ANDREW AND ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

Technion Israel Institute of Technology

“Black-box Tests for Algorithmic Stability”
Regression & Coverage Stability
Implemented on IVIM

Kfir Levi & Priel Salomon

Technion - Electrical Engineering

048100 - Reliability in ML course

Final project

Abstract

We based our project on an article "Black-box tests for algorithmic stability" (Byol & Kim, 2021). Algorithmic stability is a concept from learning theory that expresses the degree to which changes to the input data (i.e. removal of a single data point) may affect the outputs of a regression algorithm. many modern algorithms currently used in practice are too complex for a theoretical analysis of their stability properties, and thus we can only attempt to establish these properties through an empirical exploration of the algorithm's behavior on various datasets. In this project, we have used the mentioned paper formal statistical framework on a medical imaging model of IVIM used in MRI scans. We've implemented the split conformal prediction algorithm on the IVIM deep learning approach and on top of that we've conduct the paper statistical framework on the confidence interval (coverage) of the estimated parameters results of the IVIM model.

Section 1 – Background

Conformal prediction

In this work, we implemented the Conformal prediction algorithm, as detailed in [3], on a specific medical imaging estimation task. Conformal prediction (CP) is a statistical technique for producing prediction intervals without assumptions on the predictive algorithm (often a machine learning system) and only assuming exchangeability of the data. CP works by computing a nonconformity measure, often called a score function, on previously labeled data, and using these to create prediction sets (or intervals for a regression estimation) on a new (unlabeled) test data point. Conformal prediction requires a user-specified *significance level* for which the algorithm should produce its predictions. This significance level restricts the frequency of errors that the algorithm is allowed to make. For example, a significance level of 0.1 means that the algorithm can make *at most* 10% erroneous predictions. To meet this requirement, the output is an **interval prediction**, instead of a **point prediction** produced by standard supervised machine learning models. For regression tasks, this means that predictions are not a specific value, for example 34.768, but instead an interval of 31.56 – 37.67. Depending on how good the underlying model is (how well it can estimate the interval) and the specified significance level, these intervals can be smaller or larger. The output is prediction intervals, where a smaller significance level (fewer allowed errors) produces wider intervals which are less specific, and vice versa – more allowed errors produce tighter prediction intervals.

IVIM Estimation

This imaging technique has been developed with the objective of obtaining not only a functional analysis of different organs but also different types of lesions. Among many accessible tools in diagnostic imaging, IVIM MRI aroused the interest of many researchers. The intravoxel incoherent motion (IVIM) diffusion-weighted (DW) model as a possible imaging technique, using multiple b values and bi-exponential fitting for the concurrent estimation of the pure molecular water diffusion and microcirculation of blood water in randomly oriented capillaries (perfusion) was first introduced in the late 1980s by Le Bihan *et al* [1]. The idea is to use diffusion and IVIM magnetic resonance imaging (MRI) to acquire perfusion parameter maps. IVIM reflects the random microscopic motion of water molecules that occurs in each

voxel on MR images not only in intra- or extracellular space but also in microcirculation of blood. According to IVIM theory, diffusion and perfusion are affected by several tissue characteristics, including the presence of restrictive barriers within tissue, the viscosity of the fluid in which the spins are diffusing, and the velocity and fractional volume of perfusing spins. Formerly, due to degradation of images caused by cardiac, respiratory, and other motion artifacts, IVIM imaging was restricted to neuroradiologic applications. Over the last few years there has been a revival of interest in IVIM MRI and its applications in many fields, particularly in oncology.

The basic IVIM diffusion and perfusion model for the signal intensity (per pixel):

$$S(b) = F_p e^{-bD_p} + \left(1 - F_p\right) e^{-bD_t}$$

In this model we have 3 different parameters to estimate for any pixel in MRI image: D_p, D_t, F_p which in simple words are the diffusion and perfusion factors and their proportion in the physical scanned voxel.

IVIM DNN approach

A paper from 2020 by Barbieri, S., Gurney-Champion, O. J., Klaassen, R., & Thoeny, H. C. [2] proposed a Deep Learning approach to solve the IVIM model's parameters. A feed-forward backward-propagation DNN was trained to generate estimates of IVIM parameters ($\widehat{D}t, \hat{F}p, \widehat{D}p$). Training is unsupervised and needs to be repeated for data sets with different distributions (e.g., because of different acquisition protocols or imaged anatomical regions). Given that the goal is to encode a given data set, separate training and testing data sets are not required and the network was trained directly on the data set of interest. The network is composed of an input layer, 3 hidden layers, and an output layer. The passthrough input layer is made of neurons, which take the normalized diffusion-weighted signal sampled at each b-value as input. The 3 hidden layers are fully connected, with a number of neurons equal to the number of b-values of the data of interest and an exponential linear unit activation function. The output layer is made of 3 neurons, which hold the estimated parameter values. Initial network weights were set using He initialization or using a previously trained network. An Adam optimizer was used for training with the mean squared error between the observed input S(b) and the signal Ŝ(b), reconstructed based on the IVIM model and $\widehat{D}t, \hat{F}p, \widehat{D}p$ as loss function. Early stopping was implemented by terminating training after the loss function did not improve for 10 consecutive iterations. The proposed neural network architecture is essentially an autoencoder with the constraint that the input signal should be encoded by the 3 IVIM parameters. The network does not impose any restrictions on the range of fitted parameter values.

Section 2 – The chosen paper "Black-box Tests for Algorithmic Stability"

The paper conducts a black-box statistical test framework in which one can evaluate the stability properties of an algorithm to small changes in the dataset samples. The base of the method is to fit the algorithm on two very similar datasets differ only with one sample (where one dataset size is n and the other is $n-1$) and to evaluate how much change there is in the estimated output on a new and hadn't been seen $n+1$ sample point. For this framework two definitions must be put:

Definition 1 - (ϵ, δ) Stability:

Let $\mathcal{A}$ be a symmetric algorithm. Let $\epsilon \geq 0$ and $\delta \in [0,1)$ We say that $\mathcal{A}$ is (ϵ, δ) - stable with respect to training datasets of size n from a data distribution P—or, for short, the triple ($\mathcal{A}$, P, n) is (ϵ, δ)-stable—if $\mathbb{P}\{|\hat{\mu}_n(x_{n+1}) - \hat{\mu}_{n-1}(x_{n+1})| > \epsilon\} \leq \delta$
Where μ_n, μ_{n-1} are the fitted models obtained from the full training dataset and the dataset after removing the last data point, i.e., $\mu_n = \mathcal{A}\,[(X_1, Y_1), \ldots, (X_n, Y_n); \xi]$, $\mu_n = \mathcal{A}\,[(X_1, Y_1), \ldots, (X_{n-1}, Y_{n-1}); \xi]$, where the data is distributed as $(X_i, Y_i) \sim P$ and $\xi \sim$ Uniform[0, 1] is drawn independently of the data. we define stability by comparing the outputs of $\mathcal{A}$ while fixing ξ at the same value, i.e., both μ_n, μ_{n-1} are fitted using the same value ξ (e.g., the two calls to $\mathcal{A}$ are initialized with the same random seed).
The meaning of this definition is to define what is a large enough "error" stability wise and what is the probability of that error to occur.

Definition 2 - black-box test:

Consider any test $\hat{T}$ that takes as input an algorithm $\mathcal{A}$, a labeled dataset $\mathcal{D}_\ell$, and an unlabeled dataset $\mathcal{D}_u$, and returns a binary output $\hat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u) \in \{0,1\}$ whether the test succeeded or not, Then, we say that $\hat{T}$ is a black-box test if it can be defined in the following way: $\hat{T} = g[\mathcal{D}_\ell, \mathcal{D}_u, \hat{Y}, \zeta]$, where $\zeta \sim Uni[0,1]$
In simple words, the test case is dependent only on the labeled and unlabeled datasets, the algorithm output after estimation (i.e. our IVIM DNN after training process), and possibly some randomizing factor.

Notice- No knowledge on the algorithm properties is required for this method.
In the paper there are a few examples of how to obtain the probability of the unstable cases. We implemented their first "binomial test" example due to the fact that this example had been prove to be "essentially optimal among all black-boxes tests".

The binomial test:

we choose $\lfloor \kappa \rfloor$ different batches from the labeled data while $\kappa = \kappa(n, N_l, N_u) = \min\{\frac{N_l}{n}, \frac{N_l + N_u}{n+1}\}$. $\lfloor \kappa \rfloor$ represents the largest number of copies of independent datasets with size n we can use.

- For k = 1,…,$\lfloor \kappa \rfloor$, using labeled samples, construct the k-th dataset
$$\left[\left(X_{(k-1)n+1}, Y_{(k-1)n+1}\right), \ldots, (X_{kn}, Y_{kn})\right]$$
with one unlabeled data point $X_{\lfloor \kappa \rfloor n+k}$.
- Train $\hat{\mu}_n^{(k)} = \mathcal{A}\left[\left(X_{(k-1)n+1}, Y_{(k-1)n+1}\right), \ldots, (X_{kn}, Y_{kn}); \xi^{(k)}\right]$

$$\hat{\mu}_{n-1}^{(k)} = \mathcal{A}\left[\left(X_{(k-1)n+1}, Y_{(k-1)n+1}\right), \ldots, \left(X_{kn-1}, Y_{kn-1}\right); \xi^{(k)}\right]$$

- Compute

$$\Delta^{(k)} = \left|\hat{\mu}_n^{(k)}\left(X_{\lfloor\kappa\rfloor n+k}\right) - \hat{\mu}_{n-1}^{(k)}\left(X_{\lfloor\kappa\rfloor n+k}\right)\right|$$

- Calculate test statistic

$$B = \sum_{k=1}^{\lfloor\kappa\rfloor} 1\{\Delta^{(k)} > \epsilon\}$$

- $B/\lfloor\kappa\rfloor$ is the empirical proportion of

$$\mathbb{P}\{|\hat{\mu}_n(X_{n+1}) - \hat{\mu}_{n-1}(X_{n+1})| > \epsilon\}$$

- Compare B with $Binomial(\lfloor\kappa\rfloor,\ \delta)$, and if $\frac{B}{\lfloor\kappa\rfloor}$ is sufficiently below δ, return $\hat{T}_{\epsilon,\delta} = 1$, otherwise $\hat{T}_{\epsilon,\delta} = 0$.

Section 3 – Conformal stability

When observing the proposed method in [5], and also the known disadvantages of working with a model prediction score only (i.e. without a certainty coverage), it became interesting to examine a theory which would combine the black box stability tests with certainty coverages, in hope it would increase our understanding of such test and it's meanings, in the same way that estimating a certainty coverage for a prediction model increases our understanding of the model's prediction.

Our proposed method is to define $\Delta^{(k)}$ differently, while the definitions of all other elements of the process remain the same. In our proposed method we use:

$$\Delta^{(k)} = f\left(\hat{C}_n(x_{n+1}), \hat{C}_{n-1}(x_{n+1})\right), for\ some\ intersection\ calculation\ function\ f$$

Function f can be, for example,

$$f = \min(UL_n, UL_{n-1}) - \max(LL_n, LL_{n-1})$$

or

$$f = \frac{1}{2}\frac{\min(UL_n, UL_{n-1}) - \max(LL_n, LL_{n-1})}{UL_n - LL_n} + \frac{1}{2}\frac{\min(UL_n, UL_{n-1}) - \max(LL_n, LL_{n-1})}{UL_{n-1} - LL_{n-1}}$$

when UL, LL represent the upper limit and lower limit of the algorithm's prediction interval, respectively. In the second proposal of f, we calculate the average of the percentages of the intersection in relative to each of the intervals. It satisfies $f \in [0,1]$ when there is an intersection between the two intervals and $f = 1$ only if the intervals are identical. Negative values of f are the result of no intersection between the intervals, e.g. $f([1,2],[3,4]) = -1$. To prevent this from hurting our calculation we can use $\bar{f} = \max(0, f)$ so we only take the positive part of f. Then our test will be defined as evaluating if algorithm $\mathcal{A}$ satisfies $\mathbb{P}(\bar{f} > \epsilon) < \delta$.

For the interval estimation process, we need a separate dataset, independent of the training set. Therefore, it is needed to split the data into train, calibration, and stability datasets. To keep in line with the notations in [5], we will split N_l into train and calibration.

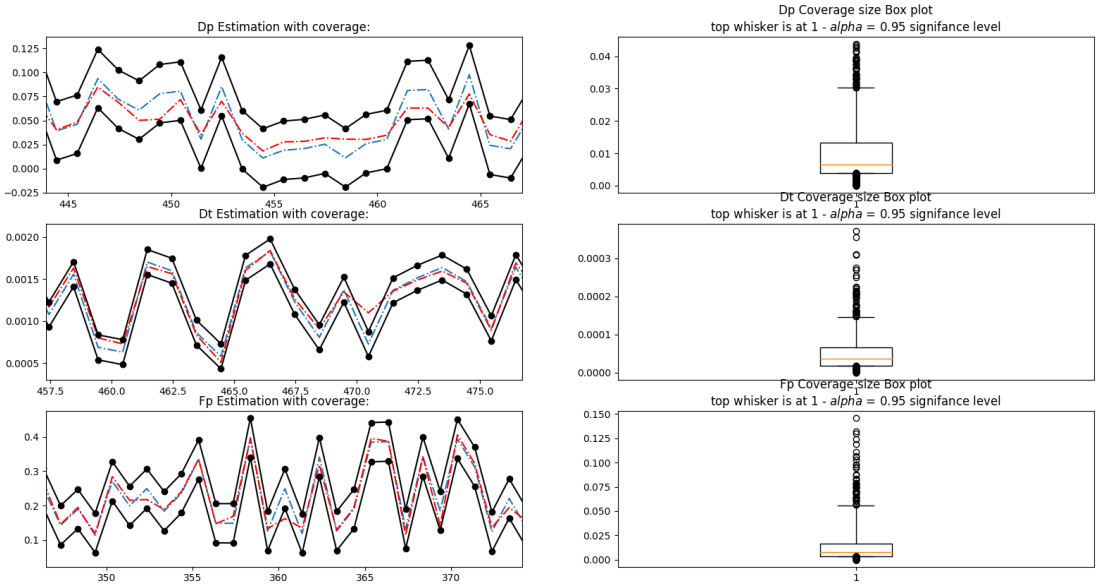
The goal for this proposal is to test the intersection of the prediction intervals of trained models μ_n, μ_{n-1}, and like in the original test, high intersection (the models produced similar intervals) means a stable solution, and not stable otherwise.

Working with intervals instead of with output alone could, as the means for evaluating stability, takes into account the model's level of certainty in the prediction, since, just like we expect similar outputs, we expect similar level of certainty (which translates to similar confidence interval).

It is also important to note that one of the most important properties of these tests are that they can be applied as black box tests, and expanding the algorithm to confidence interval does not impact this property.

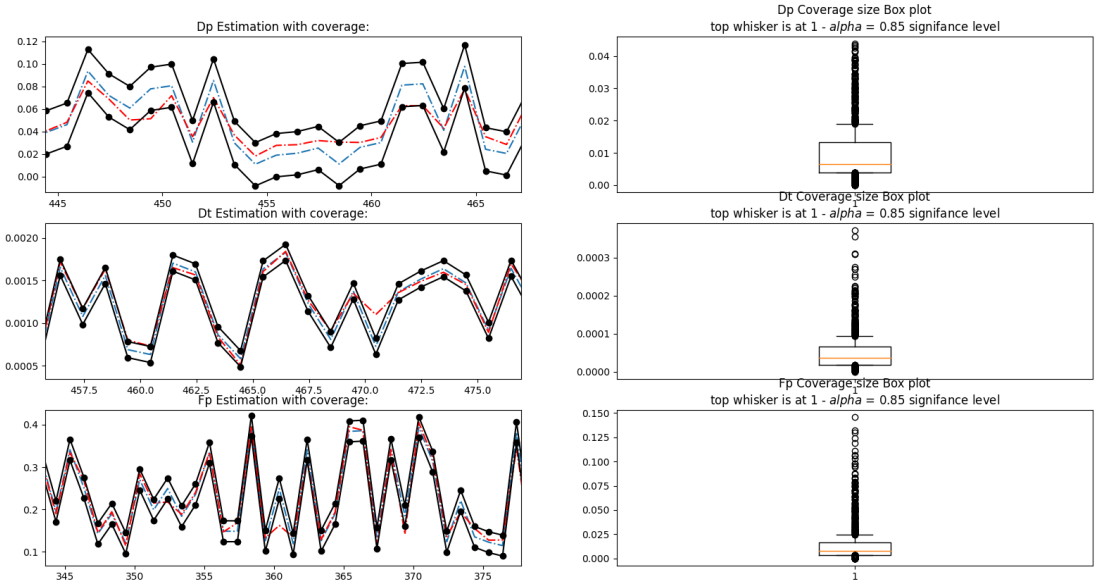
Section 4 – Results and Discussion

Our results are divided into three sections. First, we tried to evaluate a single model's confidence interval using conformal prediction algorithm. The results are presented below:



The 3 graphs on the left above show the ground truth values (in blue) of the 3 IVIM parameters per data sample (the horizontal axis), the estimated values of those parameters after the DNN inference, and the black lines are the lower and upper bounds using the conformal prediction algorithm.

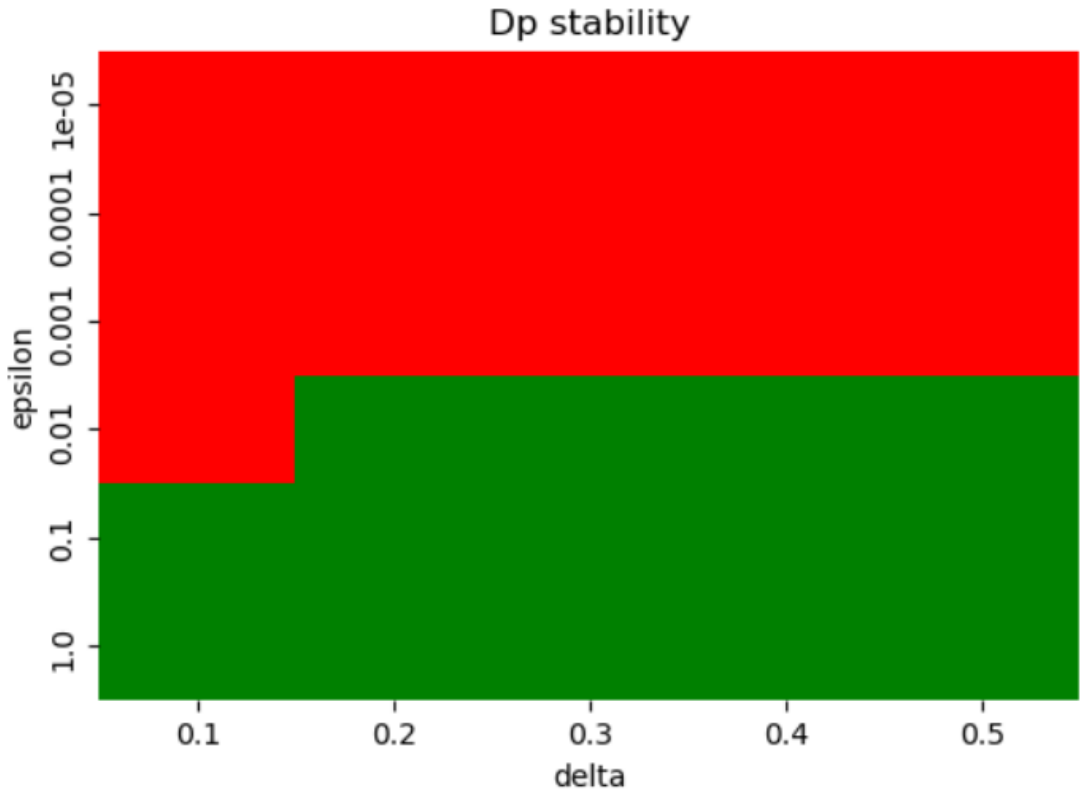
The right graphs are box plots of the non-conformity scores of the 3 IVIM model parameters, the top whisker is the $1 - \alpha$ quantile of the coverage size.

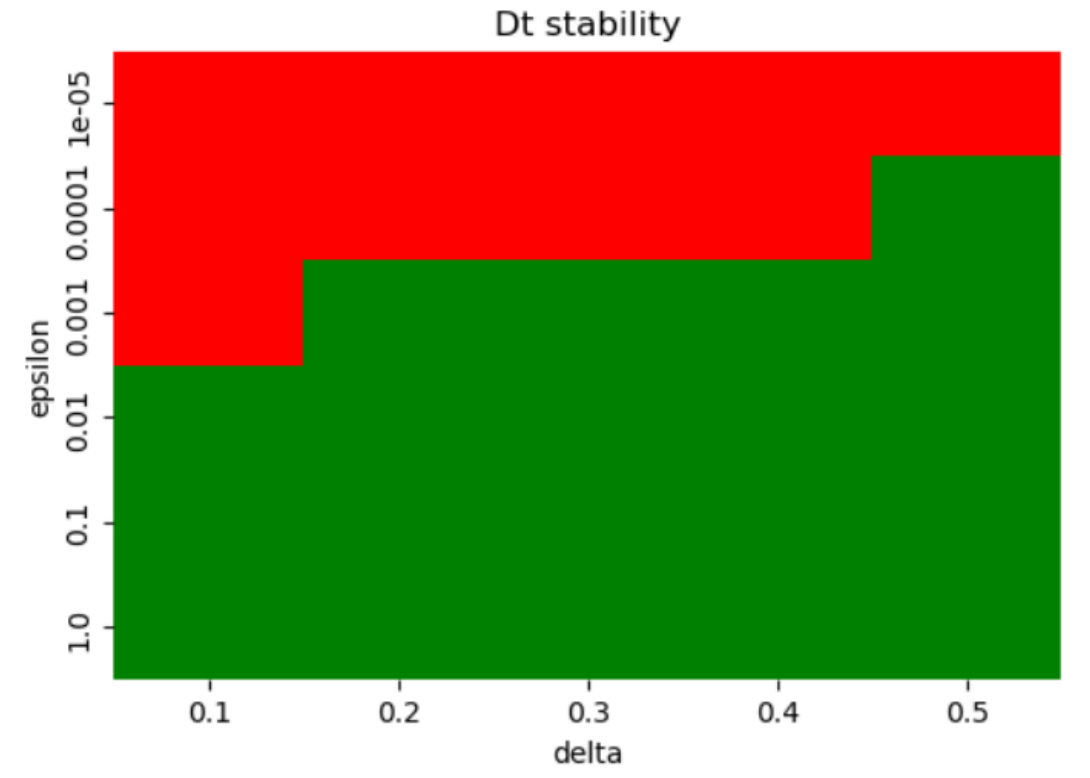


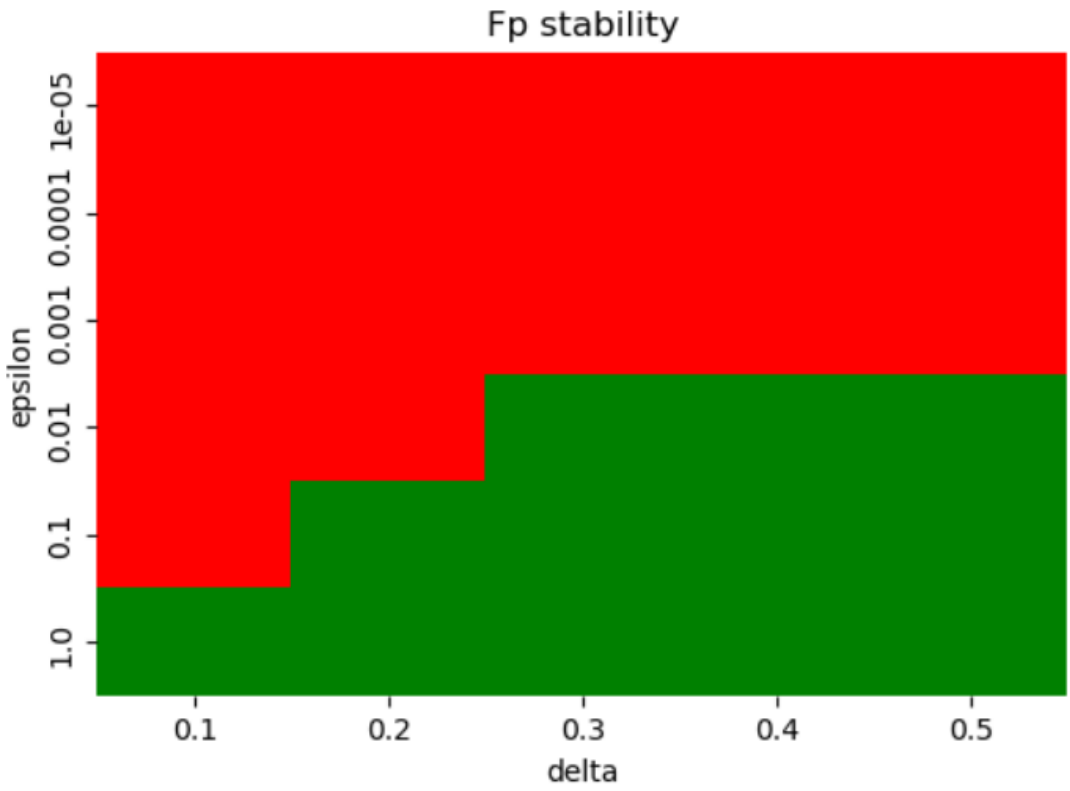
As we can see in this second figure, which we've used a lower confidence level of 85% the lower the $1-\alpha$ confidence level is, the narrower the bounds will become.

After that, we tried to test the algorithmic stability using the method proposed in [5]. We trained $\kappa = 10$ times a pair of models, μ_n, μ_{n-1}, which results in 20 training processes in total. We applied the binomial test on the 10 model pairs, testing different ϵ, δ, to test asses if $\mathbb{P}\{|\hat{\mu}_n(x_{n+1}) - \hat{\mu}_{n-1}(x_{n+1})| > \epsilon\} \leq \delta$, or in other words – if the triple ($\mathcal{A}$, P, n) is (ϵ, δ)

stable, as explained in section 2. What we got when testing stability for different (ϵ, δ) pairs, for each of the three parameters: (red means unstable, green for stable)



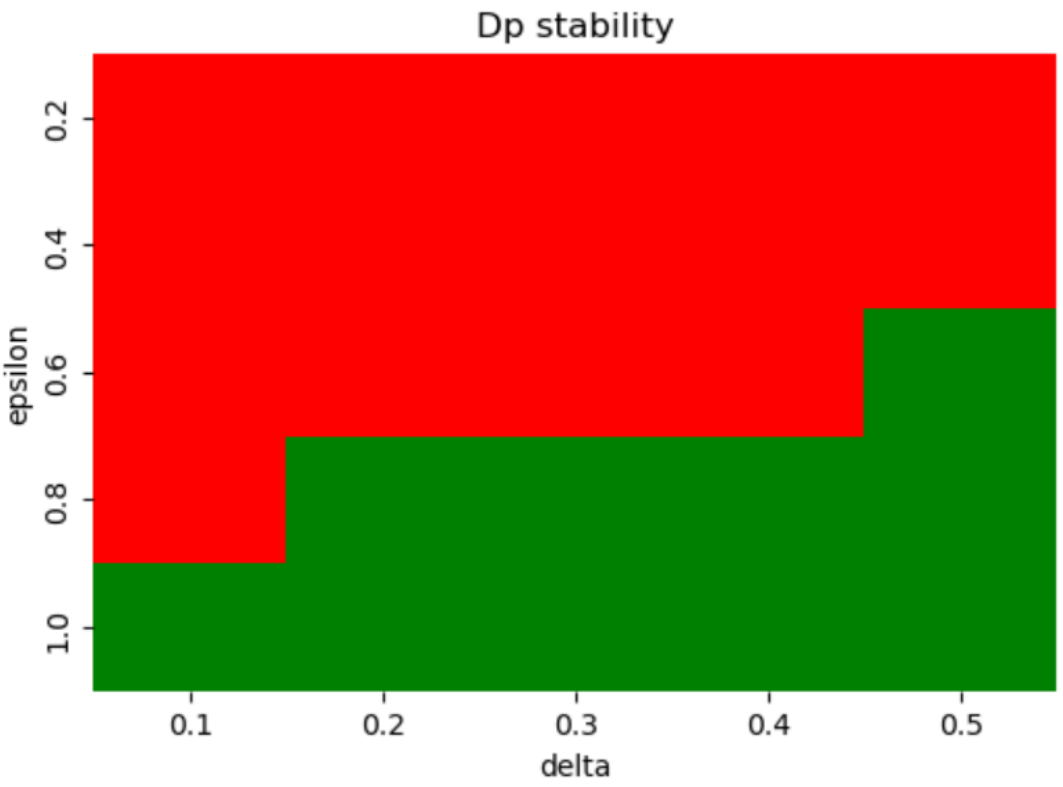


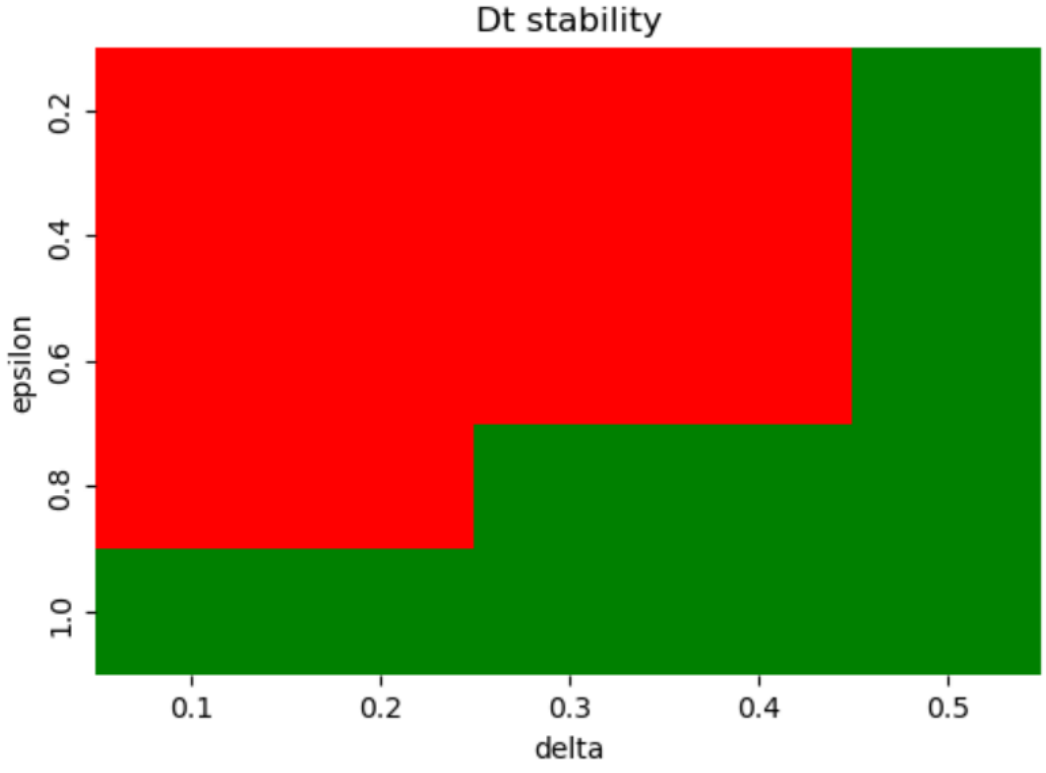


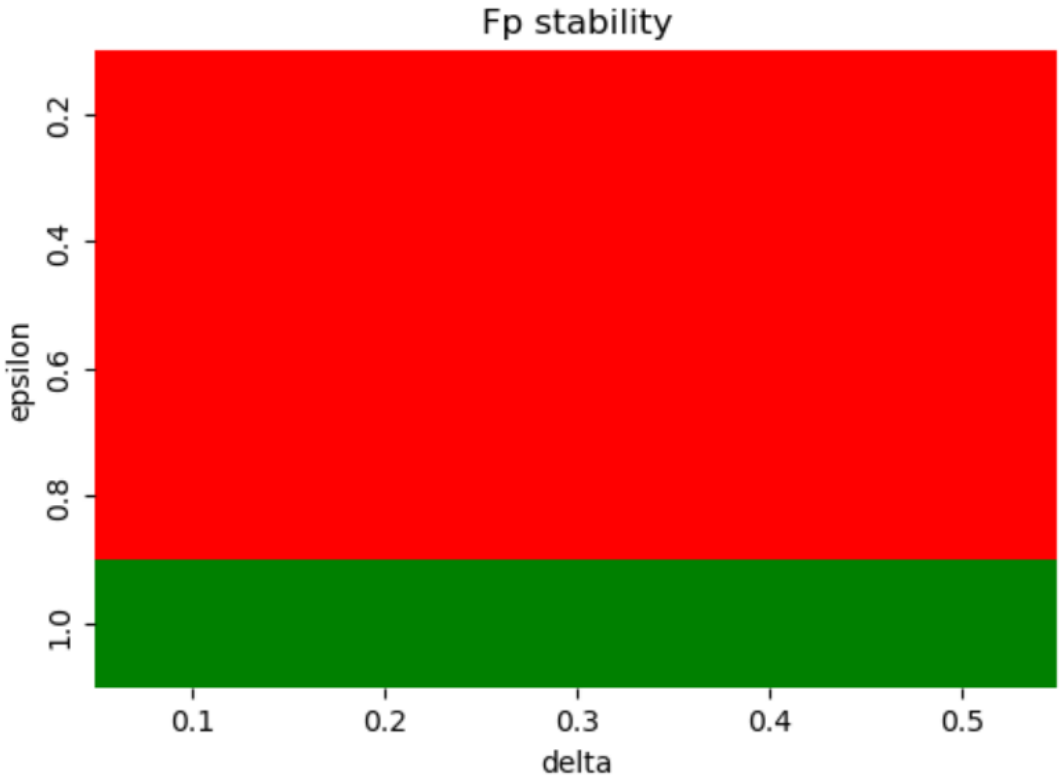
Now let's explain these results. First, let's examine along the X axis, meaning a fixed ϵ and changing δ. We got that as δ get's lower, we are less likely to be stable since our demand for stability is more strict. In simple words, we are basically testing whether the fraction of times we got $\Delta > \epsilon$ is smaller then δ, so when δ gets smaller, the answer to this question is less likely to be positive.
Now let's look at Y axis, meaning fixed δ and varying ϵ. When ϵ gets larger, we are more likely to be within ϵ distance, since we have more room for error. This is exactly what is shown in the graphs above.

The stability results sit well with the interval prediction results, since we can tell, for example, in both methods that D_t is the 'most' stable' parameter – it has the narrowest interval and also with the most green cells.

And for the third part of the results, as mentioned before, we try to test the stability of the algorithm using a confidence interval. Our score for stability is the second definition for f, introduced in section 3.


Dp stability
epsilon
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
delta


Dt stability
epsilon
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
delta



First, it's important to mention that since our definition of a stable algorithm is now very different, we used a different scale for ϵ, and also it would be very difficult, and not necessarily informative, to compare between the two cell-by-cell.
For better intuition of the results, we should take a look at a specific μ_n, μ_{n-1} model pair. For example, for a specific pair of models we got in our training process, one can notice that although the predictions of the model are very close to each other, meaning the pair's predictions would suggest a stable algorithm for a lot of ϵ values in the method suggested in [5], there is no intersection between the models intervals, even when using wider than normal intervals, meaning the pair suggest an unstable algorithm in our method: (shown, for example, for parameter D_p)

```
n interval: 0.05796015656369081, 0.057980980511733386
n-1 interval: 0.05782969700300189, 0.05785028232133893
```

Section 5 – Conclusion and Future Work

In conclusion, as mentioned in [5], testing an algorithm's stability can have a big impact on our understanding and trust in the algorithm's predictions. Extending the stability test to prediction intervals, takes the test a step further into analyzing the algorithm's reliability, since a prediction's confidence levels are now considered as well. The results support our assumption that the stability of a model is in high correlation with the confidence interval, as explained above, and combining the two methods could give us a very rich analysis of the reliability of the model, even when using a fixed length interval like we used in our work.

An obvious limitation of the conformal stability method's is the expensive computation required for the method, adding complexity for an already expensive method such as the binomial stability test. Also, as we experienced ourselves, each score function f has a

different outlook on the results, and it is not as straightforward to compare between two different functions.

In the future, one could try and develop a mathematical framework that will support our proposal, enriching the theory proposed above. One could also investigate more into the proposed method's limitations, or the test's optimality, which are two subjects discussed in [5].

Moreover, we believe our proposal could be extended to more complicated intervals, such as conformalized quantile regression (CQR), or extend the algorithm for the case of multi-class classification. Using a more complex interval can open the door to more complicated $\Delta^{(k)}$ definitions, e.g. calculating intersections and also mean length, etc. This would require an extension of the stability test itself, since a single test sample is not enough for this case.

References

[1] Le Bihan, D., Breton, E., Lallemand, D., Grenier, P., Cabanis, E., & Laval-Jeantet, M. (1986). MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. Radiology, 161(2), 401-407.

[2] Barbieri, S., Gurney-Champion, O. J., Klaassen, R., & Thoeny, H. C. (2020). Deep learning how to fit an intravoxel incoherent motion model to diffusion-weighted MRI. *Magnetic resonance in medicine*, *83*(1), 312-321.

[3] Shafer, G., & Vovk, V. (2008). A Tutorial on Conformal Prediction. Journal of Machine Learning Research, 9(3).

[4] Ortiz-Jiménez, G., Modas, A., Moosavi-Dezfooli, S. M., & Frossard, P. (2021). Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness. Proceedings of the IEEE, 109(5), 635-659.

[5] Kim, B., & Barber, R. F. (2021). Black box tests for algorithmic stability. *arXiv preprint arXiv:2111.15546*.