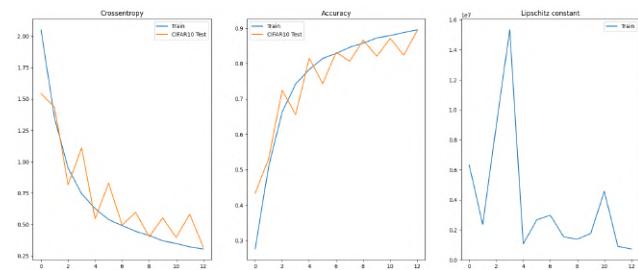


FIGURE 18 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-9}$  sans bruit sur 13 époques

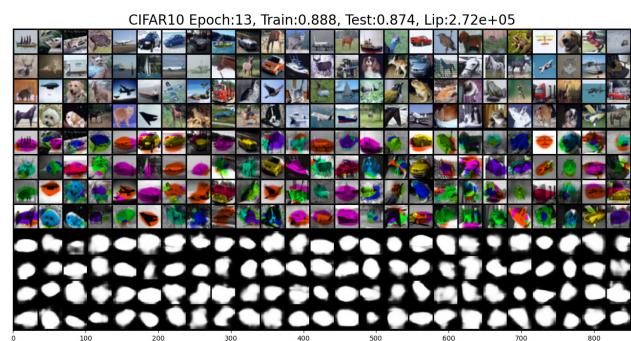


FIGURE 19 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-8}$  sans bruit sur 13 époques

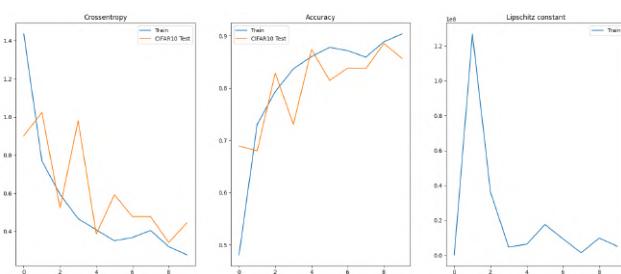


FIGURE 16 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-10}$  sans bruit sur 10 époques

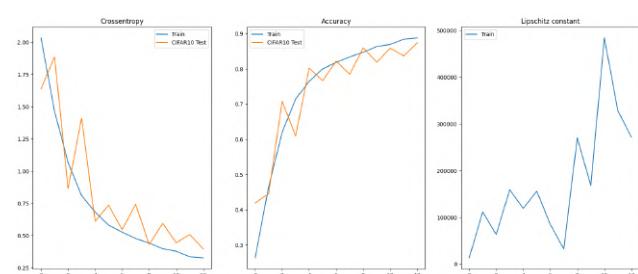


FIGURE 20 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-8}$  sans bruit sur 13 époques

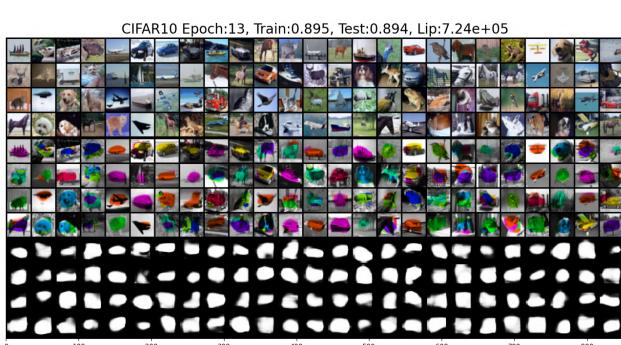


FIGURE 17 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-9}$  sans bruit sur 13 époques



FIGURE 21 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-7}$  sans bruit sur 13 époques

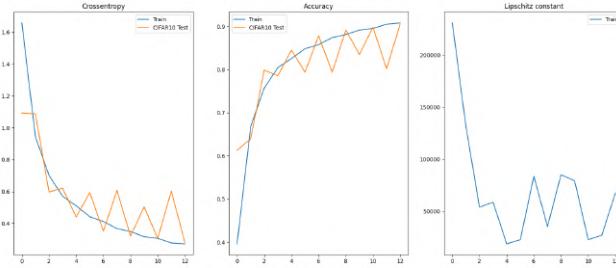


FIGURE 22 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-7}$  sans bruit sur 13 époques

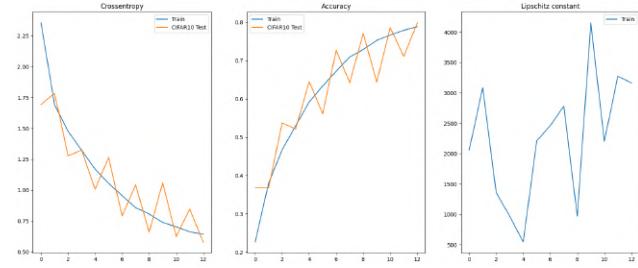


FIGURE 26 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-5}$  sans bruit sur 13 époques

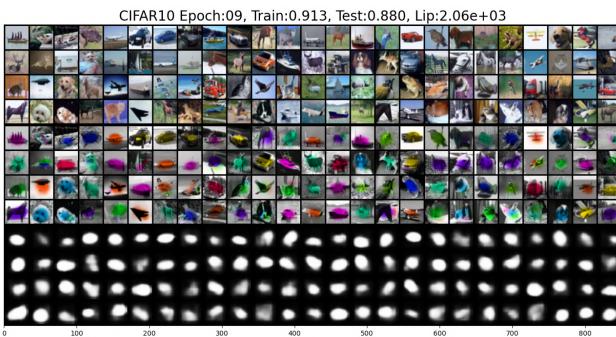


FIGURE 23 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-6}$  sans bruit sur 10 époques

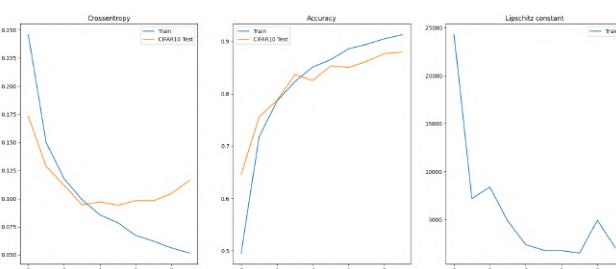


FIGURE 24 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-6}$  sans bruit sur 10 époques

## 9.2 Simulations sur instabilités numériques du modèle auto-supervisé

### 9.2.1 Masques



FIGURE 27 – Masques en sortie de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-5}$  sans bruit sur 5 époques

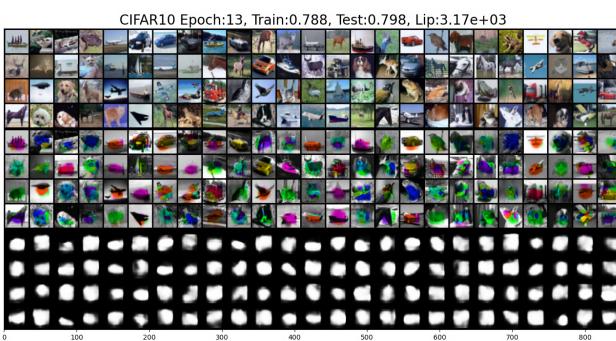


FIGURE 25 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-5}$  sans bruit sur 13 époques



FIGURE 28 – Masques en sortie de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-4}$  sans bruit sur 5 époques

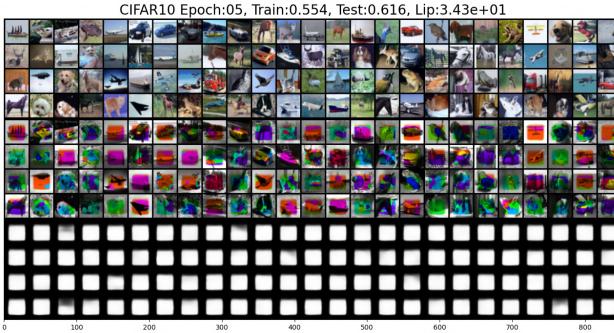


FIGURE 29 – Masques en sortie de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-3}$  sans bruit sur 5 époques



FIGURE 30 – Masques en sortie de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-2}$  sans bruit sur 3 époques

### 9.2.2 Graphiques

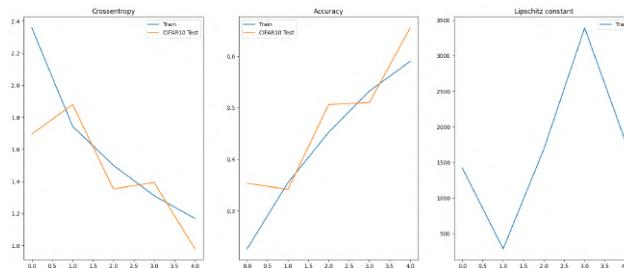


FIGURE 31 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-5}$  sans bruit sur 5 époques

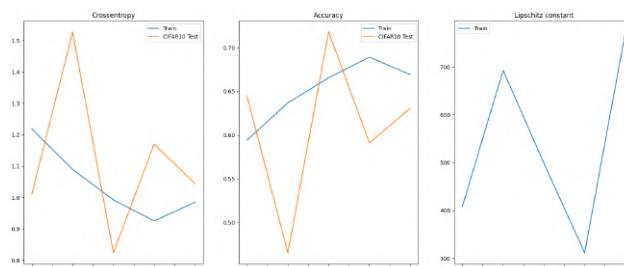


FIGURE 32 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-4}$  sans bruit sur 5 époques

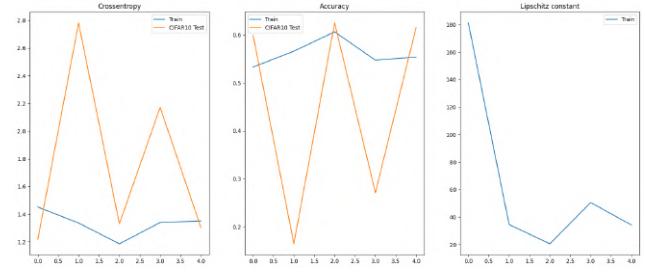


FIGURE 33 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-3}$  sans bruit sur 5 époques

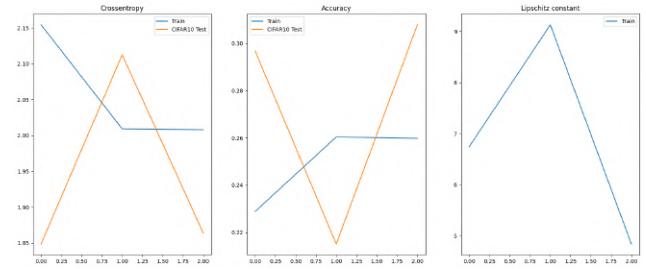


FIGURE 34 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-2}$  sans bruit sur 3 époques

## 9.3 Simulations sur la robustesse du modèle auto-supervisé

### 9.3.1 Masques

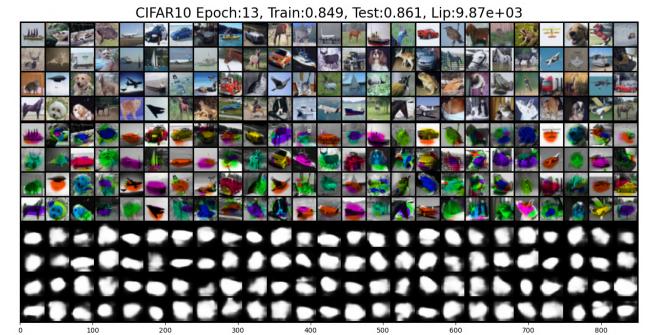


FIGURE 35 – Masques en sortie de simulation du modèle régularisé à  $\lambda = 10^{-6}$  avec bruit  $\epsilon$  sur 13 époques

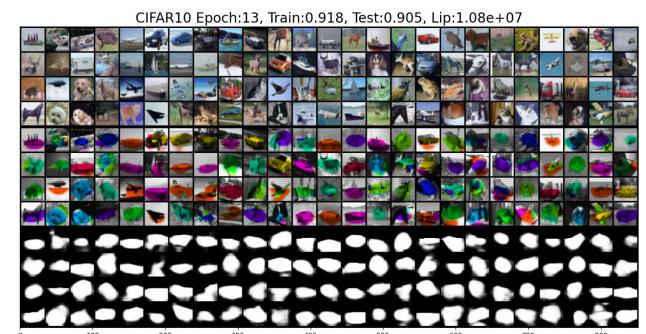


FIGURE 36 – Masques en sortie de simulation du modèle non-régularisé avec bruit  $\epsilon$  sur 13 époques

### 9.3.2 Graphiques

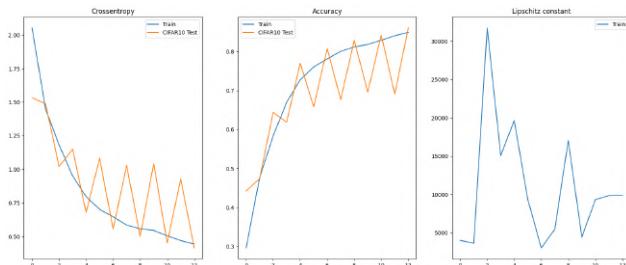


FIGURE 37 – Résultats de simulation du modèle régularisé à  $\lambda = 10^{-6}$  avec bruit  $\epsilon$  sur 13 époques

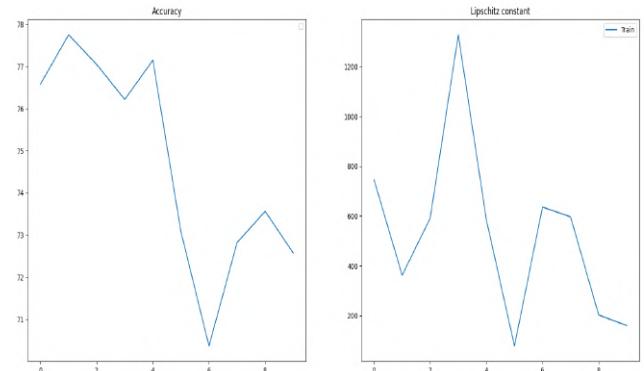


FIGURE 42 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-5}$

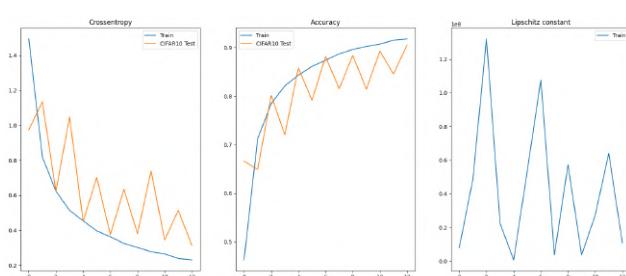


FIGURE 38 – Résultats de simulation du modèle non-régularisé avec bruit  $\epsilon$  sur 13 époques

### 9.4 Simulations pour le modèle supervisé



FIGURE 39 – Masques binaires réels d'un lot de 8 images



FIGURE 43 – Masques binaires prédits pour un lot de 8 images avec un poids de régularisation  $\lambda = 10^{-5}$

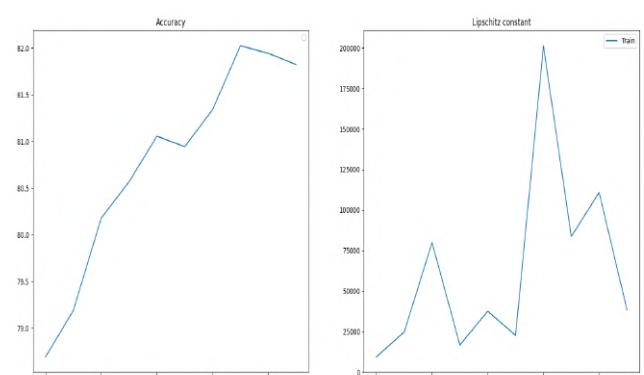


FIGURE 44 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-8}$



FIGURE 45 – Masques binaires prédits pour un lot de 8 images avec un poids de régularisation  $\lambda = 10^{-8}$

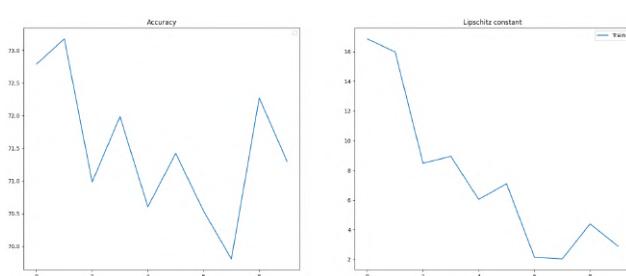


FIGURE 40 – Résultats de simulation du modèle pré-entraîné régularisé à  $\lambda = 10^{-3}$



FIGURE 41 – Masques binaires prédits pour un lot de 8 images avec un poids de régularisation  $\lambda = 10^{-3}$