Méthodes d'intelligence artificielle pour la surveillance d'une colonie de phoques gris

La Revue 3E.I
Culture Sciences de l'Ingénieur

Édité le 17/03/2025

école normale supérieure paris—saclay

Thibault NAPOLÉON[1] – Ayoub KARINE[2] – Hamza MASMOUDI[1]

[1] *ISEN Ouest, LabISEN, Vision-AD, 20 rue Cuirassé Bretagne, 29200 Brest*
[2] *Université Paris Cité, LIPADE, F-75006 Paris, France*

Cette ressource fait partie du N°115 de La Revue 3EI du deuxième trimestre 2025.

Dans le cadre de la conservation de la biodiversité marine, les technologies de surveillance et d'analyse jouent un rôle crucial. En effet, la gestion efficace des habitats naturels nécessite une compréhension approfondie des écosystèmes et de leur dynamique. Cela est particulièrement vrai dans des zones sensibles comme l'îlot de Morgol, situé dans la réserve naturelle nationale d'Iroise, qui abrite une population importante de phoques gris. Ces dernières années, les méthodes traditionnelles de suivi ont été remises en question, notamment à cause de leur impact potentiellement perturbateur sur la faune.

Cet article se concentre sur le développement d'une méthode de surveillance qui minimise les interférences avec les espèces surveillées tout en fournissant des données précises et exploitables. En particulier, nous proposons ici une approche basée sur la vision par ordinateur et les méthodes modernes d'intelligence artificielle pour : 1. Mesurer la fréquence des dérangements liés au passage de bateau ou au débarquement d'humains. 2. Estimer la densité de phoque gris sur l'îlot. La méthodologie employée s'appuie d'une part sur l'utilisation de *YOLO v8* pour la détection des perturbateurs et sur l'approche *IOCFormer* ré-entrainée pour la phase de comptage d'autre part. Comparés à la méthode précédemment utilisée, les résultats obtenus sont plus riches et montre une nette amélioration de l'estimation de la colonie pour des coûts de calcul acceptable.

1 - Introduction

La vision par ordinateur présente aujourd'hui un fort intérêt pour l'observation des espaces naturels au travers de tâches allant de la détection d'objets à la segmentation sémantique en passant par le comptage d'individus [1]. Cet attrait pour ce champ de recherche est poussé par le coût faible des capteurs de vision ainsi que par la croissance des approches d'intelligence artificielle traitant les données issues des capteurs optiques. En particulier, les réseaux de neurones profonds ont permis un développement rapide de nouvelles méthodes grâce à l'essor de deux éléments complémentaires. D'une part, les grandes bases de données d'images, telles qu'*ImageNet* [2], nécessaires à l'apprentissage des modèles d'intelligence artificielle. D'autre part, les puissances de calcul disponible au travers des processeurs graphiques (c.-à-d. GPU) qui se sont avérés particulièrement efficaces pour exécuter les approches neuronales. Ainsi, il est désormais possible d'employer ce type d'approche d'intelligence artificielle pour des projets concrets alliant robustesse des traitements et rapidité des calculs dans des applications diverses telles que la reconnaissance de visages, le déploiement de voitures autonomes ou encore la préservation de l'environnement.

1.1 - Présentation du projet

L'approche présentée ici s'inclut dans le projet de remplacement de l'observatoire qui permettait avant 2024 d'effectuer le comptage des individus d'une colonie de phoques gris après sa destruction par les intempéries. Ce dispositif, installé sur l'îlot de Morgol dans l'archipel de Molène au large des côtes Finistériennes, a pour rôle de faciliter le suivi de la faune sauvage qui s'y trouve avec comme objectif de minimiser les dérangements. L'observatoire mis en place est équipé d'une caméra, de panneaux solaires offrant l'autonomie énergétique, ainsi qu'un lien radio permettant la retransmission des images à terre, voir figure 1. La caméra disponible sur l'observatoire possède deux capteurs, dans les domaines visible et infrarouge, mais seule la première modalité est utilisée ici. Les images captées par l'observatoire sont traitées à terre à l'aide d'une carte de traitement *Jetson Nano Orin* embarquant les algorithmes mis en œuvre dans le projet. En particulier, deux réseaux de neurones ont été déployés pour le suivi de la colonie dans le temps :

1. Un réseau dédié à la détection de bateaux naviguant aux abords de l'îlot ou de personnes y débarquant.
2. Un réseau dédié au comptage s'appuyant sur l'estimation de la densité de phoques gris sur l'îlot.

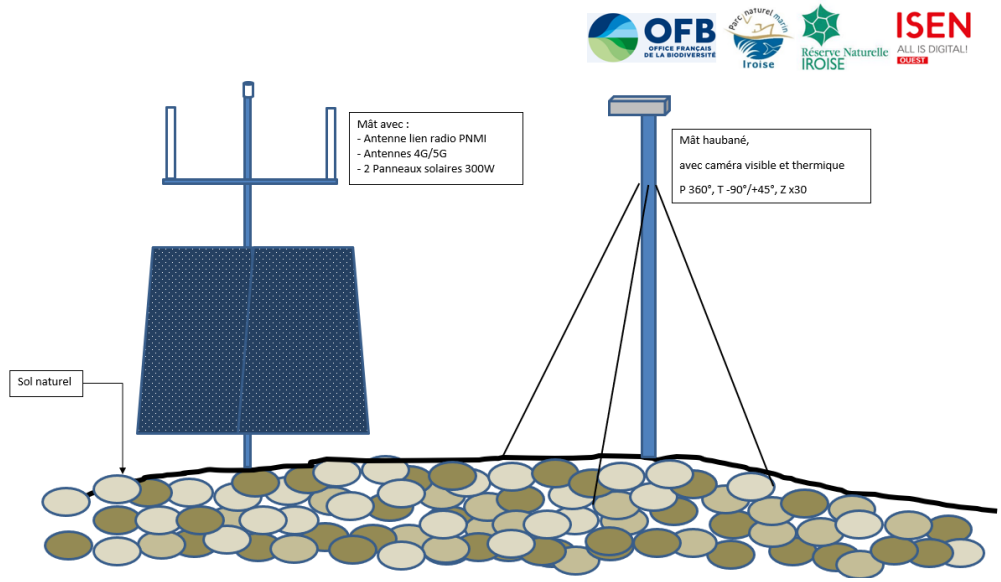

Figure 1 : Schéma de l'observatoire qui sera installé sur l'îlot de Morgol. On y trouve une caméra bispectrale, deux panneaux solaires ainsi qu'une antenne pour le lien radio.

1.2 - Travaux existants

Une première version de l'observatoire sur l'îlot de Morgol avait été réalisée en 2019 pour répondre à la problématique du suivi de la colonie de phoque. La méthode proposée par Karine et al [3] s'appuie sur une approche de classification des vidéos de phoques en utilisant l'apprentissage par transfert sur un réseau de neurones convolutifs (CNN). Grâce au transfert de connaissance apporté par cette technique, qui consiste à réutiliser les hyperparamètres d'un modèle pré-entraîné sur un large ensemble de données, il est possible de limiter la quantité d'annotations nécessaire à la maturation (c.-à-d. apprentissage) du réseau de neurone employé. La mise en œuvre de méthode proposée repose sur deux phases distinctes présentes dans la majorité des approches d'apprentissage supervisé : 1. Une phase « hors ligne » permettant l'entraînement du modèle à partir de vidéos annotées. 2. Une phase « en ligne » utilisant le modèle entraîné pour classer automatiquement les images issues des nouvelles vidéos pour en extraire les phoques et ainsi estimer l'évolution de leur nombre au cours du temps.

Phase « hors ligne »

Cette phase est appelée hors ligne, car elle est réalisée une seule fois pour apprendre au réseau de neurones profond à reconnaitre les phoques sur une image. Dans celle-ci, des imagettes de taille 70 × 100 pixels sont utilisés pour entraîner le modèle de classification. En particulier, une annotation de 353 imagettes de la classe « phoque » ont été utilisée conjointement à 357 imagettes de la classe « non-phoque » comme vérité terrain pour la phase d'apprentissage, voir figure 2. En effet, étant donné le cadre applicatif de la méthode, aucun jeu de données n'était disponible pour entraîner le modèle. Le réseau utilisé est un CNN basé (a) sur l'architecture VGG-16 [4] pré-entraîné sur la base de données *ImageNet*. Ce réseau contient 13 couches convolutives suivies de couches de *pooling* et de couches complètement connectées (c.-à-d. des couches denses). Une couche de 256 neurones ayant une fonction d'activation de type *ReLU* et avec un dropout fixé à 0.5 a été ajoutée pour éviter le surapprentissage. Enfin, la dernière couche de sortie est une classification binaire avec fonction d'activation de type *Softmax* pour permettre une prédiction probabiliste. L'apprentissage par transfert du réseau a été réalisé avec un optimiseur Adam, une taille de lot de 8 sur 50 époques. Pour cette phase, la base d'annotation a été séparée en trois parties, entrainement (60 %), validation (20 %) et test (20 %). La base de validation permet de limiter le risque du surapprentissage tandis que la base de test permet d'évaluer la pertinence du modèle sur des données annotées, mais inconnues du système d'apprentissage. Les résultats obtenus montrent une précision globale de 87.24 % sur la base de test. Les erreurs de classification sont en partie dues à la ressemblance entre les phoques et les galets présents dans les images.

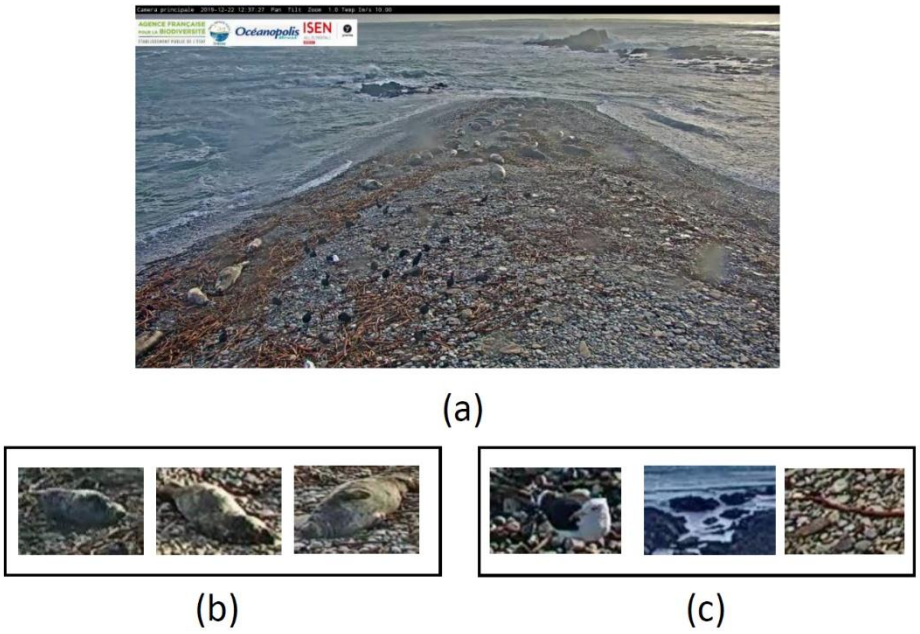

Figure 2 : (a) Exemple d'image issue de la caméra installée sur l'îlot de Morgol. (b) Exemple de vignettes de la classe « phoque ». (c) Exemple de vignettes de la classe « non-phoque ».

Phase « en ligne »

Une fois l'entraînement du modèle réalisé, le système peut alors être utilisé pour estimer le nombre de phoques sur l'îlot. Pour cette phase, une stratégie de fenêtre glissante, avec un décalage de 100 pixels, est utilisée pour classifier chaque vignette de l'image entre les classes « phoque » et « non-phoque ». Ici, la taille des vignettes est identique à celle de la phase d'apprentissage, à savoir 70 × 100 pixels. Dans cette approche, pour toutes les images captées sur l'îlot et retransmises sur le continent, le système d'intelligence artificielle va parcourir l'image pour y détecter les vignettes présentant des phoques. Conjointement à cette détection, des informations temporelles sont extraites de la vidéo pour horodater les présences/absences de phoque. En particulier, les données temporelles sont composées de la date et de l'heure d'acquisition de la vidéo ainsi que de l'instant

précis de la détection en heure, minute et seconde. Ainsi, à la suite de la phase de classification, une agrégation des résultats permet d'estimer le nombre phoque sur chaque image et de suivre ainsi l'évolution de la colonie au cours du temps.

Ce dispositif est désormais inopérant après 5 années en conditions difficiles, incluant la tempête Ciarán qui a eu lieu à la fin de l'année 2023. De ce fait, nous proposons une nouvelle implémentation de l'approche de suivi de la colonie de phoque au travers de nouvelles méthodes qui suivent l'évolution de l'état de l'art dans le domaine.

2 - Approche proposée

Dans le nouveau système, deux approches disjointes, mais complémentaires, ont été mise en place. La première est basée sur *YOLO v8* pour détecter les bateaux et les humains qui sont des perturbateurs pour l'écosystème marin tandis que la seconde s'appuie sur l'approche *IOCFormer* ré-entrainée pour estimer la densité de phoque gris.

2.1 - Mesure des perturbations : *YOLO v8*

Afin d'étudier précisément le comportement de la colonie de phoque gris, l'évaluation de l'impact des évènements extérieurs est nécessaire. Pour cela, un outil d'intelligence artificielle permettant la détection d'évènements tels que le passage d'un bateau, l'arrivée d'un kayak ou le débarquement de personnes sur l'île est mise en place. Ainsi, associé aux informations temporelles, il est possible d'évaluer le degré de perturbation de la colonie et le temps de retour à l'état normal du reposoir. L'outil développé est basé sur un réseau de neurones convolutifs « léger » permettant le traitement des vidéos en temps réel. En effet, l'une des contraintes du projet est la quantité de données à analyser, à savoir, un flux vidéo Full HD permanent. Le réseau de neurones profonds utilisé est la version 8 de la famille *YOLO* [5] (You Only Look Once) développé par Ultralytics qui améliore la précision et la vitesse par rapport aux versions précédentes grâce à des optimisations architecturales et de nouvelles techniques d'apprentissage. Ce modèle modulaire et polyvalent permet de réaliser plusieurs tâches en vision par ordinateur, dont la détection d'objets.

L'architecture de *YOLO v8* est composée de trois parties principales : *backbone*, *neck* et *head*. Le réseau de base (c.-à-d. Le *backbone*) permet d'extraire les caractéristiques visuelles de l'image telles que les contours, les textures ainsi que les motifs colorimétriques. Pour cela, il s'appuie sur des optimisations de l'architecture *CSPDarknet* permettant une meilleure efficacité. La partie *neck* de l'architecture apporte une capacité multi-échelle à la détection en fusionnant les informations issues de différentes échelles de l'image. Il est basé sur une optimisation de la méthode *PANet*. Enfin, la partie *head* du réseau génère les prédictions sous forme de boîtes englobantes associées, pour chacune, aux classes prédites avec leurs scores de confiances.

Ce réseau pré-entraîné sur la base de données *COCO* [6] permet de détecter et de reconnaître un ensemble de concept, dont les bateaux et les humains. Appliqué sur chaque image de la vidéo, il permet de détecter les évènements extérieurs pouvant perturber la colonie. Afin de rendre plus robuste le système dans les conditions difficiles, une étape de post-traitements a été mise en place. Étant donné qu'une perturbation de type « humain » ou « bateau » dure dans le temps, un lissage des détections est ajouté au système en suivant deux stratégies : 1. Une fusion des détections similaires proches dans le temps (perte de détection). 2. Une suppression des détections isolées (fausse détection). La première stratégie permet d'agréger deux détections similaires qui apparaissent à moins de 10 secondes d'intervalle tandis que la seconde élimine ensuite les détections de moins de 5 secondes.

2.2 - Comptage s'appuyant sur l'estimation de la densité : *IOCFormer*

Le modèle utilisé dans la version précédente [3] présente des lacunes significatives lorsqu'il s'agit de résoudre les ambiguïtés créées par des phoques qui sont partiellement occultés par d'autres. Cette limitation découle de la nature des données d'entraînement et de la structure même du réseau de neurones, qui ne permet pas de détecter plus d'un phoque dans une fenêtre de taille 70 x 100 pixels. Cette incapacité peut avoir des conséquences importantes pour l'estimation de la taille de la colonie, dégradant de ce fait les données fournies au parc marin pour lequel la qualité du comptage est importante pour évaluer correctement l'état de la population de phoque dans le but de conserver leur habitat. Pour pallier ces imprécisions, un changement de paradigme a été opéré dans la nouvelle version du projet. En effet, l'algorithme proposé ne repose plus sur une classification par fenêtre glissante, mais sur un comptage par estimation de densité.

Cette technique vise à convertir le problème de comptage en une tâche de prédiction de densité. Au lieu de compter directement les objets, cette méthode génère une carte de densité où chaque pixel représente la densité probable d'objets à cet emplacement. Cette technique est particulièrement utile dans des scènes où les objets sont groupés ou se chevauchent, car elle permet de modéliser la densité d'objets même dans les zones où ils ne sont pas clairement séparés. En particulier, le modèle *IOCFormer* [7] a été utilisé, car il présente une approche performante pour traiter le problème du comptage des objets dits indiscernables. Après une extraction de caractéristiques au moyen d'un encodeur (*ResNet-50* [8]), ce modèle s'appuie sur deux branches complémentaires, à savoir : 1. Une branche de densité qui estime la position des objets (c.-à-d. des phoques dans notre cas) de manière floue. 2.Une branche de régression qui prédit la position exacte des objets précédemment détectés.

Précisément, la branche de densité utilise un ensemble de convolutions pour produire une carte de densité approximative de la position des objets à partir des caractéristiques disponibles en sortie de l'encodeur. Cette carte est supervisée par une fonction de perte pour aligner la densité estimée avec le nombre réel d'objets dans l'image. La seconde branche, quant à elle, prend en entrée les caractéristiques extraites par l'encodeur ainsi que la carte de densité produite par la première branche pour affiner les prédictions, notamment en améliorant la détection des objets qui se chevauchent ou se fondent dans l'environnement. Ceci est mis en œuvre à l'aide d'un *transformer*, nommé *DETE*, supervisé par une fonction de perte qui compare les prédictions aux annotations. L'innovation clé de l'approche est la mise au point du *DETE* (c.-à-d. *Density-Enhanced Transformer Encoder*) qui joue un rôle central dans l'amélioration des performances, voir figure 3.

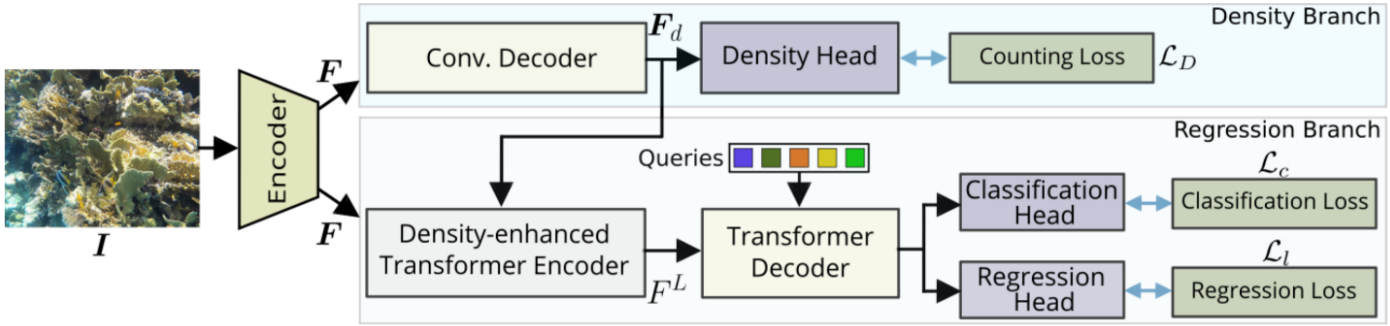

Figure 3 : Architecture de la méthode IOCFormer avec, à gauche, l'entrée est l'extraction des caractéristiques par l'encodeur ResNet-50, en haut à droite la branche de densité et en bas à droite la branche de régression (image issue de l'article original [7]).

3 - Expérimentations et résultats

Avant une utilisation en conditions réelles, les deux outils proposés doivent être amenés à maturité, quand c'est nécessaire, et validés sur un jeu de données préétabli. Pour cela, nous présentons dans cette partie, les bases de données qui ont permis l'évaluation quantitative des modèles, les détails liés à l'apprentissage de l'architecture *IOCFormer* ainsi que les résultats obtenus.

3.1 - Validation des mesures de perturbations

Pour la détection des perturbations humaines, le réseau de neurones *YOLO v8 m* (c.-à-d. la version de taille moyenne) a été utilisé tel qu'il est fourni, pré-entrainé sur la base de données d'images *COCO*. Cette version offre un bon compromis entre performance et rapidité en adéquation avec la puissance de calcul disponible à terre pour analyser les images. Afin de détecter les perturbations, seules les détections relatives aux classes « humain » et « bateau » sont conservées. Aussi, toutes les détections dont la confiance est inférieure à 0.75 (c.-à-d. 75 % de confiance) sont éliminées pour éviter les fausses alarmes. Enfin, les deux stratégies présentées dans la partie 2.1 sont appliquées pour consolider les détections. La première permet de fusionner les détections similaires proches dans le temps pour corriger les pertes de détection. La seconde limite les fausses alarmes en supprimant les détections isolées.

Base de données

Afin de valider les performances de l'approche mise en œuvre, une base de données a été constituée. Elle comprend 200 images réparties équitablement en deux jeux de données : perturbation par un humain et/ou un bateau (possiblement plusieurs), pas de perturbation. Pour chacune des images de la première catégorie, les boites minimales englobant les perturbations ont été manuellement annotées et utilisées comme vérité terrain, voir figure 4. Ces images sont issues de la précédente campagne d'acquisition qui embarquait une caméra similaire avec une résolution de 1920 × 1080 pixels filmant à 10 images par seconde.

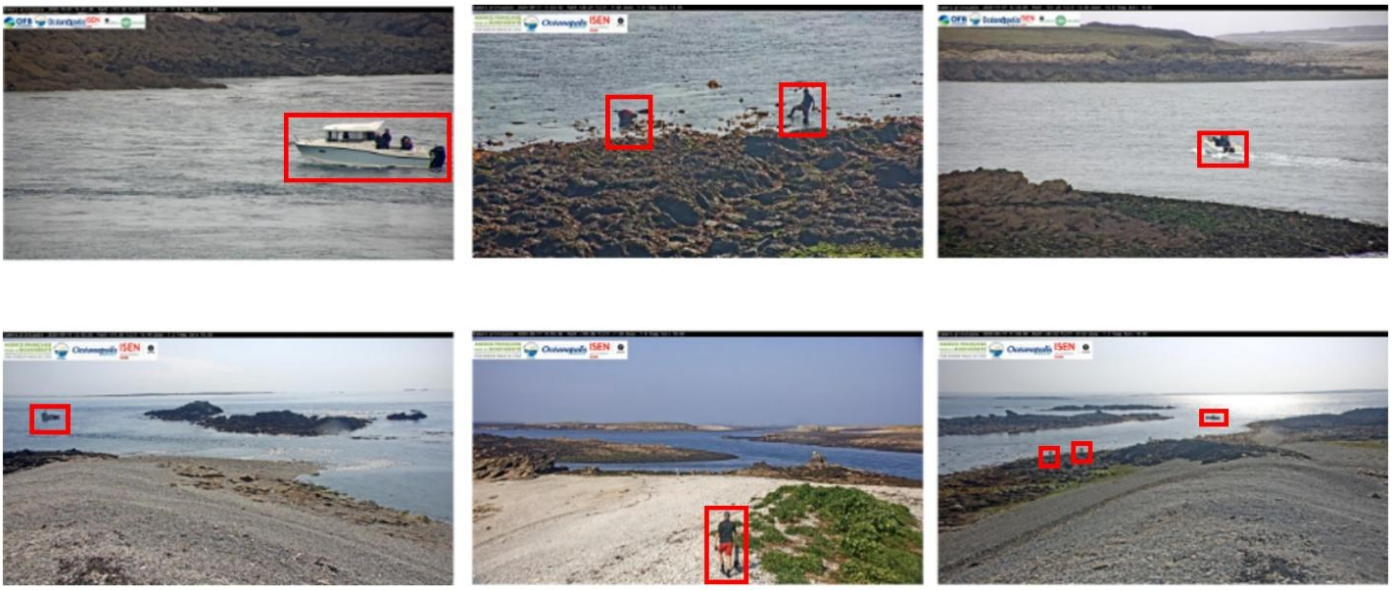

Figure 4 : Exemple de 6 images de la base de données appartenant à la catégorie « perturbation ». En rouge, les boites minimales englobantes des objets (humain ou bateau) utilisées comme vérité terrain.

Résultats

Sur la base de données de test, les résultats obtenus montrent une performance de 97 % avec la répartition suivante entre les deux jeux de données : 94 % de bonnes détections pour la catégorie « perturbation », 100 % de non-détections pour la catégorie « pas de perturbation ». En particulier,

1 erreur a été commise sur la non-détection d'un bateau et 5 sur la non-détection d'humains. Ces erreurs sont dues à une difficulté importante à différencier les pêcheurs à pied de certains rochers lorsque les individus sont très éloignés de la caméra, voir figure 4. Concernant les bateaux, la détection est plus aisée, même s'ils restent parfois difficiles à détecter lorsqu'ils sont au loin, parfois masqués à intervalles réguliers par les vagues au large de l'îlot. Pour permettre une exploitation future par les conservateurs du parc marin, les résultats obtenus sont exportés dans un fichier CSV afin d'archiver les perturbations. Un exemple d'extraction est visible dans le tableau 1. On y retrouve la date et l'heure de la perturbation, son type et sa position dans l'image. Ainsi, il est aisé de croiser les résultats des perturbations et de comptage pour estimer l'impact humain sur la colonie. À titre d'exemple, la figure 4 montre que lorsqu'il y a une perturbation, les phoques quittent généralement l'îlot.

Date et heures	Types d'objets	Position
21/09/2020 13:32:15	Bateau	[10, 12, 512, 312]
21/09/2020 14:52:10	Bateau	[150, 565, 220, 622]
21/09/2020 18:02:10	Humain	[155, 154, 223, 312]
21/09/2020 18:02:10	Humain	[256, 189, 353, 359]
22/09/2020 10:17:15	Bateau	[45, 652, 125, 732]
22/09/2020 12:22:48	Bateau	[456, 785, 589, 936]
22/09/2020 15:02:51	Bateau	[221, 375, 452, 427]

Tableau 1 : Exemple d'export au format CSV des perturbations détectées.

3.2 - Validation du comptage

Le comptage des phoques gris a été réalisé avec l'approche *IOCFormer*, initialement utilisée et évaluée sur une base de données de poissons. Étant donné les différences entre les deux contextes expérimentaux, poissons et phoques, une nouvelle base de données annotée a dû être mise en place pour permettre un réapprentissage du réseau de neurones.

Base de données

La base de données établie a été collectée à partir de 5 vidéos, chacune d'une durée de 60 minutes, capturant divers scénarios. Cette extraction vise à capturer des scènes variées afin d'assurer que le modèle puisse généraliser à différentes conditions de prise de vue ou de météo par exemple. Un échantillon de 1000 images a été sélectionné pour servir de base à l'entraînement, à la validation et au test du modèle. Afin d'établir une vérité terrain nécessaire à mesurer quantitativement la robustesse de la méthode, chacune des 1000 images a été annotée à l'aide des outils *LabelImg* [9] et *RectLabel*. En particulier, les phoques présents dans les images ont été annotés par un point situé au centre de l'individu, comme le requiert la méthode *IOCFormer*. Cette phase a permis d'obtenir un total de 6084 annotations. Afin d'entraîner le modèle, de le valider et finalement de le tester, la base de données a été divisée en 3 jeux de données répartis comme suit : 800 images pour l'entraînement, 100 images pour la validation et 100 images pour le test.

Apprentissage du modèle

L'implémentation de la méthode *IOCFormer* s'appuie sur le code officiel [11] développé avec *PyTorch* pour des GPU NVIDIA. Les spécificités de l'architecture sont les mêmes que pour la méthode originale, à savoir 4 blocs *transformer* pour la partie *DETE* avec 700 *queries*. Les caractéristiques sont extraites avec un encodeur ResNet-50 pré-entraîné sur *ImageNet*. Pour les augmentations de données, nous utilisons un redimensionnement aléatoire et un retournement horizontal. Les images sont recadrées aléatoirement pour obtenir des images d'entrées de taille 256 x 256 pixels. L'entraînement est réalisé sur 2 GPU NVIDIA Quadro RTX 8000 avec des lots de 4 images sur une

durée de 1500 *epochs* avec l'optimiseur Adam [12]. Lors de l'inférence, les images sont divisées en vignettes de taille identique à celles utilisées pendant l'entraînement. Enfin, un seuil de 0.35 est utilisé pour filtrer les prédictions [13].

Les métriques utilisées pour vérifier la qualité de l'apprentissage sont le MAE (*Mean Absolute Error*) et le MSE (*Mean Squared Error*). L'Erreur Absolue Moyenne (MAE) est la moyenne des différences absolues entre les valeurs prédites et la vérité terrain. Elle est également connue sous le nom de norme L1 ou distance de Manhattan :

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Où N est le nombre d'observation, y_i la valeur prédite pour la i^e observation et $\hat{y}_i$ la vérité terrain associée à cette même observation.

Le MSE (Erreur Quadratique Moyenne), quant à lui, est la moyenne des carrés des différences entre les valeurs prédites et la vérité terrain. Elle est également connue sous le nom de norme L2 ou distance euclidienne :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

La figure 5, montre l'évolution des métriques au cours de l'apprentissage. On remarque des résultats cohérents pour les deux métriques observées avec des erreurs qui se stabilisent à partir de 1200 *epochs*. La sélection du meilleur modèle a été réalisée en étudiant les performances du MSE sur le jeu de données de validation à chaque *epoch*.

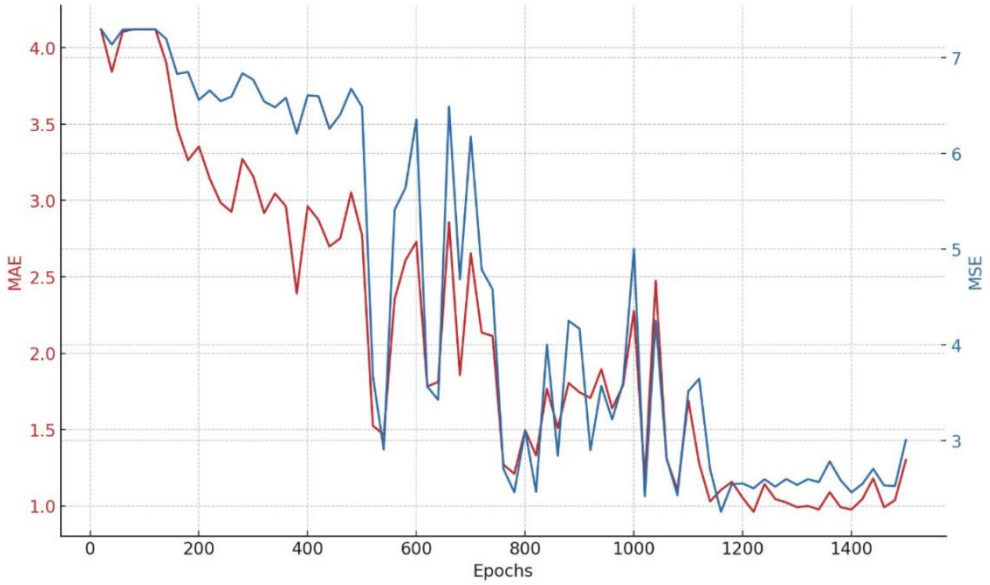

Figure 5 : Visualisation de l'évolution du MAE et du MSE au cours de l'apprentissage. On remarque une stabilisation des métriques après 1200 epochs environ.

Résultats quantitatifs

Les résultats obtenus sur la base de données de test montrent de bons résultats qualitatifs, même sur des images présentant un grand nombre de phoques, voir figure 6. La performance globale du système sur l'ensemble de test est de 91.2 %, en progression par rapport à la méthode précédente qui avait une performance de 87.24 %. On note cependant que les résultats ne sont pas

comparables, car les bases de données utilisées pour l'évaluation sont différentes. Cependant, l'approche proposée montre des perspectives intéressantes pour le comptage des phoques en situation difficile. Finalement, les résultats du comptage sont exportés dans un fichier CSV regroupant les informations visibles dans le tableau 2. On y retrouve l'horodatage du comptage ainsi que le nombre d'individus présent sur l'image.

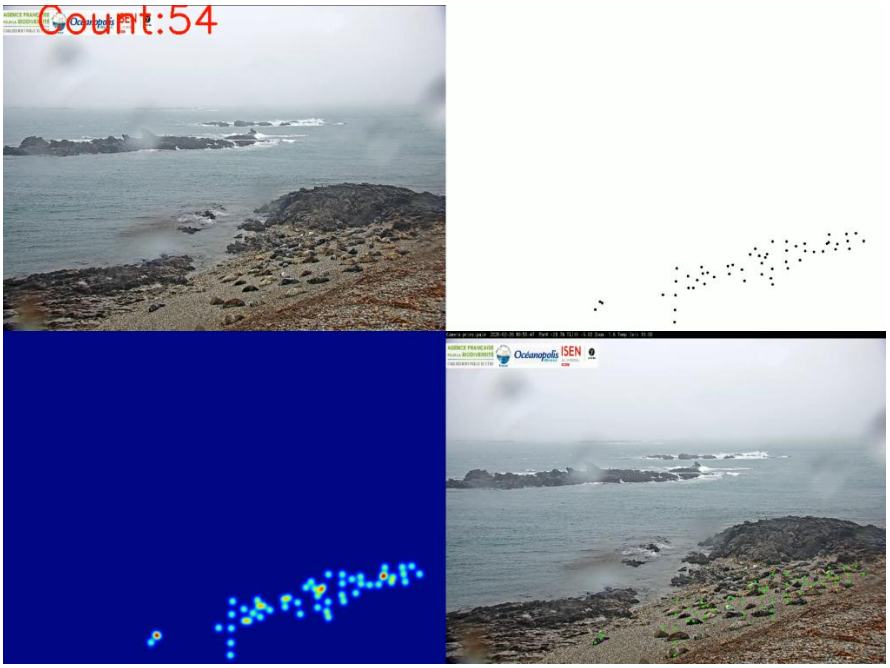

Figure 6 : Résultats de la méthode de comptage proposée avec, l'image originale avec le nombre de phoques (en haut à gauche), l'estimation de la densité (en bas à gauche), l'extraction des individus à partir de la densité (en haut à droite) et la prédiction finale (en bas à droite).

Date et heures	Nombre
22/12/2019 11:00	58
22/12/2019 11:15	55
22/12/2019 11:30	59
22/12/2019 11:45	62
22/12/2019 12:00	58
22/12/2019 12:15	46
22/12/2019 12:30	48

Tableau 2 : Exemple d'export au format CSV du nombre de phoques détectés.

4 - Conclusion

Les résultats obtenus avec les deux approches proposées montrent que l'étude des perturbations ainsi que le comptage sont possibles en utilisant la vision par ordinateur. D'une part, l'architecture *YOLO v8* a permis d'identifier avec robustesse les perturbations humaines qui peuvent survenir autour de l'îlot. D'autre part, l'approche *IOCFormer* a montré sa pertinence pour l'estimation de la densité de la colonie dans des conditions parfois difficiles. Cependant, même si les résultats obtenus répondent bien aux attentes du projet, certains points d'améliorations restent possibles. Le premier élément qui n'a pas été pris en compte dans le projet, malgré sa disponibilité, est l'information issue du capteur infrarouge de la caméra. En effet, cette nouvelle modalité pourrait permettre de robustifier l'estimation de la densité de phoques gris en réalisant une fusion des informations issues des deux types d'images. Cette fusion serait alors réalisée, soit tardivement en fusionnant les estimations de densité, soit précocement lors de l'apprentissage du réseau de neurones qui prendrait alors en entrée les deux modalités d'image plutôt que seulement l'image

dans le domaine visible. D'autre part, il faut noter que derrière les réussites liées aux grandes tâches de vision telles que la reconnaissance de visage, le développement des voitures autonomes ou la génération d'images synthétiques se cache des besoins importants en bases de données annotées, nécessaires aux phases d'entrainement, ainsi qu'en puissance de calcul. Afin de pallier ces contraintes, qui pourraient avoir un impact significatif si le système de décision était amené à devenir autonome sur l'îlot, une approche de frugalité pourrait être envisagée. En particulier, une approche de réduction de la taille, et donc du coût calculatoire, des réseaux de neurones pourrait être envisagée en s'appuyant sur les techniques de distillation de connaissances par exemple [14]. Finalement, la réussite de ce projet montre que les techniques de vision par ordinateur récentes peuvent permettre de répondre, avec une certaine simplicité, à des projets tels que la préservation de la biodiversité, en limitant les opérations humaines qui peuvent perturber les écosystèmes au travers de leurs observations sur le terrain. Ainsi, les données qui seront recueillies en 2025 et 2026 pourront permettre de valider la méthodologie mise en œuvre pour mieux comprendre les impacts humains sur la colonie de phoques gris installée sur l'îlot de Morgol dans l'archipel de Molène au large des côtes Finistérienne.

Remerciement :

Nous remercions Jean-Yves MULOT, Philippe FORJONEL et Léo-Paul PELLETIER du LabISEN pour leur aide tout au long du projet ainsi que l'Office Français de la Biodiversité et le Parc Naturel Marin d'Iroise pour le financement du projet ainsi que pour leur soutien.

Références :

[1] Zizhu FAN, Hong ZHANG, Zheng ZHANG et al. A survey of crowd counting and density estimation based on convolutional neural network. Neurocomputing. 2022. vol. 472, p. 224-251.

[2] Jia DENG, Wei DONG, Richard SOCHER et al. Imagenet: A large-scale hierarchical image database. Conference on Computer Vision and Pattern Recognition (CVPR). 2009. p. 248-255.

[3] Ayoub KARINE, Thibault NAPOLÉON, Jean-Yves MULOT et al. Video seals recognition using transfer learning of convolutional neural network. International Conference on Image Processing Theory, Tools and Applications (IPTA). 2020. p. 1-4.

[4] Karen SIMONYAN and Andrew ZISSERMAN. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[5] Glenn JOCHER, Ayush CHAURASIA and Jing QIU. Ultralytics YOLOv8. https://github.com/ultralytics/ultralytics. 2023.

[6] Tsung-Yi LIN, Michael MAIRE, Serge BELONGIE et al. Microsoft coco: Common objects in context. European Conference on Computer Vision (ECCV). 2014. p. 740-755.

[7] Guolei SUN, Zhaochong AN, Yun LIU et al. Indiscernible object counting in underwater scenes. Computer Vision and Pattern Recognition (CVPR). 2023. p. 13791-13801.

[8] Kaiming HE, Xiangyu ZHANG, Shaoqing REN et al. Deep residual learning for image recognition. Computer Vision and Pattern Recognition (CVPR). 2016. p. 770-778.

[9] https://github.com/HumanSignal/labelImg

[10] https://rectlabel.com/

[11] https://github.com/GuoleiSun/Indiscernible-Object-Counting/tree/main

[12] Diederik P. KINGMA and Jimmy BA. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[13] Dingkang LIANG, Wei XU and Xiang BAI. An end-to-end transformer model for crowd localization. European Conference on Computer Vision (ECCV). 2022. p. 38-54.

[14] Ayoub KARINE, Thibault NAPOLÉON and Maher JRIDI. Channel-spatial knowledge distillation for efficient semantic segmentation. Pattern Recognition Letters, 2024, vol. 180, p. 48-54.

Computer Vision : Introduction au calcul du volume des objets

La Revue 3E.I
Culture Sciences de l'Ingénieur

Simon PLAYE[1]

Édité le 12/03/2025

[1] *Data Scientist chez Sicara (Theodo Data & AI)*

Cette ressource fait partie du N°115 de La Revue 3EI du deuxième trimestre 2025.

Avez-vous déjà travaillé sur le calcul du volume des objets à partir de vidéos en utilisant l'IA ? Cela s'est-il avéré soit peu pertinent, soit extrêmement difficile à réaliser ? Si vous avez répondu « oui » à ces deux questions et que vous êtes passionné par le calcul du volume ou l'IA, cet article est fait pour vous. En utilisant deux vidéos, je vais vous montrer comment calculer le volume des objets.

Les cas d'utilisation concernent principalement la gestion des stocks. Alors que de plus en plus d'entreprises cherchent à réduire leur impact environnemental, un tel outil peut permettre d'améliorer l'efficacité des chaînes d'approvisionnement et optimiser la gestion des stocks. Cet outil peut également être utilisé pour le placement de meubles : design d'intérieur, planification d'événements, construction ou rénovation...

Le calcul du volume des objets soulève plusieurs défis :

- Localiser les objets dans la vidéo
- Définir leurs limites
- Mettre à l'échelle la vidéo pour convertir une distance en pixels, en mètres

Cet article passera en revue toutes les étapes ci-dessous pour expliquer comment effectuer le calcul du volume des objets :

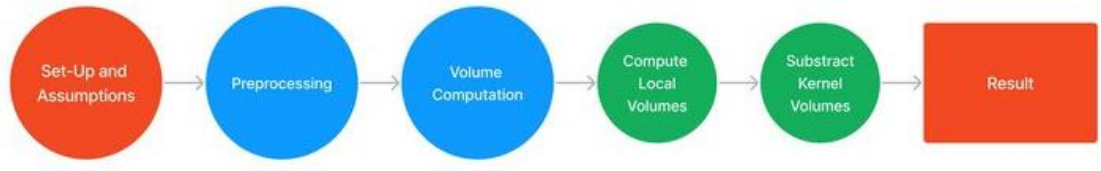

Figure 1 : Étapes du processus de calcul du volume.

1 - Configuration sous-jacente et hypothèses

Pour effectuer le calcul du volume des objets, je m'appuie sur quelques hypothèses :

- L'espace filmé est une pièce fermée.
- Cette pièce est filmée deux fois depuis le même point de vue : une fois sans les objets et une fois avec eux.
- Les vidéos sont identiques, à l'exception des objets.
- Les objets sont placés directement contre un mur, sans aucun espace.

Par souci de simplicité, dans le reste de cet article, la pièce filmée est vide, et les objets sont des boîtes rectangulaires. Par conséquent, le calcul du volume sera effectué sur ces boîtes. Voici un exemple de séquence vidéo respectant ces hypothèses :

Figure 2 : Un exemple de séquence vidéo respectant ces hypothèses. Ici, les objets étudiés sont les deux boîtes.

L'outil le plus important pour effectuer le calcul du volume est la caméra. En effet, une caméra 2D normale n'affiche que des valeurs de couleur pour tous les pixels filmés. Dans cet article, la configuration consiste en une caméra spéciale nommée Intel RealSense D435. Cette caméra est composée de deux caméras infrarouges qui offrent une représentation 3D de ce qui est filmé. Une API Python, pyrealsense2, permet de récupérer, pour chaque pixel, une coordonnée 3D. Ainsi, il est possible d'obtenir la distance en mètres par rapport à la caméra sur l'axe des x (de droite à gauche), sur l'axe des y (de haut en bas), et sur l'axe des z (de près à loin) pour tous les pixels. N'importe quelle autre caméra peut être utilisée tant qu'elle fournit ces coordonnées.

1.1 - Quelques mots sur pyrealsense2

Pyrealsense2 est l'enveloppe Python pour le SDK Intel RealSense 2.0, qui est une bibliothèque pour les caméras Intel RealSense. Le SDK peut normalement être utilisé avec librealsense, un package C++. Selon la page Github de librealsense : « Le SDK permet la diffusion de profondeur et de couleur, et fournit des informations sur la calibration intrinsèque et extrinsèque. La bibliothèque offre également des flux synthétiques (nuage de points, profondeur alignée avec la couleur et vice-versa), ainsi qu'un support intégré pour l'enregistrement et la lecture des sessions de diffusion. » Pour fournir ces services, le SDK utilise un modèle déterministe qui repose sur les entrées des deux caméras infrarouges.

1.2 - Prétraitement

Entrons maintenant dans le processus de calcul du volume. Voici la profondeur des images de ces deux vidéos :

Figure 3a : Image brute avec boîte

Figure 3b : Image brute sans boîte

Ces images illustrent un problème statistique classique : les valeurs aberrantes (outliers) :

- Les zones noires sur les images des vidéos correspondent à des pixels sans coordonnées (la profondeur est définie à 0).
- Les zones colorées correspondent à des coordonnées erronées (la profondeur est de 10 mètres, alors que le mur est à 1,5 mètre de la caméra).

1.3 - Traiter les valeurs aberrantes : bonnes pratiques

Tout d'abord, définissons plus précisément ce qu'est une valeur aberrante. Ici, une valeur aberrante est un pixel dont la valeur de la coordonnée z (correspondant à la profondeur) est égale à 0 ou supérieure à 1,6 mètre. Puisque le mur est à environ 1,5 mètre de la caméra, aucun pixel ne peut avoir une valeur z supérieure à 1,6 mètre.

Cette méthode permet également de sélectionner les pixels ayant des valeurs aberrantes pour les coordonnées x et y. En effet, un contrôle rapide montre que les pixels avec une valeur de coordonnée z égale à 0 ou supérieure à 1,6 ont aussi des valeurs aberrantes pour les coordonnées x et y. En revanche, aucun pixel n'a été trouvé avec des coordonnées x et y aberrantes, mais une valeur de coordonnée z comprise entre 0 et 1,6. Par conséquent, seule la valeur de la coordonnée z définit si un pixel est une valeur aberrante.

Prenons un pixel spécifique d'une image d'une vidéo et imaginons que ce pixel soit une valeur aberrante.

La meilleure manière de remplacer la valeur de la coordonnée z est d'utiliser la valeur de la coordonnée z du pixel non aberrant le plus proche. Ici, le « pixel non aberrant le plus proche » est le pixel ayant la distance euclidienne la plus faible par rapport à notre pixel.

La méthode utilisée pour récupérer les coordonnées x et y est plus compliquée. En nous concentrant d'abord sur x, utilisons la grille ci-dessous pour visualiser la méthode employée. Cette grille donne les coordonnées x des pixels situés sur une portion 7x7 de l'image. Les cellules rouges représentent des valeurs aberrantes, les cellules vertes représentent des pixels non aberrants. La valeur de la coordonnée x est écrite à l'intérieur des pixels non aberrants.

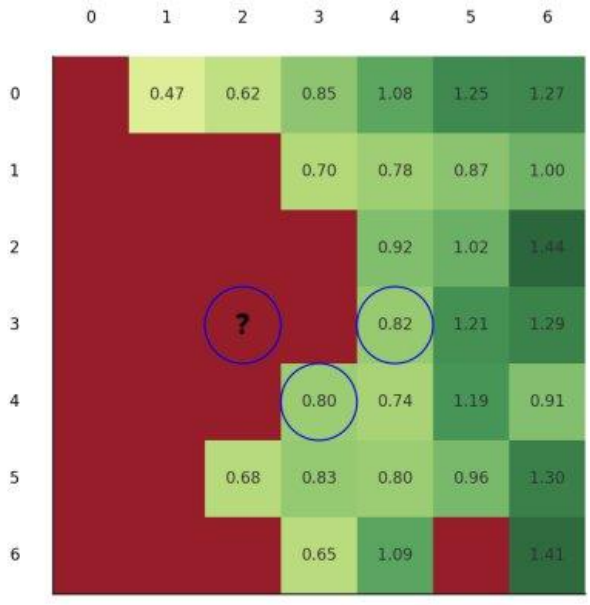

Figure 4 : Sous-ensemble de l'image : les zones rouges sont des valeurs aberrantes, les zones vertes sont des pixels avec des valeurs

Ici, je souhaite calculer la coordonnée x pour l'outlier entouré « ? » situé en (3,2). Pour ce faire, j'ai d'abord sélectionné les deux points les plus proches de cet outlier (entourés dans la grille) qui ne se trouvent pas dans la même colonne. La différence de coordonnée x entre ces deux pixels est 0.82 - 0.80 = 0.02. Ces deux pixels sont distants d'une colonne. Par conséquent, on peut estimer qu'en se déplaçant d'une colonne, la valeur de x change de 0.02. Ainsi, puisque l'outlier est situé à deux colonnes de ce pixel entouré (0.82), sa valeur estimée est 0.78 (= 0.82 - 0.02 * 2).

La même méthode est utilisée pour remplacer les valeurs des coordonnées y pour les outliers.

Ces méthodes de prétraitement permettent d'obtenir une bien meilleure représentation 3D de ce que la caméra a filmé (ici pour la profondeur) :

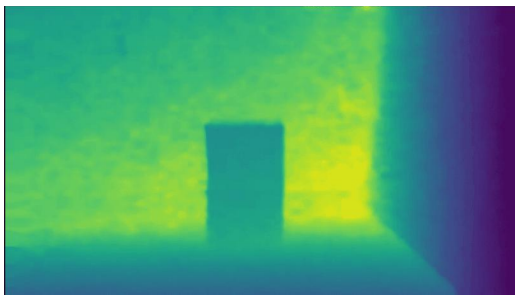

Figure 5a : Image brute avec boîte

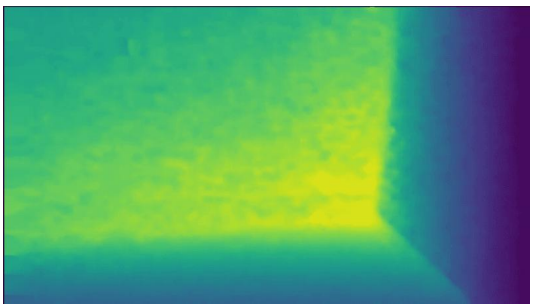

Figure 5b : Image brute sans boîte

2 - Processus de calcul du volume

2.1 - Technique pour effectuer le calcul du volume à l'aide de deux vidéos

En disposant de vidéos avec des coordonnées 3D cohérentes, je peux effectuer le calcul du volume. En lisant cet article, vous vous êtes peut-être demandé : « Pourquoi insister sur l'enregistrement d'une vidéo sans boîtes alors que seul le volume des boîtes est nécessaire ? » L'astuce pour effectuer le calcul du volume des boîtes est de calculer le volume d'une image de la vidéo avec les boîtes, puis de soustraire ce volume du volume de la même image sans boîtes. Cette différence correspond au volume calculé des boîtes.

2.2 - Calcul des volumes locaux à l'aide de noyaux

Comment effectuer le « calcul du volume » sur une seule image ? En regardant l'image ci-dessous, on peut la diviser en petits carrés, appelés noyaux :

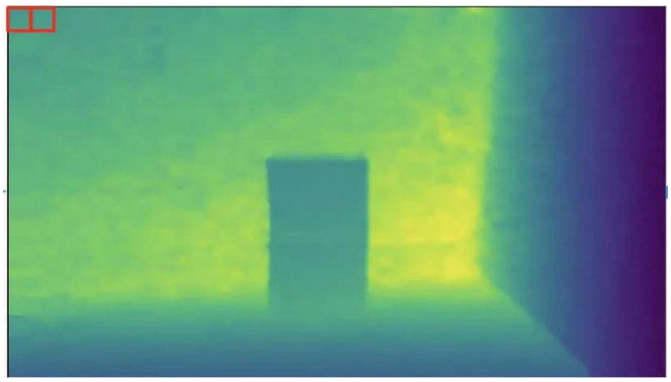

Figure 6 : Image traitée avec boîtes et noyaux

Simplifions ce problème en commençant par essayer de calculer les aires des noyaux, puis leurs volumes correspondants.

Ayant les coordonnées 3D des quatre pixels qui composent un noyau, on peut calculer l'aire du noyau de différentes manières. Pour être plus précis, nous avons approximé un noyau à un parallélogramme. Ensuite, j'ai éliminé les coordonnées z des points et utilisé cette formule :

L'aire d'un parallélogramme peut être calculée en utilisant la formule suivante :

$$A = \left\| \overrightarrow{AB} \wedge \overrightarrow{AD} \right\|$$

Formule vectorielle de l'aire d'un parallélogramme

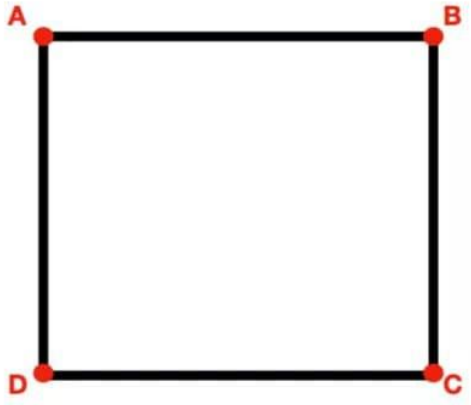

Figure 7 : Un noyau

Cette formule ignore un point (ici C), et les quatre points sont considérés comme étant dans le même plan 2D, car les coordonnées z de chaque point sont ignorées.

En se concentrant sur un seul noyau, on peut considérer ce noyau comme la base d'un prisme rectangulaire, l'autre base étant située aux coordonnées z=0 et étant identique. Une manière de l'imaginer est de penser que la caméra est située sur un plan perpendiculaire à l'axe des z. Par conséquent, le prisme rectangulaire aura une base sur ce plan et l'autre sera le noyau sur votre image. La hauteur de ce prisme rectangulaire sera la moyenne des coordonnées z de tous les points à l'intérieur du noyau, comme les points A, B, C, D, E, F et G dans l'image ci-dessous :

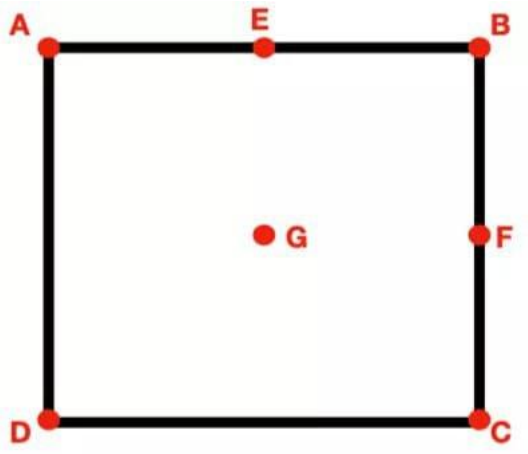

Figure 8 : E, F et G sont également utilisés pour calculer la hauteur moyenne

Ainsi, en calculant l'aire d'un noyau et en obtenant la hauteur du prisme rectangulaire correspondant, on peut effectuer le calcul du volume pour un noyau. En couvrant une image avec des noyaux et en additionnant leurs volumes correspondants, il est possible d'obtenir le volume total de l'image. Voici une représentation visuelle du volume : un pixel de chaque image représente le volume calculé avec un noyau de 2x2 pixels :

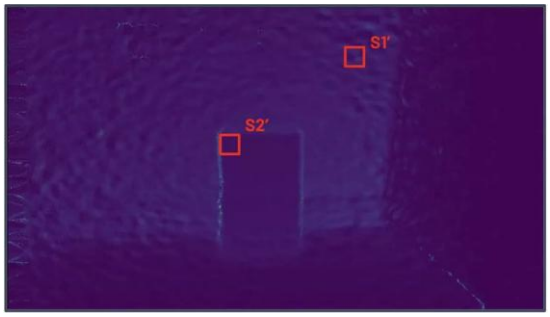

Figure 9a : Calcul du volume- Image brute avec boîte

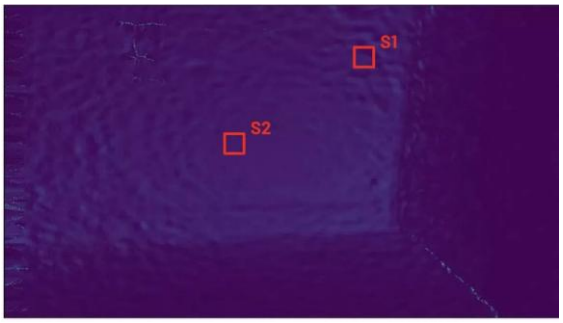

Figure 9b : Calcul du volume - Image brute sans boîte

Dans les images ci-dessus, j'ai ajouté deux carrés : S1 et S2 sur l'image sans boîtes, et S1' et S2' sur l'image avec boîtes. En comparant S1 et S1', les volumes devraient être approximativement les mêmes, car ils couvrent la même surface et ont la même profondeur. Ainsi, la différence entre S1 et S1' devrait être proche de 0. Cependant, en comparant S2 et S2', le volume de S2 devrait être plus grand que celui de S2' car S2' se trouve sur une boîte. Ainsi, même si S2 et S2' ont la même surface, la profondeur moyenne des pixels dans S2' est inférieure à celle de S2. La différence entre S2 et S2' correspond au volume occupé par la boîte. Par conséquent, en recouvrant les deux images de carrés et en calculant la différence de volume pour chaque carré correspondant, on peut obtenir le volume des boîtes.

3 - Conclusion

Pour conclure, le calcul du volume en utilisant cette technique donne de très bons résultats pour estimer le volume des objets. Pour deux boîtes, l'erreur absolue entre le volume calculé et le volume réel est inférieure à 1 %. Pour une et trois boîtes, cette erreur est inférieure à 3 %. Malgré les nombreuses hypothèses sous-jacentes concernant la pièce et les objets, cet algorithme se révèle efficace pour le calcul du volume des objets. De plus, il fournit une base pour des applications intéressantes en vision par ordinateur, telles que le calcul du volume de différents types d'objets en utilisant des algorithmes de détection d'objets.

4 - Références :

[1]: https://data-ai.theodo.com/en/technical-blog/mastering-volume-computation-of-objects-from-videos

IA Générative : Apprenez à contrôler la génération d'images avec Stable Diffusion

La Revue 3EI
Culture Sciences de l'Ingénieur

Édité le 11/03/2025

école normale supérieure paris-saclay

Simon PLAYE[1]

[1] *Data Scientist chez Sicara (Theodo Data & AI)*

Cette ressource fait partie du N°115 de La Revue 3EI du deuxième trimestre 2025.

1 - Introduction

Avez-vous déjà utilisé ChatGPT pour créer des images et avez-vous été déçu par les résultats ? Si vous avez ressenti cette frustration ou si vous êtes simplement intéressé par des outils de génération d'images plus efficaces, Stable Diffusion pourrait être la solution idéale.

Avec la popularité de ChatGPT, nombreux sont ceux qui sont curieux de découvrir ce que l'IA générative peut accomplir. Elle ne se limite pas à la création de texte ; cette technologie peut également générer des images, des vidéos et même de la musique. Pourtant, il est parfois difficile d'obtenir les images que vous souhaitez simplement à partir d'une invite textuelle.

Cet article se penche sur Stable Diffusion, un outil conçu spécifiquement pour générer des images. Ici, je vous montrerai comment utiliser une API pour contrôler la création d'images avec Stable Diffusion, en couvrant des options tant pour les non-développeurs que pour les développeurs. Les méthodes abordées dans cet article incluent :

- Diriger le processus de diffusion vers une image spécifique (IP Adapter).
- Maintenir certaines caractéristiques dans une image via le processus de diffusion (ControlNet).
- Extraire des caractéristiques à partir d'une image initiale (Image-to-Image).
- Former un modèle spécifique avec un ensemble limité d'images pour des ajustements ciblés (LORA, Model Fine-Tuning).
- Ajouter de nouveaux poids aux couches de croisement d'attention pour modifier les caractéristiques de l'image (LORA).

2 - Qu'est-ce que Stable Diffusion ?

Stable Diffusion est un modèle de diffusion spécifiquement conçu pour la génération d'images. Il commence avec un motif initial de bruit aléatoire et affine systématiquement ce bruit ou le "débruite" pour produire des images qui ressemblent de près à des photos réelles. Le modèle guide cette transformation en appliquant certaines conditions (par exemple une invite textuelle) durant le processus de débruitage.

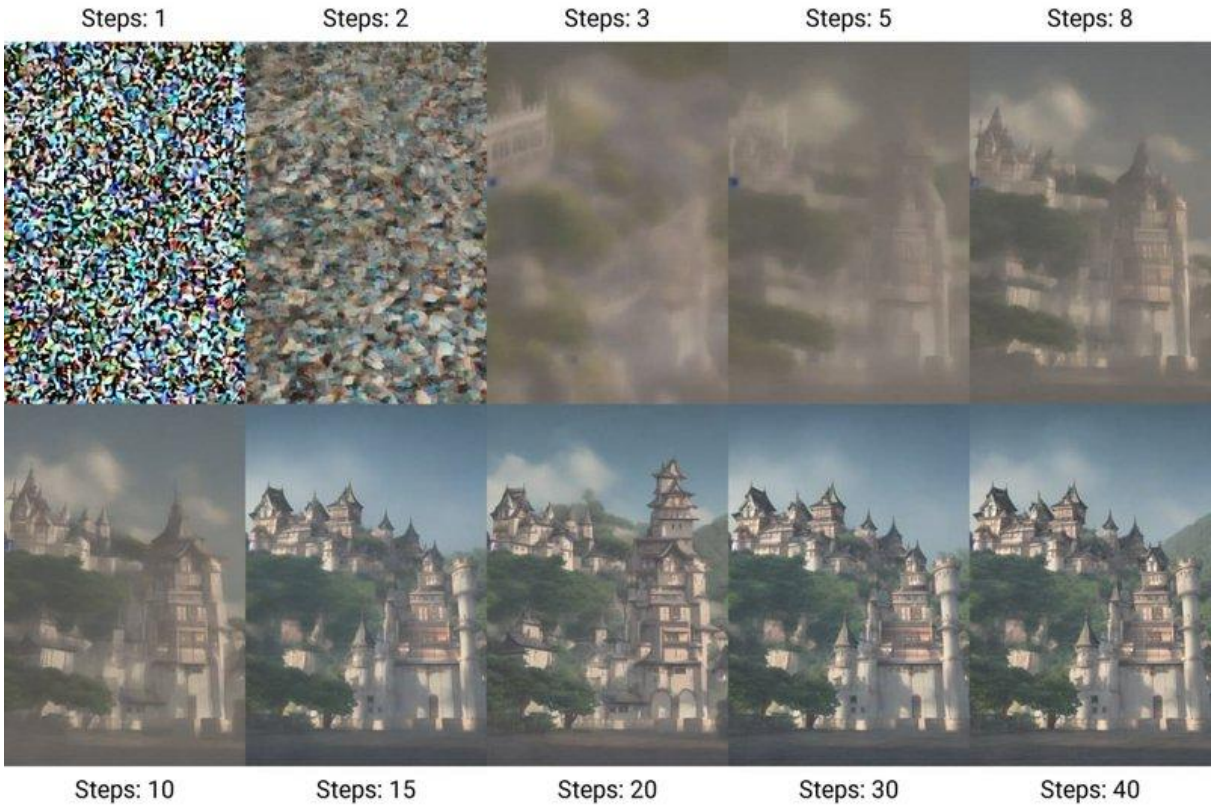

Figure 1 : Les étapes de débruitage
source: https://en.wikipedia.org/wiki/Diffusion_model#/media/File:X-Y_plot_of_algorithmically-generated_AI_art_of_European-style_castle_in_Japan_demonstrating_DDIM_diffusion_steps.png

Contrairement aux méthodes traditionnelles qui se contentent de prompts textuels, Stable Diffusion offre des contrôles plus flexibles et sophistiqués :

- Conditionnement basé sur l'image : Grâce à des fonctionnalités comme l'IP-Adapter et ControlNet, Stable Diffusion peut utiliser une image existante pour orienter la génération, permettant des modifications ou des améliorations basées sur cette image.
- Transformation Image-à-Image : Cette fonctionnalité permet au modèle de partir non pas de zéro mais d'une image existante, qu'il transforme ensuite en une nouvelle création.
- Contrôle de style : Les utilisateurs peuvent choisir des modèles générant des images dans des styles artistiques spécifiques, ou même entraîner leurs modèles pour produire des styles personnalisés.
- Conditionnement hybride : Il permet de combiner des invites textuelles, des images et différents modèles pour conditionner la génération, offrant un contrôle sans précédent sur la sortie.

Pour expérimenter avec ces diverses techniques de contrôle de la génération d'images, ModelsLab propose une API Stable Diffusion pratique. Elle est disponible sous deux formats :

- **Version Playground** : Une interface conviviale conçue pour l'expérimentation et l'exploration sans nécessiter de connaissances techniques étendues.
- **API pour Développeurs** : Offre un contrôle et une personnalisation plus détaillés, adaptés aux développeurs souhaitant intégrer ces capacités dans leurs applications.

Dans les sections suivantes, je vais approfondir chaque technique, en montrant comment utiliser la plateforme ModelsLab pour réaliser votre vision créative.

2.1 - Stable Diffusion text-to-image

Avant de plonger dans les divers outils permettant de contrôler la génération d'images, commençons par une introduction à l'API Stable Diffusion text-to-image. Cette API, semblable à Dall-E, facilite la génération d'images à partir de prompts textuels.

Pour les développeurs, il existe deux points de terminaison principaux : **text2img** et **realtime-stable-diffusion**. Le point de terminaison **text2img** est le choix principal pour la génération d'images, accessible via le playground et l'API pour développeurs. Il permet aux utilisateurs de créer des images avec des modèles de diffusion entraînés par la communauté, disponibles sous les versions **Stable Diffusion** et **Stable Diffusion XL**. La version XL offre des images plus précises mais nécessite plus de temps pour la génération.

Dans le playground, les utilisateurs peuvent personnaliser leur génération d'images en définissant différents paramètres tels que :

- **Negative Prompt** : Spécifiez ce que vous ne voulez pas voir apparaître dans l'image.
- **Guidance Scale** : Définissez l'importance de l'influence du prompt sur le débruitage.
- **Steps** : Déterminez le nombre d'étapes de génération, ce qui affecte le détail de l'image.

L'API développeur offre également des options supplémentaires, telles que :

- **enhance_style** : Choisir un style spécifique pour l'image.
- **highres_fix** : Créer des images en haute résolution.

De plus, le point de terminaison **realtime-stable-diffusion** offre moins d'options de personnalisation et ne permet pas de choisir un modèle de diffusion, mais il est plus rapide pour générer des images.

Voici un exemple rapide de fonctionnement de **text2img**.

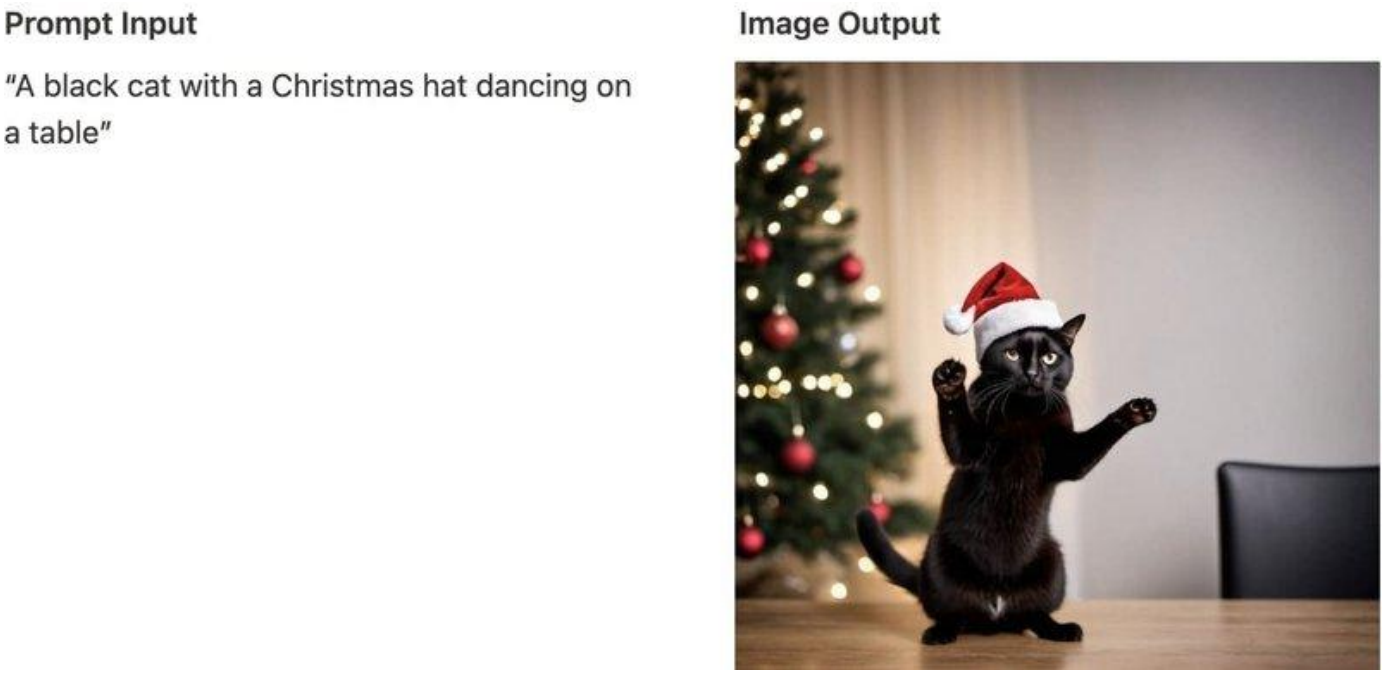

Figure 2 : Un chat dansant sur une table avec un chapeau de noël

3 - Contrôle de la génération d'images avec Stable Diffusion

Dans cette section, nous explorerons trois outils avancés fournis par Stable Diffusion qui offrent des façons alternatives de guider la génération d'images, au-delà du prompt textuel conventionnel :

- **IP-Adapter**
- **ControlNet**
- **Image-to-Image**

3.1 - IP-Adapter

Imaginez que vous puissiez guider la génération d'images non pas en donnant des instructions textuelles, mais en utilisant une image cible à la place. C'est exactement ce que fait **IP-Adapter**. Il permet de diriger le processus de génération avec une image, tout comme un prompt textuel. Cependant, l'IP-Adapter doit être utilisé avec un prompt textuel (ou une image, comme nous le verrons ci-dessous).

Cependant, l'IP-Adapter est uniquement disponible via l'API développeur sur le point de terminaison **img2img**. Dans ce paramètre, vous pouvez ajuster des paramètres tels que :

- **ip_adapter_id**, qui définit comment l'image est encodée,
- **ip_adapter_scale**, qui affecte l'impact de l'image IP-Adapter sur la réduction du bruit,
- **ip_adapter_image**, qui est l'URL de l'image IP-Adapter.

Adapter avec un prompt textuel

Figure 3 : Create a rose and cute castle with IP-Adapter (Créez un château rose et mignon avec IP-Adapter)

3.2 - ControlNet

ControlNet fonctionne de manière similaire à l'IP-Adapter, mais avec une approche plus nuancée. Contrairement à l'IP-Adapter, qui influence la génération d'images en fonction de l'image entière, ControlNet cible des caractéristiques spécifiques dans l'image pour guider le processus de génération vers ces détails.

Il existe plusieurs types de ControlNet qui permettent d'extraire des caractéristiques spécifiques d'une image :

- **softedge :** trouve les contours dans les images
- **canny :** délimite avec précision les frontières dans des environnements contrôlés
- **openpose :** détecte les points clés du corps humain, des mains, du visage et des pieds
- etc.

Vous pouvez également intégrer plusieurs modèles ControlNet en énumérant plusieurs paramètres controlnet_model, séparés par des virgules, comme "canny, softedge".

L'adaptabilité de ControlNet signifie qu'il peut fonctionner en parallèle avec des prompts textuels et des images IP-Adapter. Il est facile à utiliser pour les non-développeurs dans l'interface, tandis que les développeurs peuvent ajuster des paramètres supplémentaires, tels que :

- **controlnet_type :** Cela définit l'un des différents types acceptés de modèles ControlNet, comme canny, depth, hed, etc. Vous pouvez trouver la liste complète des types de modèles disponibles ici.
- **controlnet_model :** Cela spécifie le modèle ControlNet spécifique utilisé, qui peut être un modèle par défaut ou un modèle communautaire. Dans le cas d'un modèle par défaut, controlnet_model correspond directement à controlnet_type.
- **controlnet_conditioning_scale :** Cela détermine dans quelle mesure ControlNet influence le processus de réduction du bruit.
- **control_image (optionnel) :** Il s'agit de l'image à partir de laquelle les caractéristiques seront extraites. Si ce paramètre n'est pas spécifié et qu'une image initiale est fournie, cette image sera utilisée.

ControlNet sans prompt textuel et sans image IP-Adapter.

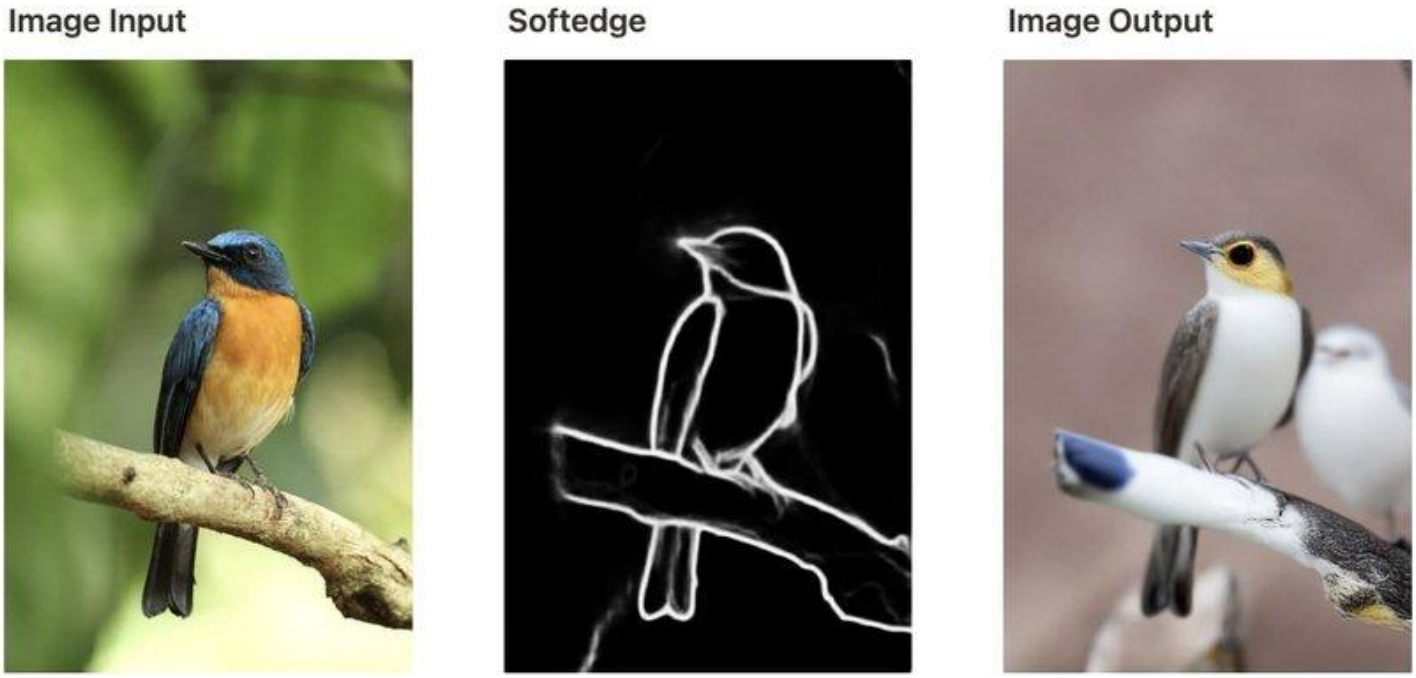

Figure 4 : Transforming a bird picture with a softedge ControlNet (Transformation d'une image d'oiseau avec un softedge ControlNet)

ControlNet avec prompt textuel et sans image IP-Adapter

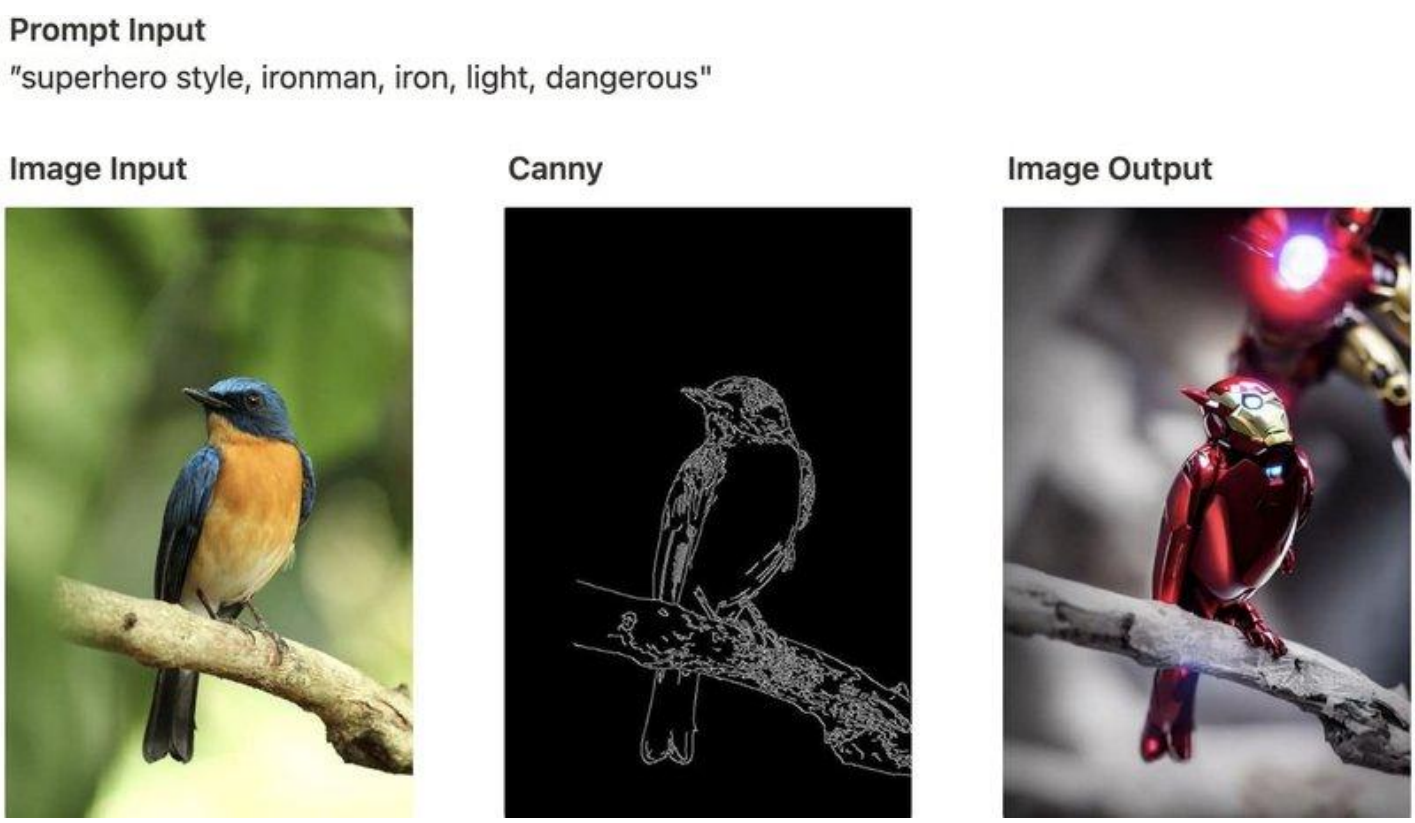

Figure 5 : Transforming a bird picture with a prompt and a canny ControlNet (Transformer une photo d'oiseau grâce à un prompt et un canny ControlNet)

3.3 - Génération d'Image-à-Image

Enfin, au lieu d'orienter la génération d'image en conditionnant la sortie, pourquoi ne pas commencer la génération à partir d'une image qui partage des caractéristiques avec votre image finale ? C'est ainsi que fonctionne l'API Stable Diffusion img2img. Elle ne part pas du bruit aléatoire complet, mais ajoute du bruit à l'image initiale. Elle est conçue pour capturer les caractéristiques générales de l'image initiale, comme sa couleur et sa composition.

La génération Image-à-Image peut être combinée avec un prompt textuel, ControlNet et/ou IP-Adapter.

Parce que la génération img2img fonctionne de manière similaire à la génération texte-à-image, les deux API partagent presque les mêmes fonctionnalités. Les modèles communautaires et realtime-stable-diffusion peuvent être appliqués à l'un ou l'autre type de génération.

Génération d'image avec un prompt textuel et sans IP-Adapter

Prompt Input

"plane, flying, blue, sky, sun"

Image Input

Image Output

Figure 6 : A plane inspired from a bird picture (Un avion inspiré d'une photo d'oiseau)

Génération d'image sans commande textuelle et sans IP-Adapter

Image Input

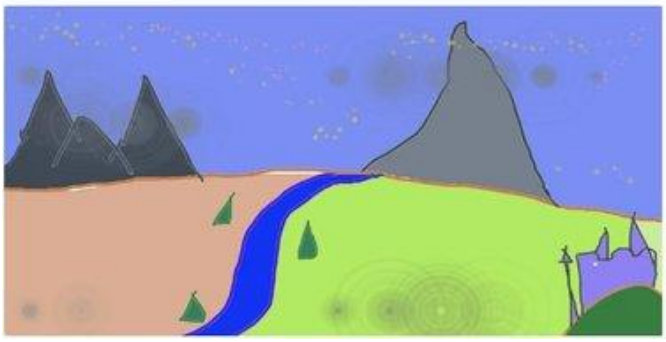

Image Output

Figure 7 : Image d'un paysage de « fantasy » inspiré d'un dessin, source : https://tinyurl.com/2eyf5ky5

3.4 - Conclusion sur le Contrôle de la Génération d'Images

En conclusion, les générateurs d'images traditionnels comme Dall-E s'appuient uniquement sur des prompts textuels pour le contrôle, mais **Stable Diffusion** introduit des outils puissants tels que **IP-Adapter**, **ControlNet** et **Image-à-Image** pour diversifier et améliorer la génération d'images. Les utilisateurs peuvent utiliser ces outils indépendamment ou en combinaison, offrant une grande flexibilité et créativité pour générer des images, avec ou sans prompts textuels.

Sélection et entraînement de modèles spécifiques

Jusqu'à présent, nous avons parlé de l'ajout de conditions pour contrôler la génération d'images. Mais pourquoi ne pas modifier totalement le processus de génération ? Derrière la génération d'images se cache un modèle de diffusion. Cependant, ces modèles peuvent être modifiés pour générer des images plus spécifiques. Ici, je vais présenter deux façons d'améliorer le processus de génération : en sélectionnant ou en affinant un modèle de diffusion ou un **LoRA**. Ces deux approches peuvent être combinées avec les outils présentés ci-dessus.

Modèles de Diffusion

Comme mentionné précédemment, **Stable Diffusion** fonctionne avec un modèle de diffusion. Un modèle de diffusion est un modèle formé pour générer des images. Pour ce faire, ce modèle est entraîné sur un ensemble d'images qu'il cherche à reproduire. Par exemple, si vous voulez un modèle qui génère des images de chiens, vous devrez entraîner un modèle de diffusion en utilisant de nombreuses images de chiens.

La puissance de **Stable Diffusion** réside dans le fait qu'il permet un accès facile aux modèles affinés par la communauté. L'**affinage** est un processus de réentraînement partiel du modèle standard de **Stable Diffusion** sur votre propre ensemble d'images. Au lieu d'utiliser le modèle standard qui génère des images classiques, vous pouvez choisir un modèle qui génère des images de pixels, par exemple.

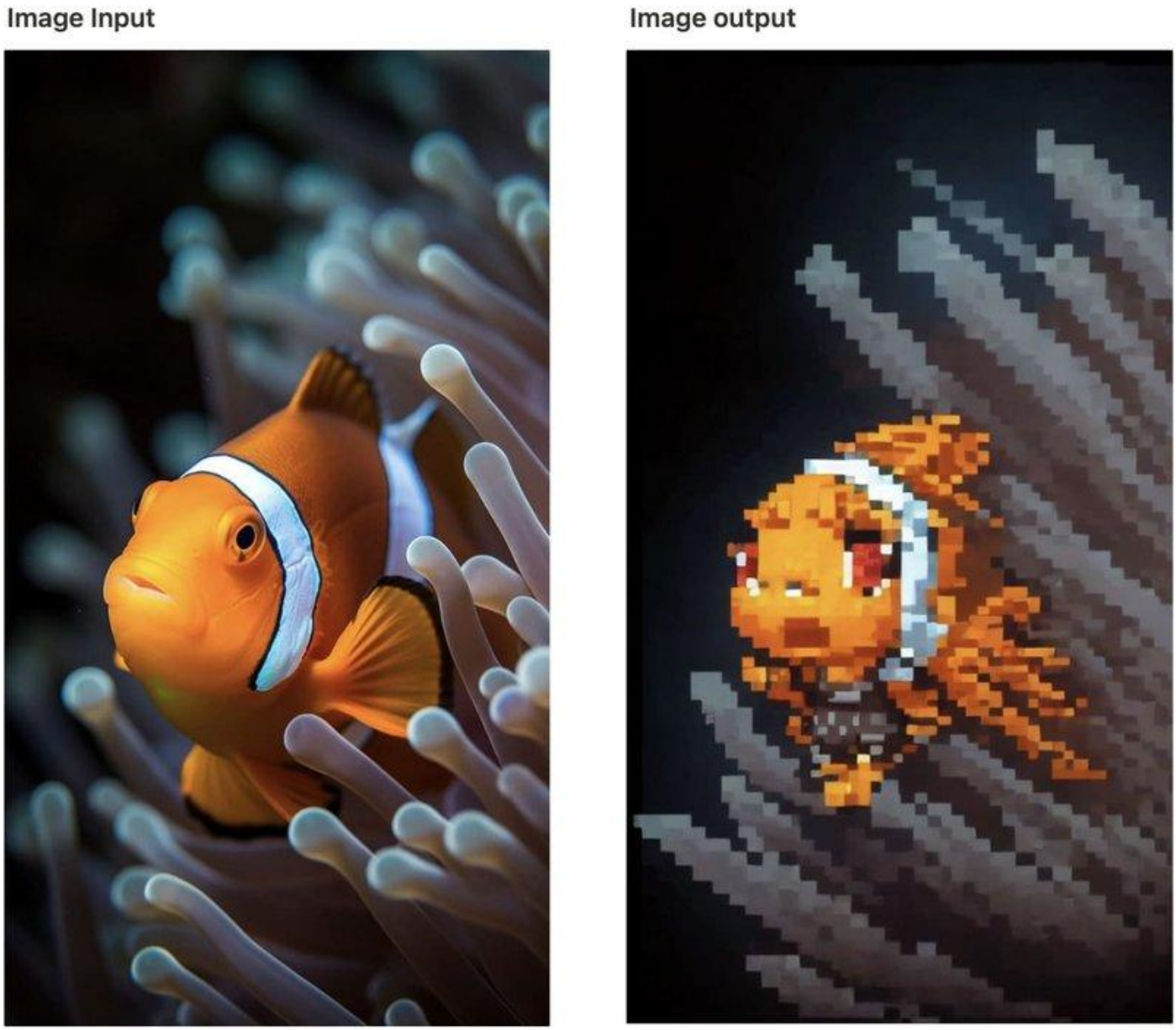

Figure 8 : A pixelated fish using model Chibi Pixel Art Style (Un poisson pixelisé utilisant le modèle Chibi Pixel Art Style)

Ou pour générer des images de cartoons :

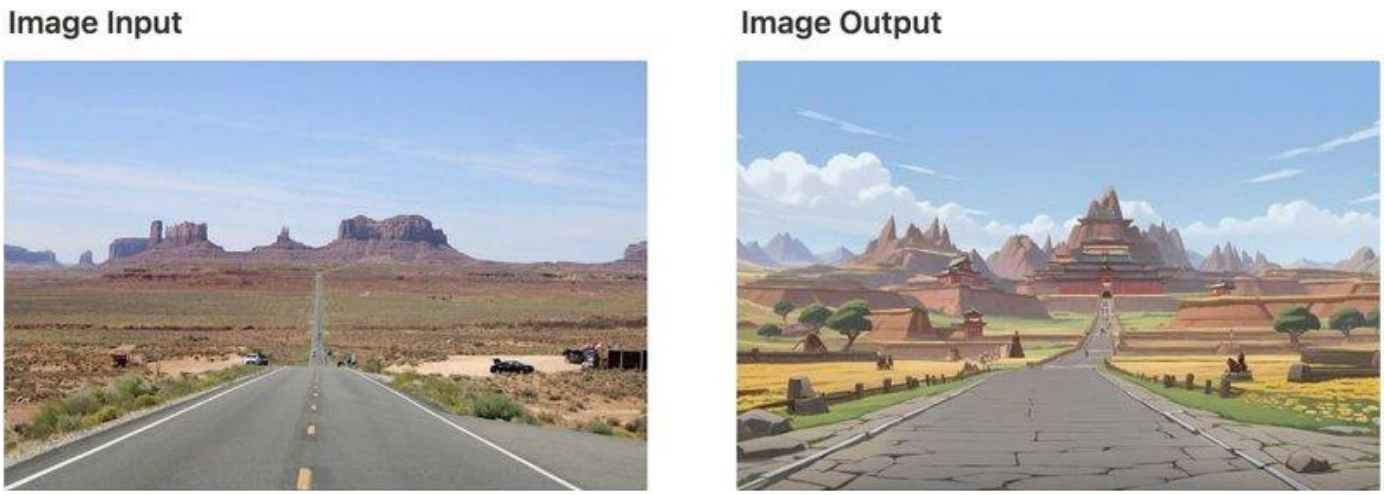

Figure 9 : Cartoon Image using model Cartoon Backgrounds (Image de bande dessinée utilisant un modèle Arrière-plans de bande dessinée)

La sélection de modèles est disponible à la fois pour les utilisateurs non-développeurs et via l'API développeur.

Enfin, si aucun modèle ne correspond à vos attentes, vous pouvez également affiner votre propre modèle. Selon la documentation, l'affinage nécessite seulement 7 à 8 images. Cependant, cela n'est disponible que via l'API développeur.

4 - LoRA

Enfin, **LoRA** est le dernier outil que je vais vous présenter pour avoir plus de contrôle sur la génération d'images. Les **LoRA** sont des versions compactes des modèles **Stable Diffusion**, généralement 10 à 100 fois plus petites. Les LoRAs ajoutent des modifications par-dessus le modèle de diffusion. Elles affinent les modèles standards, modifiant subtilement des styles spécifiques ou l'apparence générale des images générées. L'avantage des LoRAs réside dans leurs faibles besoins en calcul et leurs temps d'entraînement rapides.

Tout comme pour les modèles de diffusion, vous pouvez facilement utiliser des LoRAs affinées par la communauté. Mais, contrairement aux modèles de diffusion et de manière similaire à **ControlNet**, vous pouvez utiliser plusieurs LoRAs à la fois en les séparant par des virgules. La sélection de LoRAs est disponible tant pour les utilisateurs non-développeurs que via l'API développeur.

De plus, entraîner votre propre LoRA est simple à l'aide de l'API développeur, et cela nécessite typiquement seulement 7 à 8 images.

Vous trouverez deux nouveaux paramètres pour les LoRAs dans l'interface utilisateur et l'API développeur :

- **lora_model** : Spécifie quel modèle LoRA utiliser.
- **lora_strength** : Détermine l'ampleur de l'impact du LoRA pendant le processus de réduction du bruit.

Voici un exemple d'un LoRA intitulé "Princess" :

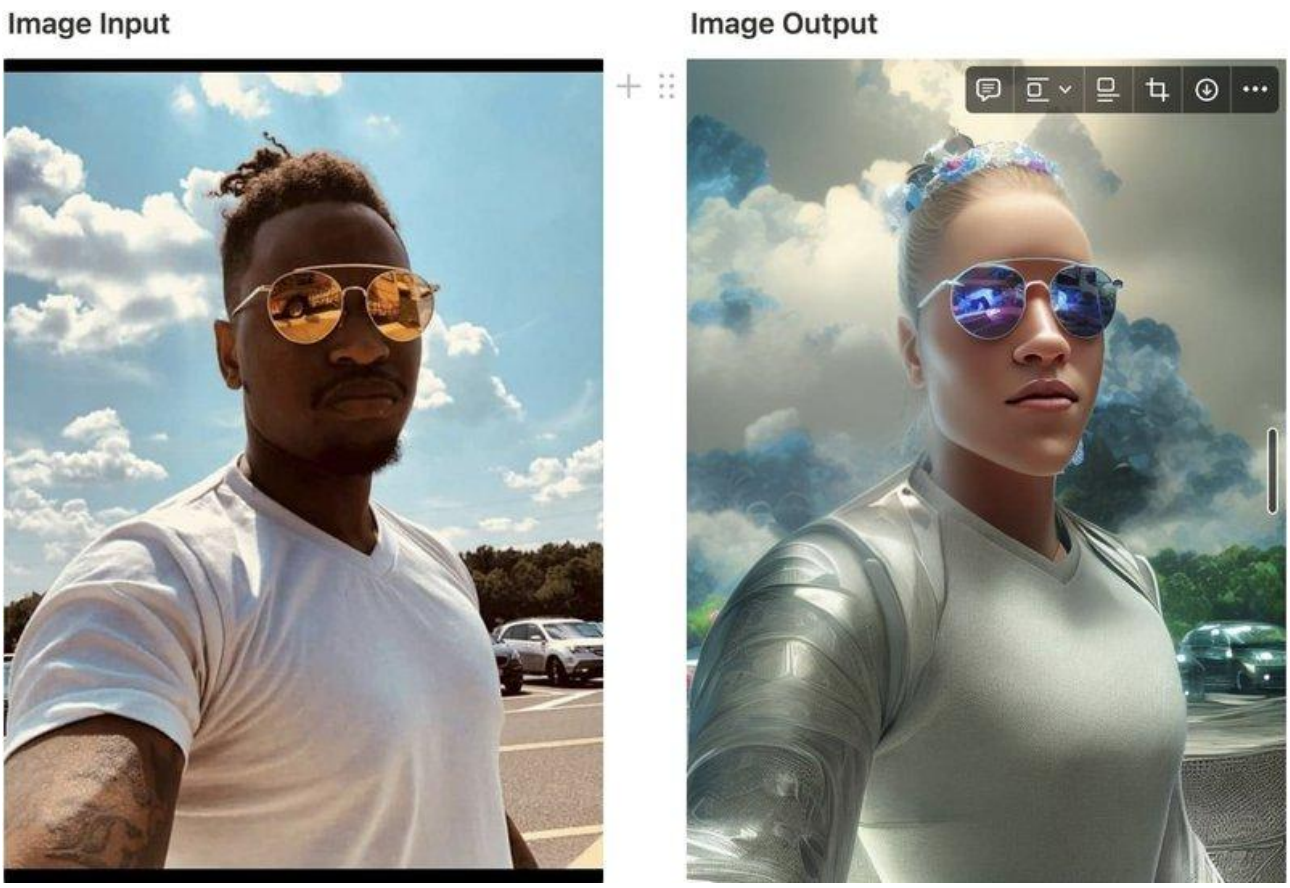

Figure 10 : LoRA : Princess

Et ici une illustration pour enfant :

Figure 11 : LoRA : Illustration pour enfant

5 - Conclusion

En résumé, Stable Diffusion offre plusieurs façons de contrôler la génération d'images au-delà de l'utilisation simple de prompts.

En guise de conclusion, l'intégration des différents composants de Stable Diffusion dans un flux de travail cohérent peut être complexe. Pour ceux qui cherchent à créer des images détaillées avec un meilleur contrôle dans un format facile à utiliser, ComfyUI offre une solution idéale.

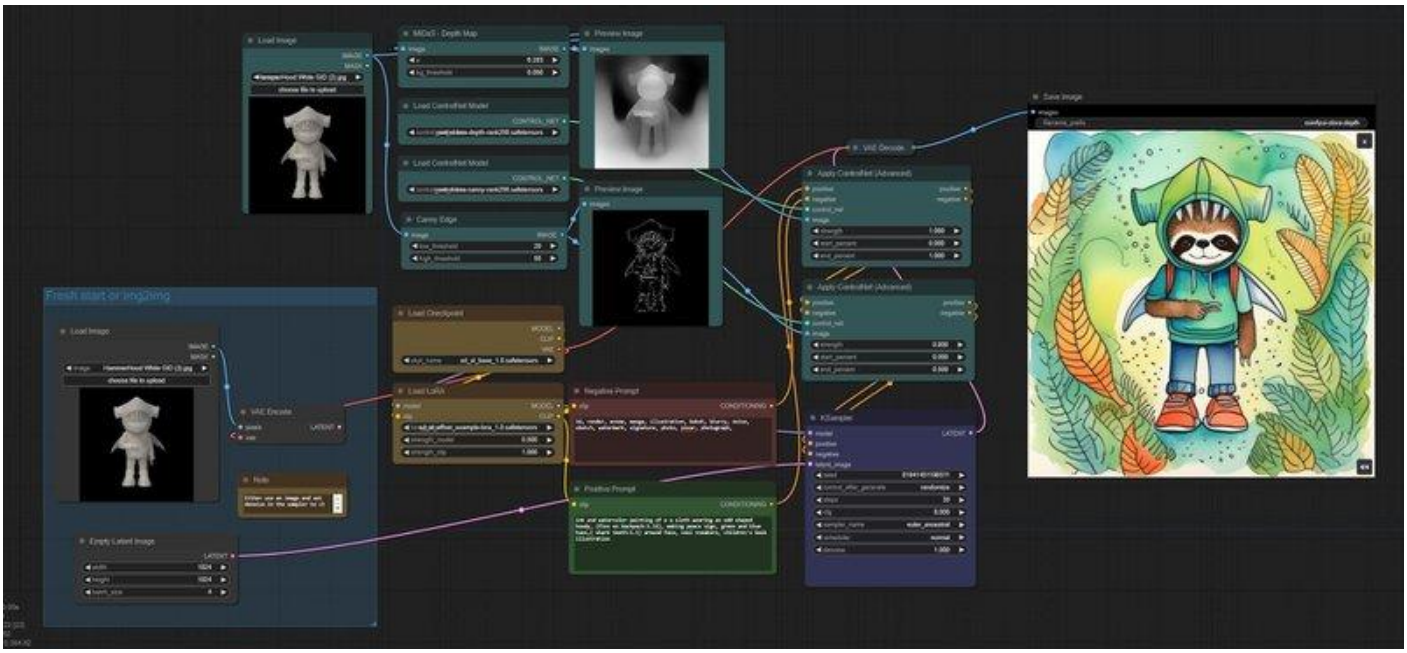

Figure 12 : Création d'une image détaillée à l'aide de ComfyUI et satble Diffsuion
source : https://tinyurl.com/2wuudfat

ComfyUI vous permet de concevoir des pipelines pour la génération d'images en utilisant **Stable Diffusion**. Pour plus de détails sur ses fonctionnalités, vous pouvez consulter la documentation. Avec les outils appropriés et les connaissances nécessaires, le potentiel de l'IA générative est illimité — profitez de votre créativité !

6 - Références :

[1]: https://data-ai.theodo.com/en/technical-blog/generative-ai-image-generation-stable-diffusion