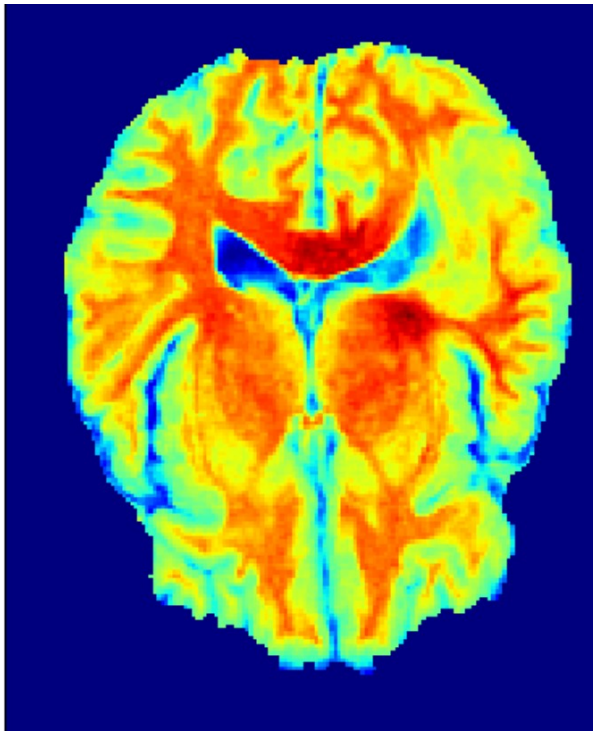
Formation Keras

Tips and tricks
Template Keras

Format de la présentation

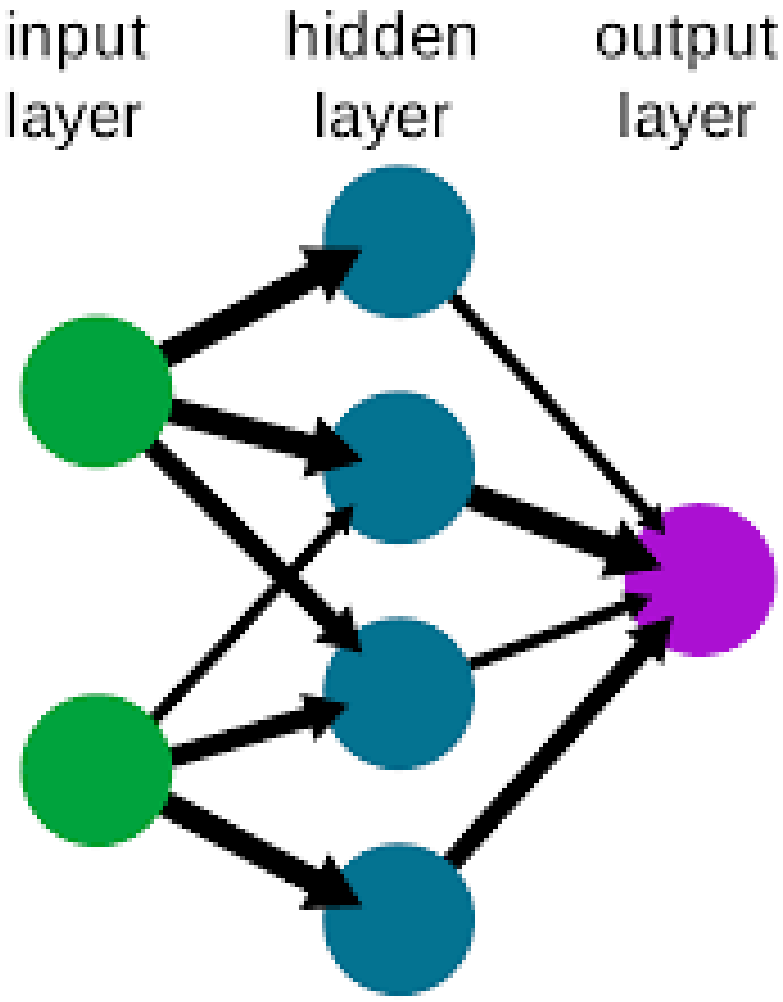
1. Introduction aux CNN
   - Layers
   - Activations
   - Data format
2. Data pre-processing
   - Normalization
3. Fonctions de coût
4. Outils de visualisation
5. Exemples et recommandations



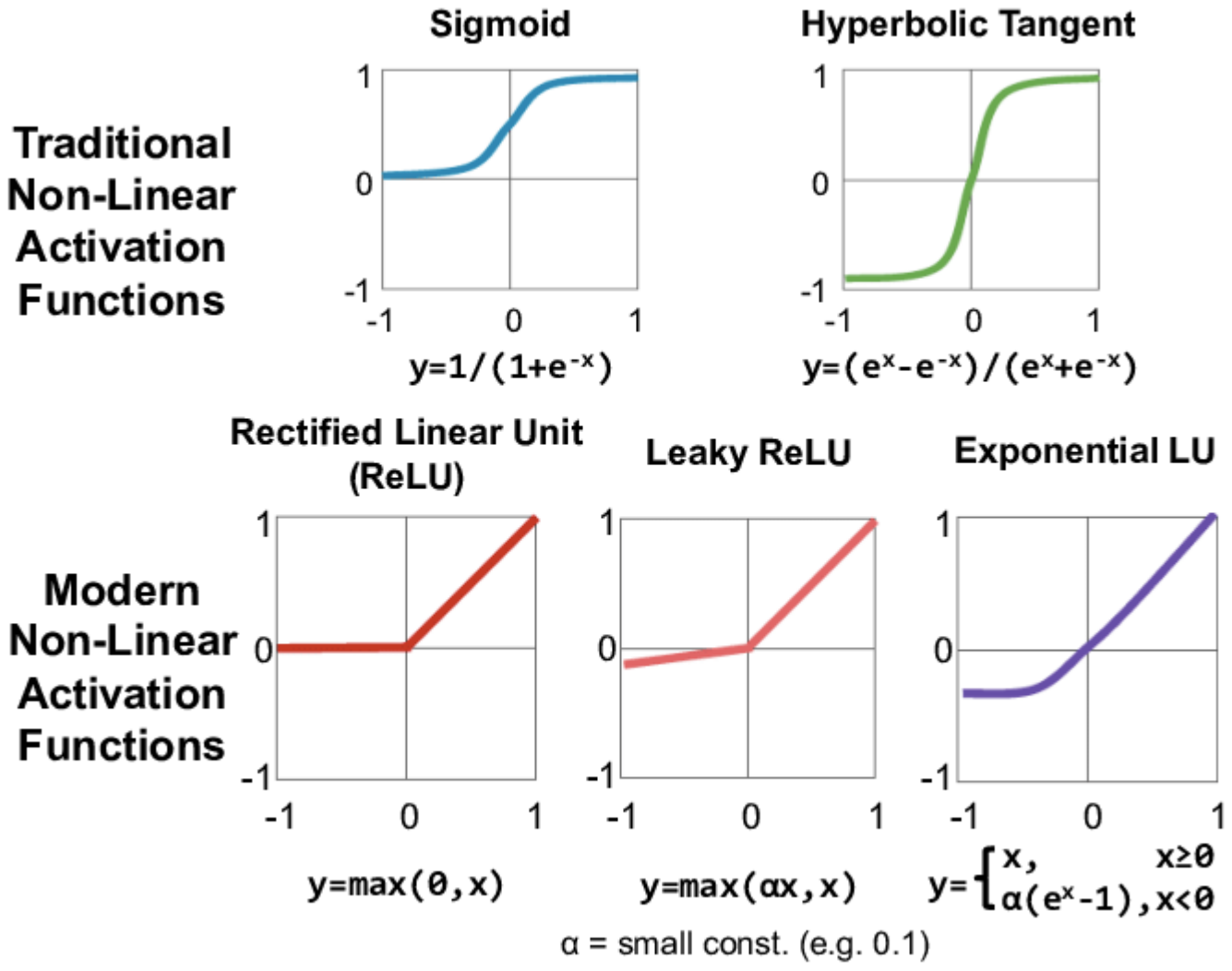
Type de couches

Couches pleinement connectées
‘Couches’ d’activation
Couches de normalisation
Couches de convolutions
Couches de ré-échantillonnage

Couche pleinement connectée



Fonctions d'activation



Couches de normalisation

La normalisation des features est nécessaire! On veut garder les features proche de 0 pour rester dans le range non-linéaire des activations

2 types principaux de normalisation

1) Batch normalization
   - Compute moving mean/variance of each feature maps across the batch, then normalize feature maps
2) Instance normalization
   - Normalize feature maps based on individual examples mean/variance

Couches de normalisation

Quand utiliser l'une ou l'autre?

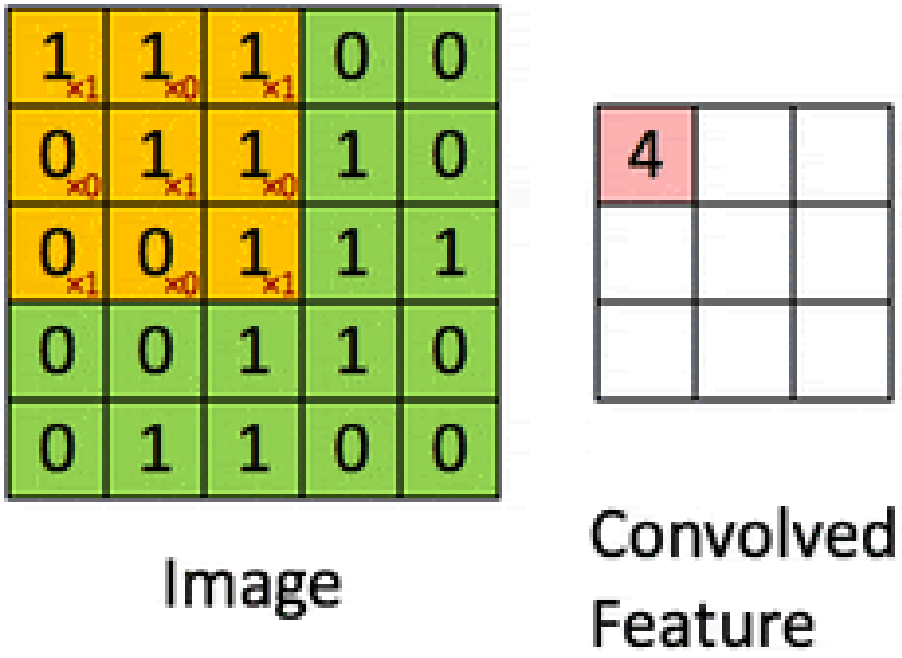
Batch Norm:

- Utiliser quand l'intensité absolue des images en input est informative (Ex: CT)
- ***Comportement différent en test!***

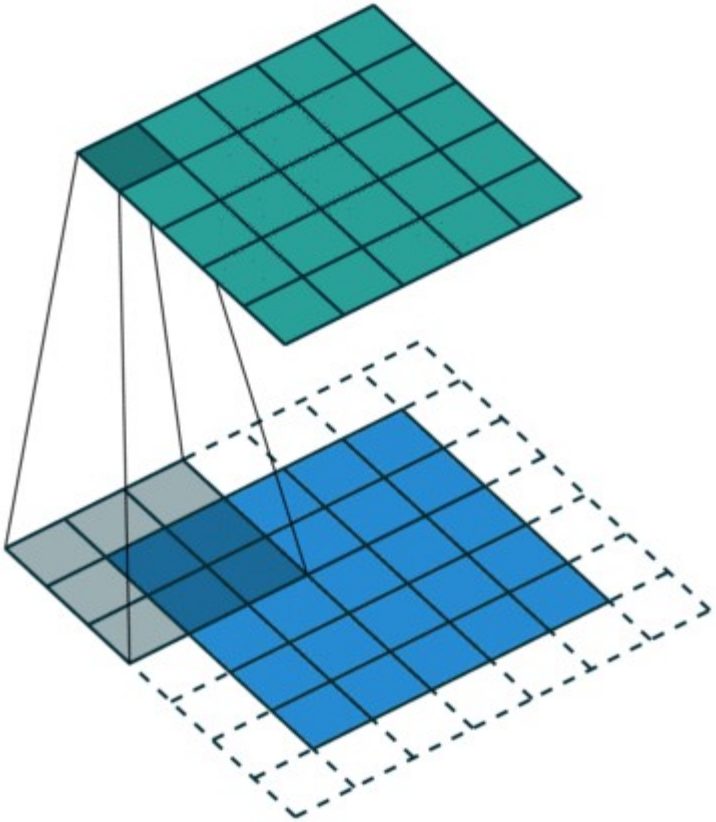
Instance Norm:

- Batch Norm peut quand même être utilisée quand l'intensité absolue des images est relative (Ex: IRM)
- Si le batch size est petit (< 4), utiliser Instance Norm

Rappel: Convolution

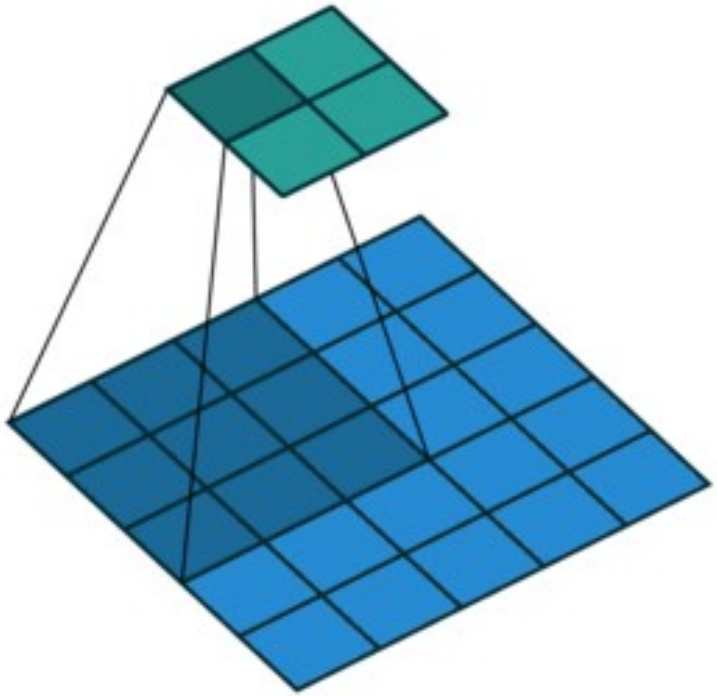


Réseaux de neurones de convolution (CNN)



‘valid’ --> sans padding
‘same’ --> avec padding

Convolution avec pas

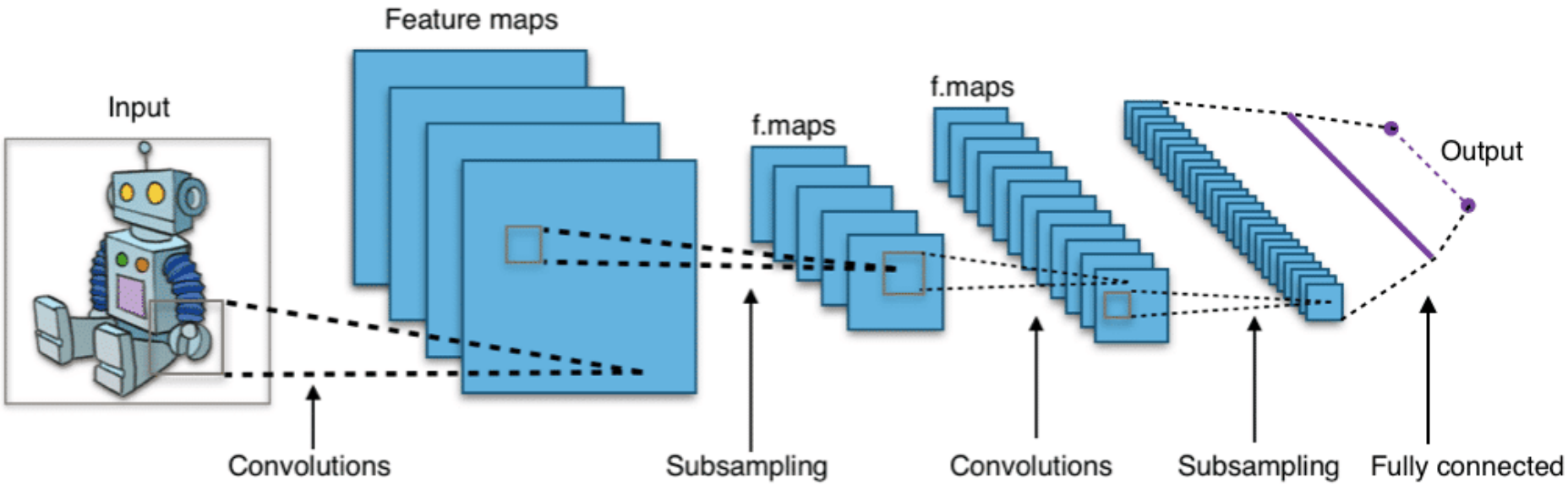


Couches de ré-échantillonnage

Utile pour augmenter la taille des feature maps (upsampling) et pour la diminuer (downsampling)

Downsampling peut être fait par convolution avec pas
Upsampling peut être fait par convolution transposée

Réseaux de neurones de convolution (CNN)



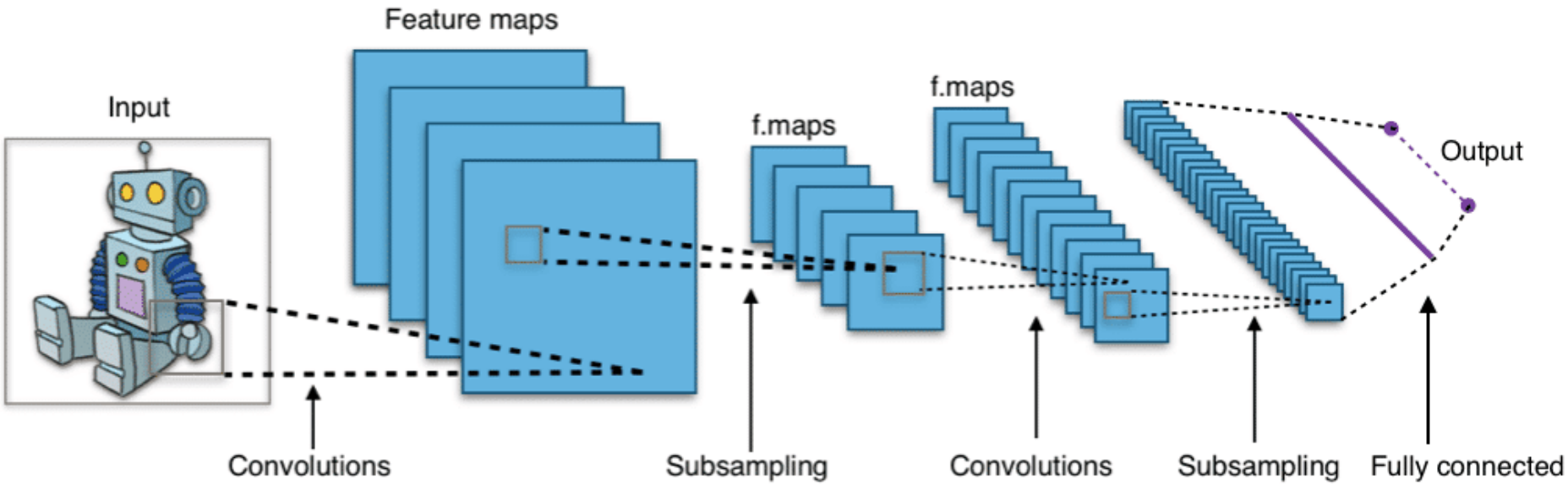
Réseaux de neurones de convolution (CNN)

Souvent, structure en blocs:

On agence une série d'opérations qu'on répète.

Ex: Dowmsampling block

Réseaux de neurones de convolution (CNN)



Dimensions: (b_size, height, width, f_m)
Dimensions weights: (f_m_0, k_size, k_size, f_m_1)

Considérations de taille

Taille des images --> doit rentrer dans la RAM

- Réduire la taille de batch
- Réduire le nombre de feature maps
- Réduire la taille de l'image (downsampling, strided convolution, cropping)

Taille des poids --> trop de paramètres peut overfitter

- Garder les convolutions à 3x3(x3)
- Alterner la dimension (7x1 suivi de 1x7)
- Réduire le nombre de feature maps
- Préférable d'avoir un réseau plus profond qu'un réseau plus large

Pre-processing

Bien préparer ses données est **crucial**

Au même titre qu'on normalise les features, il faut normaliser l'input. La normalisation peut varier, mais elle est essentielle.

- Min-max scaling
- Z-normalization
- Tissue-normalization
- Histogram equalization

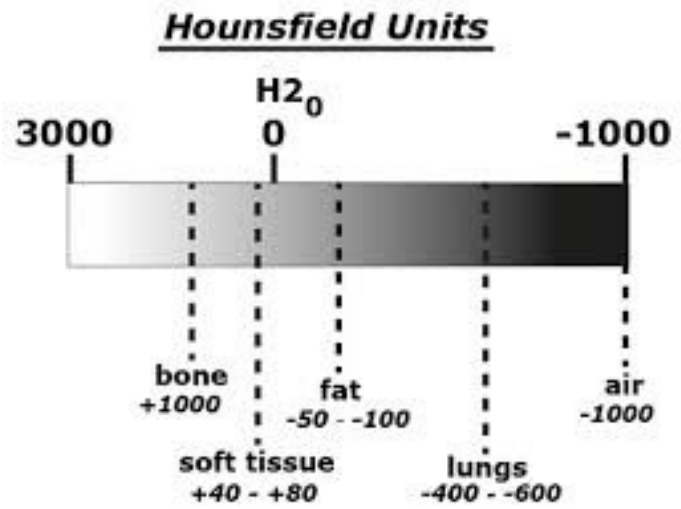
Beware of the background!

Pre-processing

Normalisation par exemple vs normalisation par dataset -->

Similaire à la considération entre batch norm et instance norm

Ex. pour dataset CT, normalisation par exemple ne fait pas de sens, mais bon pour IRM!



Pre-processing

Il est aussi important de limiter la taille de l'input

Traditionnellement, par cropping. On veut garder uniquement la partie intéressante de l'image

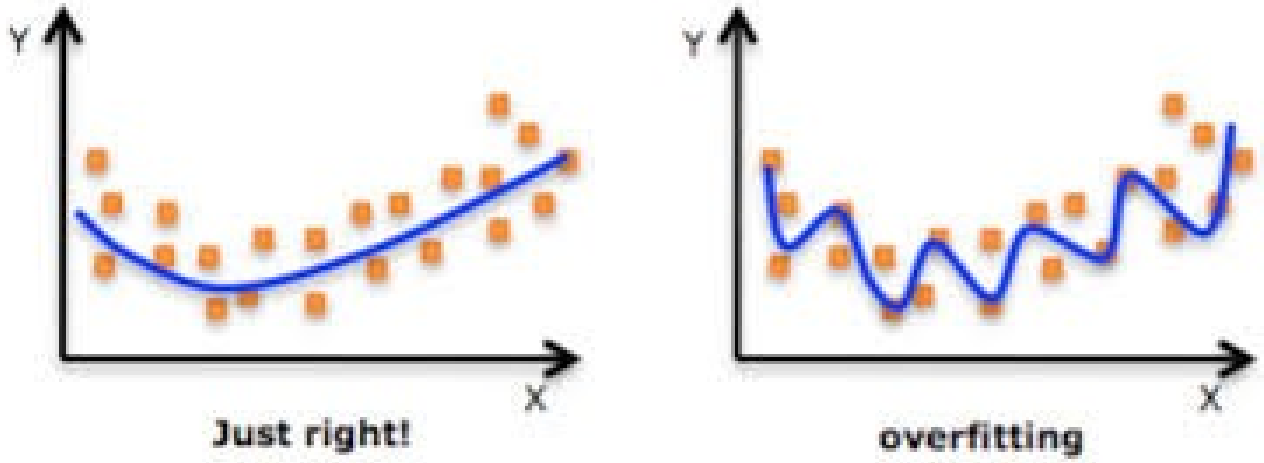
Downsampling peut être utilisé parfois, si la résolution est très élevée.

Historiquement, il s'agit de la raison pourquoi la plupart des réseaux en imagerie médicale sont 2D

Régularisation

Un réseau entraîné sur trop peu d'exemple va 'overfitter'
Plus un réseau est complexe (haut nombre de poids)

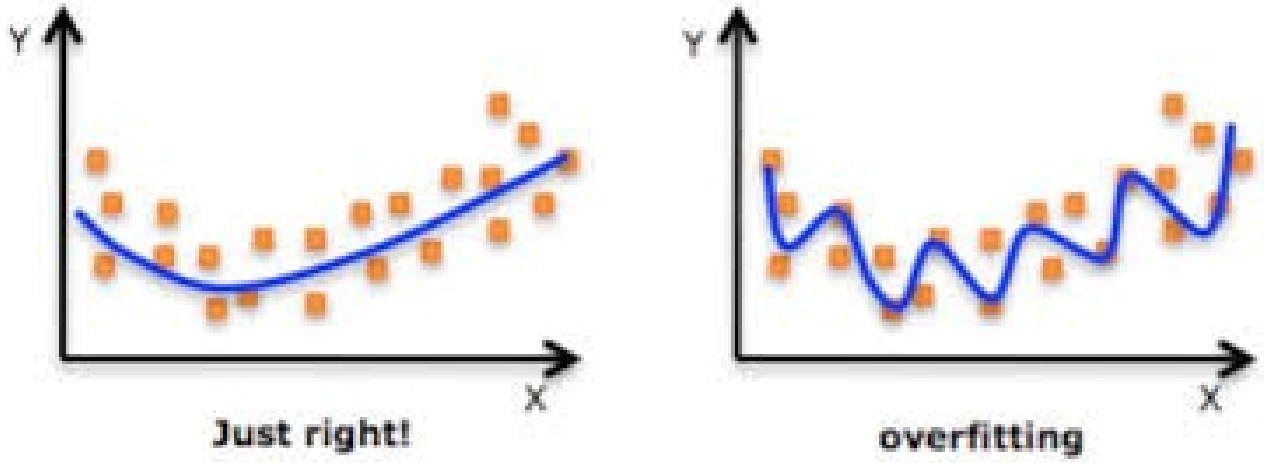
La régularisation est une méthode pour rendre la tâche plus dure pour le réseau, le forçant à adopter une fonction plus simple



Régularisation

Ex de régularisation:

- Data augmentation (rotation, translation)
- Régularisation de poids (L1, L2)
- Dropout
- Multi-task network



Cost functions

Sans bonne fonction de coût, même un super réseau n'apprendra pas.

Important d'avoir la fonction de coût appropriée

Classification: Softmax cross-entropy
Classification multi-classe: Binary cross-entropy (BCE)
Regression: MSE, MAE
Reconstruction: MAE, SSIM
Segmentation: DICE, BCE

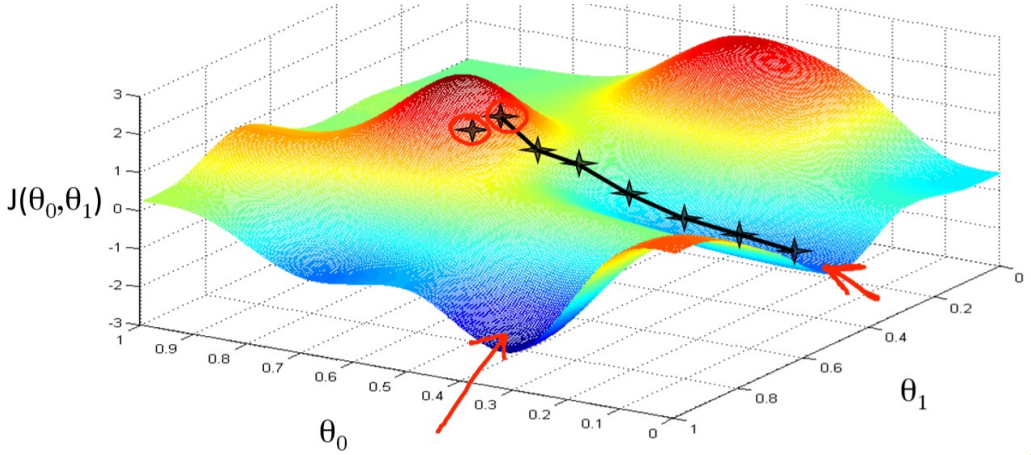
Optimiseurs

Les optimiseurs sont des algorithmes de descente de gradient qui optimisent la fonction de coût.

Optimiseur de base: SGD (stochastic gradient descent)

Optimiseur par défaut: Adam

Optimiseur à utiliser si disponible: lazy Adam

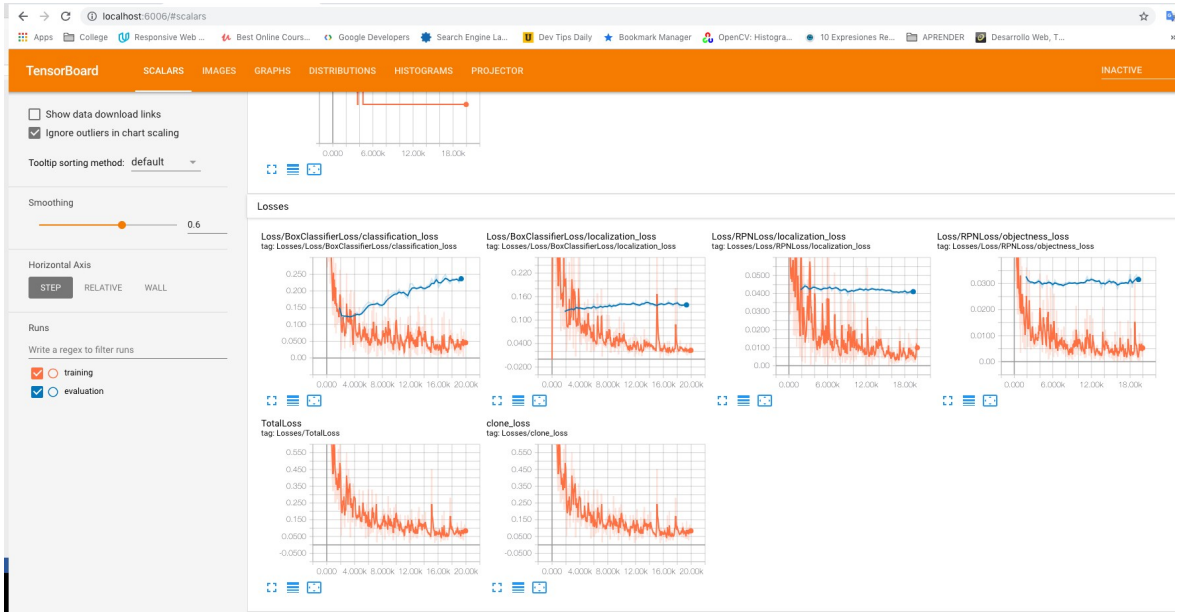


Tensorboard (X)

Logiciel de monitoring/visualisation par excellence

Permet d'enregistrer images, fonctions de coût, output du réseau, gradients, texte, graphes, licornes, etc

On le fait rouler sur un port qu'on accède par un browser



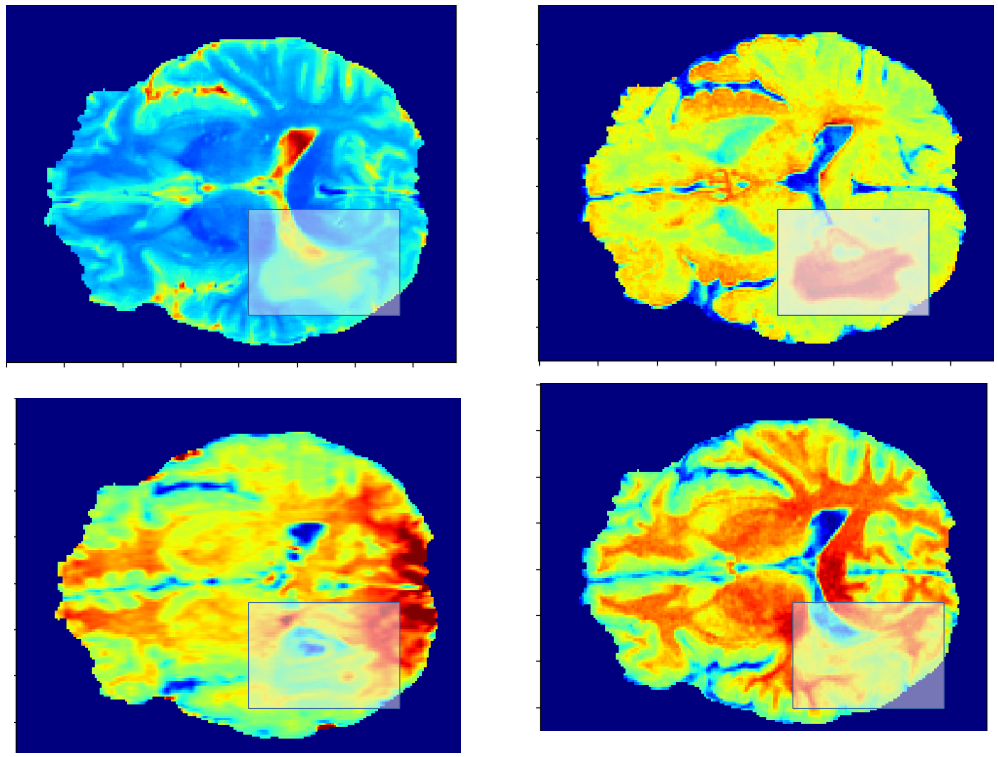
Exemple pratique en Keras

Dataset: BRATS

Objectif: segmentation

4 modalité en input

4 régions différentes



Par ou commencer?

Simplifions le problème:

Architecture simple (FCCN)

Gardons 1 modalité (T1)

Segmentons 1 classe

Faisons le en 3D!

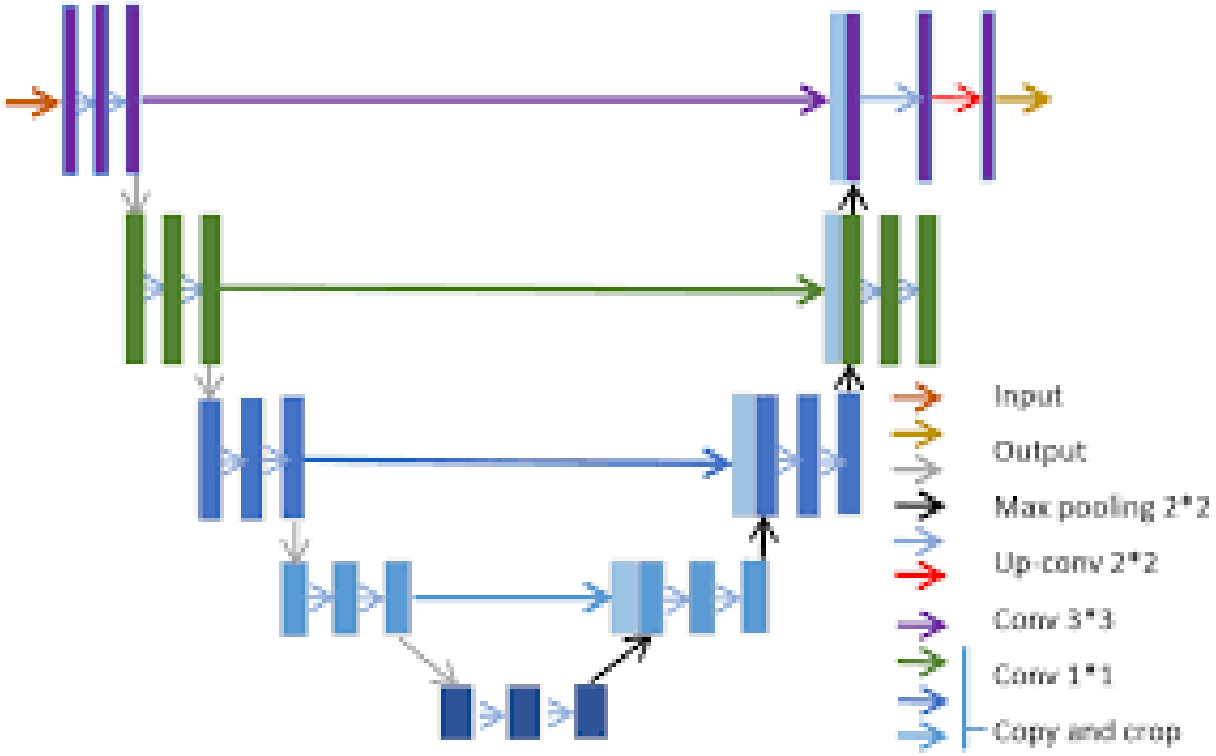
Par ou commencer?

La première chose à faire, c'est de séparer le jeu de données en train/validation/test. Le jeu de test est gardé séparé **JUSQU'À LA FIN**

Dans Keras (et TF2) on peut passer directement les arrays Numpy dans la méthode fit. Si le dataset ne rentre pas en mémoire, utiliser les pipeline de données (classe Dataset)

Modèle

En segmentation, normalement on utilise l'architecture U-Net.



Premiers résultats

On optimise une fonction de coût BCE!

Notre fonction de coût diminue steadily de 0.7 jusqu’à 0.06!

Allons voir la segmentation résultante

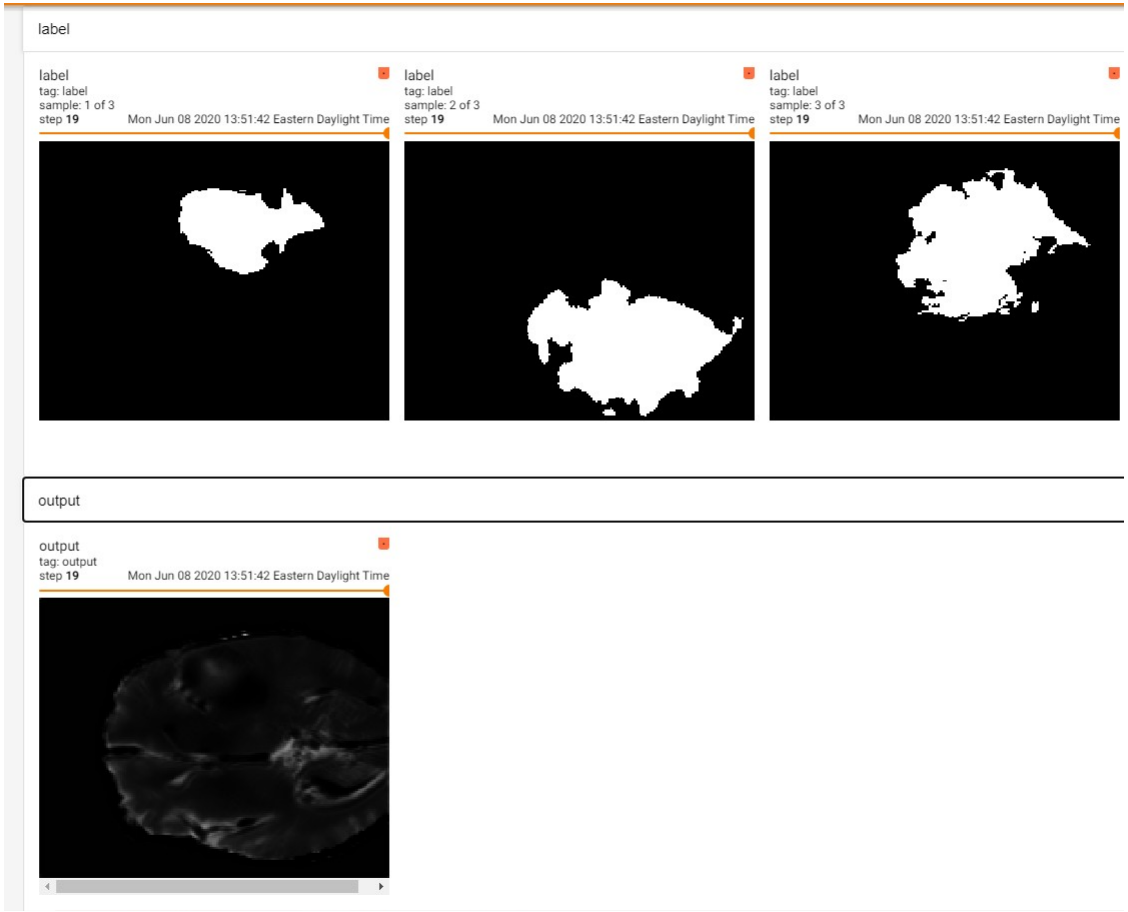
Premiers résultats

On optimise une fonction de coût BCE!

Notre fonction de coût diminue steadily de 0.7 jusqu’à 0.06!

Allons voir la segmentation résultante

Premiers résultats

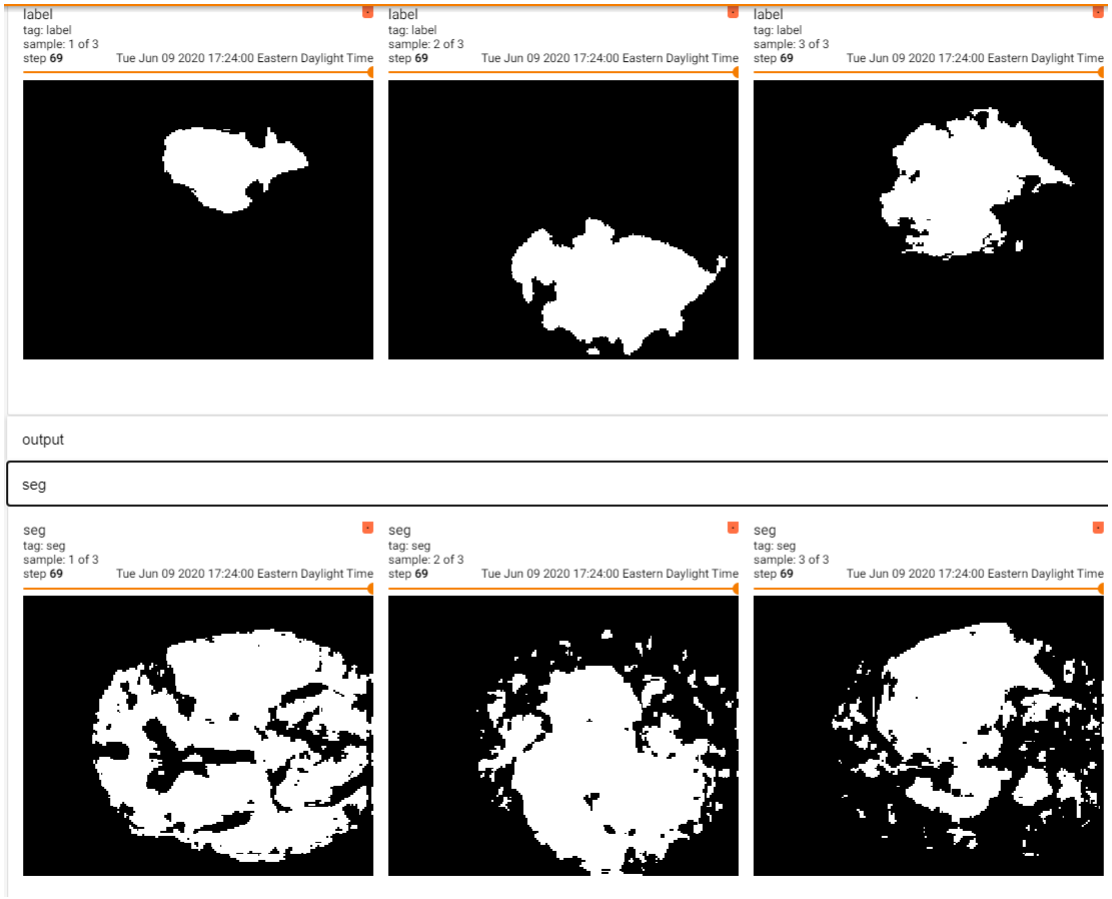


BCE pondérée

Comme la BCE normale ne segmente rien, on peut augmenter la pondération de la fonction de coût à l'intérieur de la lésion

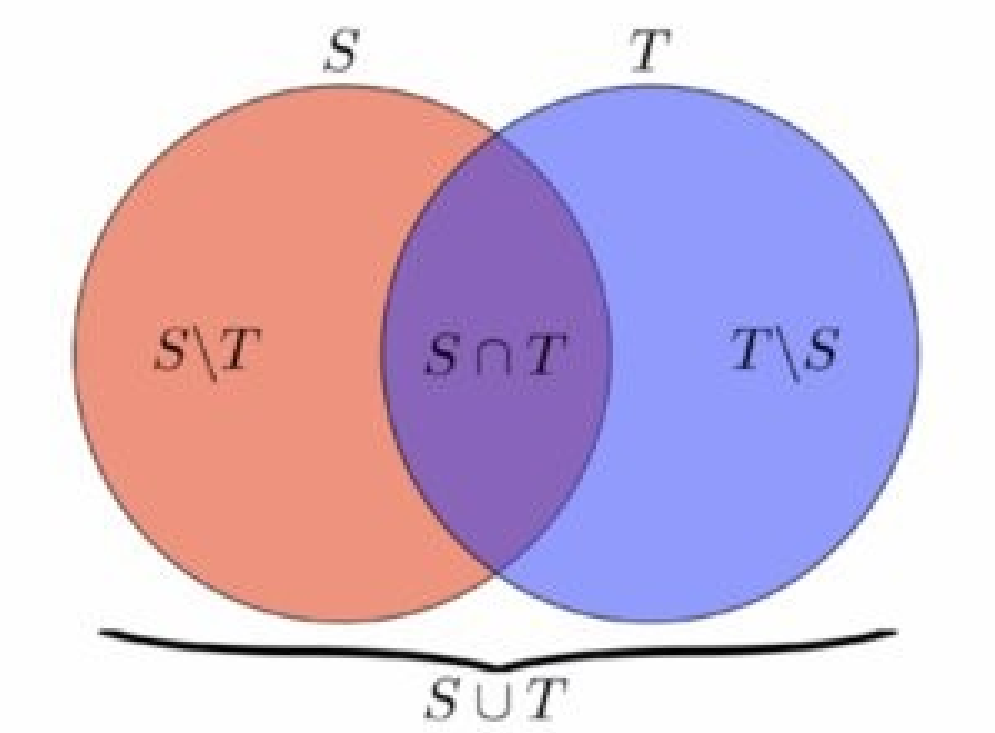
Voyons si ça aide!

BCE pondérée



Dice loss

Dice score = 2 * | S ∩ T | / | S | + | T |

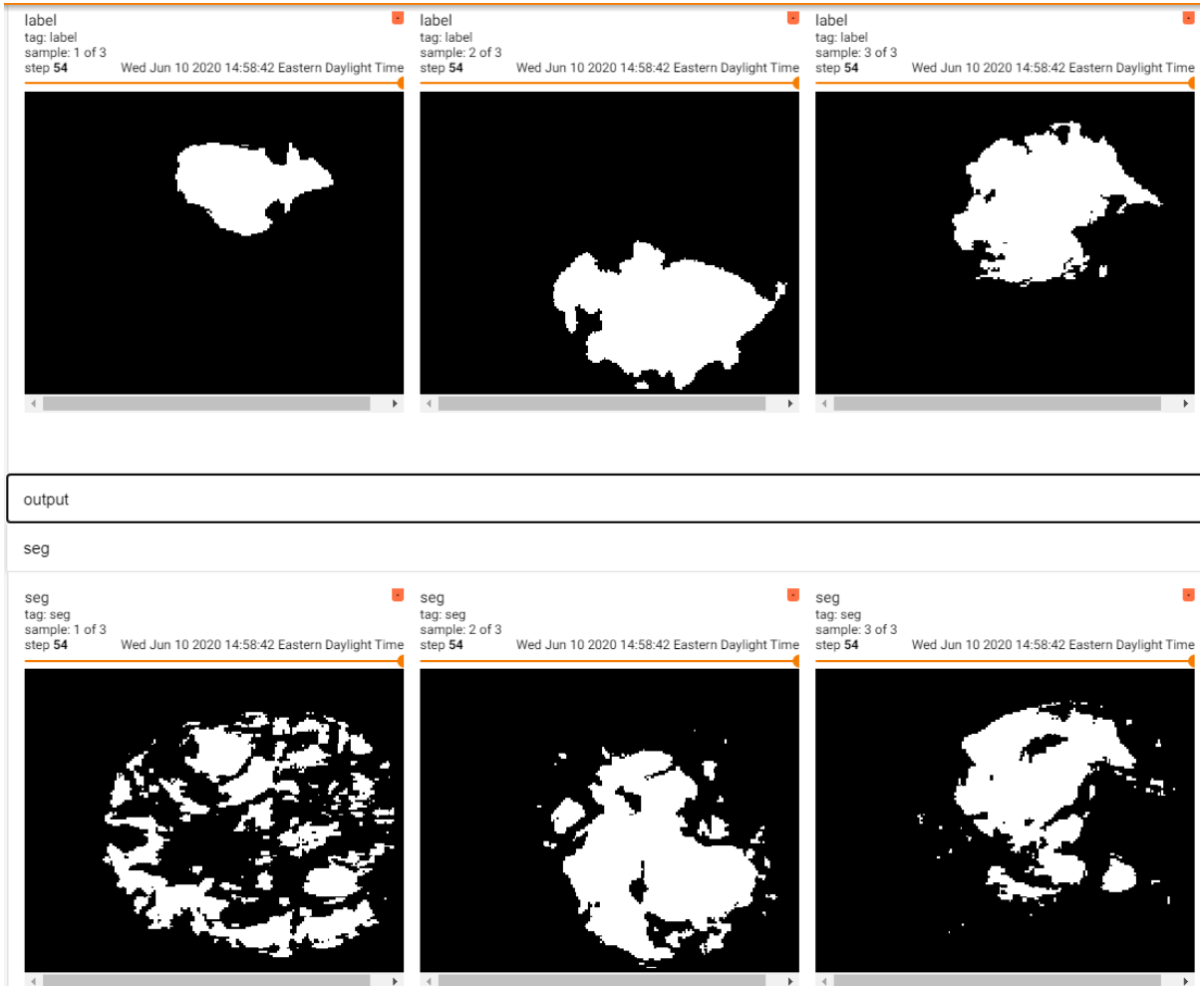


Dice loss

On a bien vu que BCE non pondéré donne des résultats médiocres.

Comment le réseau se comporte avec une Dice loss?

Dice loss


label
tag: label
sample: 1 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time
label
tag: label
sample: 2 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time
label
tag: label
sample: 3 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time
output
seg
seg
tag: seg
sample: 1 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time
seg
tag: seg
sample: 2 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time
seg
tag: seg
sample: 3 of 3
step 54
Wed Jun 10 2020 14:58:42 Eastern Daylight Time

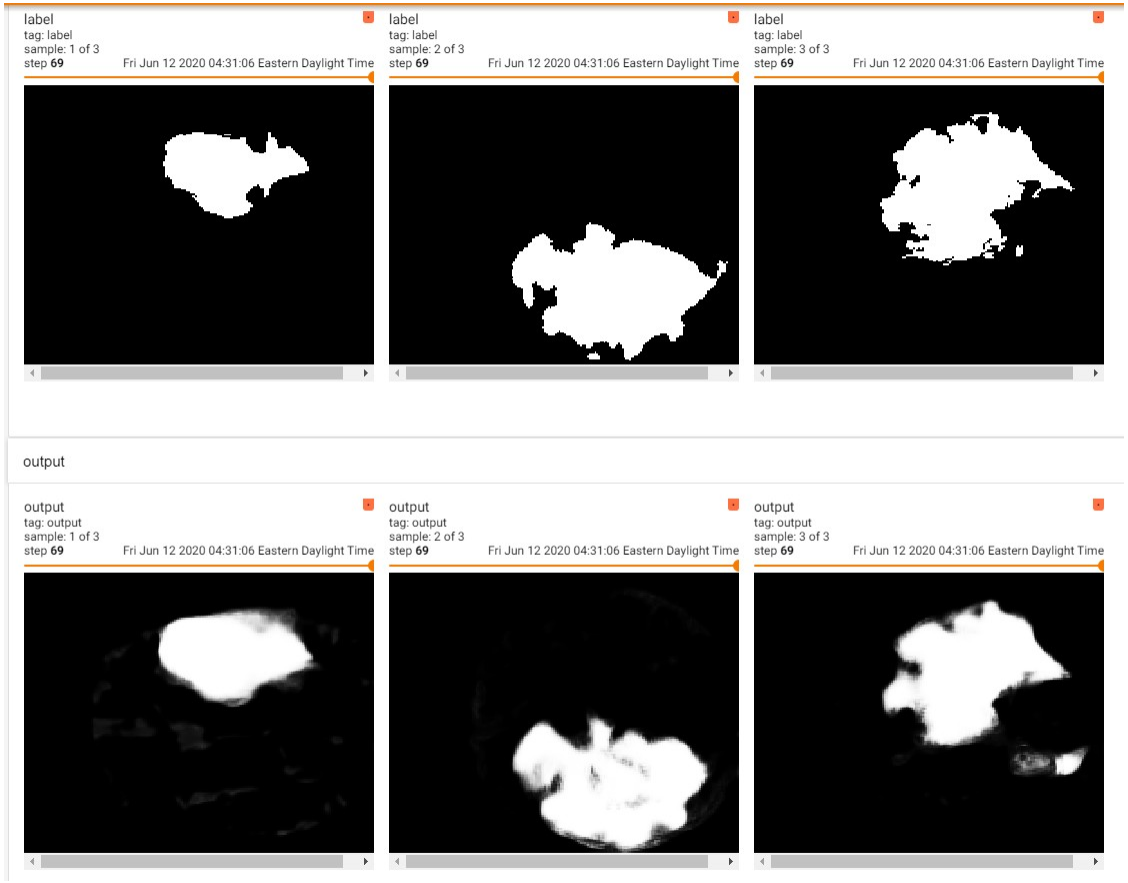
T1 est insuffisant

On voit que le réseau a de la misère à segmenter utilisant uniquement T1. Comment le faire utiliser les autres modalités?

On va avoir un input de taille
(b_size, height, width, depth, 4)

Ré-entraînons avec ça!

T1 est insuffisant



Comment continuer

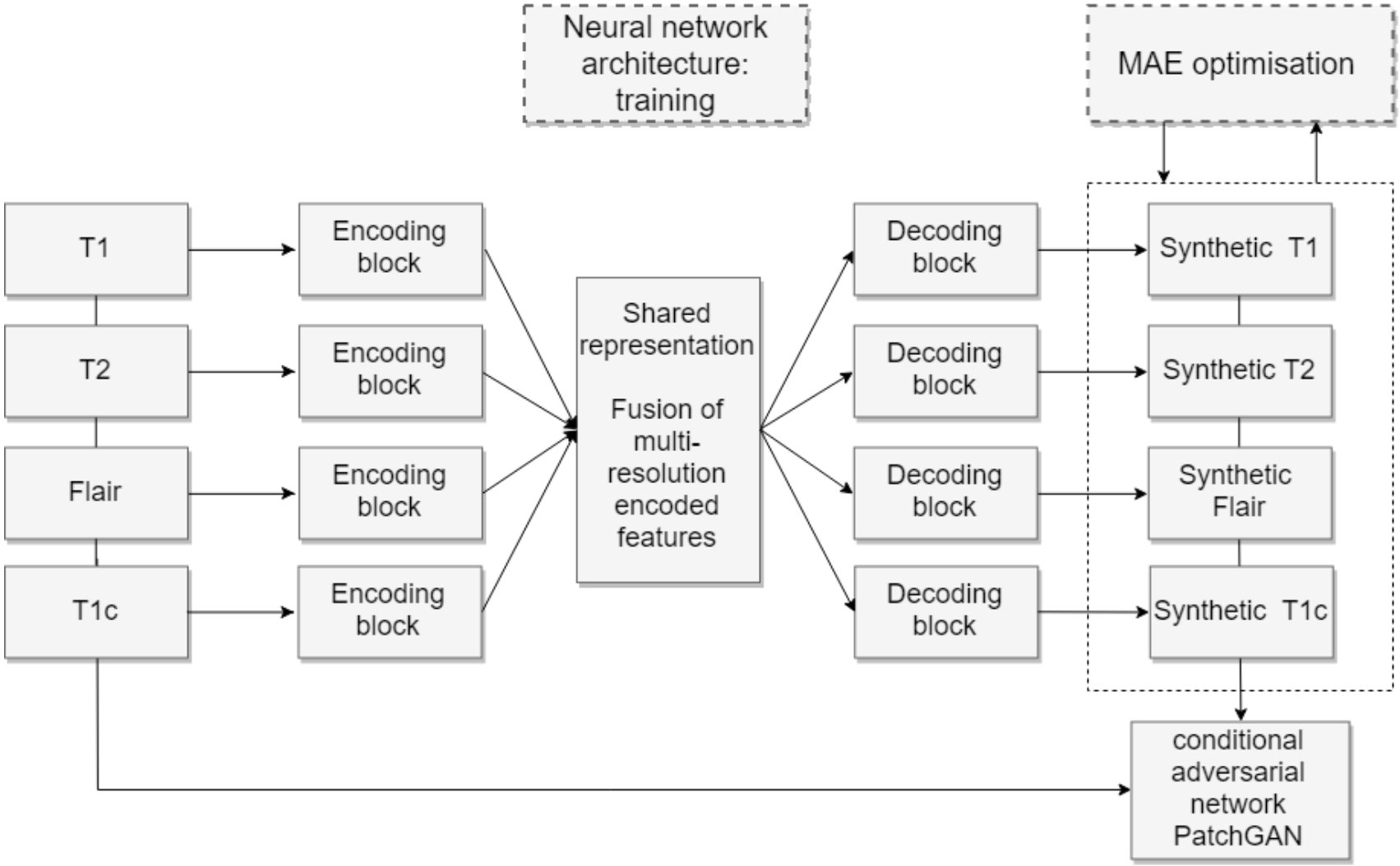
Faire la segmentation des 4 classes

Implémenter le U-Net

Ajouter de l’augmentation de données/régularisation

Découpler les modalités?

Comment continuer



Ou accéder au code?

Je vais partager le template sur le git du groupe

Surveillez vos emails!

Principalement, il vous faudra modifier la partie pour loader vos propres données!