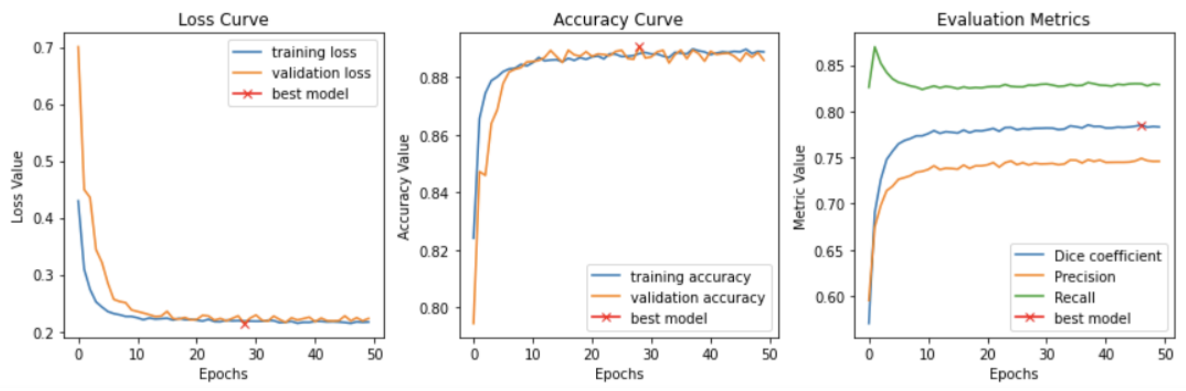
Gerard Castells and Mariona Carrasco

Deep Learning Methods for Medical Image Analysis (CM2003)

LAB 5

Task 1a

Lung segmentation in CT images: Reuse the same pipeline of task1 for segmentation of lung regions within the “CT” dataset. Please note the left and right lungs are labeled with different values in the segmentation masks. So you need to binarize the loaded masks for a binary segmentation task. Use the following parameters: Base=8, batch norm=ture, image_size=256, train_percentage=0.8, LR=1e-4, loss=dice, dropout=True(0.2),batch size=8 and augmentation: false. Please note as the number of images in this data set is quite large (>8000 images), the computational time will be a bit longer so that you can train the model for only 50 epochs. Path = ‘…./Lab4/CT/’



In order to do the binary segmentation task, we have to follow some steps. First, we binarize the mask by making all the regions that are not background white while keeping the background black. Then from here we work with the U-Net architecture and the parameters mentioned.

From the results obtained we can see that loss and accuracy values are highly acceptable (0.2 and 0.89, respectively) and that no overfitting is present. This means that the model is learning properly and working nice in images that are not the training ones.

In relation to the evaluation metrics, first we want to clarify what exactly is each of those:

- **Dice coefficient:** it is a metric that quantifies the similarity between the predicted positive instances and the true positive instances in a binary classification or segmentation problem. It is defined as 2 times the intersection of the predicted and true positive sets divided by the sum of the sizes of those sets. Mathematically, it's expressed as: 2 * (TP) / (2 * TP + FP + FN), where TP (True Positives) are the instances correctly predicted as positive, FP (False Positives) are instances incorrectly predicted as positive, and FN (False Negatives) are instances incorrectly predicted as negative. The Dice coefficient ranges from 0 (no overlap between predicted and true positives) to 1 (perfect match), with higher values indicating better model performance.
- **Precision:** Precision is a metric that measures the accuracy of a model in correctly identifying positive instances out of all instances it predicts as positive.
- **Recall:** it measures the ability of a model to correctly identify all positive instances out of all actual positive instances.

From the first task, we can observe that all of these three metrics have a value higher than 0.7, which we can consider as good.

Task 1b

Evaluating the segmentation performance is often done by using Dice coefficient; however, it is quite essential to assess the rate of false positives and false negatives too. Implement "precision" and "recall" metrics as well, and use these three metrics in your model (Metric = [dice_coef, precision, recall]) to monitor the model performance. Repeat the task 2a by including data augmentation technique (similar to task1e) and train the model for 50 epochs and interpret the observed values of the three metrics over the validation set. In general, what can be inferred from precision and recall metrics?

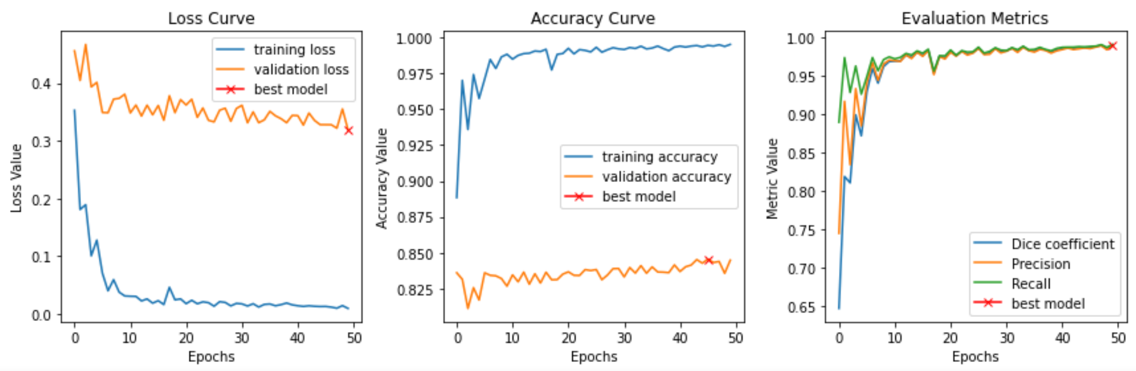


Image augmentation serves as a valuable tool to enhance the generalization capabilities of deep learning models. When used appropriately, it can lead to more robust and accurate models, especially when the training dataset is limited or when the real-world data can vary in ways that are hard to anticipate.

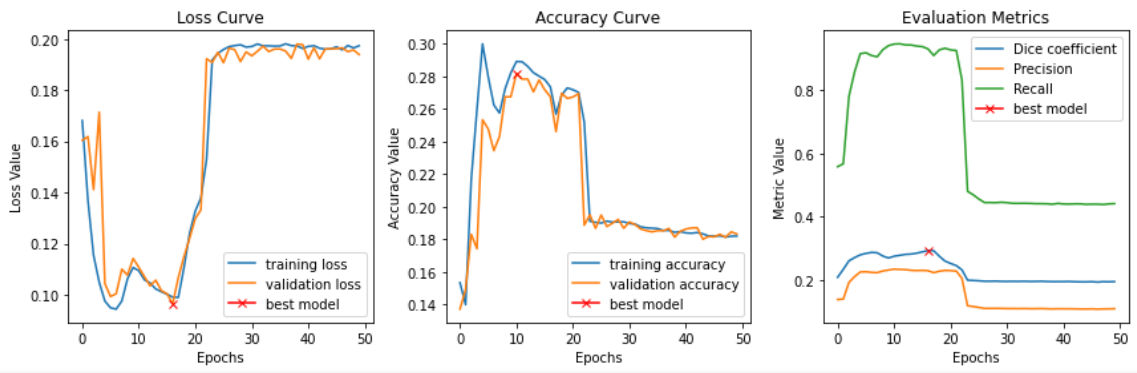
This technique also mitigates overfitting by introducing elements of randomness and noise, thus reducing the likelihood of the model merely memorizing training instances. Furthermore, augmentation equips the model to withstand real-world challenges such as changes in lighting or perspective. Additionally, it effectively expands the effective dataset size, which, in turn, can enhance model performance, even when working with a limited original dataset.

Nevertheless, maintaining a balance when employing image augmentation is crucial. Excessive augmentation or inappropriate transformations can result in excessive regularization or the introduction of unrealistic data, potentially compromising model performance.

Evaluating our learning curves, it becomes evident that when adding data augmentation techniques, the model performs even better for the training images, but poorly for the validation data. That is due to overfitting caused by over learning the training data.

Task 2

As it is already explained, the labels of the left and right lungs in the CT masks are not the same. We can consider the two lungs as two different organs and perform a multi-organ segmentation. Modify your model and adapt it for a multi-organ segmentation task and segment the left and right lungs separately in one framework. Set the image_size:256, batch norm:true, LR:1e-4, dropout=True(0.2), augmentation:true and n_epoch=80. Choose a proper loss function for multi-organ segmentation and modify the segmentation mask labels to match the problem.

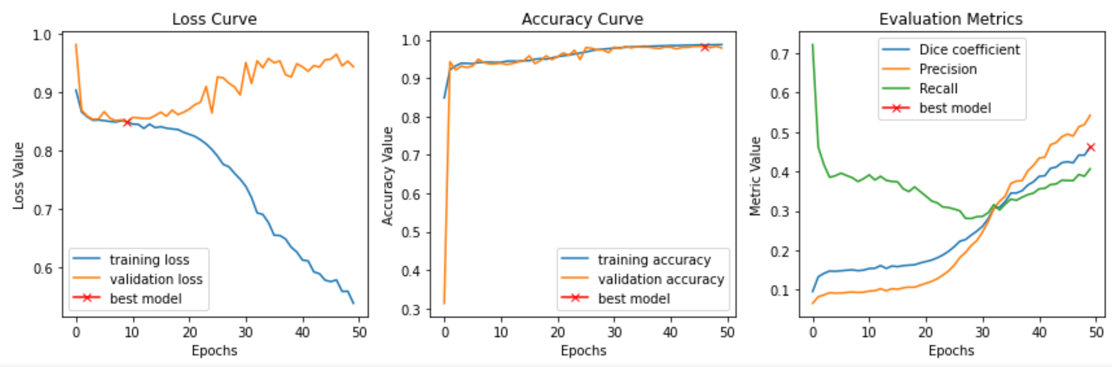


The labels for the left and right lungs in CT masks are typically different because these organs are anatomically distinct and have separate functions. It's crucial to differentiate between different structures and organs within the body for accurate diagnosis and analysis. In consequence, adapting a deep learning model for a multi-organ segmentation task where we segment the left and right lungs separately is a valuable approach.

Analyzing our learning curves, we could see that with a lot of epochs the model performance goes badly so it is not correct training so we should modify more the segmentation mask labels to match the problem and obtain best results.

Bonus task

Brain tumor segmentation in MR images: In contrast to the lung region that entails specific shape, appearance, and intensity pattern, brain tumors can be presented in a large variety of shape, appearance, texture, and intensity patterns. More importantly, they can appear in any region within the brain; therefore, they do not have a fixed location. In this exercise, you are asked to optimize your model by tuning the hyperparameters for the task of brain tumor segmentation.
This dataset is a part of a larger dataset related to a challenge named as Brain Tumor Segmentation challenge. The highest achieved dice coefficient over this dataset is around 0.96. Please compare your best-achieved result against this best performance and discuss the possible ways to improve the performance of your developed model.



Optimizing a model for brain tumor segmentation in MR images is a crucial task given the diverse shape, appearance, texture, and intensity patterns of brain tumors, as well as their unpredictable locations within the brain. To enhance the model's performance, you can consider hyperparameter tuning.

Evaluating our learning curves, the accuracy values are good around 1 so the model generalizes well to new data while also performing well on the training data. However, on the loss curve it occurs overfitting, which is problematic because it suggests that the model is becoming too specialized in the training data and is not generalizing well to unseen data.

Comparing our best-achieved Dice coefficient of 0.5 against the highest achieved Dice coefficient of approximately 0.96 in the Brain Tumor Segmentation challenge indicates that there is room for improvement in our model's performance. Achieving a Dice coefficient of 0.96 is an exceptional result and a challenging benchmark. Although reaching it may be very challenging, it's important to continually iterate, test, and refine our model to achieve the best possible results.

Task 3

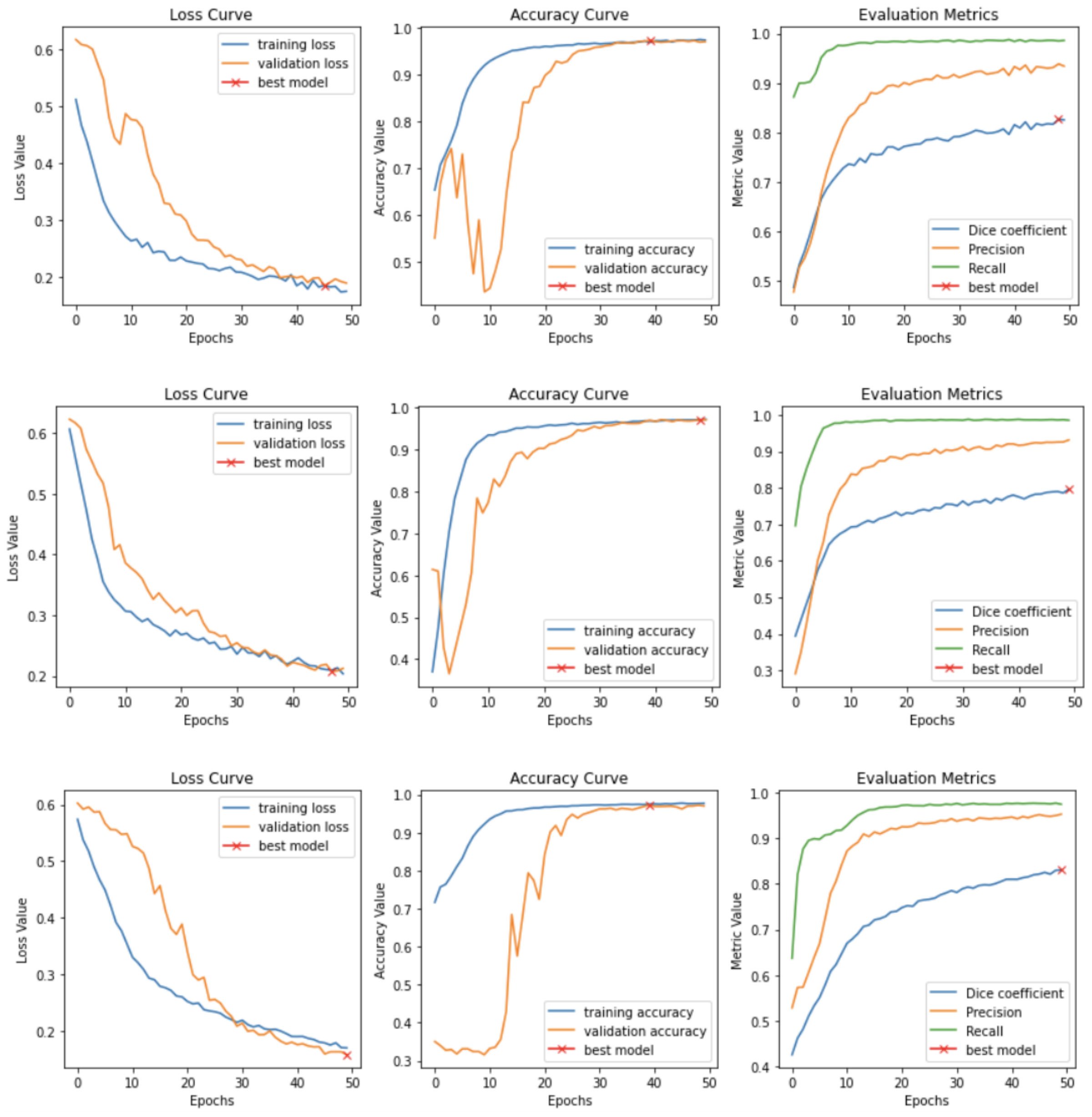
K-fold cross validation

With K-fold cross validation, you do not simply split the dataset into one training and one validation set, but rather test K different splits between training and validation sets. You can start by setting K = 3. You will need to implement a loop that goes through each of the 3 folds. At the beginning of each loop, you will call a function (which you need to define) that appropriately splits your image and mask files into the training and validation sets for that specific fold. If you have n images in total, then—for the first fold—you will use the first n/3 images as validation set and the remaining 2*n/3 as a training set. Instead, for the second fold, the first n/3 images and

the last n/3 images will be your training set. For the last fold, the first 2*n/3 images of the list will be used as training set

K-fold cross-validation is a technique in machine learning that splits the dataset into K subsets, repeatedly trains and evaluates the model on different combinations of these subsets, and provides a more comprehensive and robust assessment of a model's performance and its ability to generalize to new, unseen data.

It estimates a model's generalization, reduces bias, utilizes data efficiently, allows model comparison, and helps with hyperparameter tuning, providing a comprehensive and robust performance evaluation. It's a crucial tool for model assessment and selection.



As we can observe, by comparing the three folds, we can observe that our model works properly in the generalization of the learning. In all cases, validation accuracy gets to values really close to 1, without the presence of overfitting. Precision, recall and dice coefficient, all three parameters present high values with an increasing tendency.

In conclusion, it helps to obtain a more comprehensive and robust assessment of our model's performance by evaluating it on multiple data subsets. The loss and accuracy curves show variations across folds, but they are used to assess the overall model's performance.