# Appendix - Hyperparameters selection

In this section, we reported the experiments that we run on the 3DIrcad-dB to find the best sets of hyperparameters for both the tumor segmentation networks.

Here is the list of hyperparameters to set in our case:

* Learning rate
* Decay
* Number of epochs
* Depth of the network, modelized in our case by the number of filters used in the bottleneck of our U-Net architecture
* Initial image size
* Type and amount of data augmentation
* Dropout
* Input type, either single slice or 2.5D

Several techniques exist for the determination of the best set of hyperparameters, such as the grid search or the bayesian hyperparameters optimization. However given the number of hyperparameters and the time required to perform one training, it would have been too costly to implement those techniques that better suits small architectures and less complex problems, so we have decided to follow an heuristic approach to find the best set of parameters.

We first tried to set the best pairs of learning rate and decay, by freezing the rest of the parameters. For the chosen optimizer, the learning rate and the decay will determine how the gradient descent is performed. A too large initial learning rate will cause the network to continuously overshoot the minimum, whereas a too small one will require a large amount of iterations to reach the minimum, as depicted hereafter.

|  |
| --- |
|  |
| Figure: Illustration of the gradient descent when the learning rate is too large (left) or too small (right) |

Once the best candidates were found, we tried to modify the depth of the network. In the case of the U-Net architecture, one easy way to change the depth of the network is to add or remove one stage in both the encoding and the decoding parts. This can be directly modeled by the number of filters used in the bottleneck part of the network (originally 1024).

In our case, using 1024 filters in the bottleneck part of the network, will correspond to an architecture with almost 32M parameters.

Setting these first 3 parameters (learning rate, the decay and the depth of the network) gave us a baseline segmentation accuracy that we tried to improve with the data augmentation.

As exposed previously, the standard way to apply data augmentation is to perform geometrical transformations to the original images so that the network will further be invariant to those modifications.

The classical transformations applied are:

* Translation: each translation is performed in one randomly chosen direction, where the delta value corresponds to the amplitude of the shift in terms of number of pixels (e.g shift of 0.1 for an initial size of 256 corresponds to a shift of 26 pixels in one direction).
* Flip: Horizontal or Vertical flip of the original image.
* Rotation: apply a rotation of a given angle to the original image.

We decided in our case to only investigate the data augmentation that will prevent the aspect of the liver and its internal tissues. In the case of elastic deformation, we can not testify that the retained features will still be discriminant in our deep radiomics approach for the grade classification if the aspect of the internal liver tissues is changed.

We have performed for each of the following experiments a 5-Fold Cross-Validation training on the 3DIrcad-dB, and the reported results correspond to the mean dice per patient for the given region (liver, parenchyma or lesion). We first set the hyperparameters for the second network of our cascade, since

### Parenchyma - Tumor Network Hyperparameters

#### Learning Rate & Decay

We first evaluated the best pairs of **learning rate** and **decay** when the other parameters such as the number of epochs or the depth of the network are fixed.

Here are the results obtained with the following fixed parameters:

* Optimizer: Adam
* Number of epochs: 10
* Number of filters at bottleneck: 512
* Input image size: 256
* No data augmentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parenchyma** |  | **Decay** | | | | |
|  |  | **1.00E-01** | **1.00E-02** | **1.00E-03** | **1.00E-04** | **1.00E-05** |
| **Lr** | **1.00E-03** | 82.9 | 91.9 | 90 | 80.3 | 9.6 |
| **1.00E-04** | 81.3 | 90.8 | 90.8 | 91.9 | 93.7 |
| **1.00E-05** | 32.4 | 76.7 | 88 | 89.1 | 89.7 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tumor** |  | **Decay** | | | | |
|  |  | **1.00E-01** | **1.00E-02** | **1.00E-03** | **1.00E-04** | **1.00E-05** |
| **Lr** | **1.00E-03** | 7.9 | 23.5 | 24.5 | 16.1 | 1.5 |
| **1.00E-04** | 6 | 15.6 | 15.6 | 23.6 | 21.7 |
| **1.00E-05** | 5.4 | 5.4 | 14 | 17 | 15.2 |

Since those two metrics are highly related to the number of voxels of each class, we decided in order to combine them to incorporate the number of voxels of each one of the two classes to create a new metric that will help us select the best networks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Combined** |  | **Decay** | | | | |
|  |  | **1.00E-01** | **1.00E-02** | **1.00E-03** | **1.00E-04** | **1.00E-05** |
| **Lr** | **1.00E-03** | 11.6 | 26.9 | 27.7 | 19.3 | 1.9 |
| **1.00E-04** | 9.7 | 19.3 | 19.3 | 27.0 | 25.2 |
| **1.00E-05** | 6.7 | 8.9 | 17.6 | 20.5 | 18.9 |

We can see that using a too small initial learning rate (1e-5) will not be enough, for the given number of iterations, to approach the minimum during the gradient descent. On the contrary, when the learning rate is set at a very high value such as 1e-3, with a small decay (1e-5 or lower in our case), the network continuously overshoots the minima, ending with a very large loss.

We selected the pairs providing the best accuracy and investigated the behavior of the gradient descent when slightly changing the depth of the network.

#### Network’s depth

Once the pairs of learning rate and decay selected, we have changed the depth of the network. This setting is controlled in our pipeline by a parameter called “bottleneck filters”.

In the above mentioned experiments, this number was set to 512, and we tried to change it with the following values: **{128, 256, 512, 1024}**.

Here are the reported results for both the segmentation of the parenchyma and the tumor when changing the depth of the network:

* Parenchyma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **# Bottleneck filters** | | | |
| **Lr** | **Decay** | **128** | **256** | **512** | **1024** |
| 1.00E-03 | 1.00E-02 | 76.1 | 92.1 | 91.9 | 7.2 |
| 1.00E-03 | 1.00E-03 | 76.5 | 88.7 | 87 | 0 |
| 1.00E-04 | 1.00E-04 | 87.1 | 92.8 | 93.9 | 94.2 |
| 1.00E-04 | 1.00E-05 | 89.7 | 92.6 | 89.9 | 93.6 |

* Tumor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **# Bottleneck filters** | | | |
| **Lr** | **Decay** | **128** | **256** | **512** | **1024** |
| 1.00E-03 | 1.00E-02 | 18.5 | 17.3 | 23.5 | 2.8 |
| 1.00E-03 | 1.00E-03 | 23.5 | 22.1 | 24.5 | 0 |
| 1.00E-04 | 1.00E-04 | 14.6 | 22 | 25.9 | 27.6 |
| 1.00E-04 | 1.00E-05 | 12.2 | 19 | 15.2 | 26 |

* Combined

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **# Bottleneck filters** | | | |
| **Lr** | **Decay** | **128** | **256** | **512** | **1024** |
| 1.00E-03 | 1.00E-02 | 21.3 | 21.0 | 26.9 | 3.0 |
| 1.00E-03 | 1.00E-03 | 26.1 | 25.4 | 27.6 | 0.0 |
| 1.00E-04 | 1.00E-04 | 18.2 | 25.5 | 29.2 | 30.9 |
| 1.00E-04 | 1.00E-05 | 16.0 | 22.6 | 18.9 | 29.3 |

We can see that a too large network depth can cause problems, probably because the size of the training dataset is too small.

For the next experiments, we have decided to fix the learning rate to **1e-4** and the decay to **1e-4**, with a number of filters at the bottleneck stage of the network of **1024**, similar to the original architecture [**ref Ronnenberger**].

Once those hyperparameters were set, we investigated the amount and the type of data augmentation to apply during the training.

#### Data augmentation

We investigate the impact of geometrical data augmentation by first fixing the amount of data augmentation to 4, meaning that each image in the training set will be used to create 4 new images. The other hyperparameters are fixed as follow:

* Lr: 1e-4
* Decay: 1e-5
* Optimizer: Adam
* Number of epochs: 10
* Number of filters at bottleneck: 1024
* Input image size: 256

##### Translation

We first investigated the value brought by translations during the data augmentation process, by changing the shift from large to small deltas.

Here are the results for each of the chosen sets:

|  |  |  |  |
| --- | --- | --- | --- |
| **Shift delta** | **Parenchyma** | **Tumor** | **Combined** |
| No Augm. | 94.2 | 27.6 | 30.9 |
| [-0.1, -0.05, ..., 0.1] | 94.8 | 32.6 | 35.7 |
| [-0.1, 0, 0.1] | 95 | 32.5 | 35.6 |
| **[-0.2, 0.1, ..., 0.2]** | **95** | **34.6** | **37.6** |
| [-0.3, 0.2, ..., 0.3] | 94.6 | 32 | 35.1 |

We can see that all the experiments provided a better accuracy than the baseline, with a better accuracy obtained for images augmented with a shift in the range [-0.2, 0.2].

##### Flip

We also evaluated the value brought by flipped images during the data augmentation process. Flip can only be performed horizontally or vertically, thus a maximum of 2 images can be generated for a given initial image.

Here are the results obtained when generating only flipped images during the data augmentation process:

|  |  |  |  |
| --- | --- | --- | --- |
| **Flip** | **Parenchyma** | **Tumor** | **Combined** |
| No Augm. | 94.2 | 27.6 | 30.9 |
| **TRUE** | **94.7** | **31.6** | **34.7** |

We can see here that flip as data augmentation brings a clear value to the segmentation performances of the network.

##### Rotation

When applying rotation, we have to carefully choose the angles used during the rotation since too small angles can cause overfitting and too large angles can be useless because representing unrealistic data.

Here are the results obtained for the chosen rotation sets:

|  |  |  |  |
| --- | --- | --- | --- |
| **Rotation angle** | **Parenchyma** | **Tumor** | **Combined** |
| No Augm. | 94.2 | 27.6 | 30.9 |
| [0, 10, ..., 40] | 94.9 | 35.9 | 38.8 |
| [0, 20, ..., 80] | 87.1 | 36 | 38.5 |
| [0, 30, ..., 90] | 87.3 | 36.2 | 38.7 |
| [0, 90, ..., 270] | 94.6 | 34.7 | 37.6 |

Reported results showed that experiments involving training images obtained after rotations always improved the baseline accuracy of the network. It is worth noting that rotations performed with angles in the range [0,40] gave better results than larger ones.

Given the results of the above mentioned experiments, we have decided for the data augmentation to generate new images using **rotations with an angle included in the interval [0, 40]°, both horizontal and vertical flips, and translation in the range [-0.2, 0.2]**.

These 3 geometrical transformations will randomly be combined for the next experiments.

Here is the list of hyperparameters set for the rest of the experiments:

* Lr: 1e-4
* Decay: 1e-5
* Optimizer: Adam
* Number of filters at bottleneck: 1024
* Input image size: 256
* Data augmentation
  + Rotations with an angle in the interval [0, 40]
  + Translation with a shift in the interval [-0.2, 0.2]
  + Horizontal and vertical flip

We first investigated what happened when both augmentation functions are combined, by incrementally increasing the augmentation factor, and reported the results hereafter:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epochs** | **Augm. Factor** | **Parenchyma** | **Tumor** | **Combined** |
| 10 | 4 | 94 | 26.6 | 29.9 |
| 10 | 8 | 94.6 | 32.3 | 35.4 |
| 10 | 12 | 94.8 | 34.2 | 37.2 |
| 10 | 16 | 94.9 | 34.9 | 37.9 |
| 10 | 20 | 94.8 | 40.5 | 43.2 |

We can see that increasing the number of generated images improves the segmentation performances of the network.

For this network, we haven’t seen any value when adding a dropout with a rate of 0.2, however this might be discussed because with a higher number of epochs, the addition of dropout might be beneficial.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epochs** | **Augm. Factor** | **Dr** | **Parenchyma** | **Tumor** | **Combined** |
| 10 | 20 | 0 | 94.8 | 40.5 | 43.2 |
| 10 | 20 | 0.2 | 94.7 | 38.4 | 41.2 |

Using the original image size (512x512) seems also to improve the accuracy of the network, but in these experiments, we trained our networks with a reduced resolution (256x256) to reduce the training time.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epochs** | **Augm. Factor** | **ImgSize** | **Parenchyma** | **Tumor** | **Combined** |
| 10 | 20 | 256 | 94.8 | 40.5 | 43.2 |
| 10 | 20 | 512 | 95.1 | 42.7 | 45.3 |

We finally increased the number of training epochs, and changed the augmentation factor, to realize that the best accuracy was obtained with an augmentation factor of 20 and a network trained with 30 epochs.

Augmenting the data with a factor higher than 20 (e.g 24 in the table below) tends not to improve the performances of the network. The same phenomenon is observed when the number of training epochs is increased at more than 30, probably because the network will start to overfit, due to a too large number of training iterations, or due to the presence of images that are too similar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epochs** | **Augm. Factor** | **ImgSize** | **Parenchyma** | **Tumor** | **Combined** |
| 10 | 20 | 256 | 94.8 | 40.5 | 43.2 |
| 20 | 16 | 256 | 95 | 43.4 | 45.9 |
| 30 | 16 | 256 | 95.1 | 43.2 | 45.8 |
| 20 | 20 | 256 | 95 | 41.1 | 43.8 |
| 30 | 20 | 256 | 95.1 | 44.9 | 47.4 |
| 36 | 20 | 256 | 95 | 42.8 | 45.4 |
| 30 | 24 | 256 | 95.2 | 45.2 | 47.7 |

**The final set of hyperparameters for the segmentation of both the parenchyma and the tumor** is the following:

* Lr: 1e-4
* Decay: 1e-5
* Number of epochs: 30
* Optimizer: Adam
* Number of filters at bottleneck: 1024
* Input image size: 256
* Data augmentation
  + Rotations in the interval [0, 40]
  + Horizontal and vertical flip
* Augmentation factor: 20