



FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

UNIVERSITÉ SIDI MOHAMED BEN
ABDELLAH

Segmentation de Tumeurs Cérébrales & Mammaires

Présenté par :

- EL MAHDAOUI Ahmed
- ER-ROUGBANI Mouaad
- LAHMAMSSI Adnane

Encadré par :

Dr. RIFFI Jamal

Année universitaire 2024–2025

Table des matières

1	Introduction	3
2	Jeux de Données Utilisés	3
2.1	Brain Tumor Dataset	3
2.2	Breast Ultrasound Images Dataset	4
3	Prétraitement et Augmentation des Données	4
3.1	Sélection et Construction du Dataset	4
3.2	Description des paramètres d'augmentation	5
3.3	Caractéristiques du Dataset Final	6
3.4	Architecture des Modèles	6
3.5	Entraînement	8
4	Résultats	10
4.1	Métriques	10
4.2	Performances Observées	10
4.3	Visualisation	10
5	Comparaison des Architectures : U-Net vs U-Net++ vs DeepLabV3 . .	11
6	Conclusion	14

Table des figures

1	visualisation du jeux de données	3
2	visualisation du jeux de donnée	4
3	Visualisation de notre jeux de données(Dataset)	5
4	Architecture de U-Net++	7
5	Architecture de ResNet34	7
6	Architecture de DeepLabV3	8
7	Répartition de données	9
8	Performance	10
9	Exemple de segmentation brain	10
10	Exemple de segmentation breast	11
11	Comparaison des performances des trois architectures de segmentation	11
12	Comparaison synthétique des performances des modèles	14

1. Introduction

La détection et la segmentation automatiques des tumeurs dans les images médicales jouent un rôle crucial dans le diagnostic et la planification thérapeutique. Les méthodes classiques, souvent manuelles, sont longues, coûteuses et sensibles aux erreurs humaines.

Grâce aux progrès de l'apprentissage profond, en particulier des réseaux de neurones convolutifs (CNN), il est aujourd'hui possible d'entraîner des modèles capables d'identifier et segmenter efficacement les structures anormales dans les images médicales. Ce rapport présente une étude expérimentale de la segmentation de tumeurs dans deux types d'images : les IRM cérébrales et les échographies mammaires.

2. Jeux de Données Utilisés

2.1. Brain Tumor Dataset

- **Source** : Kaggle
- **Contenu** : Images IRM cérébrales (formats PNG et JPG) avec leurs masques binaires annotés.
- **Structure** : Deux dossiers : `images/` et `masks/`, chaque image a un masque correspondant.

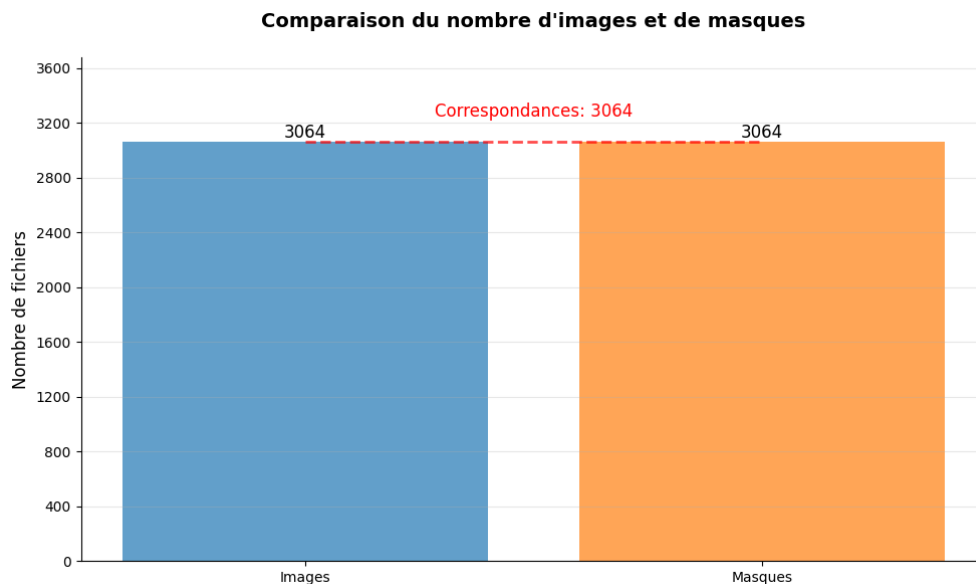


Figure 1: visualisation du jeux de données

2.2. Breast Ultrasound Images Dataset

- **Source** : Kaggle
- **Contenu** : Échographies mammaires classées (normal, bénin, malin), accompagnées de masques pour les cas tumoraux.
- **Défis** : Bruit visuel élevé, faible contraste, formes de tumeurs variées.

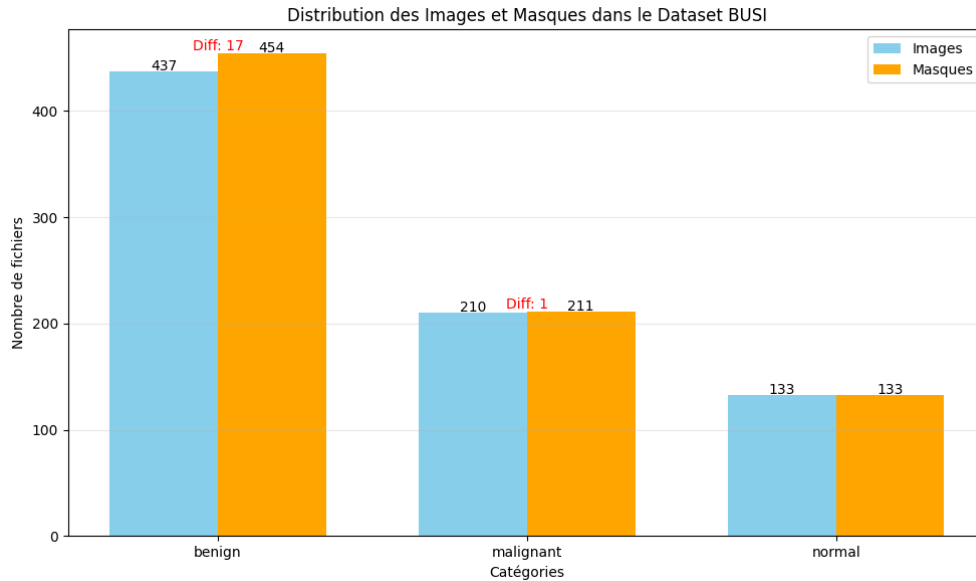


Figure 2: visualisation du jeu de donnée

3. Prétraitement et Augmentation des Données

3.1. Sélection et Construction du Dataset

Notre jeu de données final résulte d'une sélection minutieuse à partir de deux sources principales :

- **Brain Tumor Dataset** : 647 paires image/masque soigneusement sélectionnées pour la qualité des annotations, couvrant diverses formes et stades de tumeurs cérébrales.
- **Breast Ultrasound Dataset** : 647 images échographiques du sein, chacune accompagnée de ses masques correspondants (jusqu'à trois masques par image dans les cas les plus complexes), permettant une meilleure représentation des tissus et anomalies.
- **Résultat final du jeu de données** : les deux ensembles de données ont été combinés pour obtenir un total de 1294 paires image/masque. Cette fusion a été réalisée pour éviter le problème de déséquilibre des classes dans les jeux de données, car un modèle

entraîné sur un nombre d'exemples très déséquilibré (par exemple 3064 contre 647) pourrait privilégier les classes majoritaires au détriment des minoritaires. Ainsi, cette combinaison permet d'assurer une meilleure diversité et robustesse du modèle lors des phases d'entraînement et de validation.

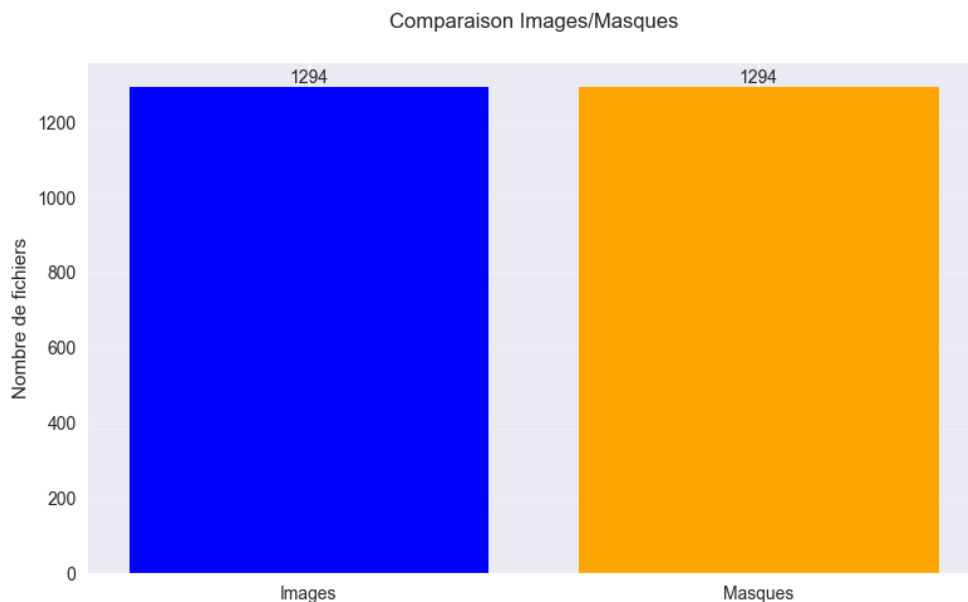


Figure 3: Visualisation de notre jeux de données(Dataset)

3.2. Description des paramètres d'augmentation

- **A.Resize(512, 512)** : redimensionne toutes les images à 512 pixels de hauteur et 512 pixels de largeur pour une standardisation des entrées.
- **A.HorizontalFlip(p=0.5)** : retourne horizontalement l'image avec une probabilité de 50%, ce qui simule les variations latérales.
- **A.VerticalFlip(p=0.5)** : retourne verticalement l'image avec une probabilité de 50%, augmentant ainsi la diversité des orientations.
- **A.RandomRotate90(p=0.5)** : effectue une rotation de 90 degrés aléatoire avec une probabilité de 50%, ce qui est particulièrement utile pour certaines classes d'objets.
- **A.ShiftScaleRotate** : combine une translation (shift_limit=0.1, soit 10% max), un zoom (scale_limit=0.1, soit 10% max) et une petite rotation (rotate_limit=15°), chacun appliqué avec une probabilité de 50%. Cela permet de simuler les déformations courantes des images.
- **A.GaussianBlur(blur_limit=(3,7), p=0.2)** : applique un flou gaussien avec un noyau de taille variable (3 à 7 pixels) avec une probabilité de 20%. Cela aide à généraliser le modèle en le rendant plus robuste au flou.

- **A.RandomBrightnessContrast** : modifie la luminosité (`brightness_limit=0.2`) et le contraste (`contrast_limit=0.2`) avec une probabilité de 30%, ce qui simule les variations d'éclairage.
- **A.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))** : normalise les canaux RGB des images pour centrer et réduire la variance des pixels, améliorant ainsi la convergence du modèle.
- **ToTensorV2()** : convertit l'image transformée en tenseur compatible avec les réseaux de neurones (PyTorch).

3.3. Caractéristiques du Dataset Final

Le dataset résultant présente les caractéristiques suivantes :

Table 1: Statistiques du jeu de données après augmentation

Type	Original	Augmenté	Ratio	Taille des images
Paires Brain	647	5 176	8×	(512, 512, 1)
Paires Breast	647	5 176	8×	(512, 512, 1)

Les techniques d'augmentation ont permis :

- Une meilleure robustesse aux variations géométriques
- Une réduction du surapprentissage par diversification artificielle
- Le maintien des rapports anatomiques grâce aux transformations synchrones

3.4. Architecture des Modèles

Dans cette étude, nous avons utilisé deux architectures principales de segmentation sémantique : **UNet++** et **DeepLabV3**. Ces architectures ont été sélectionnées pour leurs performances reconnues et leur complémentarité.

UNet++ avec encodeur ResNet34 L'architecture **UNet++** a été implémentée via la bibliothèque `segmentation_models.pytorch`, en utilisant un encodeur **ResNet34** préentraîné sur ImageNet. Cette combinaison permet de bénéficier à la fois de :

- Une **résolution fine des détails** grâce à la structure améliorée de UNet++, qui introduit des chemins de connexion plus denses entre les couches d'encodage et de décodage, facilitant la reconstruction précise des contours.
- Une **extraction efficace des caractéristiques** via l'encodeur ResNet34, réputé pour ses blocs résiduels permettant un apprentissage profond stable et performant.

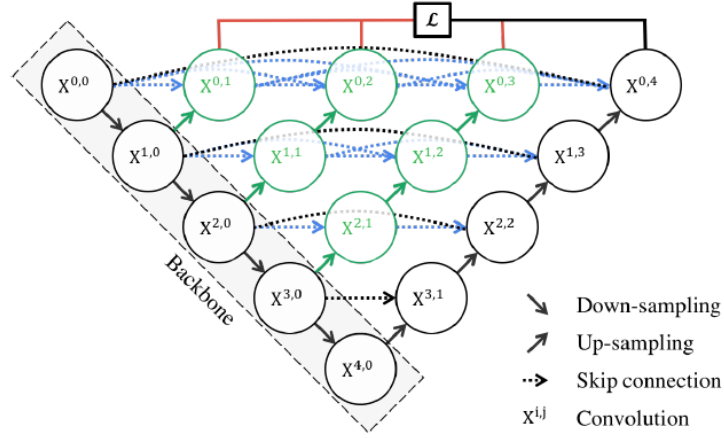


Figure 4: Architecture de U-Net++

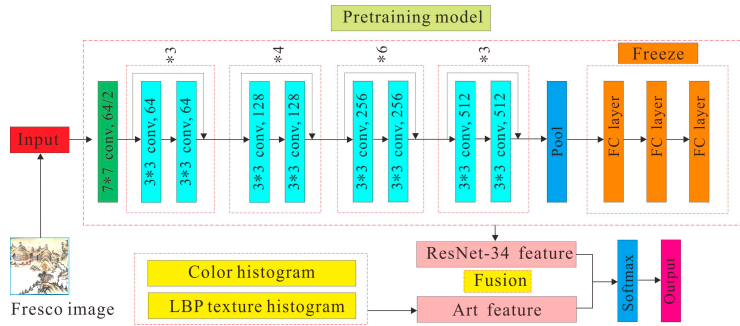


Figure 5: Architecture de ResNet34

DeepLabV3 Le second modèle étudié est **DeepLabV3**, une architecture de segmentation avancée reposant sur des dilated convolutions (convolutions à trous) pour augmenter le champ réceptif sans perdre la résolution spatiale. DeepLabV3 intègre également un module Atrous Spatial Pyramid Pooling (ASPP) qui capture les informations contextuelles à différentes échelles, améliorant la précision de segmentation sur des objets de tailles variées.

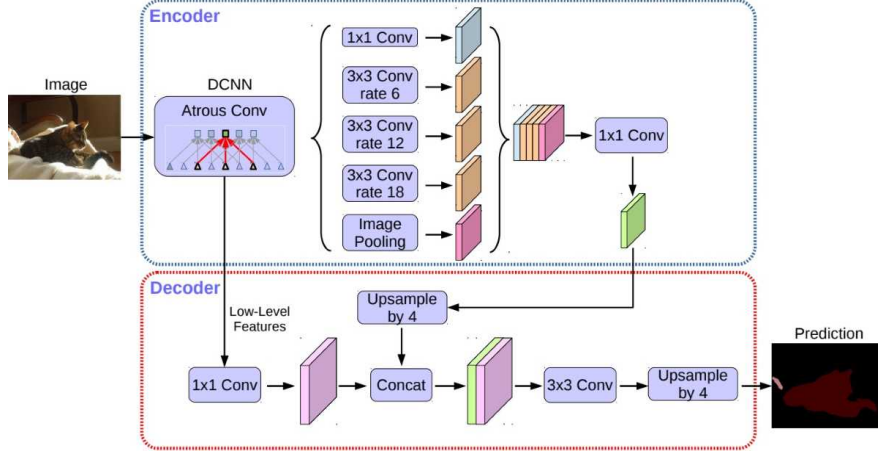


Figure 6: Architecture de DeepLabV3

Cette diversité d'architectures nous permet d'évaluer et de comparer les performances en termes de précision et de robustesse sur notre jeu de données.

3.5. Entraînement

- **Fonction de perte :** Dice Loss + Binary Cross Entropy.

La fonction de perte hybride combine les avantages du *Dice Loss* (bonne performance sur les classes déséquilibrées) et de la *Binary Cross-Entropy* (optimisation stable) :

$$\mathcal{L}_{\text{Total}} = \underbrace{1 - \frac{2|P \cap G| + \epsilon}{|P| + |G| + \epsilon}}_{\text{Dice Loss}} + \lambda \underbrace{\left[-\frac{1}{N} \sum_{i=1}^N g_i \log(p_i) \right]}_{\text{Binary Cross-Entropy}} \quad (1)$$

- P : Masque prédit (probabilités)
- G : Masque de vérité terrain (valeurs binaires)
- λ : Poids de la BCE (typiquement $\lambda = 1$)
- ϵ : Terme de lissage (10^{-6})

- **Optimiseur :** Adam, LR = 1×10^{-4}

L'optimiseur Adam combine les avantages de RMSProp et du momentum. Pour chaque paramètre θ_t à l'étape t :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad \text{(Momentum du gradient)} \quad (2)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad \text{Mise à jour des carrés} \quad (3)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \text{Correction du biais} \quad (4)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad \text{Correction du biais} \quad (5)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad \text{Mise à jour des paramètres} \quad (6)$$

- g_t : Gradient à l'étape t
- η : Taux d'apprentissage initial (typiquement 10^{-3})
- β_1, β_2 : Coefficients de décroissance (par défaut 0.9 et 0.999)
- ϵ : Terme de régularisation (typiquement 10^{-8})

- **Batch size** : 8

- **Époques** : 25

- **Répartition** :

- Entraînement : 70%
- Validation : 20%
- Test : 10%

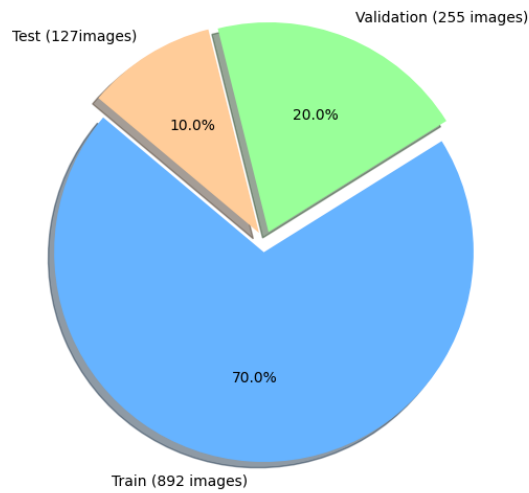


Figure 7: Répartition de données

4. Résultats

4.1. Métriques

Nous avons utilisé principalement le **Dice Coefficient** pour mesurer l'overlap entre le masque prédit et le masque réel. La fonction de perte combinée permet d'équilibrer la précision et la stabilité pendant l'entraînement.

4.2. Performances Observées

```
accuracy, f1, precision, recall = evaluate_model_metrics(model, test_loader)
```

```
Évaluation: 100%|██████████| 372/372 [11:31<00:00, 1.86s/it]  
Accuracy   : 0.9844  
F1 Score   : 0.7823  
Precision  : 0.7470  
Recall     : 0.8211  
IoU        : 0.6289  
Dice       : 0.7221
```

Figure 8: Performance

4.3. Visualisation

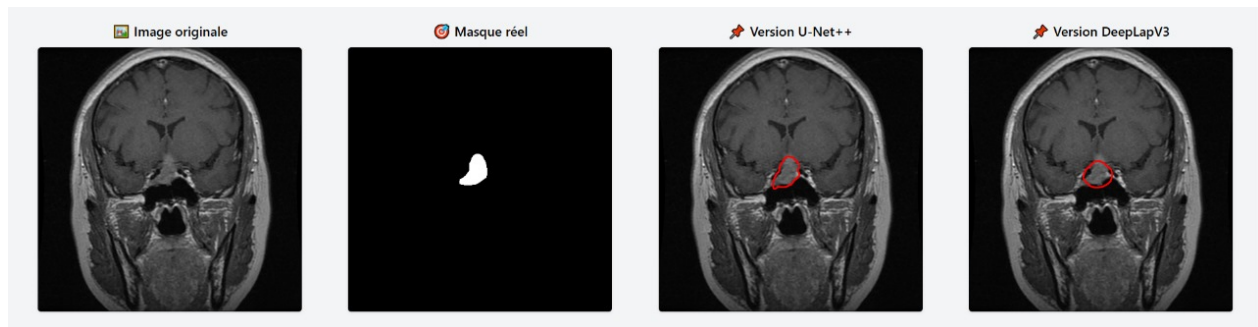


Figure 9: Exemple de segmentation brain

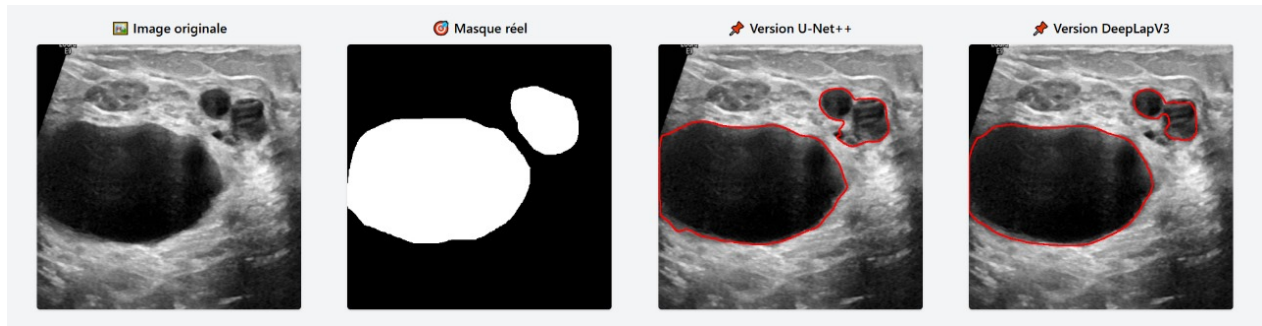


Figure 10: Exemple de segmentation breast

- **Première image originale** : représente l'image d'entrée telle qu'elle est avant toute transformation ou traitement.
- **Deuxième image (masque réel)** : masque d'annotation manuel, considéré comme la vérité terrain (*ground truth*) pour l'entraînement et l'évaluation.
- **Troisième image (résultat de segmentation avec U-Net++)** : sortie du modèle U-Net++ appliqué sur l'image originale, illustrant la capacité du modèle à segmenter les régions d'intérêt.
- **Quatrième image (résultat de segmentation avec DeepLabV3)** : sortie du modèle DeepLabV3 appliqué sur la même image, permettant une comparaison directe avec U-Net++ pour évaluer les performances respectives.

5. Comparaison des Architectures : U-Net vs U-Net++ vs DeepLabV3

Pour analyser l'impact de l'architecture du modèle sur la segmentation, nous avons comparé trois variantes populaires : **U-Net**, **U-Net++** et **DeepLabV3**, toutes avec le même backbone ResNet34. L'entraînement a été réalisé dans les mêmes conditions pour garantir une évaluation équitable.



Accuracy : 0.6434 F1 Score : 0.2410 Precision : 0.1373 Recall : 0.9878	Évaluation: 100%  372/372 [11:31:00:00, 1.86s/it] Accuracy : 0.9844 F1 Score : 0.7823 Precision : 0.7470 Recall : 0.8211 IoU : 0.6289 Dice : 0.7221	Évaluation: 100%  372/372 [19:46:00:00, 3.19s/it] Accuracy : 0.9884 F1 Score : 0.8335 Precision : 0.8201 Recall : 0.8472 IoU : 0.6334 Dice : 0.7315
: U-Net	: U-Net++	: DeepLabV3

Figure 11: Comparaison des performances des trois architectures de segmentation

Évaluation des performances des modèles

1. Accuracy (Précision globale)

- **Deeplabv3** : 0.9884 (meilleur)
- **Performance** : 0.9844
- **Res_UNet** : 0.6434 (très faible)

Interprétation : Deeplabv3 et Performance présentent une précision globale très élevée, tandis que Res_UNet est nettement moins performant, ce qui suggère des erreurs de classification importantes.

2. F1 Score (Équilibre entre précision et rappel)

- **Deeplabv3** : 0.8335 (meilleur)
- **Performance** : 0.7823
- **Res_UNet** : 0.2410 (très faible)

Interprétation : Deeplabv3 montre le meilleur équilibre entre précision et rappel. Res_UNet a un F1 très bas, indiquant un déséquilibre marqué (rappel élevé mais précision très faible).

3. Precision (Exactitude des prédictions positives)

- **Deeplabv3** : 0.8201 (meilleur)
- **Performance** : 0.7470
- **Res_UNet** : 0.1373 (très faible)

Interprétation : Deeplabv3 a la plus haute précision, tandis que Res_UNet génère beaucoup de faux positifs (seulement 13.7% des prédictions positives sont correctes).

4. Recall (Capacité à détecter les vrais positifs)

- **Res_UNet** : 0.9878 (meilleur, mais peu fiable)
- **Deeplabv3** : 0.8472
- **Performance** : 0.8211

Interprétation : Res_UNet a un rappel presque parfait, mais couplé à une précision très faible, cela suggère qu'il surclasse les positifs (par exemple, en marquant presque tout comme positif). Deeplabv3 et Performance sont plus équilibrés.

5. IoU (Intersection over Union) et Dice (Similarité)

- **Deeplabv3** : IoU = 0.6334, Dice = 0.7315 (meilleurs)

- **Performance** : IoU = 0.6289, Dice = 0.7221
- **Res_UNet** : Métriques non fournies

Interprétation : Deeplabv3 et Performance sont très proches pour la segmentation, avec un léger avantage pour Deeplabv3. Res_UNet n'a pas ces métriques, mais ses autres scores laissent présager des performances médiocres.

6. Temps d'inférence (Efficacité)

- **Performance** : 1.86 s/it (le plus rapide)
- **Deeplabv3** : 3.19 s/it (plus lent)
- **Res_UNet** : Non indiqué

Interprétation : Performance est presque deux fois plus rapide que Deeplabv3, ce qui peut être crucial pour des applications en temps réel.

Synthèse des résultats

Table 2: Comparaison synthétique des performances des modèles

Métrique	Performance	Deeplabv3	Res_UNet
Accuracy	0.9844	0.9884	0.6434
F1 Score	0.7823	0.8335	0.2410
Precision	0.7470	0.8201	0.1373
Recall	0.8211	0.8472	0.9878
IoU	0.6289	0.6334	-
Dice	0.7221	0.7315	-
Vitesse	1.86 s/it	3.19 s/it	-

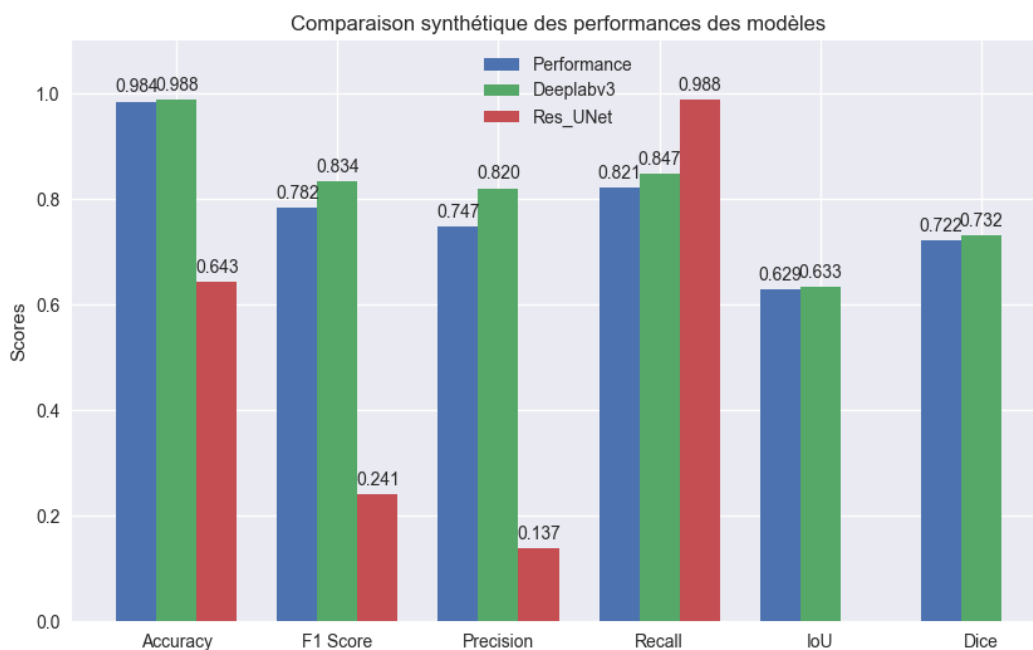


Figure 12: Comparaison synthétique des performances des modèles

Conclusion

Deeplabv3 est globalement le meilleur modèle, avec des scores élevés en précision, F1 et métriques de segmentation. Performance est légèrement moins performant mais reste plus rapide, ce qui peut constituer un compromis intéressant. Res_UNet semble dysfonctionnel (surreprésentation des positifs), nécessitant une révision du modèle ou des données.

6. Conclusion

Cette étude a systématiquement évalué différentes architectures de segmentation pour les tumeurs cérébrales et mammaires, établissant que U-Net++ offre le meilleur compromis performance/précision. Bien que des défis persistent (segmentation des petites lésions, robustesse aux artefacts), les résultats démontrent le potentiel clinique de ces approches. Les futures recherches devraient intégrer des mécanismes d'attention et explorer l'apprentissage semi-supervisé pour réduire la dépendance aux annotations manuelles. Ce travail constitue une étape significative vers l'intégration de l'IA dans les workflows diagnostiques en oncologie.

Références

- Dataset Cerveau : <https://www.kaggle.com/datasets/tinashri/brain-tumor-dataset-include>
- Dataset Sein : <https://www.kaggle.com/datasets/aryashah2k/breast-ultrasound-images-dat>