Histopathologic cancer detection : premiers essais

L'approche pour les débuts sur la détection histopathologique du cancer était la suivante : commencer petit, et s'améliorer au fur et à mesure.

Premiers tests

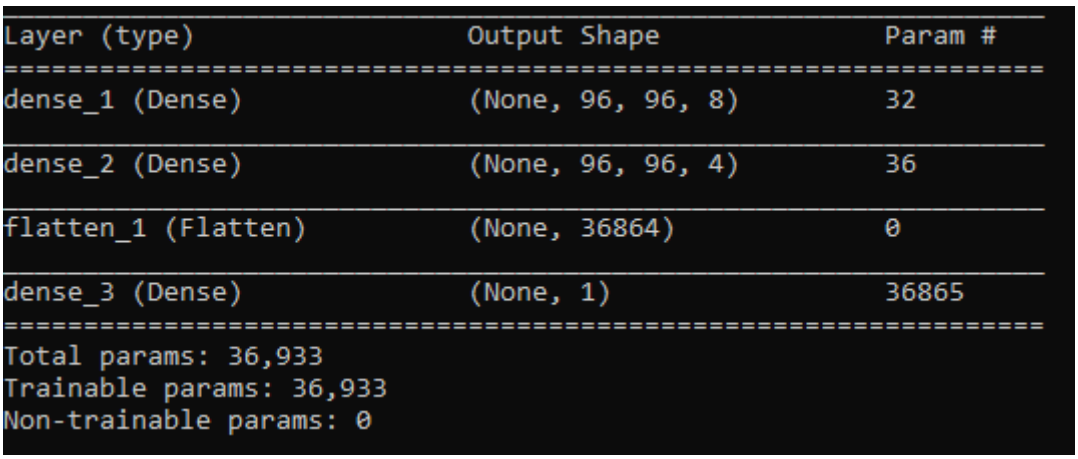
Nous avons donc commencé avec des modèles simples, sur des bases de données simples :

- 20 données d'entrainement
- 10 données de validation
- 10 données de test.

Concernant le réseau, il était lui aussi très simple : deux couches denses, avec peu de neurones. Le but d'une telle implémentation n'est pas d'obtenir des résultats valides, mais plutôt de valider le workflow, et donc :

- de mettre au point les procédés de récupération des données (mise en forme du jeu de données)
- d'implémenter la récupération et le préprocess des données (augmentation, normalisation, mise en place de générateurs)
- d'implémenter un réseau de neurone valide pour ces données (inputs et outputs compatibles)
- de mettre en place les procédés d'entrainement, de tests, et de suivi des performances.

On commence donc avec un tout petit réseau pour le travail de ce workflow :



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 96, 96, 8)         32
_________________________________________________________________
dense_2 (Dense)              (None, 96, 96, 4)         36
_________________________________________________________________
flatten_1 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 36865
=================================================================
Total params: 36,933
Trainable params: 36,933
Non-trainable params: 0
```

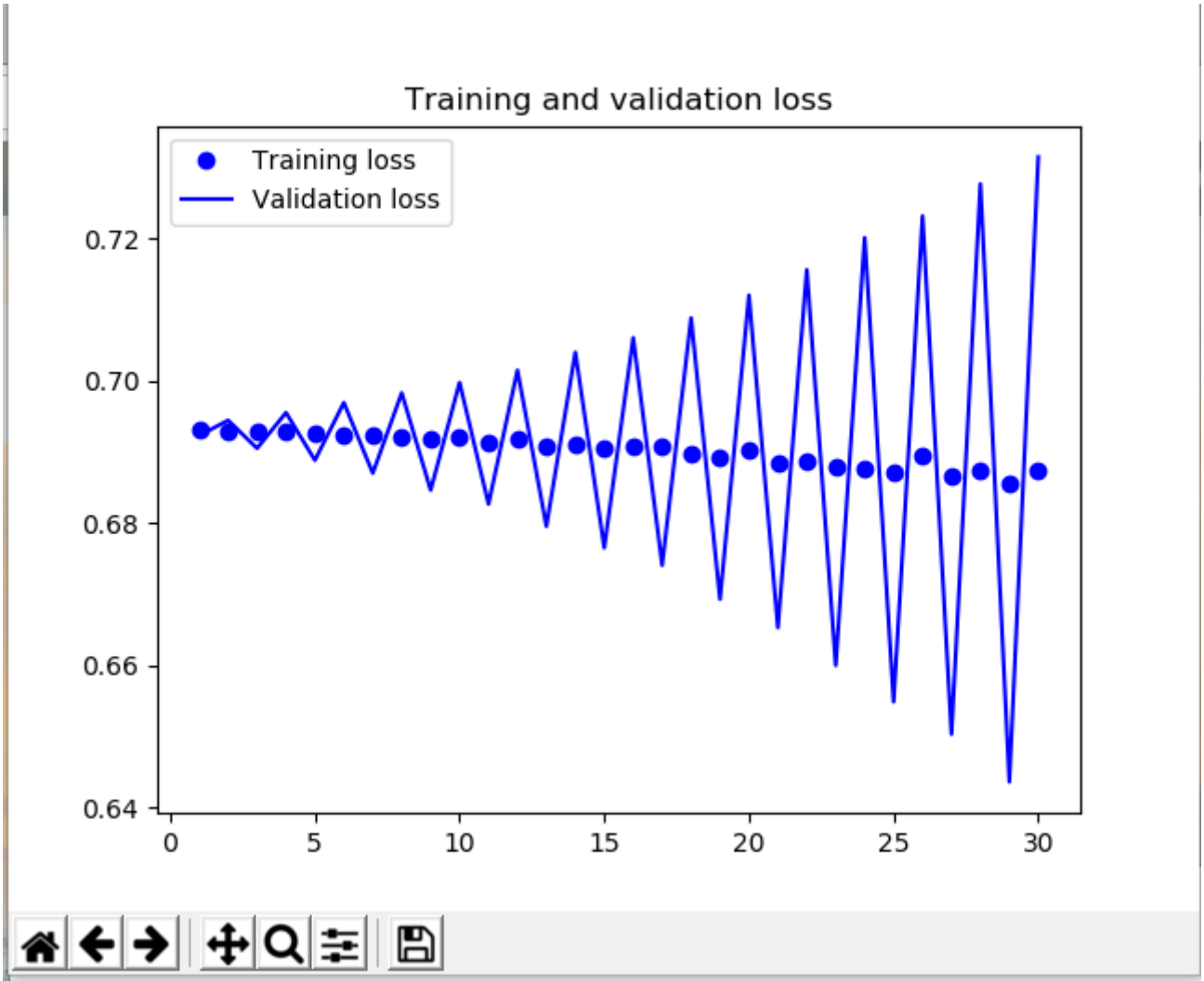
Les résultats obtenus avec ce genre de réseau sur des données aussi peu fournies ne sont pas intéressants ; nous avons donc commencé à utiliser des bases de données plus fournies, sur des réseaux plus adaptés et plus gros :

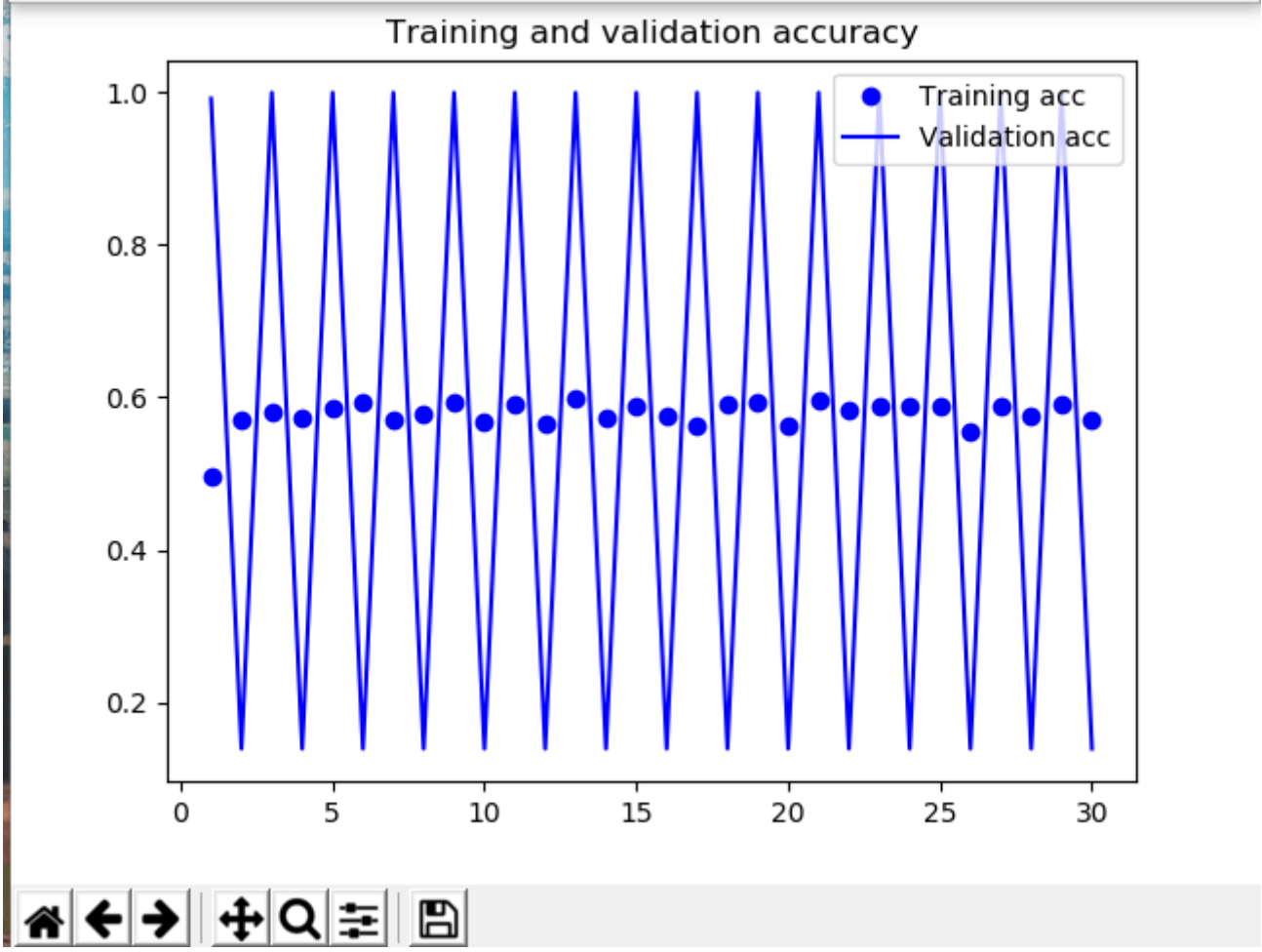
- 800 données en entrainement
- 200 données en validation
- 1000 données en test

  Le réseau utilisé est un petit réseau de convolution 2D. Ces réseaux sont particulièrement efficaces pour traiter les problèmes relatifs au traitement d'images. Il est de la forme suivante :

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 94, 94, 64)        1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 47, 47, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 45, 45, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6400)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6400)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                204832
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 280,513
Trainable params: 280,513
Non-trainable params: 0
```

Les résultats obtenus, après entrainement, sont observables sur les graphes suivants :


Training and validation loss
Training loss
Validation loss
0.72
0.70
0.68
0.66
0.64
0
5
10
15
20
25
30


Training and validation accuracy
Training acc
Validation acc
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
25
30

On observe une valeur pratiquement constante pour les données d'entrainement, tandis que les scores de validation sont divergents. Le réseau tel quel n'apprends donc pas. Nous avons avancés plusieurs hypothèses pour expliquer ces résultats :

- Les données sont mal régularisées (erreur dans l'augmentation ou la normalisation)
- Nous travaillons sur trop peu de données
- La taille du batch perturbe l'apprentissage
- Le modèle est trop simple pour représenter efficacement les données
- les paramètres d'entrainement (learning rate, optimizer, ...) sont mauvais
- Les données telles quelles(96x96) ne permettent pas d'extrapoler la présence de cellules cancéreuses en leur centre (32x32).

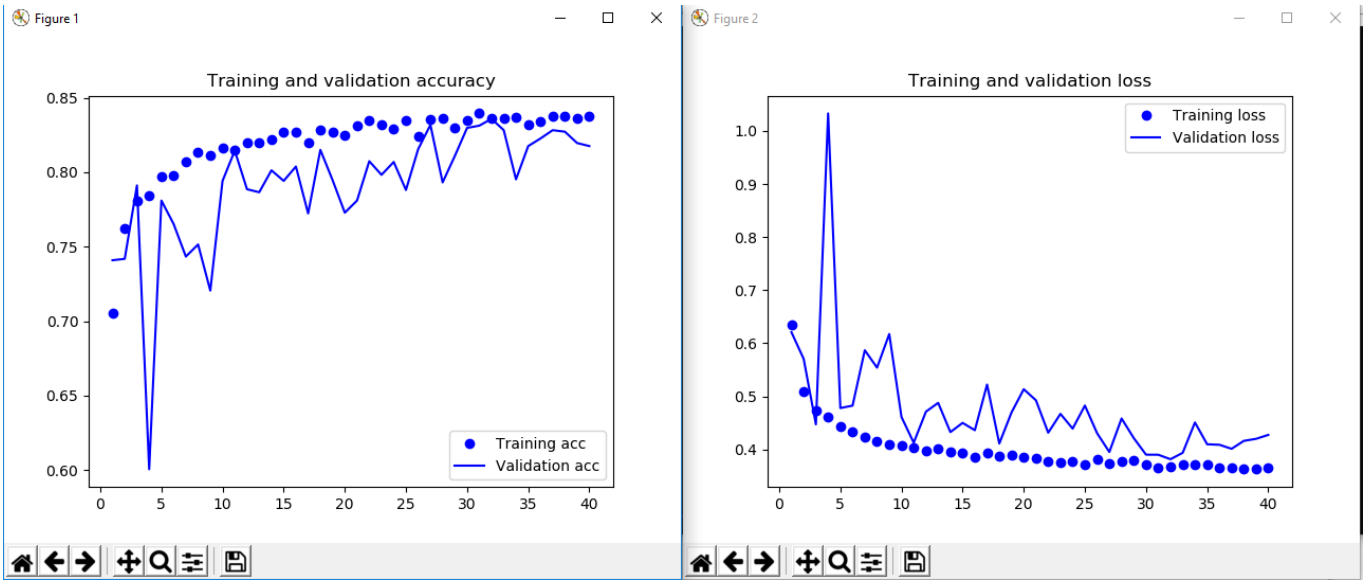
Différentes hypothèses sont rapidement testées (régularisation des données, taille du batch, paramètres d'entrainement, ...) avant de tester la complexification du réseau (plus de données, modèle plus complexe. L'état des tests est alors le suivant :

- 8 000 données en entrainement
- 2 000 données en validation
- 2 000 données en test

Le modèle utilisé est toujours un modèle de convolution, mais plus développé que le précédent :

```
batch_normalization_1 (Batch (None, 96, 96, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 32)        0
_________________________________________________________________
separable_conv2d_2 (Separabl (None, 48, 48, 64)        2400
_________________________________________________________________
activation_2 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
separable_conv2d_3 (Separabl (None, 48, 48, 64)        4736
_________________________________________________________________
activation_3 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
separable_conv2d_4 (Separabl (None, 24, 24, 128)       8896
_________________________________________________________________
activation_4 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
separable_conv2d_5 (Separabl (None, 24, 24, 128)       17664
_________________________________________________________________
activation_5 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
separable_conv2d_6 (Separabl (None, 24, 24, 128)       17664
_________________________________________________________________
activation_6 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
batch_normalization_6 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               4718848
_________________________________________________________________
activation_7 (Activation)    (None, 256)               0
_________________________________________________________________
batch_normalization_7 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
_________________________________________________________________
activation_8 (Activation)    (None, 2)                 0
=================================================================
Total params: 4,774,077
Trainable params: 4,772,477
Non-trainable params: 1,600
```

Les résultats obtenus après entrainement sont observables sur les graphes suivants :



Ce que l'on peut comprendre de ces graphes, c'est :

- que ce réseau apprend bien à reconnaitre des cellules cancéreuses dans les images
- qu'il y a un léger overfit des données d'entrainement par rapport aux données de tests

On obtient dès lors une précision de l'ordre de 0.8 sur notre jeu de données, ce qui est un début correct. Les pistes d'amélioration sont les suivantes :

- Un réseau plus gros serait probablement plus performant
- Travailler sur un plus gros jeu de données pourrait augmenter la précision finale
- Travailler la régularisation (Dropout, Batch normalization, ...) pour éviter l'overfit
- Faire varier la learning rate au fur et à mesure de l'entrainement du réseau
- Augmenter encore le nombre d'épochs pour l'entrainement du réseau
- Télécharger des réseaux pré-entrainés et procéder à du transfert-learning et fine-tuning pour gagner encore de la précision