

I. Introduction

II. Méthodes classiques de détection de contours

II. Détection de contours avec des méthodes de Machine Learning et Deep Le

III. Conclusion

IV. Références

Détection de contours

Rémi Colin, Karima Ghamnia, Cassandra Mussard, Mickaël Song

23/06/2023



1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

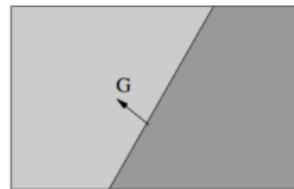
4 III. Conclusion

5 IV. Références

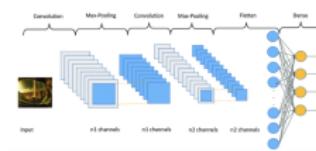
Introduction

- * Détection de contours : une étape fondamentale en traitement d'images dans plusieurs applications : Segmentation, compression, ...
- * Un contour est une rupture (discontinuité) d'intensité du niveau de gris dans l'image suivant une/plusieurs direction(s).

Dans ce projet :



(a) Méthodes classiques



(b) ML

Figure – Différentes méthodes utilisées

1 I. Introduction

2 II. Méthodes classiques de détection de contours

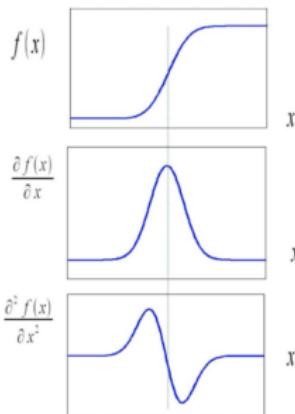
Méthodes basées sur les dérivées

Reconstruction des formes par la transformée de Hough

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

4 III. Conclusion

5 IV. Références



Pour calculer les approximations des dérivées numériquement, on fait des convolutions entre l'image et des masques.

Filtres

- * Les poids capturent les changements d'intensité dans les directions horizontales et verticales.
- * Les valeurs positives et négatives des poids permettent de différencier les transitions ascendantes et descendantes d'intensité.
- * Reposent tous sur une recherche d'un extremum de la dérivée première.
- * Généralement composés de 2 masques pour les contours horizontaux et verticaux .

$$\begin{array}{|c|c|} \hline 1 & \\ \hline -1 & \\ \hline \end{array}$$

gradient

$$\begin{array}{|c|c|} \hline 1 & \\ \hline & -1 \\ \hline \end{array}$$

Roberts

$$\begin{array}{|c|c|c|} \hline 1 & & -1 \\ \hline 1 & & -1 \\ \hline 1 & & -1 \\ \hline \end{array}$$

Prewitt

$$\begin{array}{|c|c|c|} \hline 1 & & -1 \\ \hline 2 & & -2 \\ \hline 1 & & -1 \\ \hline \end{array}$$

Sobel

Figure – Différents filtres

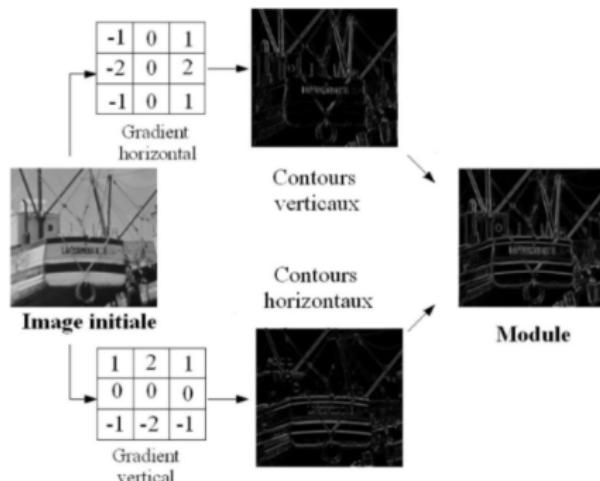


Figure – Exemple du filtre de Sobel

On peut voir que l'un est la transposée de l'autre.

Filtre Gaussien (Etapes de pré-traitements)

Pour enlever les bruits et les détails indésirables :

- * Effet de lissage doux : On applique une fenêtre de pondération gaussienne sur les pixels environnents lors de la convolution (on donne plus d'importance aux pixels proche du pixel central et inversement). Cela crée un effet de flou progressif et doux.
- * Contrôle du flou : Contrôle le niveau de flou en ajustant le paramètre de l'écart-type (sigma) de la distribution gaussienne (sigma élevé = beaucoup de flou et inversement). Cela permet une flexibilité dans le choix du niveau de flou souhaité en fonction des besoins spécifiques de l'application.

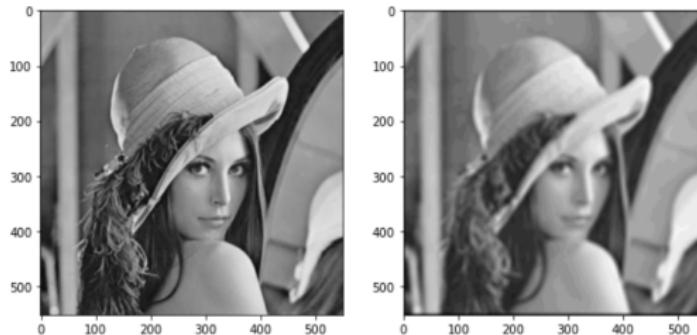


Figure – Résultat du filtre gaussien sur Lenna

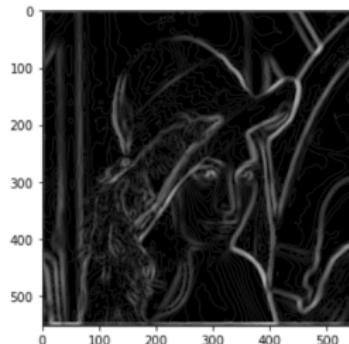
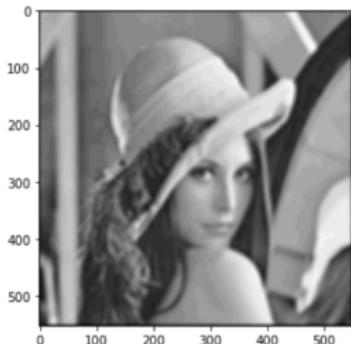


Figure – Résultat du filtre Roberts sur Lenna

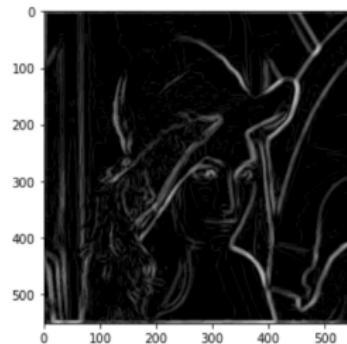
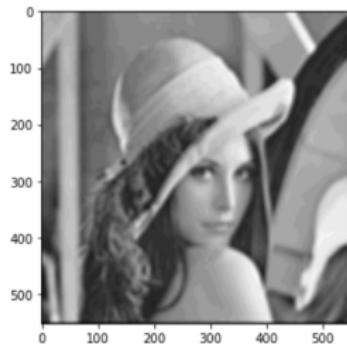


Figure – Résultat du filtre Prewitt sur Lenna

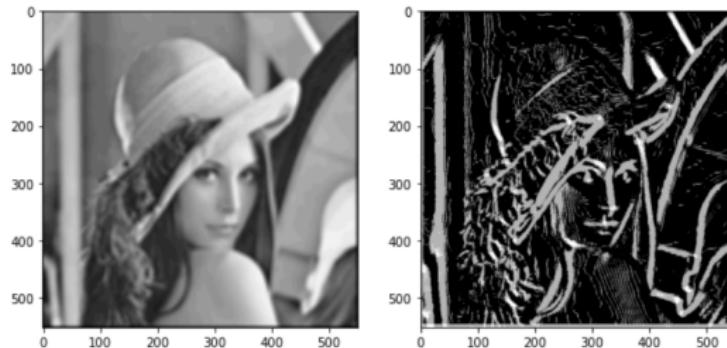


Figure – Résultat du filtre Sobel sur Lenna

On peut remarquer beaucoup plus de détails.

Canny

- * Un algorithme de détection de contour développé par John Canny en 1986.
- * C'est la méthode la plus utilisée et précise dans la détection de contours.
- * Elle est basée sur le calcul du gradient avec un filtre de Sobel.

Etapes pour Canny

- * Lissage de l'image à l'aide d'un filtre gaussien pour réduire le bruit et les détails indésirables.
- * Calcul des gradients en utilisant le filtre de Sobel.
- * Suppression des non-maximaux en ne conservant que les valeurs de gradient maximales dans chaque direction locale.
- * Seuillage à double seuil pour classer les pixels de contour en pixels forts, faibles ou non liés au contour.
- * Seuillage par hystérésis pour former des contours complets en considérant les pixels de contour faible qui sont connectés à des pixels de contour fort comme contour fort.

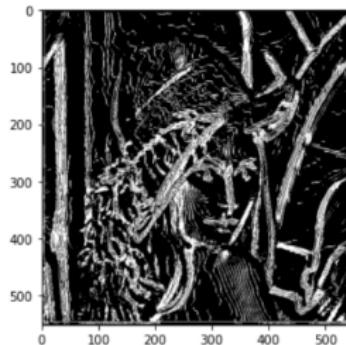
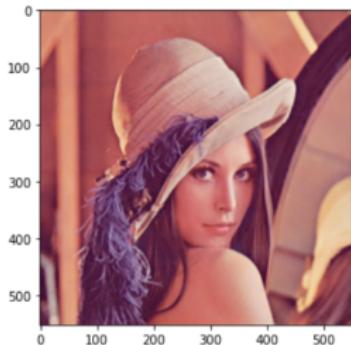


Figure – Résultat de Canny sur Lenna

Opérateur Laplacien

- * méthode du gradient : contour dans une image = maximum du gradient dans la direction orthogonale au contour
- * méthode du laplacien : contour dans une image = passage par zéro de la dérivée seconde

La dérivée seconde déterminée par le calcul du Laplacien qui peut être appliquée à une image par l'intermédiaire d'un masque :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacien d'une image :

$$I = \nabla^2 I(x, y) = \left(\frac{\delta^2 I(x, y)}{\delta^2 x}, \frac{\delta^2 I(x, y)}{\delta^2 y} \right)^t$$

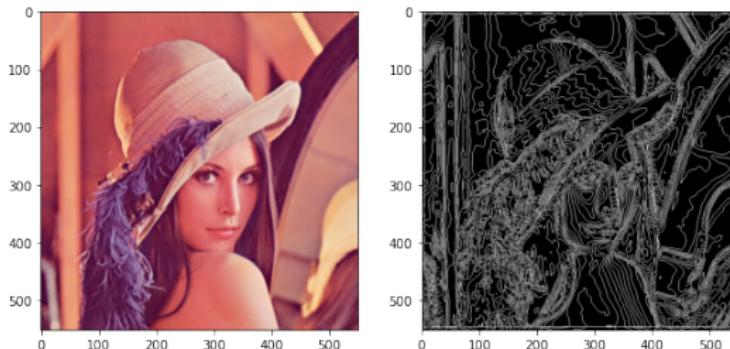


Figure – Résultat des contours par calcul de la dérivée seconde sur Lenna

1 I. Introduction

2 II. Méthodes classiques de détection de contours

Méthodes basées sur les dérivées

Reconstruction des formes par la transformée de Hough

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

4 III. Conclusion

5 IV. Références

Canny : image de contours.

Coordonnées polaires d'une droite :

$$\rho = x \cos(\theta) + y \sin(\theta)$$

Espace de Hough = plan (ρ, θ) .

Point/pixel dans l'image de contours -> courbe sinusoïdale dans l'espace de Hough.

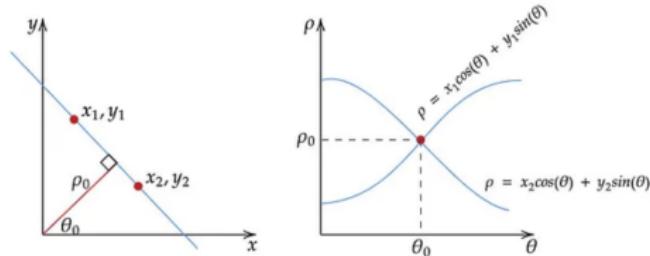


Figure – Une ligne et sa représentation dans l'espace de Hough

Détection des lignes : paires (ρ, θ) qui ont un nombre d'intersections \geq seuil.

Seuil : sensibilité de la détection de lignes.

Seuil élevé : seules les lignes avec une forte accumulation de pixels seront détectées => détection plus précise mais avec moins de lignes détectées.

Seuil bas : plus de détections mais moins fiables ou du bruit.

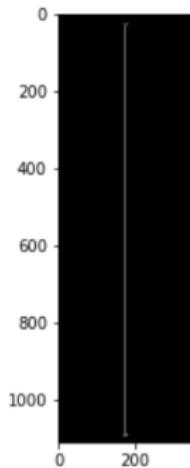
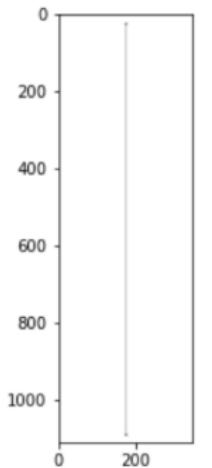


Figure – Image d'un trait à gauche et son image contour correspondant (par filtre Canny) à droite

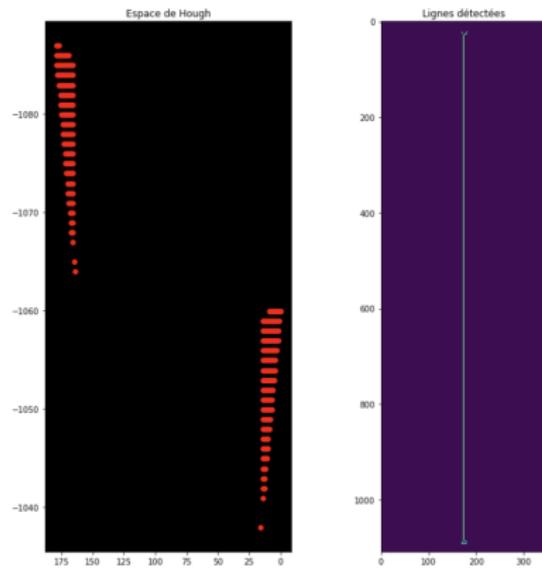


Figure – Représentation dans l'espace de Hough de l'image précédente à gauche et le trait détecté après l'algorithme de transformée de Hough à droite

Autres algorithmes

D'autres transformées de Hough existent pour d'autres formes géométriques.

- * exemple : cercle d'équation $(x - a)^2 + (y - b)^2 = r^2$
-> espace de Hough (a,b,r).
- * application à Lenna

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

Dataset BIPED

Random Forest

CNN

Résultats du CNN

Fine Tuning

Résultats avec du Fine-Tuning

4 III. Conclusion

5 IV. Références

Présentation du dataset

- * Images des rues de Barcelone avec des labels binaires (1 si le pixel est un contour 0 sinon).
- * Dataset composé de 200 images d'entraînement et 50 images de test.
- * Images de taille 1280*720 redimensionnées en 200*200.
- * Images normalisées entre 0 et 1 avec un seul canal.

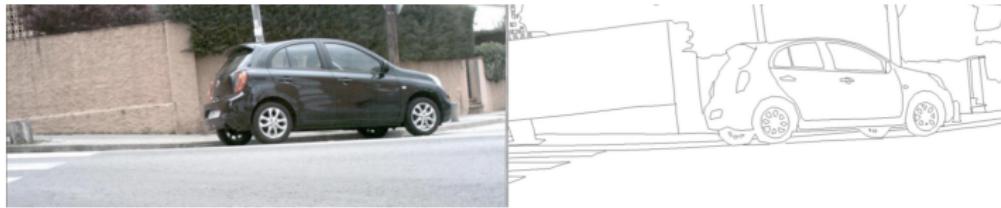


Figure – Image d'entraînement et le label associé

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

Dataset BIPED

Random Forest

CNN

Résultats du CNN

Fine Tuning

Résultats avec du Fine-Tuning

4 III. Conclusion

5 IV. Références

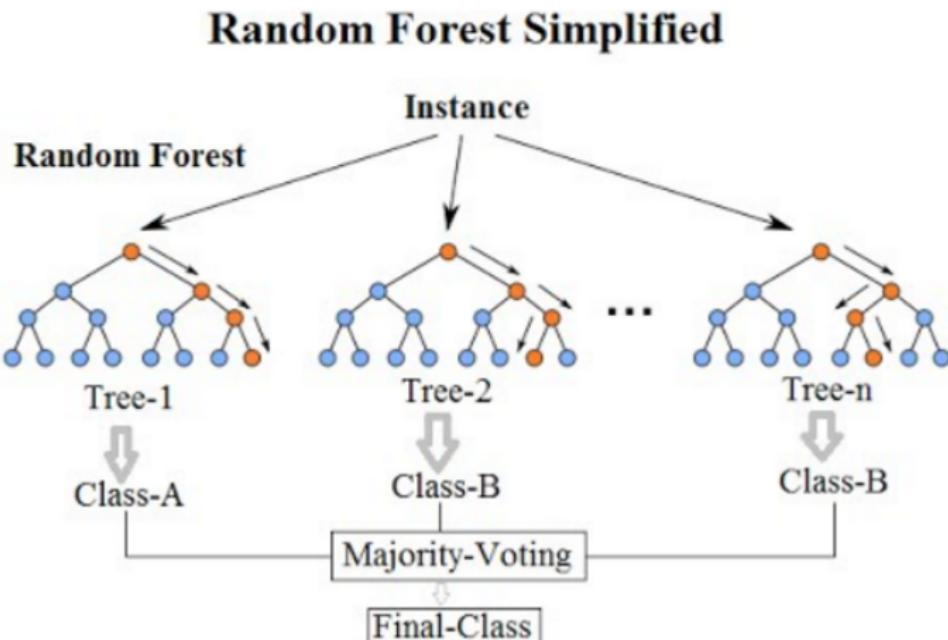
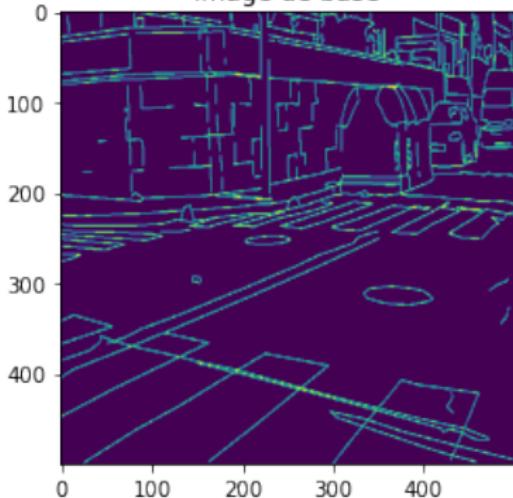


Figure Random Forest

Différents algorithmes

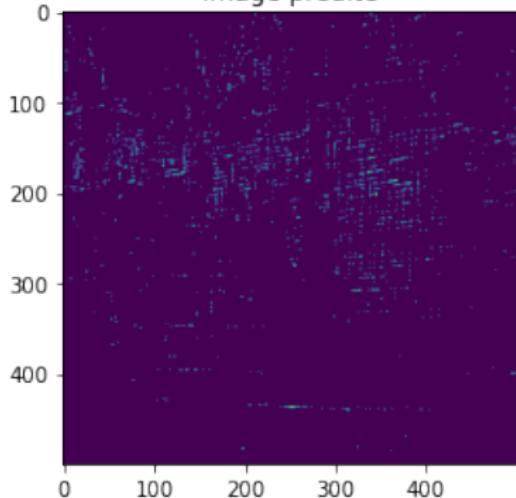
- * Random Forest Classifieur avec 20 arbres, toutes les images d'entraînement (accuracy de 93%).
- * Random Forest Classifieur avec un nombre d'arbre variant de 1 à 15 (accuracy de 93%).
- * Random Forest Regressor avec 50 arbres et 50 images d'entraînement et une pour le test (accuracy de 82%).

Image de base



(a) Image de base

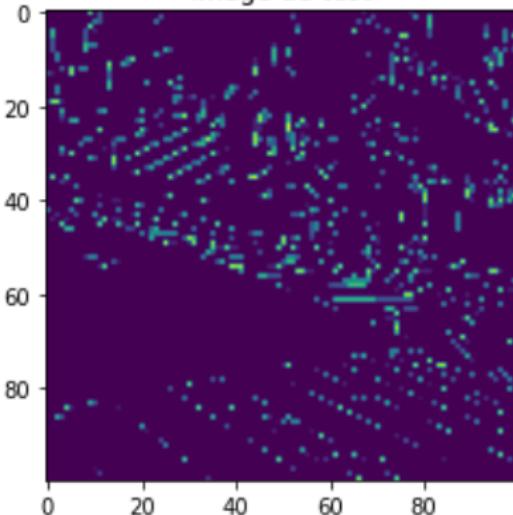
Image prédictive



(b) Image prédictive

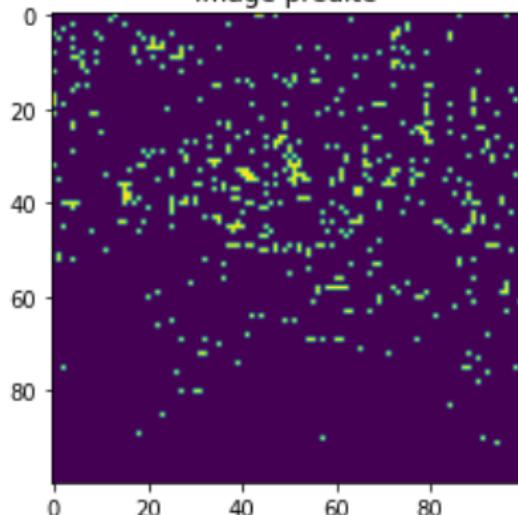
Figure – Résultat du Random Forest Classifieur avec 20 arbres

Image de test



(a) Image de base

Image prédictive



(b) Image prédictive

Figure – Résultat du Random Forest Regressor avec 50 arbres

Pistes d'amélioration

- * Un arbre par pixel.
- * Améliorer les données d'entrée pour le Random Forest Regressor.

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

Dataset BIPED

Random Forest

CNN

Résultats du CNN

Fine Tuning

Résultats avec du Fine-Tuning

4 III. Conclusion

5 IV. Références

Architecture d'autoencodeur

- * Permet de reconstruire une image de même taille que celle en entrée.
- * Partie encodeur qui permet de construire de nouvelles représentations dites encodées. (3 couches de convolution).
- * Partie décodeur reçoit ces représentations et les traitent (3 couches de déconvolution).

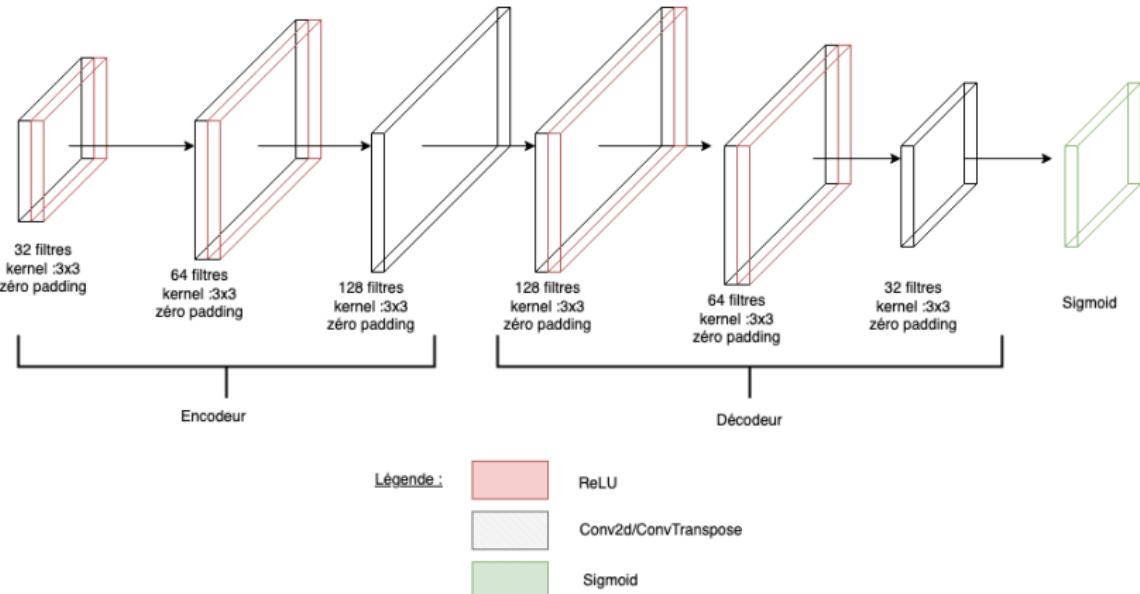


Figure – Architecture autoencodeur utilisée

Hyperparamètres

- * Classification binaire.
- * Batch de 8 pour l'entraînement et batch de 1 pour le test.
- * Algorithme SGD avec moment avec un pas de 0.1 pour la descente de gradient.
- * BCELoss comme fonction de coût.
- * Seuillage en sortie de la loss pour créer les contours.

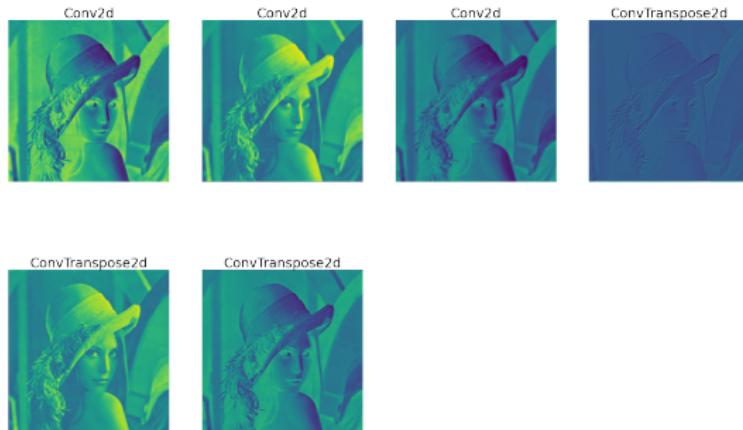
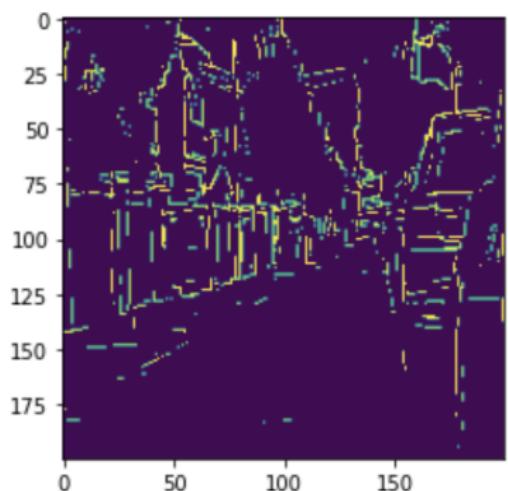
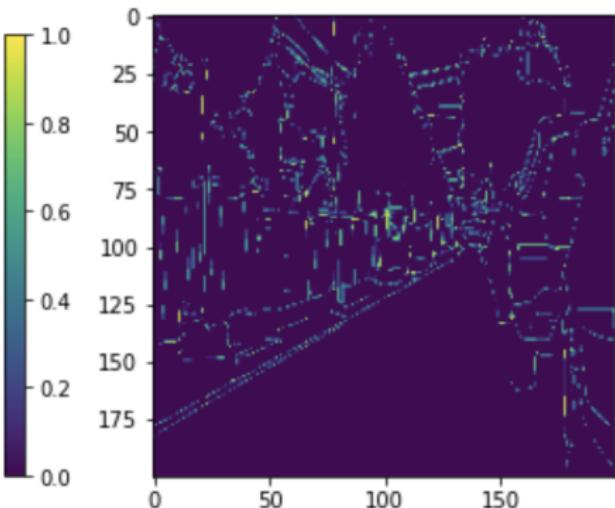


Figure – Première feature maps de chaque couche de convolution du réseau



(a) Image prédictive dataset test



(b) Label test

Figure – Résultat sur une image du dataset test

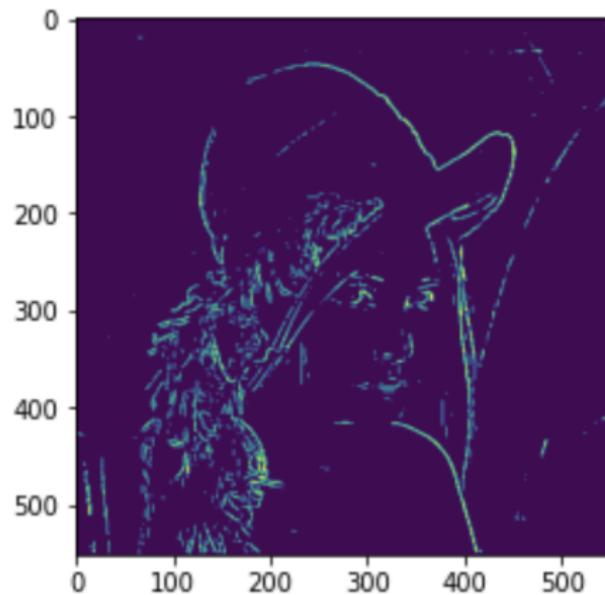


Figure – Résultat sur une Lena (une image différente de celle du dataset)

Pistes d'amélioration

- * Modification de l'architecture
- * Ajout de plus de couches dans le réseau autoencodeur.
- * Réaliser de l'augmentation de données.

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

Dataset BIPED

Random Forest

CNN

Résultats du CNN

Fine Tuning

Résultats avec du Fine-Tuning

4 III. Conclusion

5 IV. Références

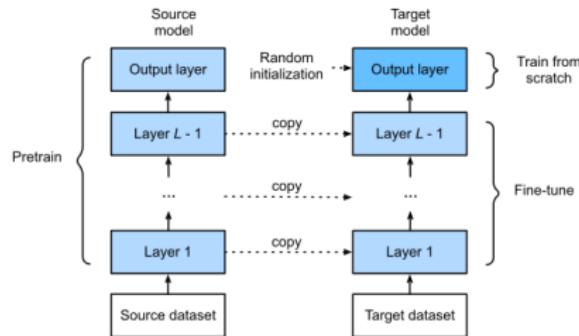


Figure – Fine Tuning

Avantages du fine-tuning

- * Permet de gagner du temps et des ressources.
- * Améliore les performances d'un modèle en utilisant les poids pré-entraîné.
- * On ne reprend pas l'apprentissage de zéro.

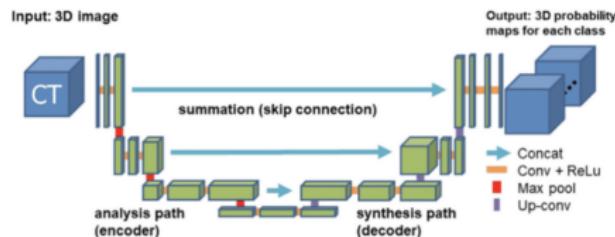


Figure – Architecture U-Net utilisée pour le fine tuning

Modification de l'architecture

- * Modification de la première couche (couche Conv2d avec un seul canal en entrée).
- * La modification de la première couche entraîne la modification de la dernière (couche Conv2d avec un seul canal en sortie).

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

Dataset BIPED

Random Forest

CNN

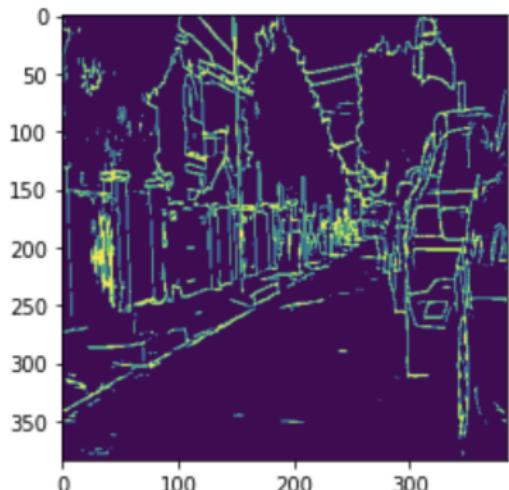
Résultats du CNN

Fine Tuning

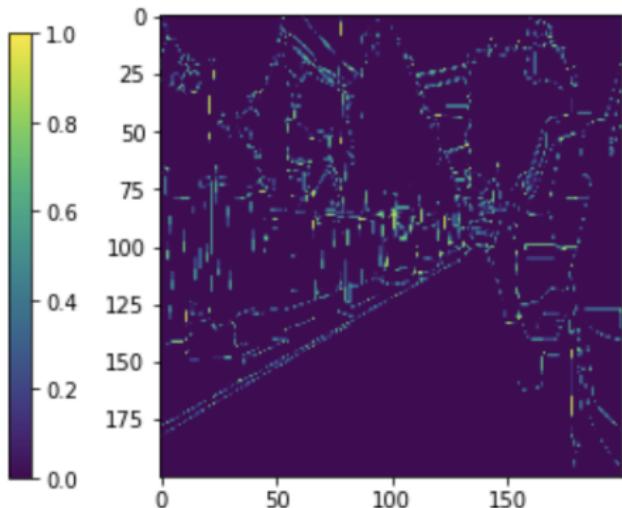
Résultats avec du Fine-Tuning

4 III. Conclusion

5 IV. Références



(a) Image prédite par U-Net après fine-tuning sur une image du dataset test



(b) Label test

Figure – Résultat sur une image du dataset test avec U-Net modifié

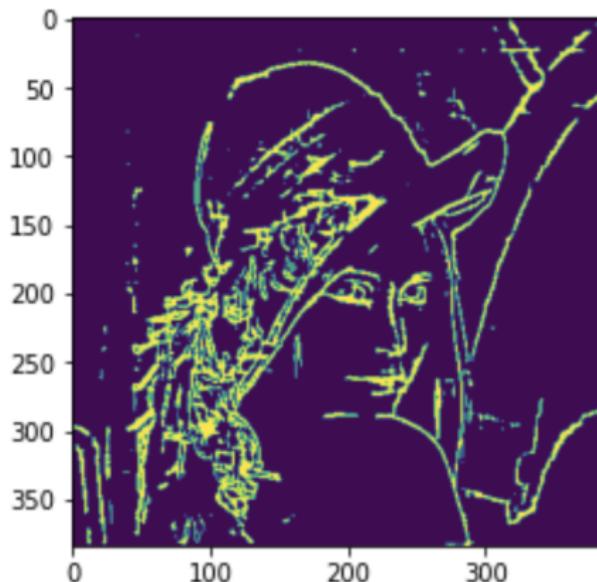
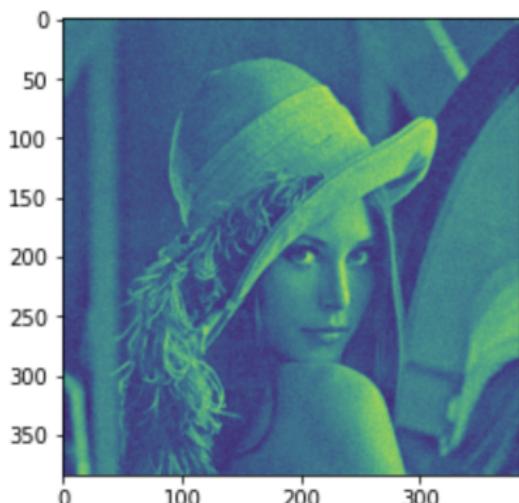
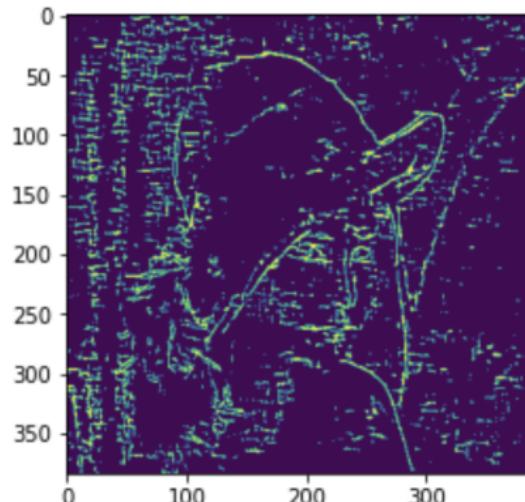


Figure – Image prédictive par U-Net après fine-tuning sur Lenna



(a) Image bruitée initiale



(b) Prédiction de U-Net sur Lenna bruitée

Figure – Résultat avec ajout de bruit

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et
Deep Learning

4 III. Conclusion

5 IV. Références

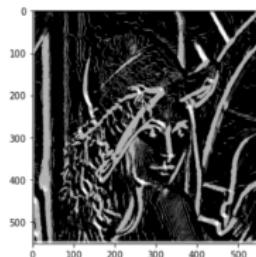


Figure – Canny

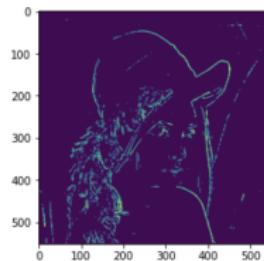


Figure – Autoencodeur

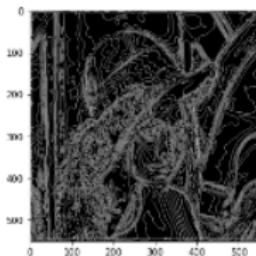


Figure – Laplacien

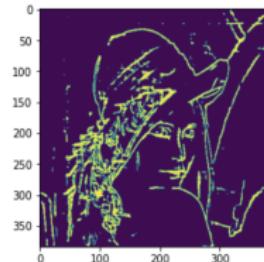


Figure – U-Net

1 I. Introduction

2 II. Méthodes classiques de détection de contours

3 II. Détection de contours avec des méthodes de Machine Learning et Deep Learning

4 III. Conclusion

5 IV. Références

- * http://www.optique-ingenieur.org/fr/cours/pdf/OPI_fr_M04_C05.pdf
- * https://perso.telecom-paristech.fr/bloch/TDI/poly_contours.pdf
- * <https://github.com/xavysp/DexiNed/blob/master/datasets.py>
- * <https://ravivaishnav20.medium.com/visualizing-feature-maps-using-pytorch-12a48cd1e573>