# EchoCem : évaluation de la qualité du ciment par segmentation d'images ultrasoniques par SLB

Laura Choquet, Thomas Gravier

## Abstract

L'évaluation de l'intégrité des puits de forage repose sur l'analyse des interfaces entre le casing, le ciment et la formation rocheuse. Une détection précise de ces interfaces est essentielle pour garantir la stabilité du puits et prévenir les risques d'instabilité structurelle, de fuites ou d'éruptions. Dans le cadre du défi *EchoCem: Third Interface Echo Segmentation for Cement Quality Assessment*, nous avons développé une approche basée sur l'apprentissage automatique pour segmenter avec précision le casing et le *Third Interface Echo* (TIE) dans des images ultrasoniques de puits. Notre méthodologie repose sur l'utilisation d'architectures de segmentation avancées et d'un traitement optimisé des données. Ce rapport détaille les étapes de notre travail, depuis l'analyse initiale des données jusqu'à l'amélioration progressive de notre modèle et l'évaluation des performances obtenues.

## Plan



## 1 Introduction

L'intégrité des puits de forage est un enjeu majeur dans l'industrie énergétique, car elle conditionne à la fois la sécurité des opérations et l'efficacité des extractions. Les puits sont constitués de tubes métalliques (*casing*), entourés de ciment destiné à stabiliser la structure et à isoler les différentes formations géologiques. Une mauvaise qualité du ciment peut entraîner des défaillances structurelles, des fuites ou même des incidents environnementaux majeurs.
Pour évaluer cette intégrité, des scanners ultrasoniques sont utilisés afin de détecter les interfaces entre le ciment et les autres composants du puits. Plus particulièrement, deux interfaces sont d'intérêt : celle entre le ciment et le *casing*, et celle entre le ciment et la formation rocheuse, appelée *Third Interface Echo* (TIE). La segmentation précise de ces interfaces dans les images ultrasoniques est une tâche complexe en raison du bruit dans les données et de la variabilité des formations géologiques.

Dans le cadre de ce rapport, nous présentons notre participation au défi *EchoCem*, qui consiste à développer un modèle de segmentation performant pour identifier le *casing* et le *TIE* avec une précision optimale. Nous détaillons notre approche méthodologique, les choix algorithmiques réalisés et les améliorations successives apportées à notre modèle afin d'optimiser ses performances.

## 2 Présentation du Problème et Analyse des Données

### 2.1 Défis Techniques et Exploration des Données

La segmentation d'images ultrasoniques des puits de forage présente des défis spécifiques par rapport aux images classiques (ex. ImageNet). Ces images sont souvent bruitées, ont un faible contraste et des contours peu définis, rendant difficile la distinction entre les différentes interfaces (*casing*, *Third Interface Echo (TIE)*, et le fond).
Les données fournies pour ce challenge sont des images ultrasoniques segmentées en patchs de dimensions $160 \times 160$ et $160 \times 272$ pixels, accompagnées de masques d'annotation définissant trois classes :

- **0 : Fond**
- **1 : Casing**
- **2 : Third Interface Echo (TIE)**

Un exemple est présenté figure 1.
L'analyse statistique des données montre un fort déséquilibre des classes, avec une majorité de pixels appartenant au fond, ce qui peut biaiser l'entraînement du modèle de segmentation. Un diagramme présente le déséquilibre entre les classes 2
Nous avons également remarqué que certaines masques d'un puit étaient mal labellisés, nous les avons supprimer des données d'entraînements, voir sur la figure 3

### 2.2 Formulation Mathématique du Problème de Segmentation

Le problème de segmentation peut être modélisé comme une classification pixel-par-pixel où l'objectif est d'attribuer à chaque pixel une étiquette parmi $\{0, 1, 2\}$.
Soit $I$ un ensemble de pixels d'une image ultrasonique et $C = \{0, 1, 2\}$ l'ensemble des classes possibles. La segmentation consiste à apprendre une fonction de décision $f : I \to C$ qui minimise la fonction de coût suivante :

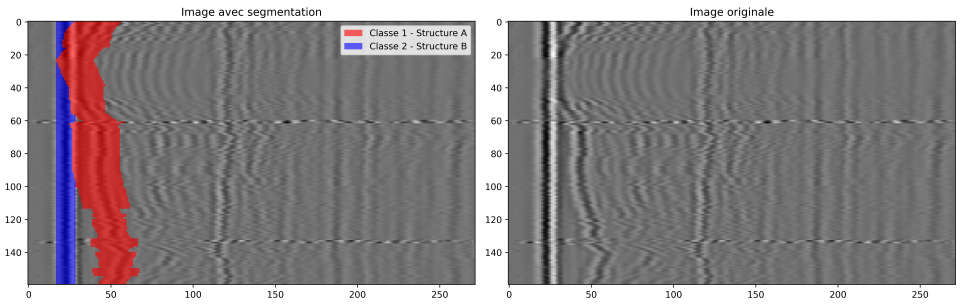


**FIGURE 1.** *Image du dataset d'entraînement avec les labels associés à chaque pixel*

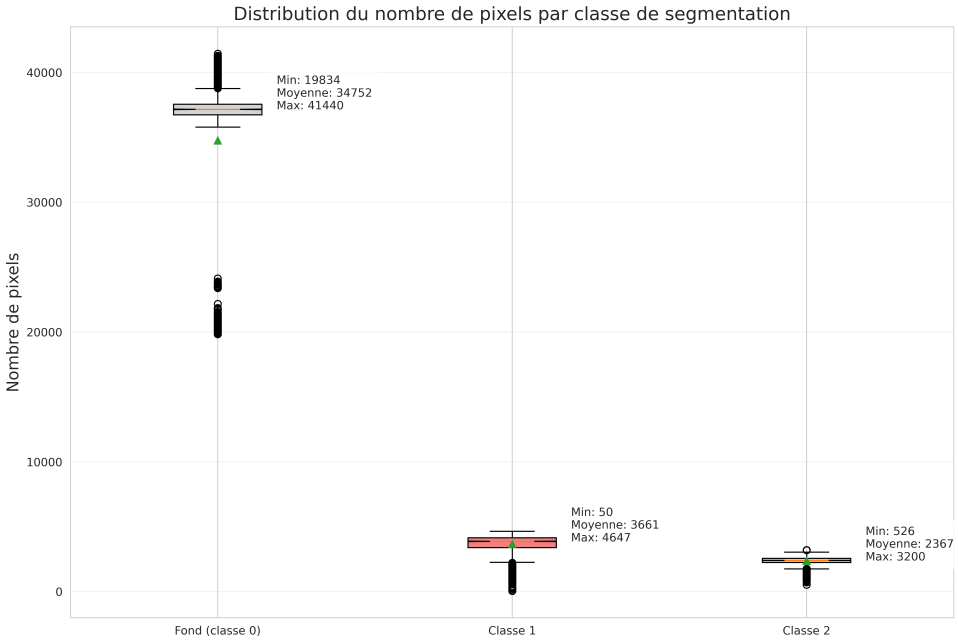


**FIGURE 2.** *Distribution du nombre de pixels par classe de segmentations*

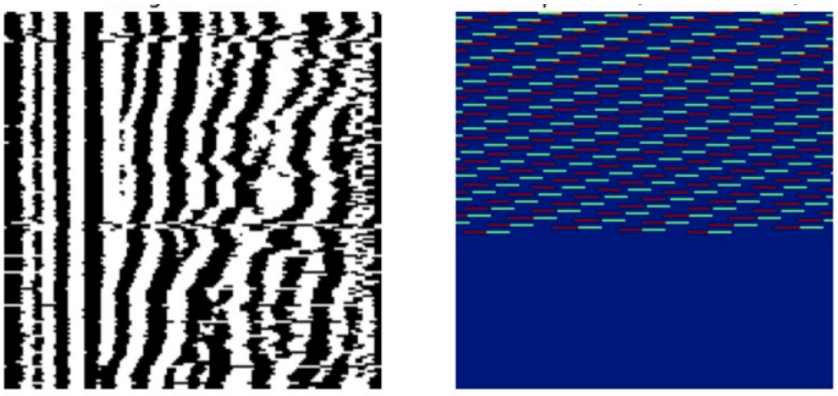

**FIGURE 3.** *Exemple d'image mal labellisée*

$$J(f) = \sum_i L(f(i), y_i) \tag{1}$$

où $L$ est une fonction de perte, $f(i)$ la classe prédite pour le pixel $i$, et $y_i$ la classe réelle du pixel $i$.

### 2.3 Représentation des Données et Apprentissage des Features $\Phi(x)$

Dans les tâches de segmentation, la construction d'une bonne représentation $\Phi(x)$ des données est essentielle pour capturer les structures pertinentes. On distingue deux grandes approches :

#### 2.3.1 Méthodes Classiques (Feature Engineering)

Avant l'essor du deep learning, la segmentation se basait sur des extractions de caractéristiques comme :

- Les histogrammes d'intensité pour capturer la répartition des niveaux de gris.
- Les filtres de Gabor pour détecter des textures spécifiques.
- Les transformées de Fourier ou d'ondelettes [5] pour analyser les fréquences spatiales.

Ces méthodes peuvent être utiles mais manquent de généralisation pour les images ultrasoniques.

#### 2.3.2 Apprentissage de Représentations avec des Réseaux de Neurones

Les architectures de segmentation modernes, comme U-Net[6] ou DeepLabV3+ [1], apprennent directement $\Phi(x)$ via un réseau de neurones convolutif [3] (Convolutional neural network - CNN). Le U-Net qui doit son nom à sa forme en U, est composé d'un premier bloc appelé, encodeur, qui extrait le contexte et les détails sémantiques des images. Dans ce blocs, se suivent des couches convolutionnels et des opérations de max-pooling, la taille de l'image est froncièrement réduite. Le pont ou le goulot d'étranglement qui suit l'encodeur traite les données extraites qui sont ensuite passées dans le décodeur. Ce deuxième bloc permet de faire la classification pixel par pixel, se suivent des opérations de up-sampling ainsi que des convolutions transposées pour retrouver la taille d'origine de l'image.

### 2.4 Fonction de Perte et Métrique d'Évaluation

#### 2.4.1 Gestion du Déséquilibre des Classes

Pour compenser la sous-représentation du *casing* et du *TIE*, nous utilisons une combinaison de fonctions de perte adaptées :

- **Dice Loss** ($\mathcal{L}_{Dice}$) : pénalise les erreurs sur les petites classes.
- **Focal Loss** ($\mathcal{L}_{Focal}$) : ajuste le poids des erreurs en fonction de leur fréquence.
- **Combinaison des deux :**

$$\mathcal{L} = \alpha \mathcal{L}_{Dice} + \beta \mathcal{L}_{Focal} \tag{2}$$

avec $\alpha$ et $\beta$ des coefficients ajustables.

#### 2.4.2 Métrique d'Évaluation : Intersection over Union (IoU)

La performance du modèle est évaluée à l'aide du coefficient d'Intersection over Union (**IoU**), défini comme :

$$\text{IoU} = \frac{\text{Aire de l'intersection}}{\text{Aire de l'union}} \tag{3}$$

Cette métrique est robuste par rapport aux déséquilibres de classes et permet de pénaliser les faux positifs. Elle permet également de comparer l'efficacité de prédiction de l'algorithme par classe.

## 3 Présentation des algorithmes d'essais

### 3.1 Algorithme de Référence (Benchmark)

Le benchmark est basé sur une architecture de type encodeur-décodeur avec des réseaux de convolutions adaptés à l'analyse d'images. L'entraînement d'un tel modèle est rendue difficle par plusierus facteurs :

- **Manque d'interprétabilité** : Les CNNs sont des "boîtes noires".
- **Besoins en données** : Ces modèles nécessitent de grandes quantités de données annotées, nous ne possédons que 4410 images.
- **Entraînement coûteux** : Une architecture profonde est nécessaire pour extraire les caractéristiques intéressantes des images.

Avant d'utiliser un modèle sur des réseaux de neurones nous avons tenté d'utiliser des réseaux de machine learning plus classique, une Random Forest.

### 3.2 Développement d'un Modèle Basé sur Random Forest

Les random forests, un outil puissant de machine learning utilisé pour la régression et la classification, peuvent également être adaptées à la segmentation d'images en classant individuellement chaque pixel sur la base de caractéristiques locales et contextuelles.

#### 3.2.1 Extraction des Caractéristiques

- **Echelle globale :** Nous avons commencé par filtrer les données et extraire les caractéristiques d'intérêt. Les images ont d'abord été partiellement débruitées à l'aide d'un **filtrage médian**, qui consiste à remplacer la valeur d'un pixel par la médiane de ses voisins. Nous avons ensuite appliqué un **filtrage de Gabor**, particulièrement efficace pour extraire des lignes d'orientations variées tout en réduisant le bruit. Nous aurions également pu utiliser une décomposition en ondelettes afin d'extraire d'autres caractéristiques pertinentes.
- **Echelle locale :** A l'échelle du pixel nous avons utilisé sa valeur ainsi que des informations prises dans une fenêtre centrée sur celui-ci. Nous avons appliqué un filtre de Sobel dans la direction verticale et horizontale, et calculer la norme du gradient. Les caractéristiques associées à un pixel était donc un vecteur des valeurs de ses pixels voisins, la moyenne et l'écart-type de l'intensité des pixels de la fenêtre, la moyenne et le maximum des normes des gradients calculés.

### 3.2.2 Construction et Entraînement du Modèle

Une Random Forest est un ensemble de plusieurs arbres de décision, chaque arbre est entraîné sur un sous-ensemble aléatoire de données, à chaque noeud seul un sous-ensemble aléatoire de caractéristiques est considéré. Chaque arbre vote pour une classe d'un pixel.
Soit $I$ une image et $p$ un pixel, on note $f(p)$ les caractéristiques extraites pour ce pixel. On note $t_1, t_2, \ldots, t_n$ les $n$ arbes qui composent la random forest. La probabilité que le pixel p appartienne à la classe c est :

$$P(c|p) = \frac{1}{N} \sum_{i=1}^{N} P_i(c|f(p)) \tag{4}$$

où $P_i(c|f(p))$ est la prédiction du $i$-ième arbre.

Notre **Random Forest Classifier** a été entraîné avec :
- **Nombre d'arbres** : 500
- **Critère** : Impureté de Gini
- **Bootstrap** : Activé

### 3.2.3 Limites

Le choix des filtres, déterminé par l'utilisateur et facilement **visualisable**, permet **d'adapter** leur sélection **aux spécificités des données**. Cette flexibilité est un atout, mais elle présente aussi des limites : les caractéristiques extraites restent dépendantes des choix manuels et peuvent ne pas capturer toute la richesse des images. De plus, le **coût computationnel** peut devenir prohibitif, notamment avec le filtrage de Gabor, dont l'application à plusieurs orientations et longueurs d'onde spatiales entraîne un temps de calcul important.
La prédiction de la classe d'un pixel en fonction de ses caractéristiques est également **coûteuse en temps et en mémoire**. Chaque image compte 36 000 pixels, et la Random Forest doit traiter un grand nombre de caractéristiques par pixel, ralentissant l'inférence. Nous avons donc réduit l'échantillonnage à 10 000 pixels par image, trouvant un compromis entre coût et qualité de segmentation. Toutefois, cette réduction peut affecter la précision si l'échantillonnage ne reflète pas bien la distribution des classes (cf. Fig. 4). Une alternative serait un échantillonnage plus stratégique, par exemple en privilégiant les pixels proches des contours ou des zones à forte variabilité.

**FIGURE 4.** *Prédiction de classe avec une random forest. Image de gauche : Image de base, Image du milieu : Masque, et Image de droite : Prédiction*

# 4 Développement de l'Algorithme principal

## 4.1 Amélioration Basée sur l'Analyse et l'Expérience

### 4.1.1 Motivation : Limite du Baseline et Passage au Fine-Tuning

Le modèle proposé dans le benchmark était entraîné sur trop peu d'images, limitant sa capacité de généralisation. En apprentissage supervisé, un échantillon insuffisant entraîne un surapprentissage et réduit la performance sur de nouvelles images. De plus, la Random Forest, bien que performante pour la classification, restait dépendante du choix des descripteurs et impliquait un temps de calcul prohibitif, notamment lorsque le nombre de pixels analysés augmentait.
Face à ces limitations, nous avons implémenté un **U-Net**, une architecture de deep learning conçue pour la segmentation. Pour améliorer la robustesse et accélérer l'apprentissage, nous avons intégré **un backbone pré-entraîné sur ImageNet**, permettant d'exploiter des représentations d'images généralistes et de réduire le besoin en données annotées.

### 4.1.2 Justification Théorique du U-Net et Apprentissage d'une Représentation Adaptée

Un **U-Net** est une architecture spécialisée en segmentation, composée d'un **encodeur** et d'un **décodeur**. La fonction de transformation des données par le modèle peut être formulée comme :

$$\phi(x) = D(E(x)) \tag{5}$$

où :
- $E(x)$ est l'encodeur qui extrait une représentation comprimée des caractéristiques importantes de l'image.
- $D(E(x))$ est le décodeur qui reconstruit l'image segmentée.

La différence avec le modèle du benchmark réside donc dans l'apprentissage par transfert que nous réalisons avec ce modèle basé sur l'architecture U-Net. Nous utilisons un réseau préalablement entraîné sur un grand dataset, ce qui résout ainsi le problème de la taille relativement limitée de l'ensemble d'images annotées. De plus, U-Net intègre des connexions résiduelles, permettant d'utiliser des couches profondes sans rencontrer le problème du vanishing gradient.

**Choix du Backbone Pré-entraîné:**

L'encodeur E(x) est basé sur un ResNet34 [2], un modèle pré-entraîné sur ImageNet, qui permet d'apprendre une représentation robuste des images. Grâce à l'architecture résiduelle de ResNet, qui intègre des blocs résiduels, ce modèle facilite l'entraînement de réseaux très profonds en atténuant les problèmes de dégradation des performances. ResNet34 exploite les couches profondes du réseau pour capter efficacement des motifs de bas niveau, tels que les bords et textures, ainsi que des motifs de haut niveau, comme les formes et objets, déjà appris dans le cadre du pré-entraînement sur ImageNet. Cette approche permet d'améliorer la capacité de généralisation du modèle sur de nouvelles tâches de vision par ordinateur.

**Choix de la loss:**

La Focal Dice Loss combine le coefficient de Dice, qui mesure le chevauchement entre prédictions et vérités terrain, avec la Focal Loss qui se concentre sur les cas difficiles à classifier. En additionnant ces deux composantes ($\alpha \cdot$ Focal Loss + $\beta \cdot$ Dice Loss), cette fonction donne plus

d'importance aux pixels mal segmentés, particulièrement ceux des classes minoritaires. Cette approche est particulièrement efficace pour notre segmentation à trois classes déséquilibrées, où la classe "fond" domine largement l'image.

#### 4.1.3 Préprocessing et Augmentation des Données

**Choix de la Taille d'Image :** Notre dataset est composé d'images de taille 160x160 et d'images 160x272. Nous devions choisir une taille d'images uniques pour tout le dataset pour l'entraînement des dernières couches du modèle.

- **160 × 160** : Moins coûteux en mémoire, mais perte d'information. (Down sampling)
- **160 × 272** : Préserve mieux les détails, mais ralentit l'apprentissage. (Up sampling)

Le choix final a été **160 × 160** après analyse de la performance en validation.

**Augmentation des Données avec Albumentations :** Pour améliorer la robustesse et éviter l'overfitting, nous avons utilisé des transformations garantissant l'**invariance de la représentation** :

$$\phi(T(x)) \approx \phi(x) \tag{6}$$

où $T(x)$ est une transformation de l'image. Les transformations appliquées sont :

- **Rotation** ($T_r(x)$) : Permet de capturer des orientations variées.
- **Flip Horizontal/Vertical** ($T_f(x)$) : Assure l'invariance par symétrie.
- **Affine Transformations** ($T_a(x)$) : Légères déformations pour augmenter la variabilité.
- **Gaussian Noise** ($T_n(x)$) : Simule du bruit sur l'image.

Cette augmentation permet de **renforcer la généralisation** sur des images bruitées, et d'**éviter l'overfitting** sur des caractéristiques spécifiques, tout en**améliorerant la robustesse** du modèle en couvrant des cas non présents dans l'entraînement.

#### 4.1.4 Optimisation de l'Architecture du Réseau

**Fine-Tuning du Modèle :**

- Geler les couches inférieures du backbone au début de l'apprentissage.
- Débloquer progressivement ces couches après stabilisation des premières epochs.

**Optimisation du Learning Rate :**

- Utilisation d'un **scheduler dynamique** (ReduceLROnPlateau).
- Meilleur compromis entre convergence rapide et stabilité d'apprentissage.

**Validation et Sélection du Modèle :** L'évaluation s'est faite sur :

- **L'IoU** sur le jeu de validation.
- **La perte d'entraînement** pour détecter l'overfitting.

#### 4.1.5 Post-Traitement des prédictions

**Post-processing manuel :**

Après avoir analysé les labels, nous avons remarqué que la classe 2, qui correspond au casing, avait une forme très régulière, souvent un rectangle. Or, nous avons remarqué lors de l'inférence, que ce soit en validation ou en test que certaines prédictions donnaient plusieurs groupes de

pixels disjoints pour cette classe. Nous avons appliqué un post-processing qui consistait à garder la plus grande composante connexe de la classe 2. Cette amélioration a permis sur l'ensemble de test d'améliorer l'IoU.
Les labels de la classe 1 suivaient en majorité la même propriété mais certains cas montraient deux sous-ensembles connexes. Ils étaient ainsi difficiles d'appliquer un règle global.

**Post-processing appris par réseau de neurones :**

Ne pouvant appliquer de règle générale pour la classe 1, nous avons choisi d'entraîner un réseau de neurones basé sur une architecture U-net pour améliorer les masques.

- **Input :** Probabilités d'appartenance aux classes associées à chaque pixel
- **Loss :** Focal Dice Loss, identique à la loss de l'entraînement du modèle initial
- **Critère de sélection du modèle :** Nous avons choisi de garder le modèle qui améliorait le plus l'IoU moyen.

Ce U-net assez simple nous a permis d'améliorer de 1% l'IoU moyen sur les données de validation, ce qui nous aussi permis de passer en tête du classement. Sur la figure 5 on observe que le modèle de post-traitement est capable de combler certains trous, de supprimer le bruit.

**Post-processing morphologique :**

En dernier lieu, nous avons appliqué une fonction de post-traitement morphologique aux masques de segmentation pour améliorer sa qualité. Elle utilise une ouverture morphologique pour supprimer les petits bruits, suivie d'une fermeture morphologique pour combler les petits trous dans les objets segmentés. Ces opérations, effectuées avec un noyau structurant $3 \times 3$, permettent d'affiner la segmentation en rendant les contours plus cohérents et homogènes.

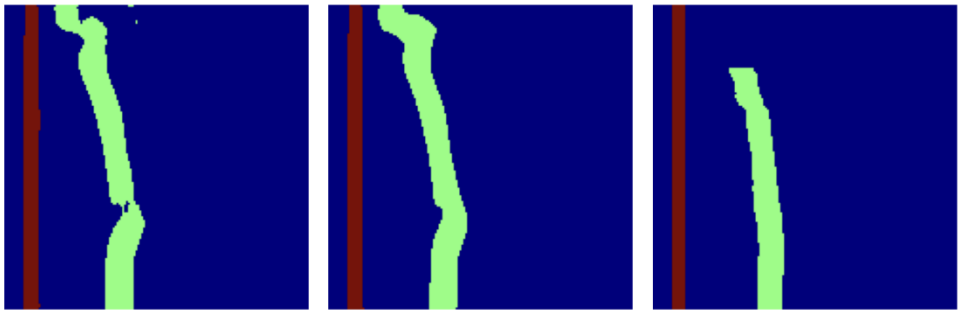

**FIGURE 5.** *Post-traitement automatique.*
*A gauche : Prédiction du modèle principale, Au milieu : Prédiction améliorée par le modèle secondaire, A droite : Le ground truth*

## 5 Résultats et analyse des soumissions

### 5.1 Évolution des performances

L'évolution de nos soumissions a montré une progression constante, partant d'un simple *fine-tuning* de ResNet avec un score de **0,5909**, jusqu'à l'obtention d'un score final de **0,6821**. Cette amélioration a été rendue possible grâce à une combinaison de techniques avancées, incluant

l'augmentation des données, l'optimisation des hyperparamètres, l'utilisation d'un modèle de post-traitement basé sur un *U-Net*, et l'introduction de stratégies adaptées de filtrage.
La première tentative reposait sur un *fine-tuning* d'un *ResNet pré-entraîné*, dans l'objectif de segmenter les interfaces du ciment dans les images ultrasoniques. Malgré un résultat encourageant, cette approche s'est révélée insuffisante en raison de l'absence d'une stratégie de régularisation et d'augmentation des données.
Des ajustements ont ensuite été réalisés en introduisant des techniques d'augmentation, permettant d'améliorer progressivement le score. L'agrandissement des images, bien qu'envisagé, n'a pas apporté d'amélioration significative. En revanche, l'introduction d'un post-traitement avec *U-Net* et la fonction de perte Lovász Softmax a permis de franchir un palier, atteignant un score de **0,6802**.
Afin d'affiner davantage la segmentation, plusieurs stratégies de post-traitement ont été testées, notamment une correction heuristique manuelle et l'utilisation de composantes connexes. Ces méthodes n'ont toutefois pas toujours abouti à des améliorations notables. Finalement, l'intégration d'un post-traitement automatique basé sur la sélection de la plus grande composante connexe s'est avérée être la solution la plus efficace, permettant d'obtenir le **meilleur score du challenge (0,6821)**.

### 5.2 Classements publics et privés

La performance de notre modèle a été validée par sa capacité à se classer **premier dans tous les classements**, aussi bien sur les données publiques que privées. Cette stabilité des performances démontre l'absence de sur-apprentissage et confirme la généralisation efficace du modèle.

| Rang | Date | Participants | Score final (privé) |
|---|---|---|---|
| 1 | 14 mars 2025 | **Looora & Litr0ck** | **0,6818** |
| 2 | 9 mars 2025 | othman.hicheur | 0,6654 |
| 3 | 6 mars 2025 | BarthélémyBrégeon & kvert | 0,5696 |
| 4 | - | **Benchmark** | 0,4086 |

**TABLE 1.** Classement final sur les données privées

| Rang | Date | Participants | Score public |
|---|---|---|---|
| 1 | 14 mars 2025 | **Looora & Litr0ck** | **0,6821** |
| 2 | 9 mars 2025 | othman.hicheur | 0,6738 |
| 3 | 16 mars 2025 | mathis.wauquiez & pikoglu | 0,6222 |
| 4 | 17 mars 2025 | BarthélémyBrégeon & kvert | 0,5915 |
| 5 | - | **Benchmark** | 0,4302 |

**TABLE 2.** Classement académique public

### 5.3 Analyse des performances

L'obtention de la première place dans tous les classements, aussi bien sur les données publiques que privées, montre que notre approche évite le sur-apprentissage et assure une bonne généralisation.

L'utilisation d'un *U-Net* en post-traitement a permis de corriger des erreurs systématiques du modèle principal, améliorant ainsi la segmentation des interfaces. De plus, l'implémentation d'une validation croisée stricte a joué un rôle clé dans la stabilisation des performances. L'optimisation des hyperparamètres, notamment via le choix de la fonction de perte Lovász Softmax et l'adoption de techniques avancées de régularisation, a également eu un impact significatif.
Enfin, l'approche progressive d'amélioration, intégrant des tests successifs sur différentes méthodes de post-traitement et d'augmentation des données, a permis d'atteindre un modèle optimal, performant sur l'ensemble des évaluations. Cette méthodologie pourrait être appliquée à d'autres problématiques nécessitant une segmentation fine d'images bruitées.

## 6 Conclusion

Dans ce rapport, nous avons exploré différentes approches pour la segmentation d'images ultrasoniques dans le cadre du défi *EchoCem*. En commençant par une analyse approfondie des données et des défis associés, nous avons progressivement affiné notre méthodologie, en passant d'une approche classique basée sur des forêts aléatoires à un modèle de deep learning performant basé sur l'architecture **U-Net avec un backbone pré-entraîné**.
Nos résultats montrent que l'utilisation d'un modèle de segmentation avancé, couplée à une **stratégie d'augmentation des données adaptée**, un **choix optimal de la fonction de perte** et un **post-traitement efficace**, a permis d'obtenir une **IoU de 0,6821 sur le classement public** et **0,6818 sur le classement académique privé**. Le post-processing basé sur un second réseau de neurones a apporté une amélioration notable, permettant d'affiner les prédictions et d'optimiser les performances du modèle principal.
Malgré ces résultats encourageants, plusieurs pistes d'amélioration peuvent être envisagées pour des travaux futurs :

- **Utilisation de modèles plus avancés**, tels que les **transformers** (ex. Swin Transformer) [4], qui pourraient mieux capturer les relations spatiales complexes des images ultrasoniques.
- **Exploitation de la continuité temporelle entre images**, en intégrant des techniques de traitement de séries temporelles ou de modèles 3D pour une meilleure cohérence spatiale.
- **Affinement des méthodes de post-traitement**, en développant des algorithmes adaptés à la morphologie des interfaces détectées.

En conclusion, ce travail met en évidence l'importance d'une approche hybride combinant **apprentissage profond, optimisation algorithmique et traitement des données** pour relever les défis posés par l'analyse d'images ultrasoniques en milieu industriel. Ces avancées ouvrent la voie à des applications concrètes dans l'évaluation de l'intégrité des puits de forage et au-delà, dans d'autres domaines où la segmentation d'images bruitées est un enjeu critique.

## References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[5] Stephane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.