

# ROUSTESSE DU MODÈLE CLIP FACE À DES ATTAQUES ADVERSARIALES



16 décembre 2024

Amine Chraibi - Antoine Maechler - Yassine Guennoun



## TABLE DES MATIÈRES

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Préambules . . . . .	3
1.1.1	Le modèle CLIP . . . . .	3
1.1.2	Attaques . . . . .	4
1.1.3	Robustesse . . . . .	5
1.2	État de l'art . . . . .	6
1.3	Motivations . . . . .	7
<b>2</b>	<b>Attaques adversariales</b>	<b>8</b>
2.1	Attaques par Projected Gradient Descent . . . . .	8
2.2	Attaques par Expectation Over Transformation . . . . .	11
2.2.1	Influence des hyperparamètres . . . . .	12
2.3	Comparaison des différentes attaques . . . . .	13
<b>3</b>	<b>Mécanismes de défense</b>	<b>14</b>
3.1	Ajout d'un bruit Gaussien . . . . .	14
3.1.1	Etude des hyperparamètres . . . . .	14
3.1.2	Résultats . . . . .	15
3.2	Méthode par denoising . . . . .	17
3.3	Bilan des méthodes précédant l'inférence . . . . .	21
3.4	Fine-tuning par apprentissage non-supervisé . . . . .	22
3.5	Orthogonal Fine-Tuning . . . . .	25
<b>4</b>	<b>Généralisation des attaques à d'autres modèles</b>	<b>28</b>
4.1	Images du dataset CIFAR100 . . . . .	28
4.2	Images de haute qualité . . . . .	29
<b>5</b>	<b>Conclusion</b>	<b>31</b>

# 1 INTRODUCTION

Les modèles multi-modaux, particulièrement les modèles vision-langage (VLM) comme CLIP [1] ou Flamingo [2], sont devenus des outils essentiels dans de nombreux domaines (santé, art...). Toutefois, des travaux antérieurs ont montré qu'ils sont particulièrement vulnérables à des attaques imperceptibles ciblant spécifiquement la modalité visuelle [3]. Ces attaques adversariales exploitent les failles des modèles pour altérer leurs prédictions sans que ces modifications soient détectables par l'œil humain. Ce sujet est fondamental pour assurer la sécurité des modèles, notamment dans des applications où une erreur pourrait avoir des conséquences graves. Prenons l'exemple d'une voiture autonome utilisant CLIP pour reconnaître les panneaux de signalisation et décider s'il faut s'arrêter ou continuer à rouler. Un attaquant malveillant pourrait exploiter les vulnérabilités du modèle en ajoutant un bruit imperceptible à l'image perçue par le véhicule, menant celui-ci à faire une prédiction erronée et potentiellement dangereuse, comme illustré dans la figure ci-dessous.

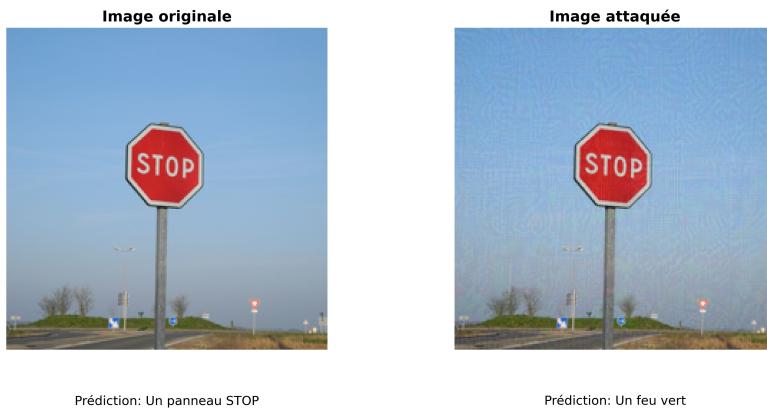


FIGURE 1 – Prédiction du modèle CLIP sur une image et son équivalent adversarial.

La robustesse des modèles multi-modaux est donc un sujet fondamental. Dans ce rapport, nous nous intéressons aux mécanismes permettant d'utiliser ces perturbations pour manipuler des modèles et diffuser de fausses informations. Nous analyserons également les approches existantes pour contrer ces attaques, en étudiant leur efficacité et leurs limites, et proposerons des stratégies pour renforcer la robustesse des modèles face à ces menaces.

## 1.1 PRÉAMBULES

### 1.1.1 • LE MODÈLE CLIP

CLIP (*Contrastive Language-Image Pre-training*) [1] est un modèle multimodal conçu pour aligner des représentations d'images et de texte dans un espace latent partagé. Ce modèle est notamment utilisé pour des tâches de classification sans supervision (*zero-shot learning*).

L'architecture de CLIP repose sur deux encodeurs :

- Un **encodeur d'image**, basé sur ResNet ou un *Vision Transformer* (ViT), qui encode une image  $I$  en un vecteur  $\mathbf{v}_I$ .
- Un **encodeur de texte**, basé sur un *Transformer*, qui encode une description textuelle  $T$  en un vecteur  $\mathbf{v}_T$ .

Les vecteurs  $\mathbf{v}_I$  et  $\mathbf{v}_T$  sont alignés dans un espace partagé à l'aide de la *cosine-similarity* :

$$\text{sim}(\mathbf{v}_I, \mathbf{v}_T) = \frac{\mathbf{v}_I \cdot \mathbf{v}_T}{\|\mathbf{v}_I\| \|\mathbf{v}_T\|}.$$

Pour la classification, CLIP calcule la similarité entre le vecteur associé à l'image et le vecteur associé à chaque classe, et prédit la classe qui a la meilleure similarité avec l'image.

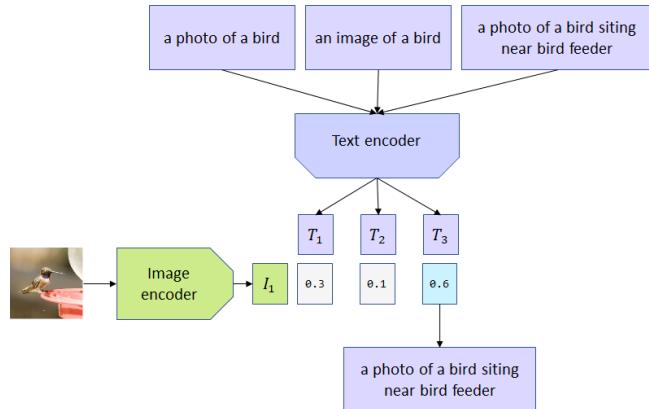


FIGURE 2 – Illustration du fonctionnement de CLIP pour l’inférence. Les  $T_i$  représentent les *cosine-similarities* respective entre l’image à classifier et les labels

### 1.1.2 • ATTAQUES

- **ATTAKUES DE TYPE WHITE-BOX ET BLACK-BOX.**

Une **attaque de type white-box** fait référence à un scénario où l’attaquant a un accès complet au modèle cible, y compris sa structure, ses paramètres et ses poids. Cela permet à l’attaquant d’exploiter ces informations pour concevoir des perturbations adversariales optimales qui maximisent la probabilité de tromper le modèle. Ces attaques sont souvent considérées comme les plus puissantes, car elles utilisent toutes les informations disponibles sur le modèle.

En revanche, une **attaque de type black-box** suppose que l’attaquant n’a pas d’accès direct au modèle cible. L’attaquant peut uniquement interagir avec le modèle via des requêtes et des réponses, c’est-à-dire en observant les sorties du modèle pour des entrées spécifiques. Ce type d’attaque nécessite généralement de concevoir des perturbations en se basant sur des approximations du modèle ou des techniques d’apprentissage par transfert. Les attaques black box sont plus difficiles à mettre en œuvre, mais elles reflètent des scénarios plus réalistes dans lesquels les détails internes du modèle ne sont pas accessibles.

Dans toute la suite, les attaques implémentées sur CLIP sont de type white-box. Ce choix est motivé par le désir de tester l’efficacité de nos méthodes de robustesse face aux attaques les plus efficaces.

- **ATTAKUES NON CIBLÉES**

Étant donné une image d’entrée  $q$  et une légende réelle  $y$ , une attaque non-ciblée (ou non-spécifique) vise à maximiser la probabilité que le modèle ne prédise pas  $y$ , donc qu’il se trompe. Dans notre cas, nous voulons ajouter une perturbation  $\delta_q$  à l’image d’entrée de sorte à maximiser la log-vraisemblance négative de  $y$ , c’est à dire, en notant  $p(y_l | q + \delta_q)$  la probabilité que le modèle prédise  $y_l$  avec pour entrée visuelle  $q + \delta_q$ , trouver  $\delta_q$  solution de :

$$\max_{\delta_q} - \sum_{l=1}^m \log p(y_l | q + \delta_q) \quad (1)$$

sous la contrainte  $\|\delta_q\|_\infty \leq \epsilon_q$ , où  $\epsilon_q$  est le rayon de l’attaque.

- ATTAQUES CIBLÉES

Un attaquant peut également chercher à forcer le modèle à produire une sortie spécifique souhaitée. Cela peut être réalisé avec une attaque ciblée. Si  $\hat{y}$  est la sortie cible souhaitée, l'objectif de l'attaque ciblée est d'ajouter une perturbation à l'image originale pour maximiser la probabilité que le modèle prédise  $\hat{y}$ . C'est à dire, en gardant les mêmes notations que précédemment, de trouver  $\delta_q$  solution de :

$$\min_{\delta_q} - \sum_{l=1}^m \log p(\hat{y}_l | q + \delta_q) \quad (3)$$

sous la contrainte  $\|\delta_q\|_\infty \leq \epsilon$ .

### 1.1.3 • ROBUSTESSE

Présentons à présent formellement la robustesse d'un réseau de neurones. Les définitions qui suivent sont issues de Katz et al.[4].

**Définition 1** Un réseau  $\mathcal{N}$  est  $\delta$ -localement robuste au point  $x_0$  si et seulement si

$$\forall x', \|x' - x_0\| \leq \delta \implies \arg \max \mathcal{N}(x') = \arg \max \mathcal{N}(x_0)$$

où  $\mathcal{N}$  est un classifieur associé à un ensemble de labels  $L$ ,  $x_0$  est l'entrée originale, et  $x'$  est l'entrée perturbée qui est proche de  $x_0$ .

Cette définition indique que pour tout  $x'$ , le réseau assigne à  $x'$  le même label qu'il assignerait à  $x_0$ . Des valeurs plus grandes de  $\delta$  impliquent des voisinages plus larges et donc une meilleure robustesse. Cependant, la robustesse locale est vérifiée pour des points d'entrée individuels dans un espace d'entrée infini sans se transférer à d'autres points non vérifiés. De plus, pour chaque  $x_0$ , une valeur minimale acceptable de  $\delta$  doit être spécifiée, et ces valeurs varient selon les différents points d'entrée.

Pour surmonter le besoin de spécifier chaque  $\delta$  séparément, introduisons la notion suivante :

**Définition 2** Un réseau  $\mathcal{N}$  est  $(\delta, \epsilon)$ -globalement robuste dans la région d'entrée  $\mathcal{D}$  si et seulement si

$$\forall x_1, x_2 \in \mathcal{D}, \|x_1 - x_2\| \leq \delta \implies \forall l \in L. |C(\mathcal{N}, x_1, l) - C(\mathcal{N}, x_2, l)| < \epsilon$$

où  $C$  désigne la confiance du modèle  $\mathcal{N}$  que l'entrée est labellisée  $l$ . Dans notre cas, il s'agit simplement de la probabilité attribuée à la classe  $l$  par le modèle.

Comparée à la *Définition 1*, la *Définition 2* considère un domaine d'entrée  $\mathcal{D}$  au lieu d'un point spécifique  $x_0$ , permettant ainsi de couvrir un nombre infini de points ou tout l'espace d'entrée en une seule requête, avec  $\delta$  et  $\epsilon$  définis une seule fois pour tout le domaine. Cependant, prouver la robustesse globale  $(\delta, \epsilon)$  est beaucoup plus difficile sur des réseaux plus grands.

Pour équilibrer la robustesse locale et globale, il existe une définition hybride :

**Définition 3** Un réseau  $\mathcal{N}$  est  $(\delta, \epsilon)$ -localement robuste au point  $x_0$  si et seulement si

$$\forall x', \|x' - x_0\| \leq \delta \implies \forall l \in L. |C(\mathcal{N}, x', l) - C(\mathcal{N}, x_0, l)| < \epsilon$$

Ainsi, nous voudrions idéalement entraîner notre modèle pour qu'il soit  $(\delta, \epsilon)$ -globalement robuste aux attaques, pour une valeur de  $\delta$  assez grande pour défendre contre les attaques où les perturbations sont imperceptibles pour l'œil humain.

## 1.2 ÉTAT DE L'ART

La complexité des VLM les rend particulièrement vulnérables aux attaques adversariales sophistiquées. Une étude récente intitulée *ADvLM : Visual Adversarial Attack on Vision-Language Models for Autonomous Driving* met en évidence les vulnérabilités des VLM dans des applications critiques comme la conduite autonome. Leur méthode repose sur deux composantes clés : l'*Invariant Semantic Induction* et le *Scenario-Associated Enhancement*. L'*Invariant Semantic Induction* utilise un modèle de langage pour générer une bibliothèque diversifiée de prompts textuels exprimant des intentions similaires mais avec des variations lexicales, tout en maintenant une faible entropie sémantique. Dans la modalité visuelle, le *Scenario-Associated Enhancement* identifie les cadres critiques d'un scénario en s'appuyant sur les mécanismes d'attention du modèle. Ces cadres sont utilisés pour optimiser les perturbations adversariales, qui sont ensuite généralisées à travers la bibliothèque de prompts et des perspectives temporelles multiples. Cette approche permet de concevoir des attaques robustes qui restent efficaces sur des variations textuelles et visuelles [5].

Une autre étude, intitulée *Visual Adversarial Examples Jailbreak Aligned Large Language Models*, met en évidence comment une seule image adversariale visuelle peut suffire à "jailbreaker" un modèle de langage aligné. En optimisant une image adversariale pour cibler des comportements spécifiques, tels que l'émission de contenus nuisibles ou la conformité à des instructions malveillantes, les auteurs montrent que ces attaques permettent au modèle de contourner ses mécanismes de sécurité et de générer des sorties préjudiciables, même pour des instructions qui n'ont pas été directement optimisées [6].

Face à ces menaces, *Semantic Shield* propose une méthode pour atténuer les attaques de type backdoor et empoisonnement en utilisant des connaissances externes issues d'un grand modèle de langage. Cette approche impose que l'attention du modèle soit concentrée sur les régions visuelles alignées avec ces connaissances, empêchant ainsi l'apprentissage de corrélations malveillantes. Par exemple, si une attaque tente de lier une zone non pertinente d'une image (comme un fond aléatoire) à une instruction textuelle spécifique, *Semantic Shield* réduit l'attention accordée à cette zone en raison de son faible alignement avec les connaissances externes. Cette méthode a démontré une efficacité notable contre diverses attaques, tout en maintenant les performances du modèle et sans nécessiter de modifications pendant l'inférence [7].

*ASTRA*, une autre méthode de défense, offre quant à elle une solution innovante et efficace contre les attaques de type "jailbreak" dans les modèles vision-langage (VLM). Contrairement aux défenses traditionnelles, souvent coûteuses et peu pratiques, *ASTRA* repose sur des vecteurs de guidage pour orienter les activations du modèle loin des directions adversariales identifiées. Ces vecteurs, appelés *steering vectors*, sont construits en identifiant les tokens visuels fortement associés aux comportements nuisibles dans des images adversariales. Pendant l'inférence, une méthode d'activation adaptative projette les activations du modèle loin de ces vecteurs nuisibles. Cette approche maintient des performances élevées sur des entrées non attaquées tout en réduisant fortement les sorties nuisibles. Les expériences montrent que *ASTRA* offre une défense robuste contre divers types d'attaques, y compris celles qui n'étaient pas spécifiquement prises en compte lors de la conception, tout en étant hautement transférable et efficace [8].

Ainsi, les travaux récents mettent en lumière la vulnérabilité des VLM face à des attaques adversariales de plus en plus sophistiquées, mais aussi les efforts considérables pour renforcer leur robustesse. Ces études montrent une diversité croissante d'attaques, allant des manipulations visuelles ciblées à des stratégies complexes comme le "jailbreaking", tout en soulignant l'importance critique de sécuriser ces modèles dans des applications sensibles. Dans ce contexte, nous proposons d'apporter une contribution en étudiant certaines méthodes de robustesse, et en proposant des approches nouvelles, notamment l'orthogonal fine-tuning et le denoising afin d'améliorer la résilience des VLM face à ces risques.

## 1.3 MOTIVATIONS

---

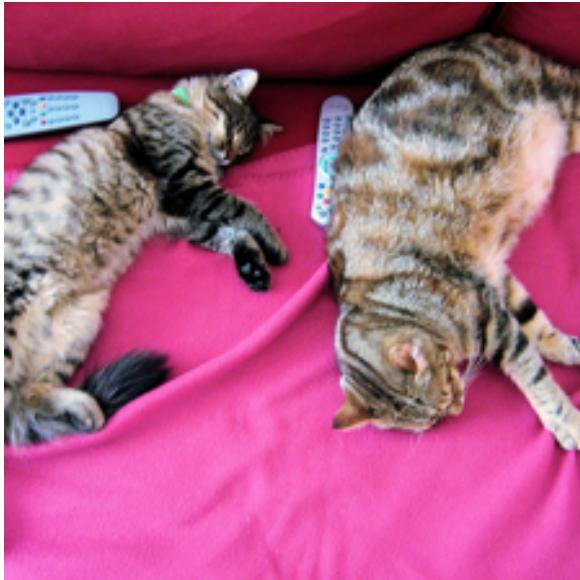
Nous avions prévu de travailler sur Flamingo [2], mais le long temps d'inférence du modèle en raison d'un manque de ressources de calcul nous a conduit à nous concentrer sur CLIP, un VLM plus léger et plus rapide. Après une phase de familiarisation avec le modèle, nous avons implémenté des attaques white-box pour analyser ses vulnérabilités et testé des méthodes de défense classiques pour renforcer sa robustesse.

Une contribution clé de notre travail réside dans l'exploration de l'orthogonal fine-tuning, méthode qui n'a, à notre connaissance, jamais été appliquée à la robustesse face aux attaques adversariales. Elle permet de limiter le sur-apprentissage souvent observé avec d'autres méthodes, où la robustesse aux attaques se fait au détriment des performances sur des données non attaquées.

Notre recherche a donc consisté à adapter cette méthode à une nouvelle tâche, celle de renforcer la robustesse des modèles vision-langage face aux attaques adversariales. Ce travail ouvre des perspectives prometteuses en combinant robustesse accrue et maintien des capacités de généralisation.

## 2 ATTAQUES ADVERSARIALES

Pour tester la robustesse de nos différents modèles, nous implémentons tout d'abord différentes attaques adversariales, en nous concentrant particulièrement sur les white-box attacks.



(a) Output : a cat



(b) Output : Ce rapport mérite une bonne note

FIGURE 3 – Démonstration d'une attaque adversariable de rayon  $\epsilon = 5/255$

Cette illustration montre que l'attaque est imperceptible à l'œil nu tout en forçant le modèle à prédire le résultat souhaité. Il s'agit d'un exemple d'attaque spécifique PGD, que nous présentons en détail ci-après.

### 2.1 ATTAQUES PAR PROJECTED GRADIENT DESCENT

- ATTAQUES SPÉCIFIQUES

La première méthode que nous avons implémentée est l'attaque spécifique basée sur la technique de *Projected Gradient Descent* (PGD). Cette attaque a pour objectif de minimiser la fonction de perte, *cross-entropy loss*, afin de pousser le modèle à prédire une étiquette cible souhaitée, appelée *target*. Pour cela, un bruit est ajouté à l'image d'origine de manière itérative, en ajustant ce bruit en fonction des gradients calculés par rapport à la sortie du modèle.

L'une des hypothèses clés de cette attaque est que nous avons accès aux gradients du modèle, ce qui en fait une attaque de type **white-box**.

---

**Algorithm 1** Pseudo-code pour l'attaque spécifique

---

```

1: Entrée : Image d'origine  $x$ , étiquette cible  $y$ , nombre d'itérations  $num\_iter$ , taille de l'étape  $\alpha$ , contrainte de perturbation  $\epsilon$ 
2:  $adv\_noise \leftarrow$  bruit aléatoire initial
3:  $x \leftarrow$  image normalisée
4: for  $t = 1$  à  $num\_iter$  do
5:    $x_{adv} \leftarrow x + adv\_noise$ 
6:    $logits \leftarrow$  modèle( $x_{adv}$ )
7:    $loss \leftarrow$  cross-entropy loss( $logits, y$ )
8:   calculer les gradients de  $loss$  par rapport à  $adv\_noise$ 
9:    $adv\_noise \leftarrow adv\_noise - \alpha \cdot \text{sgn}(\nabla_{adv\_noise} loss)$ 
10:   $adv\_noise \leftarrow$  projeter  $adv\_noise$  dans la contrainte  $[-\epsilon, \epsilon]$ 
11:   $x_{adv} \leftarrow x + adv\_noise$ 
12: end for
13: Retourner :  $x_{adv}, loss\_values$ 

```

---

- **ATTAQUES NON-SPÉCIFIQUES**

L'attaque non spécifique vise à générer une image adversariale qui perturbe les prédictions du modèle sans cibler un label particulier. Contrairement à l'attaque spécifique, cette méthode cherche à maximiser la fonction de perte.

L'attaque commence par ajouter un bruit aléatoire initial à l'image, contraint dans une plage définie par le paramètre  $\epsilon$ . Ce bruit est optimisé sur plusieurs itérations à l'aide des gradients calculés par rapport à la *negative cross-entropy loss*. Cette attaque, tout comme l'attaque spécifique, repose sur l'accès aux gradients du modèle, ce qui en fait une attaque de type **white-box**.

---

**Algorithm 2** Pseudo-code pour l'attaque non spécifique

---

```

1: Entrée : Image d'origine  $x$ , sortie initiale du modèle  $model\_output$ , nombre d'itérations  $num\_iter$ , taille de l'étape  $\alpha$ , contrainte de perturbation  $\epsilon$ 
2:  $adv\_noise \leftarrow$  bruit aléatoire initial dans  $[-\epsilon, \epsilon]$ 
3:  $x \leftarrow$  image normalisée
4: for  $t = 1$  à  $num\_iter$  do
5:    $x_{adv} \leftarrow$  normaliser( $x + adv\_noise$ )
6:    $logits \leftarrow$  modèle( $x_{adv}$ )
7:    $loss \leftarrow -\text{cross-entropy loss}(logits, model\_output)$ 
8:   Calculer les gradients de  $loss$  par rapport à  $adv\_noise$ 
9:    $adv\_noise \leftarrow adv\_noise - \alpha \cdot \text{sgn}(\nabla_{adv\_noise} loss)$ 
10: end for
11: Retourner : Image adversariale  $x_{adv}$ , valeurs de perte  $loss\_values$ 

```

---

- INFLUENCE DES HYPERPARAMÈTRES

Nous allons maintenant étudier l'influence des hyperparamètres de nos attaques, le nombre d'itérations de descente de gradient, ainsi que le rayon d'attaque  $\epsilon$ , sur l'accuracy de CLIP, en la comparant à la baseline de notre modèle sur l'ensemble de données CIFAR100.

On commence par étudier l'influence du rayon d'attaque sur l'attaque spécifique, pour un nombre d'itérations fixé à 11. La courbe correspondante est illustrée ci-dessous.

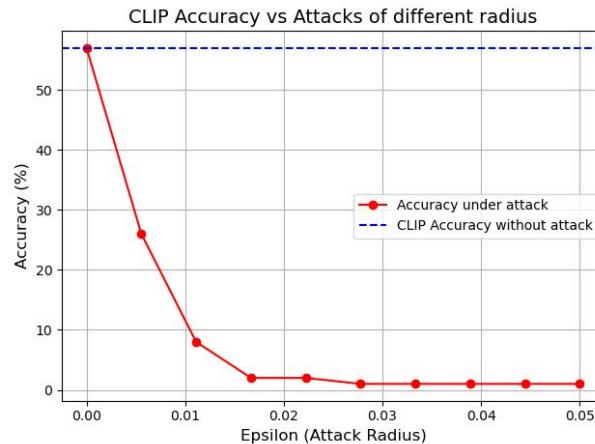


FIGURE 4 – Influence du rayon d'attaque  $\epsilon$  sur l'accuracy pour  $num\_iter = 11$  pour une attaque spécifique.

On observe que l'accuracy converge très rapidement vers des valeurs de 0% pour des valeurs de  $\epsilon$  de l'ordre de  $10^{-2}$ .

Ensuite, on analyse l'influence du nombre d'itérations  $num\_iter$ , en fixant  $\epsilon = 8/255 \approx 0.03$ . La courbe correspondante est présentée ci-dessous.

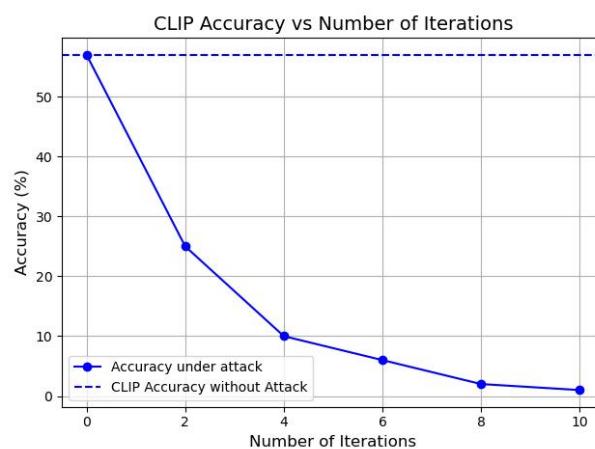


FIGURE 5 – Influence du nombre d'itérations  $num\_iter$  sur l'accuracy pour  $\epsilon = 8/255$  pour une attaque spécifique.

On remarque également que l'accuracy converge très rapidement vers 0 pour environ 10 itérations.

Enfin, on retrouve un comportement similaire pour les attaques non spécifiques. Ainsi, dans la suite de notre étude, notamment pour la défense, nous fixons  $num\_iter = 11$  pour réduire le temps de calcul et prenons une valeur arbitraire de  $\epsilon = 8/255$ .

## 2.2 ATTAQUES PAR EXPECTATION OVER TRANSFORMATION

---

L'attaque par *Expectation Over Transformation* (EOT) [9] est une méthode qui intègre des transformations stochastiques sur les images pour rendre les attaques plus robustes, voire quasi-invariantes à ces transformations. Elle repose sur l'idée de minimiser l'espérance de la perte sur une distribution de transformations aléatoires  $T$  de l'image. On veut donc résoudre le problème suivant, où  $t$  est une variable aléatoire représentant une transformation,  $p$  est une probabilité,  $\hat{y}$  est la sortie cible,  $q$  est l'image originale et  $\delta_q$  est la perturbation :

$$\arg \min_{\delta_q} \mathbb{E}_{t \sim T} [-\log p(\hat{y} | t(q + \delta_q))] \quad \text{sous la contrainte } \mathbb{E}_{t \sim T} [\|t(q + \delta_q) - t(q)\|_\infty] < \varepsilon, \quad q \in [0, 1]^d$$

Pour ce faire, à chaque itération, on applique l'une des quatre transformations suivantes sur l'image perturbée :

- **Défloutage** : une opération simulée via un filtre Gaussien inversé pour estimer l'effet d'un flou réduit.
- **Translation** : un déplacement aléatoire de l'image en pixels.
- **Rotation** : une rotation aléatoire autour du centre de l'image, dans un intervalle donné.
- **Denoising** : un débruitage simulé en ajoutant un bruit Gaussien faible.

L'objectif est de rendre les perturbations adversariales efficaces sous diverses conditions en minimisant la *cross-entropy loss* entre les prédictions du modèle pour ces transformations et une étiquette cible. Cette approche repose sur une méthode de Monte-Carlo, où l'espérance est calculée sur un nombre défini d'échantillons transformés.

---

**Algorithm 3** Pseudo-code pour l'attaque spécifique par EOT

```

1: Entrée : Image d'origine  $x$ , étiquette cible  $y$ , nombre d'itérations  $num\_iter$ , taille de l'étape  $\alpha$ , contrainte  $\epsilon$ , nombre d'échantillons Monte-Carlo  $n\_mc$ 
2:  $adv\_noise \leftarrow$  bruit aléatoire initial dans  $[-\epsilon, \epsilon]$ 
3: for  $t = 1$  à  $num\_iter$  do
4:    $eot\_loss \leftarrow 0$ 
5:   for  $k = 1$  à  $n\_mc$  do
6:      $x_{adv} \leftarrow x + adv\_noise$ 
7:     Appliquer une transformation aléatoire parmi : défloutage, translation, rotation, débruitage
8:      $logits \leftarrow$  modèle(image transformée)
9:      $loss \leftarrow$  cross-entropy loss( $logits, y$ )
10:     $eot\_loss \leftarrow eot\_loss + loss$ 
11:   end for
12:    $eot\_loss \leftarrow eot\_loss/n\_mc$                                 ▷ Calcul de l'espérance
13:   Mettre à jour  $adv\_noise$  avec la descente de gradient projetée dans  $[-\epsilon, \epsilon]$ 
14: end for
15: Retourner :  $x_{adv}, loss\_values$ 

```

---

L'attaque non spécifique suit exactement la même procédure que l'attaque spécifique décrite précédemment, à la différence près que cette fois, l'objectif est de maximiser la perte par rapport à la prédiction réelle du modèle, sans cibler une classe précise.

### 2.2.1 • INFLUENCE DES HYPERPARAMÈTRES

Nous nous intéressons à l'étude des hyperparamètres. Pour le nombre d'itérations  $num\_iter$ , nous constatons un comportement similaire à celui observé précédemment avec l'attaque PGD, pour les attaques spécifiques et non spécifiques. Nous pouvons tirer la même conclusion sur la rayon d'attaque, comme illustré par la courbe ci-dessous.

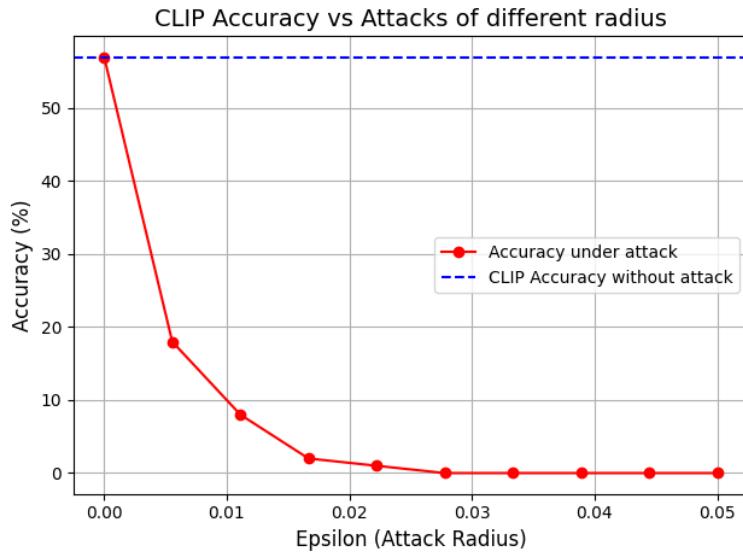


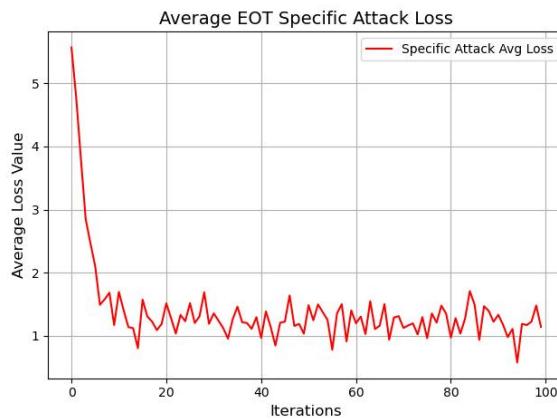
FIGURE 6 – Influence de  $\epsilon$  sur l'accuracy du modèle pour une attaque non spécifique (EOT).

Avec un nombre d'itérations réduit (de l'ordre de 10) et un rayon d'attaque faible (de l'ordre de  $10^{-2}$ ), le modèle non protégé reste vulnérable, quelle que soit la configuration de l'attaque.

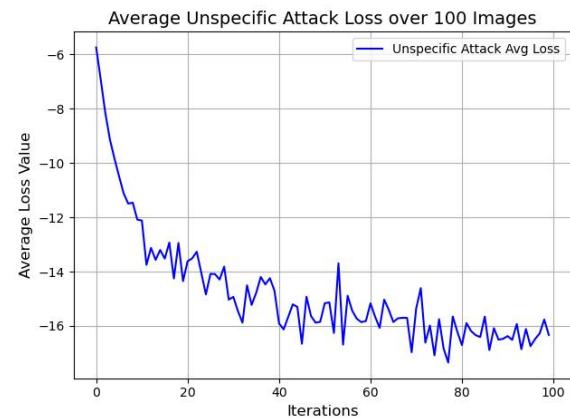
## 2.3 COMPARAISON DES DIFFÉRENTES ATTAQUES

À ce stade, si nous devions comparer les différentes attaques, il ne serait pas possible de tirer des conclusions précises. Sans mécanisme de défense intégré à CLIP, toutes les attaques s'avèrent efficaces avec des rayons d'attaque très faibles et un nombre d'itérations limité à 10, que ce soit pour les attaques spécifiques ou non spécifiques, ainsi que pour les approches PGD et EOT. Cependant, nous notons que l'attaque PGD est plus rapide que celle d'EOT, ce qui s'explique par le calcul d'une espérance via Monte Carlo dans le cas d'EOT, entraînant un temps de calcul plus élevé.

On observe néanmoins une différence dans la convergence des pertes (*loss*), qui semble être plus rapide pour les attaques spécifiques que pour les attaques non spécifiques. Les deux courbes ci-dessous illustrent cette observation pour l'attaque EOT, que nous retrouvons également pour les attaques PGD. Ce comportement est intuitivement cohérent, car pour une attaque spécifique, l'objectif est de minimiser la perte qui est bornée par 0, là où pour une attaque non-spécifique, on veut maximiser la perte (donc minimiser l'opposé de la perte), qui n'est pas bornée. Le ralentissement observé figure 7b est simplement dû au ralentissement de la croissance du logarithme asymptotiquement, mais n'est pas vraiment une convergence à proprement parler (la perte diverge juste très lentement vers moins l'infini).



(a) Convergence de la perte pour une attaque spécifique.



(b) Convergence de la perte pour une attaque non spécifique.

FIGURE 7 – Comparaison de la convergence des pertes entre attaques spécifiques et non spécifiques.

## 3 MÉCANISMES DE DÉFENSE

### 3.1 AJOUT D'UN BRUIT GAUSSIEN

- MOTIVATIONS ET CODE

Nous implémentons ici une méthode de Randomized Smoothing [10]. Nous construisons pour ce faire un nouveau classifieur *lissé*  $g$  à partir d'un classifieur arbitraire de base  $f$ . Considérons un problème de classification où les entrées  $x \in \mathbb{R}^d$  doivent être associées à des classes dans un ensemble  $\mathcal{Y}$ . Lorsqu'on interroge  $g$  à un point  $x$ , le classifieur lissé  $g$  renvoie la classe  $c$  qui a la plus grande probabilité d'être prédite par  $f$  lorsque  $x$  est perturbé par un bruit gaussien isotrope, soit :

$$g(x) = \arg \max_{c \in \mathcal{Y}} \mathbb{P}(f(x + \varepsilon) = c), \quad \text{où } \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

Une définition équivalente est que  $g(x)$  renvoie la classe  $c$  dont la pré-image  $\{x' \in \mathbb{R}^d : f(x') = c\}$  a la plus grande mesure de probabilité sous la distribution  $\mathcal{N}(x, \sigma^2 I)$ . Le paramètre  $\sigma$  est un hyperparamètre, contrôlant le compromis entre robustesse et précision ; il est constant pour toutes les entrées  $x$ . Nous supposons que le comportement de  $g$  est indéfini lorsque l'*argmax* n'est pas unique.

---

**Algorithm 4** Pseudo-code pour Randomized Smoothing

```

1: Entrée : Image d'origine  $x$ , classifieur de base  $f$ , bruit gaussien  $\sigma$ , nombre d'échantillons  $num\_samples$ , classes  $\mathcal{Y}$ 
2:  $counts \leftarrow$  vecteur de zéros de taille  $|\mathcal{Y}|$ 
3:  $x \leftarrow$  normaliser l'image originale
4: for  $i = 1$  à  $num\_samples$  do
5:    $noise \leftarrow \mathcal{N}(0, \sigma^2 I)$                                 ▷ Générer un bruit gaussien isotrope
6:    $x_{noisy} \leftarrow x + noise$                                ▷ Ajouter le bruit à l'image originale
7:    $prediction \leftarrow f(x_{noisy})$                             ▷ Prédire avec le classifieur de base  $f$ 
8:    $counts[prediction] \leftarrow counts[prediction] + 1$           ▷ Mettre à jour le compte pour la classe prédite
9: end for
10:  $predicted\_class \leftarrow \arg \max(counts)$                   ▷ Classe avec le plus grand score
11: Retourner :  $predicted\_class, counts$ 

```

---

#### 3.1.1 • ETUDE DES HYPERPARAMÈTRES

Le seul hyperparamètre autre que  $\sigma$  dans la méthode de Randomized Smoothing est le nombre d'échantillons, noté  $num\_samples$ . Comme l'illustre la figure 8, pour des valeurs variant entre 1 et 100, la précision reste globalement constante face à une attaque spécifique PGD. Cependant, la variance de la précision diminue avec l'augmentation du nombre d'échantillons, traduisant une meilleure stabilité des résultats. En revanche, le temps d'inférence par image augmente de manière linéaire. Ainsi, pour des jeux de 100 images, le temps de calcul devient significatif. Afin de maintenir un compromis entre la précision, la stabilité, et le coût temporel, nous fixerons dans la suite de l'étude le nombre d'échantillons à  $num\_samples = 50$ .

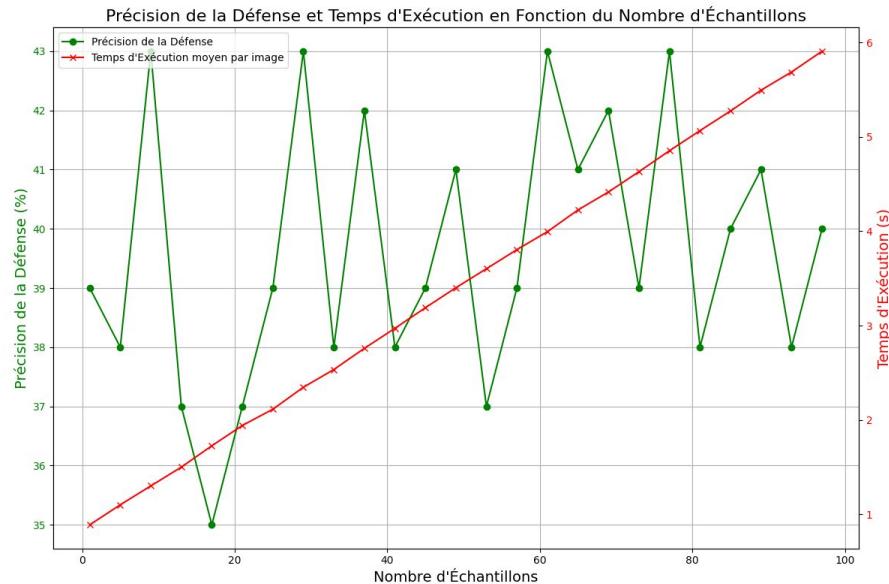


FIGURE 8 – Influence de *num\_samples* sur l’accuracy du modèle contre une attaque spécifique PGD.

### 3.1.2 • RÉSULTATS

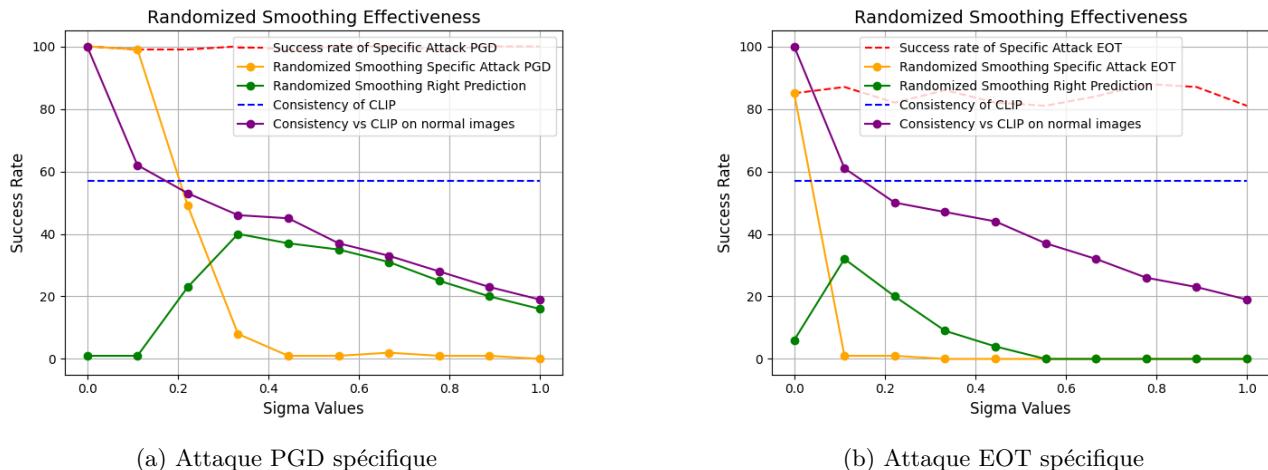
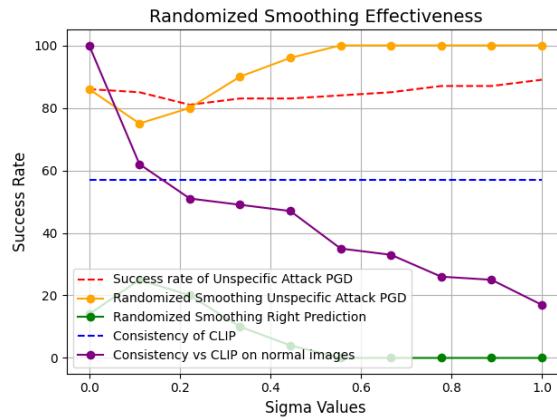
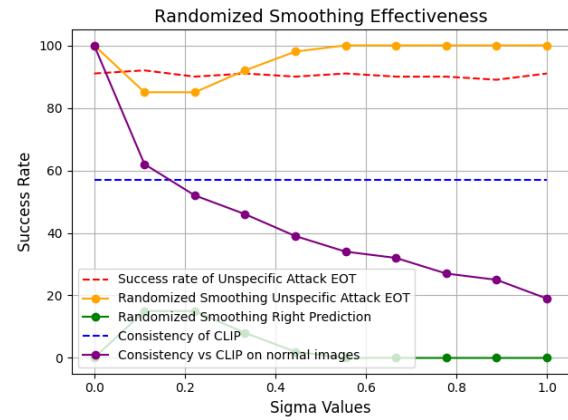


FIGURE 9 – Performances du Randomized Smoothing face aux attaques adversariales spécifiques (PGD et EOT).

Face aux attaques spécifiques, l’augmentation du paramètre  $\sigma$  permet une réduction significative de la capacité de l’attaquant à imposer une classe cible (courbe jaune). Pour l’attaque PGD spécifique, une valeur plus élevée de  $\sigma$  est nécessaire afin d’améliorer la robustesse, mais cela entraîne une dégradation notable de la précision globale du modèle sur des images non corrompues (courbe verte) et réduit la cohérence entre les prédictions du modèle et celles de la défense (courbe violette). L’attaque EOT spécifique, conçue pour s’adapter au bruit, demeure plus efficace, tout en présentant une fenêtre optimale de  $\sigma$  plus réduite, suggérant une robustesse moindre du modèle dans ce scénario. Ainsi, le choix d’un  $\sigma$  approprié apparaît comme un compromis délicat entre robustesse accrue et préservation des performances.



(a) Attaque PGD non spécifique



(b) Attaque EOT non spécifique

FIGURE 10 – Performances du Randomized Smoothing face aux attaques adversariales non spécifiques (PGD et EOT).

Pour les attaques non spécifiques, un comportement similaire est observé, avec l’existence d’un  $\sigma$  optimal permettant d’atteindre un compromis entre robustesse et précision. Cependant, les performances globales sont plus faibles que dans le cas des attaques spécifiques. L’attaque PGD non spécifique présente un maximum d’accuracy plus bas, et l’attaque EOT non spécifique se révèle encore plus perturbante, réduisant fortement la précision du modèle. Cet écart de performance reflète le caractère plus général des attaques non spécifiques, plus difficiles à contrer par une simple injection de bruit. Malgré tout, l’augmentation de  $\sigma$  permet de modérer leur impact, tout en soulevant la question d’une perte inévitable en précision.

Le Randomized Smoothing offre une forme de robustesse face aux attaques adversariales, qu’elles soient spécifiques ou non, mais cette robustesse s’obtient au prix d’une baisse significative de la précision sur des données propres. L’attaque EOT, conçue explicitement pour contourner le Randomized Smoothing, démontre une efficacité accrue, confirmant que sa meilleure performance face à cette défense n’est pas fortuite, mais résulte de son adaptation intrinsèque aux transformations stochastiques. Ces observations soulignent la nécessité de calibrer avec soin le paramètre  $\sigma$  afin de trouver un compromis optimal entre robustesse et performance, et d’explorer des stratégies défensives plus complexes, mieux adaptées aux attaques de type EOT.

## 3.2 MÉTHODE PAR DENOISING

---

- MOTIVATIONS ET CODE

Si le Random Smoothing est déjà une méthode de défense relativement efficace, il présente des faiblesses. En particulier, la décroissance de la performance générale du modèle avec  $\sigma$  (cf courbe violette Figure 10) survient avant que le Smoothing puisse pleinement faire effet, ce qui empêche une défense optimale. C'est pourquoi nous avons étudié une autre piste permettant d'utiliser de plus grandes valeurs de  $\sigma$  en préservant autant que possible l'image. Nous avons pour ce faire implémenté le **Bilateral Blur** [11] [12]. C'est également une méthode de Smoothing, mais où le poids pour chaque pixel ne dépend pas que de la distance au pixel central mais aussi des différences d'intensité et de couleur. Cela permet de préserver les contours (edge-preserving) et donc de conserver un sens à l'image malgré le Smoothing.

En posant  $f$  le classifieur original du modèle où les entrées  $x \in \mathbb{R}^d$  doivent être associées à des classes dans un ensemble  $\mathcal{Y}$ , on a donc maintenant le classifieur :

$$g(x) = \arg \max_{c \in \mathcal{Y}} \mathbb{P}(f \circ T(x) = c),$$

où  $T$  est la transformation appliquée par le Bilateral Filter.

$T$  dépend de 3 hyperparamètres :

1.  $\sigma_{space}$  l'écart-type pour la gaussienne spatiale, similaire au  $\sigma$  du Gaussian Smoothing
2.  $\sigma_{color}$  l'écart-type pour la gaussienne de couleur/intensité. On mesure les écarts à l'aide de la norme  $\|\cdot\|_1$ . Un faible  $\sigma_{color}$  entraîne donc des contours mieux délimités.
3. kernel\_size la taille du noyau, i.e. la région concernée par le denoising.

Formellement, on a :

$$T[I]_p = \frac{1}{W_p} \sum_{q \in \mathcal{S}} G_{\sigma_{space}}(\|p - q\|) G_{\sigma_{color}}(\|I_p - I_q\|) I_q,$$

où  $I_q$  est l'intensité du pixel  $q$ ,  $W_p$  un facteur de normalisation,  $\mathcal{S}$  est l'ensemble des pixels du noyau et  $G_\sigma$  une Gaussienne de variance  $\sigma^2$

**Remarque :** Les normes sont ici  $\|\cdot\|_2$ . On utilise aussi une norme pour la différence d'intensité, car la couleur est en 3 dimensions.

---

**Algorithm 5** Pseudo-code pour Denoising

---

- 1: **Entrée** : Image d'origine  $I$ , sigma\_space  $\sigma_{space}$ , sigma\_color  $\sigma_{color}$ , kernel\_size  $(k, k)$ , classes  $\mathcal{Y}$
  - 2:  $J \leftarrow I$
  - 3: **for**  $p \in \text{pixels}$  : **do**
  - 4:      $J_p \leftarrow T[I]_p$  ▷ Appliquer le filtre à chaque pixel
  - 5: **end for**
  - 6:  $predicted\_class \leftarrow f(J)$  ▷ On applique le classifieur sur  $J$
  - 7: **Retourner** :  $predicted\_class$
- 

Une difficulté de cette méthode est qu'elle introduit de 3 paramètres. Une optimisation des hyperparamètres s'avère alors nécessaire pour l'adapter aux différentes attaques.

## • OPTIMISATION DES HYPERPARAMÈTRES

On prend ici comme métrique la proportion des images attaquées puis défendues par denoising et dont la prédiction est juste, soit, avec les mêmes notations que précédemment :  $\frac{1}{N} |\{i \in [1; N], f \circ T(x_i) = \text{label}_i\}|$

On utilise ici  $N = 100$ . Pour des raisons de temps de calcul, on utilise un échantillon de 400 éléments basés sur une recherche aléatoire avec Optuna pour déterminer kernel\_size.

On obtient dans le cas des deux attaques une **taille du noyau de** (5; 5), qui est donc celle qu'on utilisera désormais.

On cherche maintenant à optimiser  $\sigma_{color}$  et  $\sigma_{space}$ . On utilise pour cela une GridSearch. En raison de la taille des plages de recherche, on se place sur une échelle logarithmique.

On obtient les résultats suivants :

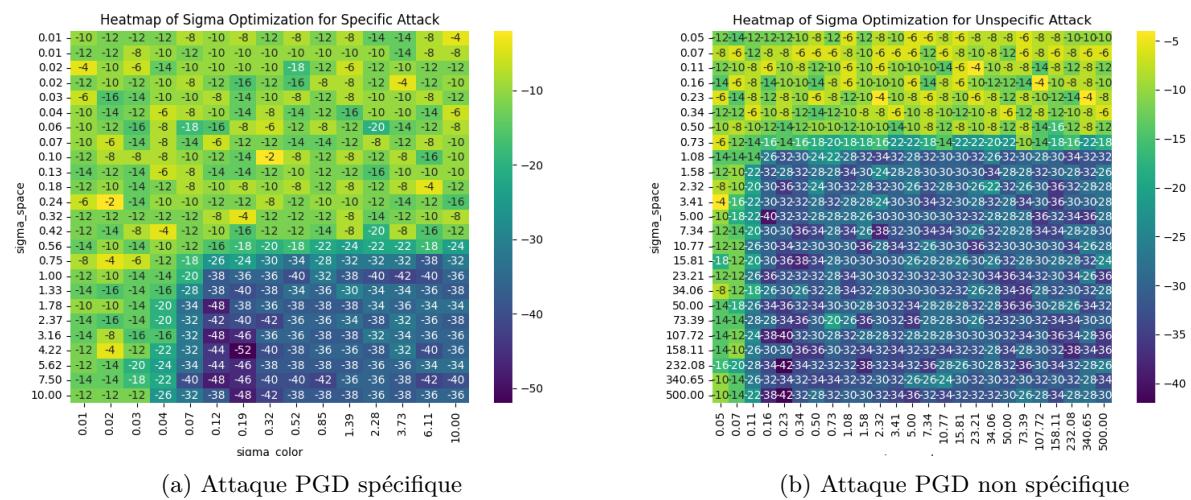


FIGURE 11 – Optimisation des hyperparamètres face à l'attaque PGD

On peut faire plusieurs observations. D'abord, on voit bien l'impact de la défense arriver progressivement. Ainsi, pour l'attaque spécifique, il n'y a qu'un faible effet pour  $\sigma_{space} \leq 0,75$  et  $\sigma_{color} \leq 0,07$ . On voit ensuite qu'à partir de ce seuil, les performances atteignent un certain plateau. Cela peut s'expliquer par la taille du noyau, qui est fixée à (5, 5) : à partir d'un certain  $\sigma$ , l'entièreté du noyau est fortement affecté et augmenter le  $\sigma$  n'a plus d'impact.

On observe néanmoins avant le plateau une zone plus foncée et donc où le modèle est plus performant, correspondant à  $\sigma_{space} = 4,22$  pour l'attaque spécifique et  $\sigma_{space} = 5,0$  pour l'attaque non-spécifique. Ce sont ces valeurs que l'on fixera désormais dans nos graphiques pour  $\sigma_{space}$ . Il est cependant clair, surtout pour l'attaque non-spécifique que l'essentiel est d'être au-dessus des seuils indiqués, plutôt que d'avoir une valeur précise de  $\sigma_{space}$ .

On peut également noter que les ordres de grandeur des deux valeurs dans les plages optimales sont très différents,  $\sigma_{space}$  étant de l'ordre de 5 tandis que  $\sigma_{color}$  est de l'ordre de 0,2.

## • RÉSULTATS

En utilisant les valeurs de  $\sigma_{space}$  précédentes, on peut désormais tracer l'évolution de notre méthode suivant  $\sigma_{color}$ , afin de comprendre l'impact de la préservation des contours sur l'efficacité de la défense.

Pour l'attaque PGD spécifique et l'attaque EOT spécifique :

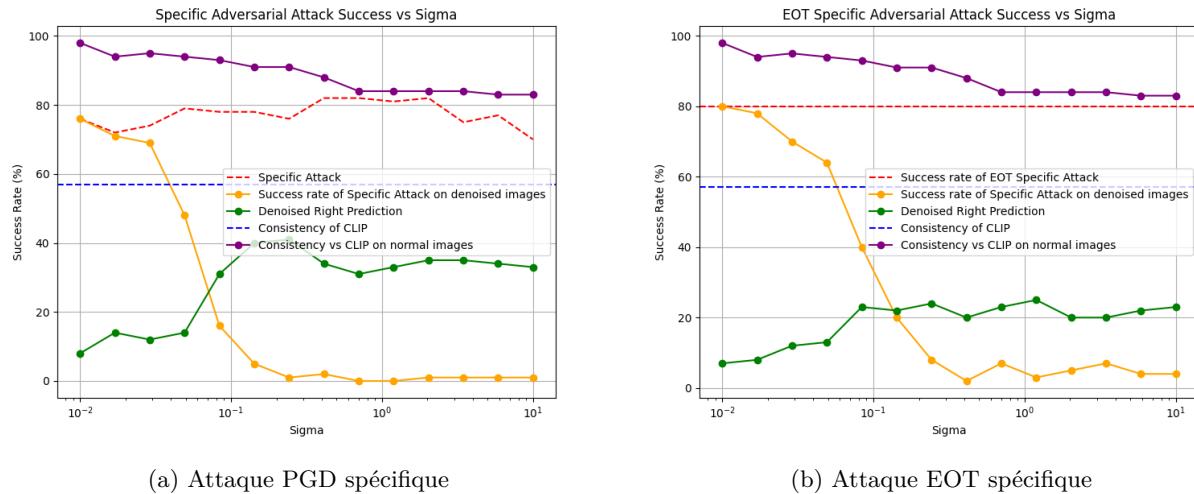


FIGURE 12 – Efficacité du Denoising face à une attaque spécifique

On observe plusieurs résultats notables pour l'attaque PGD spécifique :

- Si l'efficacité de l'attaque spécifique est élevée au départ (courbe rouge, proche de 80%), on observe rapidement un décrochage après une défense par denoising, avec une efficacité (courbe orange) qui devient vite presque nulle. Cela nous confirme que le denoising défend bien contre l'attaque spécifique.
- Dans le même temps, et contrairement à la méthode de bruit gaussien, on observe un décrochage limité par rapport aux performances initiales de CLIP (courbe violette). La courbe reste toujours au dessus des 80% de similarité avec la prédiction de CLIP sur l'image non attaquée.
- Le décrochage de l'efficacité de l'attaque, couplé au maintien de performances proches de CLIP permet de défendre correctement les attaques tout gardant un haut niveau de performance.

On obtient la meilleure performance pour  $\sigma_{color} = 0.3$ , avec 42% de précision sur des images attaquées et 90% sur des images non attaquées.

L'attaque EOT montre cependant déjà quelques limites de la défense par denoising :

- La chute de l'efficacité de l'attaque est moins brutale suivant  $\sigma_{color}$ , avec une efficacité qui reste à minima d'au moins 5%.
- Cela a pour effet de réduire le nombre de bonnes prédictions, avec un pic d'efficacité de la défense à 26% à la place des 42% précédemment trouvés.

Ainsi, on a montré que le denoising permettait de défendre les attaques spécifiques avec une meilleure performance que le Gaussian smoothing. Cependant, l'attaque EOT arrive à faire baisser cette performance, en empêchant de défendre totalement l'attaque notamment.

**Remarque :** Ici, on a utilisé  $\epsilon = 8/255$  alors que les courbes de Gaussian smoothing étaient réalisées avec  $\epsilon = 12/255$ . Cette différence influence le point de départ de l'attaque (courbe rouge) mais ne change pas les trajectoires observées.

On trace désormais l'évolution de notre méthode face à l'attaque PGD et EOT non-spécifique :

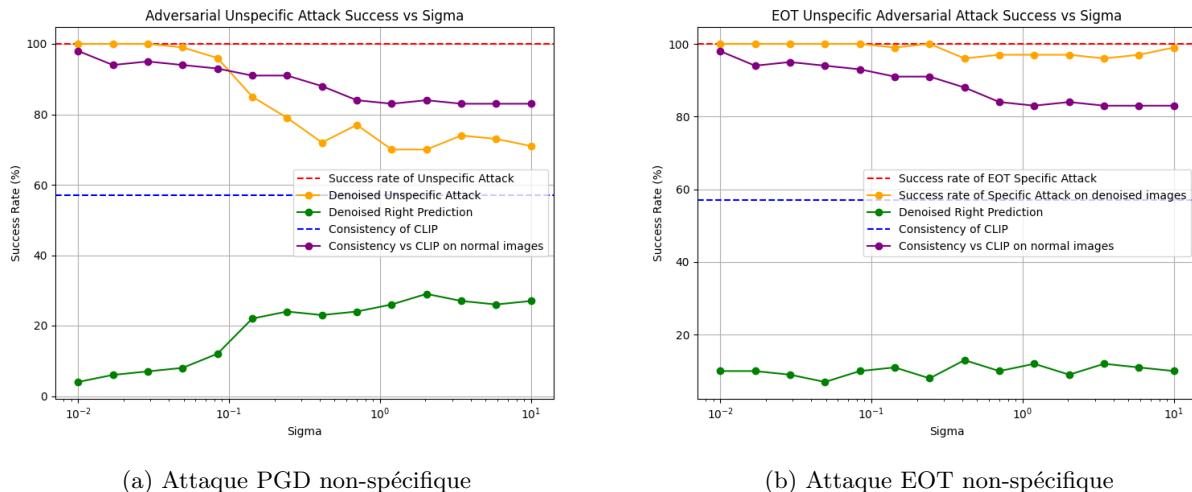


FIGURE 13 – Efficacité du Denoising face à une attaque non-spécifique

Plusieurs éléments sont remarquables :

1. On observe que l'attaque non-spécifique est beaucoup plus puissante que la spécifique, avec une efficacité sans défense de 100% en permanence (courbe rouge) et une efficacité après défense toujours au-dessus de 70% (courbe orange).
2. Cependant et comme pour l'attaque spécifique, le modèle défend bien grâce au denoising, avec la proportion de bonnes prédictions qui atteint les 30% (courbe verte).
3. Malgré la défense, on observe de manière générale des performances bien plus basses face à l'attaque non-spécifique, ce qui est cohérent car celle-ci est plus simple à réaliser : il est plus facile de faire dire n'importe quoi au modèle plutôt que quelque chose de précis.

On obtient la meilleure performance pour  $\sigma_{color} = 2, 2$ , avec 29% de précision sur des images attaquées et 85% sur des images non attaquées.

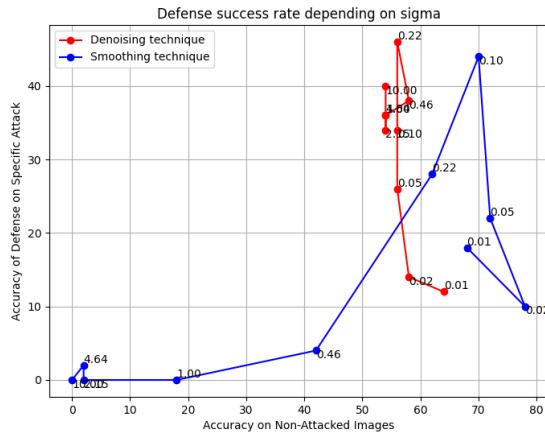
À nouveau, l'attaque EOT influence davantage la défense par denoising, puisqu'elle a été conçue pour la rendre inefficace. On observe notamment :

- Une efficacité de l'attaque très proche de 100% même après la défense.
- Un léger effet de la défense pour  $\sigma_{color} \geq 0,5$ , mais avec une efficacité de l'attaque qui ne descend pas en-dessous de 90%.
- Une proportion de bonnes prédictions qui par conséquent reste très faible, illustrant les effets de l'EOT sur les capacités de défense du modèle.

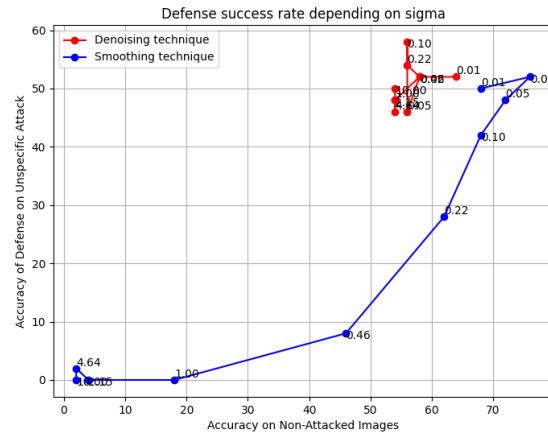
En conclusion, on voit que la méthode de défense par denoising permet de réellement augmenter la robustesse du modèle, à la fois face à des images attaquées mais aussi face à des images non-attaquées dont on arrive à conserver la substance grâce à la préservation des contours. Cependant, la technique d'attaque EOT permet de contourner cette méthode de défense, ce qui justifiera l'exploration de nouvelles méthodes, notamment de fine-tuning.

### 3.3 BILAN DES MÉTHODES PRÉCÉDANT L'INFÉRENCE

Ces deux méthodes de défense se distinguent des suivantes car elles utilisent des opérations avant l'inférence du modèle. Afin de les comparer, on trace leur évolution suivant  $\sigma$  dans le plan, avec pour axe des abscisses la précision sur des images non-attaquées et en ordonnée la précision sur des images attaquées. Cela permet d'avoir une vision claire des compromis faits par chacun des modèles, et de les comparer efficacement.



(a) Défense de l'attaque spécifique



(b) Défense de l'attaque non-spécifique

FIGURE 14 – Comparaison des défenses face à l'attaque PGD

Plusieurs éléments se dégagent :

1. De manière générale, la technique de bruit gaussien a une plus forte précision pour les images non-attaquées, quand la technique de denoising est meilleure face aux images attaquées.
2. On observe une décroissance rapide avec  $\sigma$  de l'efficacité de la méthode de bruit gaussien vers 0, quand la méthode de denoising aura plutôt tendance à converger vers une performance moyenne (dû à la taille du kernel fixée à (5; 5) qui induit une saturation).
3. De manière logique, on observe une convergence des deux méthodes lorsque  $\sigma \rightarrow 0$ . En effet, les deux défenses n'influent plus et sont équivalentes. Pour  $\sigma = 0,00001$ , on ne distingue plus les deux points (non indiqué sur le graphe).

### 3.4 FINE-TUNING PAR APPRENTISSAGE NON-SUPERVISÉ

- MOTIVATIONS

Jusqu’ici, les solutions de défense que nous proposions étaient des étapes préalables à l’évaluation par le modèle, que ce soit par l’ajout de bruit gaussien ou d’un denoising préliminaire. L’objectif sera maintenant de défendre les attaques adversariales en modifiant directement les poids du modèle, en passant par une approche de finetuning.

Nous adoptons ici la méthode du **Unsupervised Adversarial Finetuning** [13]. Cela consiste à récupérer le modèle préentraîné sur des images non-attaquées (ici entraîné sur ImageNet) et de le réentraîner sur des images attaquées.

- ENTRAÎNEMENT DU MODÈLE

L’objectif principal du finetuning adversarial est de rapprocher les représentations adversariales des images (les embeddings) des représentations originales, en optimisant un modèle modifié `model` à partir d’une version originale non modifiée `model_orig`.

Pour  $\mathbf{x}$  une image d’entrée, et  $\mathbf{x}_{\text{adv}}$  sa version modifiée par une attaque adversariale (par exemple l’attaque PGD), on définit les embeddings suivants, où `output_normalize` correspond à normaliser où non la sortie du modèle :

$$\mathbf{e}_{\text{adv}} = \text{model}(\mathbf{x}_{\text{adv}}, \text{output\_normalize} = \text{True}), \quad (1)$$

$$\mathbf{e}_{\text{orig}} = \text{model\_orig}(\mathbf{x}, \text{output\_normalize} = \text{True}). \quad (2)$$

L’idée est de minimiser la distance entre ces deux représentations dans l’espace d’embedding, ce qui revient à minimiser la fonction de perte suivante :

$$\mathcal{L} = \|\mathbf{e}_{\text{adv}} - \mathbf{e}_{\text{orig}}\|_2 \quad (3)$$

Pour stabiliser l’apprentissage, nous appliquons une moyenne sur les pertes des différents exemples du batch. Enfin, on applique une rétropropagation de la perte  $\mathcal{L}$  sur les poids du modèle.

En résumé, le fine-tuning est réalisé par :

- Génération des exemples adversariaux  $\mathbf{x}_{\text{adv}}$  par une attaque (ex. PGD) à partir des données d’entraînement non labellisées  $\mathbf{x}$ .
- Calcul des embeddings adversariaux  $\mathbf{e}_{\text{adv}}$  et originaux  $\mathbf{e}_{\text{orig}}$  via les modèles respectifs `model` et `model_orig`.
- Minimisation de  $\mathcal{L}$  par mise à jour des paramètres du modèle `model`.

Cette approche permet de ne se focaliser que sur l’entraînement de l’encodeur visuel, en se passant de labels pour uniquement chercher à approcher les embeddings originellement produits par le modèle. Cela permet notamment de ne pas réentraîner l’encodeur textuel.

**Remarque :** Pour des problèmes techniques relevant de la facilité à fine-tuner, on utilisera ici l’implémentation OpenClip [14] de CLIP. Cette implémentation présente deux défauts majeurs : d’abord ses performances sont sensiblement plus basses que la version optimisée d’OpenAI (de environ 60% à environ 50% ici). Ensuite, OpenClip a de bien meilleures performances face aux attaques adversariales, car il a été entraîné sur des images attaquées également. On observe ici pour OpenClip une précision d’environ 30% sur des images attaquées par PGD spécifique, contre environ 20% pour l’implémentation originale de CLIP.

- RÉSULTATS

Nous avons réalisé l'entraînement du modèle avec 9000 pas (dont 1000 de warmup), sur le dataset CIFAR100 avec des attaques de  $\epsilon = 8/255$  et pour optimiseur AdamW.

On obtient les résultats suivants, sur 100 images non présentes dans le dataset d'entraînement :

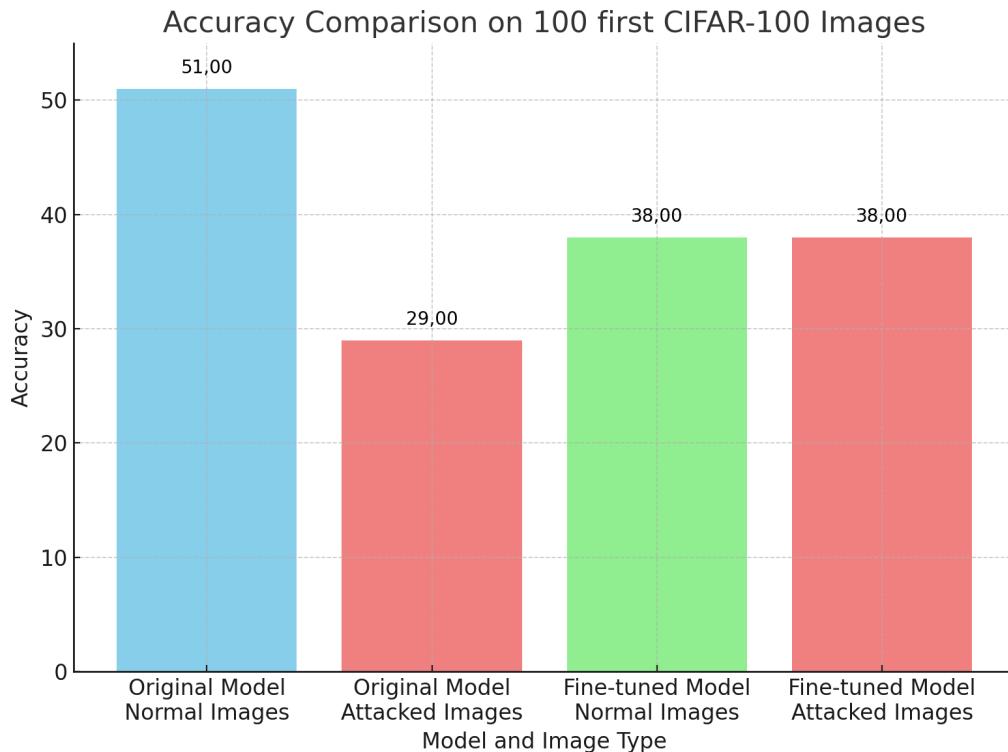


FIGURE 15 – Efficacité de l'approche du fine-tuning face à l'attaque spécifique PGD

Le fine-tuning du modèle a eu plusieurs effets notables :

1. On observe d'abord une nette augmentation de la défense après le fine-tuning : le modèle monte jusqu'à 38% de précision sur les images attaquées, contre 29% précédemment.
2. Ce gain de performance se fait malheureusement au détriment de l'efficacité du modèle sur des images non attaquées, qui tombe de 51% à 38%.
3. Enfin, on remarque que le modèle performe maintenant de manière similaire sur des images attaquées ou non attaquées. Il traite donc désormais bien, ou a minima de manière égale les images saines ou attaquées.

Cependant, la forte baisse de performance du modèle sur des images non-attaquées nous pousse à chercher comment conserver des performances satisfaisantes tout en augmentant la robustesse.

- APPROCHES COMPLÉMENTAIRES

Nous avons mis en œuvre différentes techniques afin d'augmenter la robustesse ou de mieux conserver la performance sur les images non attaquées, sans succès malheureusement.

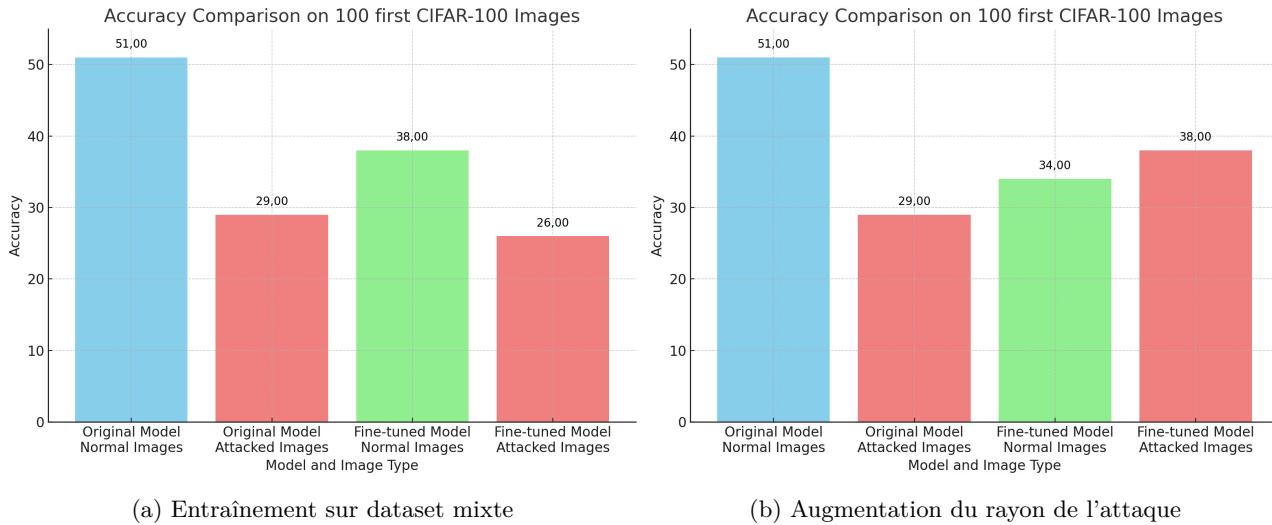


FIGURE 16 – Tentatives d'amélioration des performances du fine-tuning

Nous avons testé deux méthodes :

1. D'abord nous avons cherché à utiliser un dataset mixte d'images attaquées et non attaquées, afin de préserver la performance du modèle. Nous utilisons 50% d'images non attaquées, et 50% d'images attaquées, en tirant à chaque étape au hasard. Nous espérons qu'ainsi le modèle "oubliera" moins ses compétences sur les images non-attaquées et cherchera à concilier les deux. En pratique, cela ne fonctionne pas, ralentissant surtout l'apprentissage du modèle (les gradients sur une image non-attaquée étant en quelque sorte opposés aux gradients des images attaquées, cherchant à se rapprocher du modèle original).
2. Nous avons ensuite cherché à augmenter le rayon de l'attaque (de 8/255 à 16/255) sur les images servant à l'entraînement du modèle, en espérant qu'être exposé à des attaques plus importantes améliorerait la robustesse de celui-ci face à des attaques plus faibles. Cela n'a malheureusement pas fonctionné, avec une performance sur les images attaquées qui reste haute mais des pertes sur les images saines. Nous interprétons ceci comme le fait que le modèle s'éloigne davantage du modèle original face aux attaques plus importantes, sans y gagner cependant en robustesse.

Cette incapacité à conserver la performance du modèle sur des images non-attaquées est ce qui nous a poussé à explorer la piste de l'orthogonal fine-tuning.

### 3.5 ORTHOGONAL FINE-TUNING

- PRÉSENTATION DE LA MÉTHODE

La méthode précédente permet de grandement améliorer la robustesse du modèle, mais au coût d'une baisse de performance sur les images non-attaquées. Nous avons interprété cela comme une sorte de sur-apprentissage sur les images attaquées, et tenté de trouver une méthode qui permet de maintenir les capacités de généralisation du modèle original. *Enhancing Robustness of Vision-Language Models through Orthogonality Learning and Self-Regularization* [15] propose d'adapter des VLMs préentraînés, à des tâches spécifiques tout en préservant leur généralisabilité. L'approche repose, comme un LoRa, sur le gel des poids originaux du modèle et l'apprentissage d'une nouvelle matrice entraînable permettant d'actualiser les poids du modèle original pour qu'il s'adapte mieux à une tâche spécifique. Toutefois, contrairement à LoRa, au lieu d'ajouter une matrice d'actualisation des poids aux poids originaux, on les multiplie par une matrice orthonormale. Nous avons tenté d'implémenter cette méthode à la tâche de robustesse sur les images attaquées, ce qui constitue la dimension réellement novatrice de notre travail.

Les poids du modèle préentraîné, notés  $\mathbf{W}_0$ , sont gelés et ne sont pas mis à jour directement lors du *finetuning*. Pour adapter le modèle à une nouvelle tâche, les poids effectifs du modèle sont obtenus par la multiplication de  $\mathbf{W}_0$  avec une matrice orthonormale  $\mathbf{Q}$ , pour obtenir les nouveaux poids :

$$\mathbf{W} = \mathbf{Q}\mathbf{W}_0,$$

où  $\mathbf{Q}$  est contraint à être orthogonale, c'est-à-dire  $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$ . Cette stratégie garantit que la norme des poids est préservée, c'est à dire que

$$\|\mathbf{W}\| = \|\mathbf{W}_0\|$$

ce qui est garantit une meilleure stabilité lors de l'entraînement, empêchant le modèle finetuné de trop "s'éloigner" des performances du modèle original, ce qui améliore donc sa capacité à être généralisé.

Pour garantir l'orthogonalité de  $\mathbf{Q}$ , nous utilisons la **paramétrisation de Cayley**. Cette paramétrisation exprime  $\mathbf{Q}$  en fonction d'une matrice antisymétrique  $\mathbf{A}$  ( $\mathbf{A}^\top = -\mathbf{A}$ ) comme suit :

$$\mathbf{Q} = (\mathbf{I} - \mathbf{A})(\mathbf{I} + \mathbf{A})^{-1}.$$

Cette paramétrisation permet d'entraîner  $\mathbf{Q}$  tout en maintenant l'orthogonalité de manière efficace et stable.

L'objectif global d'optimisation combine une perte pour la tâche spécifique ( $\mathcal{L}_{\text{tâche}}$ ), ici la robustesse, et une régularisation croisée ( $\mathcal{L}_{\text{distillation}}$ ) pour préserver les connaissances globales du modèle préentraîné :

$$\mathcal{L}_{\text{ortho}} = \mathcal{L}_{\text{tâche}} + \lambda \mathcal{L}_{\text{distillation}},$$

où  $\lambda$  est un coefficient de régularisation.

La perte pour notre tâche spécifique est définie comme suit :

$$\mathcal{L}_{\text{ce}} = -\log p(\hat{y} = y \mid X)$$

où  $X$  est l'image en entrée,  $\hat{y}$  la classe prédite par le modèle, et  $y$  la prédiction du modèle sur l'image non attaquée. Ainsi, nous finetunons notre modèle sur un ensemble d'image attaquées, en essayant de lui faire prédire la même chose que le modèle original sur l'image non attaquée.

Pour préserver les connaissances du modèle préentraîné, la perte de distillation est définie à l'aide de la divergence de Kullback-Leibler entre les sorties du modèle préentraîné ( $z^{\text{pre}}$ ) et celles du modèle ajusté ( $z^{\text{fine-tune}}$ ) :

$$\mathcal{L}_{\text{distillation}} = \frac{1}{N} \sum_{i=1}^N \text{KL} \left( \sigma \left( \frac{z_i^{\text{pre}}}{T} \right) \| \sigma \left( \frac{z_i^{\text{fine-tune}}}{T} \right) \right),$$

où  $\text{KL}(\cdot \| \cdot)$  est la divergence de Kullback-Leibler,  $\sigma(\cdot)$  est la fonction softmax, et  $T$  est un hyperparamètre de température pour adoucir les distributions.

En gelant les poids originaux  $\mathbf{W}_0$  et en les combinant avec des matrices orthonormales, le modèle préserve les relations sémantiques au sein de ses représentations. Combiné à la régularisation croisée via la perte de distillation, cela permet d'adapter le modèle à des tâches spécifiques sans le sur-ajuster, garantissant ainsi une généralisabilité élevée.

## • RÉSULTATS

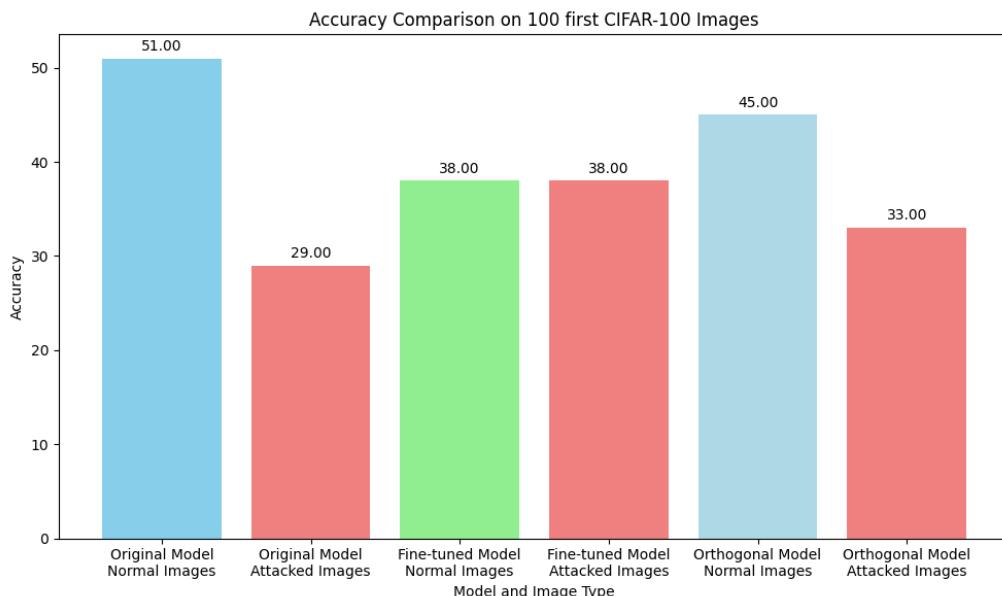


FIGURE 17 – Comparaison des méthodes de fine-tuning orthogonal et par apprentissage non supervisé

On observe sur la figure 17 qu'on a effectivement mieux maintenu les performances du modèle sur les images non-attaquées, au coût d'une moins bonne robustesse. On obtient une baisse relative des performances sur les images non attaquées de 6% contre 13% pour la méthode précédente, et un gain de robustesse de 4% contre 9% précédemment. Ainsi, il s'agit de trouver un compromis entre maintien des performances et robustesse. Si les attaques adversariales peuvent avoir des conséquences graves, l'utilisation du fine-tuning par apprentissage non-supervisé semble être la meilleure option. En revanche, si maintenir une grande précision dans la majorité des applications est notre priorité, la méthode d'orthogonal fine-tuning permet de gagner en robustesse sans trop sacrifier les performances globales.

Une perspective intéressante à laquelle nous avons pensé, mais que nous n'avons pas réussi à implémenter dans le temps imparti, est d'essayer de coupler les deux méthodes en entraînant le modèle avec la perte suivante, en gardant les mêmes notations que précédemment :

$$\mathcal{L} = \|\mathbf{e}_{\text{adv}} - \mathbf{e}_{\text{orig}}\|_2 + \lambda \mathcal{L}_{\text{ortho}} \quad (4)$$

où  $\lambda$  est un hyperparamètre qui permet de mieux contrôler l'influence relative d'un type de fine-tuning par rapport à l'autre. Cela pourrait permettre d'arriver à un compromis plus satisfaisant, obtenant de meilleures performances de généralisation que le fine-tuning par apprentissage non-supervisé seul, et une meilleure robustesse que le fine-tuning orthogonal seul.

Notons que l'avantage de ces méthodes de fine-tuning est enfin de permettre d'améliorer la rapidité de la prédiction, puisqu'il n'y a plus d'étape préalable à l'inférence comme dans le cas du bruit gaussien ou du denoising. Cela le rend plus applicable à des situations concrètes, comme l'exemple du panneau STOP donné en introduction.

## 4

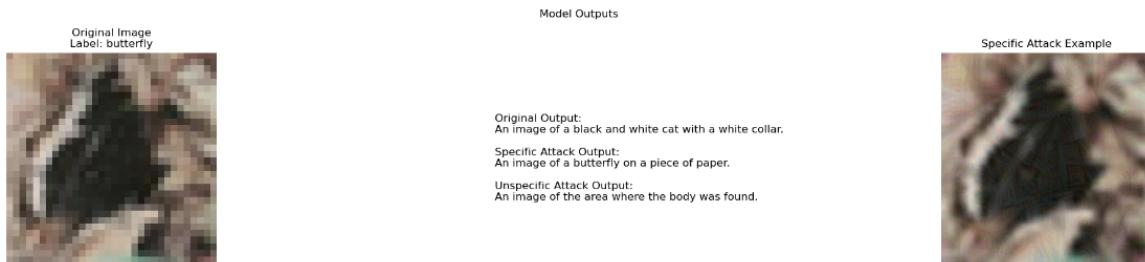
# GÉNÉRALISATION DES ATTAQUES À D'AUTRES MODÈLES

Cette section, plus qualitative, vise à examiner dans quelle mesure une attaque conçue pour tromper un modèle, en l'occurrence CLIP, peut être efficace sur un autre modèle, ici OpenFlamingo.

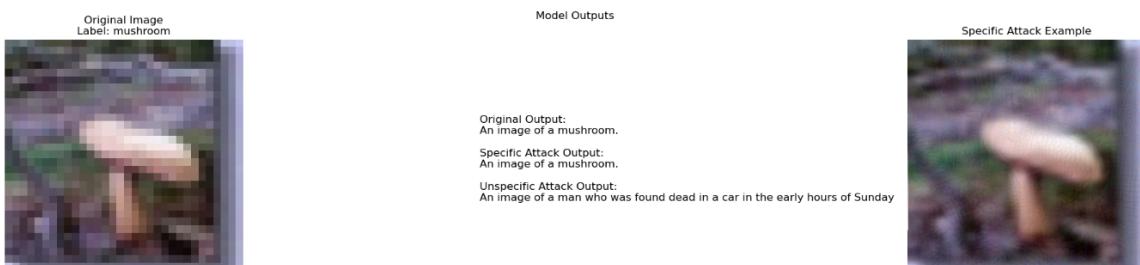
## 4.1 IMAGES DU DATASET CIFAR100

Dans un premier temps, nous avons perturbé des images du dataset public CIFAR100 (résolution 32x32) pour les soumettre en entrée à OpenFlamingo et analyser sa réponse. Les attaques spécifiques ont été conçues pour maximiser la probabilité que CLIP prédise "pomme", tandis que les attaques non spécifiques visaient à éloigner la prédiction de CLIP de celle sur l'image originale. Toutes les attaques ont été réalisées avec un rayon  $\epsilon = 8/255$  et 25 itérations.

Les résultats obtenus sont présentés à la figure 18. La colonne de droite montre les images attaquées données en entrée à OpenFlamingo, mettant en évidence que les perturbations restent imperceptibles à l'œil humain.



(a) Attaque spécifique.



(b) Attaque non spécifique.

FIGURE 18 – Résultats des attaques spécifiques et non spécifiques sur OpenFlamingo pour des images du dataset CIFAR100.

On observe que, même si l'attaque spécifique ne fonctionne pas, l'attaque non spécifique permet de produire des prédictions absurdes sur OpenFlamingo. Par exemple, dans le premier cas, le modèle prédit un résultat sans rapport avec un champignon. Les résultats des attaques EOT conduisent à des conclusions similaires et ne sont donc pas présentés ici pour des raisons de concision.

Ces résultats montrent la vulnérabilité des modèles à des attaques simples, même conçues pour d'autres architectures. Toutefois, ces conclusions doivent être nuancées par la faible résolution des images, comme illustré par le cas de la deuxième image, où même nous, humains, avons eu des difficultés à identifier un papillon.

## 4.2 IMAGES DE HAUTE QUALITÉ

---

Nous avons ensuite étudié des images de meilleure qualité, en résolution 256x256, tout en maintenant constants les autres paramètres des attaques. Les résultats sont présentés dans le figure 19 et ???. Les résultats montrent que la qualité des images joue un rôle déterminant dans l'efficacité des attaques. Avec des images haute résolution, OpenFlamingo n'est peu ou pas affecté par les attaques. Même lorsque le résultat diffère, il reste proche du sens initial de la phrase. Nous avons également essayé avec des attaques EOT, mais les résultats n'étaient pas meilleurs. Enfin, nous avons également essayé d'augmenter le rayon d'attaque, mais même avec  $\epsilon = 16/255$ , les résultats étaient légèrement améliorés. Au delà de cette valeur, les images deviennent trop bruitées pour que la perturbation soit considérée comme imperceptible pour l'oeil humain.

Adversarial Attacks and OpenFlamingo Outputs (Images 1-2)



Original Output:  
An image of the Golden Gate Bridge.

Specific Attack:  
An image of the Golden Gate Bridge.  
Unspecific Attack:  
An image of the Golden Gate Bridge.



Original Output:  
An image of a starfish on the beach.

Specific Attack:  
An image of a starfish on the beach.  
Unspecific Attack:  
An image of a starfish on the beach.

Adversarial Attacks and OpenFlamingo Outputs (Images 3-4)



Original Output:

An image of a person standing on a cliff with a sunset in the background.

Specific Attack:

An image of a golfer silhouetted against a sunset.

Unspecific Attack:

An image of a silhouette of a man standing on a hill in the middle of a sunset.



Original Output:

An image of a desk with a laptop, a tablet, a phone, and a printer.

Specific Attack:

An image of a desk with a computer, a laptop, a monitor, and a printer.

Unspecific Attack:

An image of a desk with a laptop, a monitor, a keyboard, and a mouse.

FIGURE 19 – Résultats des attaques spécifiques et non spécifiques sur OpenFlamingo pour des images haute résolution.

## 5 CONCLUSION

Ce rapport a exploré les vulnérabilités du modèle CLIP face aux attaques adversariales, en mettant en lumière l'efficacité des mécanismes de défense et les perspectives d'amélioration. Les attaques, qu'elles soient spécifiques ou non, ont démontré la fragilité de CLIP dans des contextes critiques, avec des perturbations imperceptibles qui compromettent sa robustesse.

Les approches défensives étudiées, telles que le bruit gaussien, le denoising, et le fine-tuning, ont apporté des solutions variées. Cependant, chaque méthode présente des limites, notamment des compromis entre robustesse et précision sur des données non attaquées. Le fine-tuning orthogonal se distingue en maintenant un équilibre acceptable entre généralisation et robustesse, tandis que le fine-tuning non supervisé maximise la défense au détriment des performances générales.

En outre, les résultats ont révélé que la défense présente des performances inférieures face à des attaques sophistiquées telles que l'Expectation Over Transformation (EOT). Notre étude s'est concentrée sur le modèle CLIP, et nous avons observé que les attaques perdent de leur efficacité lorsqu'elles sont transférées vers d'autres modèles, comme OpenFlamingo, notamment pour des images à haute résolution. Ces constatations soulignent l'importance d'explorer la transférabilité des attaques adversariales comme piste de recherche.

Ainsi, ce travail ouvre des perspectives prometteuses pour l'amélioration de la sécurité des modèles multi-modaux dans des applications sensibles, tout en soulignant l'importance d'un compromis adapté entre robustesse et performances.

## RÉFÉRENCES

- [1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv :2103.00020*, 2021.
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Simon W. Kornblith, Mohamed El Banani, Po-Yao Huang, Michal Valko, Florian Bordes, and et al. Flamingo : a visual language model for few-shot learning. *arXiv preprint arXiv :2204.14198*, 2022.
- [3] Christian Schlarmann and Matthias Hein. On the adversarial robustness of multi-modal foundation models. *arXiv preprint arXiv :2308.10741*, 2023.
- [4] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex : An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv :1702.01135*, 2017.
- [5] Tianyuan Zhang, Lu Wang, Xinwei Zhang, Yitong Zhang, Boyi Jia, Siyuan Liang, Shengshan Hu, Qiang Fu, Aishan Liu, and Xianglong Liu. Advlm : Visual adversarial attack on vision-language models for autonomous driving. *arXiv preprint arXiv :2411.18275*, 2024.
- [6] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. *arXiv preprint arXiv :2306.13213*, 2023.
- [7] Alvi Md Ishmam and Christopher Thomas. Semantic shield : Defending vision-language models against backdooring and poisoning via fine-grained knowledge alignment. *arXiv preprint arXiv :2411.15673*, 2024.
- [8] Han Wang, Gang Wang, and Huan Zhang. Astra : Steering away from harm : An adaptive approach to defending vision language model against jailbreaks. *arXiv preprint arXiv :2411.16721*, 2024.
- [9] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv :1707.07397*, 2017.
- [10] Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv :1902.02918*, 2019.
- [11] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. Bilateral filtering : Theory and applications. *Foundations and Trends® in Computer Graphics and Vision*, 4(1) :1–73, 2009.
- [12] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 501–509, 2019.
- [13] Christian Schlarmann, Naman Deep Singh, Francesco Croce, and Matthias Hein. Robust clip : Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. *arXiv preprint arXiv :2402.12336*, 2024.
- [14] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. *arXiv preprint arXiv :2212.07143*, 2022.
- [15] Jinlong Li, Dong Zhao, Zequn Jie, Elisa Ricci, Lin Ma, and Nicu Sebe. Enhancing robustness of vision-language models through orthogonality learning and self-regularization. *arXiv preprint arXiv :2407.08374*, 2024.