Fast Uncertainty Quantification for Deep Learning-based MR Brain Segmentation

Benjamin Lambert, Florence Forbes, Senan Doyle, Alan Tucholka, Michel Dojat

Fast Uncertainty Quantification for Deep Learning-based MR Brain Segmentation

Benjamin Lambert[*,***], Florence Forbes[**], Senan Doyle[***], Alan Tucholka[***], Michel Dojat[*]

[*]Univ. Grenoble Alpes, Inserm, U1216, Grenoble Institut Neurosciences, 38000, FR
[**]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, FR
[***]Pixyl, Research and Development Laboratory, 38000 Grenoble, FR

Abstract. Quantifying the uncertainty attached to Deep Learning models predictions can help their interpretation, and thus their acceptance in critical fields. Yet, current standard approaches rely on multi-steps approaches, increasing the inference time and memory cost. In clinical routine, the automated prediction has to integrate into the clinical consultation timeframe, raising the need for faster and more efficient uncertainty quantification methods. In this work, we propose a novel model, named as BEHT, and evaluate it on an automated segmentation task of White-Matter Hyperintensities from T2-weighted FLAIR MRI sequences of Multiple-Sclerosis (MS) patients. We demonstrate that this approach outputs predictive uncertainty much faster than the state-of-the-art Monte Carlo Dropout approach, with a similar — and even slightly better — accuracy. Interestingly, our approach distinguishes 2 distinct sources of uncertainties, namely aleatoric and epistemic uncertainties.

1 Introduction

The ever-growing usage of Deep Learning (DL) models in the industry, as well as their potential impact on human lives, is raising concerns regarding the opacity of their predictions. Thus, many studies have been focusing on producing explainable DL models to facilitate their interpretation by end-users (Arrieta et al., 2018). Among the large variety of explanations that can be provided along with a given prediction, Uncertainty Quantification (UQ) techniques stand out as one of the most popular amongst clinicians (Tonekaboni et al., 2019). UQ methods complement a prediction with an uncertainty score, providing the user with additional information regarding the model confidence in its own prediction.

Predictive uncertainty, i.e. the uncertainty attached to the output of a DL model for a given query input, is traditionally decomposed in 2 parts: (i) aleatoric and (ii) epistemic uncertainties (Gal, 2016). Aleatoric uncertainty (i) relates to random effects impacting the prediction, such as the presence of noise or artifacts in the input data. This part is irreducible, meaning that introducing extra data will not diminish it. Aleatoric uncertainty can be further split in two categories : homoscedastic uncertainty, which is constant for each input, and heteroscedastic that depends on the input. On the other hand, epistemic uncertainty (ii) is linked to the choice

of the model parameters. It represents the lack of knowledge of the model and can be reduced given additional and complementary data.

Various methods have been proposed in the literature to quantify uncertainty attached to DL predictions (Abdar et al., 2021). Popular approaches rely on a multi-step process, either based on the computation of multiple predictions from the same stochastic model (Blundell et al., 2015; Gal and Ghahramani, 2016), or alternatively based on the aggregation of predictions from multiple models (Lakshminarayanan et al., 2017). This results in a prolonged inference process, representing a significant obstacle for the full acceptance of these solutions in the clinical field, where time is a critical variable.

In this work, we propose a new DL model that quantifies both aleatoric and epistemic uncertainties attached to DL prediction, with a single forward pass. We illustrate our framework on an automatic segmentation task to detect White-Matter Hyperintensities (WMH) from T2-weighted FLAIR MRI sequences of Multiple-Sclerosis (MS) patients, and compare it with the Monte Carlo Dropout (MC-Dropout) state-of-the-art approach.

2 Related Work

2.1 Aleatoric and Epistemic Uncertainty

Distinguishing between aleatoric and epistemic uncertainties is challenging, as both sources are usually mixed up in the final estimated uncertainty (Hüllermeier and Waegeman, 2021). Yet, this distinction is particularly important in the medical domain. For example, a radiologist could react differently in response to a high predictive uncertainty if informed that it is caused by a lack of knowledge of the model about the observed pathology (epistemic uncertainty), or by the presence of an artifact or noise in the MRI input data (aleatoric uncertainty) (Senge et al., 2014).

Previous work has attempted to make such a distinction. Kendall and Gal (2017) implemented a Monte Carlo Dropout network with 2 outputs: one for the predictive mean $\hat{y}$ and one for the predictive variance $\hat{\sigma}^2$. Such a model is trained with Dropout, which is kept on at inference. As a result, multiple forward passes through the Neural Network (NN) yield to different predictions, as the Dropout mask is stochastically sampled at each pass. Additionally, they used a special loss to learn the variance $\hat{\sigma}^2$ during training and interpret it as the aleatoric uncertainty. At inference, T Monte Carlo Dropout samples were computed for each input query x, resulting in a set $\{\hat{y}_t, \hat{\sigma}_t^2\}_{t=1}^T$. The final voxel-wise predictive uncertainty $PU(x)$, combining an epistemic and an aleatoric part, was obtained with :

$$PU(x) = \underbrace{\frac{1}{T}\sum_{t=1}^{T}\hat{y}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{y}_t\right)^2}_{epistemic} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2}_{aleatoric} \tag{1}$$

Similarly, Kwon et al. (2020) used a MC-Dropout model to quantify the two types of uncertainty. They proposed 2 distinct estimators to evaluate epistemic and aleatoric uncertainties based solely on the output predicted probabilities $\{\hat{p}_t\}_{t=1}^T$, obtained after applying the Softmax function to a set of Monte Carlo Dropout samples. For a given input x, the predictive uncertainty is obtained as :

$$PU(x) = \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \overline{p}_t)^{\otimes 2}}_{epistemic} + \underbrace{\frac{1}{T}\sum_{t=1}^{T} diag(\hat{p}_t) - \hat{p}_t^{\otimes 2}}_{aleatoric} \tag{2}$$

where $\overline{p}_t = \sum_{t=1}^{T} \hat{p}_t / T$ and $v^{\otimes 2} = vv^T$.

Depeweg et al. (2018) implemented a Bayesian NN by placing distributions over the model weights w. At inference, they approximated the predictive distribution $p(y|x, D)$ associated with the training dataset D by sampling the weight distribution using a Monte Carlo approach. The authors further computed the total predictive uncertainty as the entropy of the predictive distribution $H[p(y|x)]$. This term includes both the epistemic and aleatoric sources. By fixing a set of weights, the epistemic term disappears, meaning that the expectation over the entropies can be used as an estimator of the aleatoric uncertainty. Finally, both sources can be obtained as follows:

$$H[y|x] - \underbrace{\mathbb{E}_{w|D}\, H[p|y, x)}_{aleatoric} = \underbrace{I(y, w)}_{epistemic} \tag{3}$$

2.2 Efficient Uncertainty Quantification Methods

Methods presented in the previous section — such as MC-Dropout or Bayesian NN — require sampling at inference, highly extending the inference time. In a similar way, Deep Ensemble (Lakshminarayanan et al., 2017), another popular approach for UQ based on the inference of multiple models, drastically increases the computational and memory cost during both training and testing. In clinical routine, e.g for MS follow-up, the DL-based assistance is useful only when the automatic analysis is fast enough to be integrated into the clinical consultation. This motivates the need for efficient and fast UQ methods. In the following, we review several techniques that tend toward this goal. Each of these methods, when applied to a 3D image segmentation task, provide one measure of uncertainty per voxel.

Wen et al. (2020) proposed an efficient alternative to Deep Ensemble called BatchEnsemble, which highly reduces its memory cost. The method proposes to train an ensemble of NN within a single architecture. To achieve this result, the weights of the NN are expressed as the Hadamard product of a shared weight among all ensemble members (called as slow weights) and member-specific rank-one matrices (called as fast weights). This special formulation results in different weight configurations for each member, hence modeling the possible variance of the weights, and thus the epistemic uncertainty. For an ensemble of 4 models, BatchEnsemble reduces test time and memory by up to 3 times, compared to Deep Ensemble.

Similarly, an effective approach to quantify heteroscedastic aleatoric uncertainty was proposed in McKinley et al. (2020). As in Kendall and Gal (2017), the model output is divided in two parts: the prediction and the uncertainty. Authors proposed a Labelflip loss, which is used to learn the uncertainty associated with input images without labels. This forces the model to learn the Labelflip probability, *i.e.* the probability the predicted label and the ground truth label differ. This method thus outputs the prediction and the associated aleatoric uncertainty in a single forward step.

Alternative works have directly used the output predicted probabilities as a measure of uncertainty, with no further computation. This very simple and efficient approach was successfully applied to detect misclassified and out-of-distribution images (Hendrycks and Gimpel, 2017), as well as adversarial examples (Zhang et al., 2020).

These methods, although being more efficient regarding inference time or memory cost, yet fail at distinguishing between aleatoric and epistemic uncertainties. In the next section, we propose a new method addressing this challenge, that we call BEHT for BatchEnsemble Heteroscedastic Tiramisu.

3 Methods

In this section, we introduce a novel approach to quantify both epistemic and aleatoric uncertainties in an fast and efficient manner. This is achieved by combining a BatchEnsemble approach with a Heteroscedastic model, starting from a baseline Tiramisu 2.5D architecture. We named this compound model as the BatchEnsemble Heteroscedastic Tiramisu, or BEHT.

3.1 Dataset

We illustrate our proposed approach on a supervised segmentation task. We use a proprietary brain dataset composed of 238 T2-weighted FLAIR MRI sequences of MS patients, with ground truth segmentations of WMH. The dataset is split into 187 scans for training and 51 for testing. The images are rigidly registered to a template and resampled to a 1mm isotropic resolution of $160 \times 192 \times 160$ to focus on brain tissue. Intensities are normalized to zero mean and unit variance.

3.2 Baseline 2.5D Architecture

We start with a standard Tiramisu Convolutional Neural Network (CNN) for image segmentation, originally proposed for 2D images semantic segmentation (Jégou et al., 2017). Processing 3D biomedical image is challenging, as the input data has a high dimensionality and requires large memory capabilities. To circumvent this limitation, we implement an intermediate 2.5D version of the Tiramisu network. In this setting, the segmentation model focuses on one slice, with several upper and lower slices provided to the model. More precisely, 3D images of shape $H \times W \times D$ are processed sequentially. At each step, C consecutive 2D slices are extracted from the volume and stacked as distinct image channels, resulting in a $C \times H \times W$ input. We use $C = 5$ in our experiments. Our Tiramisu 2.5D networks is composed of the same construction blocks than the original 2D Tiramisu, with a succession of Dense Blocks (DB), Transition Down (TD) and Transition Up (TU) layers (see Jégou et al. (2017) for implementation details). This baseline model is made of a total of 51 convolutional layers and 950k parameters. The model produces for each voxel a unique probability of being in the lesion class. Probabilities are obtained by applying the Sigmoid function to the predicted logits. Details of the proposed architecture are presented in Table 1.

The Tiramisu 2.5D network and following models are implemented using the Pytorch DL library (Paszke et al., 2019). Training is performed using the ADAM optimizer (Kingma and

Architecture Details
Input
3x3 Conv + DB block (1 layer)
DB block (3 layers) + TD
DB block (5 layers) + TD
DB block (7 layers) + TD
TD + DB block (10 layers) + TU
TU + DB block (7 layers)
TU + DB block (5 layers)
TU + DB block (3 layers)
DB block (1 layer) + 1x1 Convolution
Sigmoid

TAB. 1 – ***Architecture details of the baseline Tiramisu 2.5D model used in the experiments.*** *Conv=Convolution 2D, DB=Dense Blocks, TD=Transition Down, TU=Transition UP*

Ba, 2015) until convergence, with a learning rate of $5e^{-4}$. We use a batch-size of 48, and train on a single NVIDIA T4 GPU.

3.3 Modeling Epistemic Uncertainty with BatchEnsemble

Building on the baseline architecture, we model epistemic uncertainty using the BatchEnsemble framework presented in Wen et al. (2020). We obtain a BatchEnsemble of $N = 6$ members from a single Tiramisu 2.5D by replacing each convolutional weights by :

$$W_i = W \circ F_i \tag{4}$$

where W is a common weight matrix shared by each ensemble member (slow weight) and F_i is a rank-one matrix specific to each of the i-th ensemble members (fast weight). Learning of these weights is parallelized within a single GPU device using a mini-batch approach. We divide the batch of length $b = 48$ in $N = 6$ sub-batches of length $M = 8$, such as $b = N \times M$. Each member's weight $\overline{W_i}$ receives a single sub-batch during training, allowing to train the BatchEnsemble with an optimal vectorized approach. At inference, the input is repeated i times, so that each member processes it in parallel. Lastly, we take the mean of the prediction as the final output. These steps are illustrated in Figure 1

Writing $\{p_n\}_{n=1}^{N}$ the set of lesion probabilities obtained with each member of the BatchEnsemble for a given input image x, the epistemic uncertainty $Ep(x)$ is obtained by computing the variance of this set :

$$Ep(x) = \frac{1}{N}\sum_{n=1}^{N} p_n^2 - \left(\frac{1}{N}\sum_{n=1}^{N} p_n\right)^2 \tag{5}$$

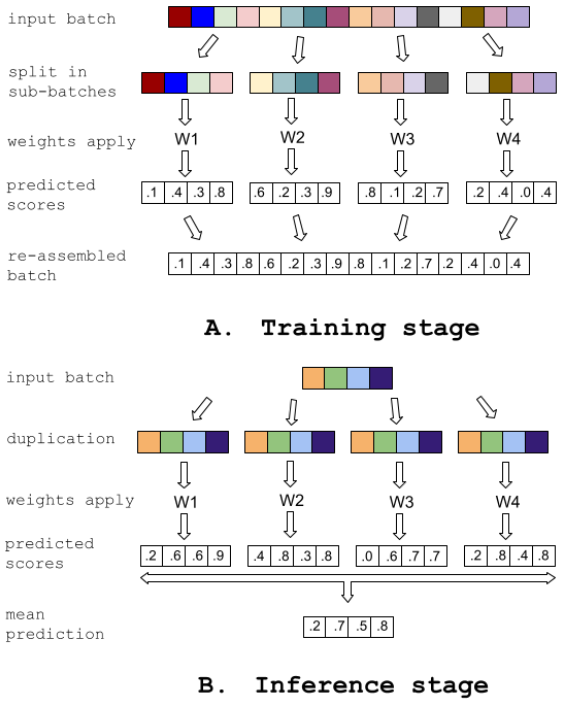

FIG. 1 – ***Illustration of the training and inference stages in a BatchEnsemble model.*** *In this example, a batch of 16 samples is distributed to 4 members during training. At inference, a batch of 4 is duplicated and distributed to each member weight.*

3.4 Modeling Aleatoric Uncertainty with the Labelflip loss

We transform our model into a heteroscedastic one by following the steps of (McKinley et al., 2020). In addition to the lesion probability for each voxel, the NN also predicts its attached uncertainty q in a single forward pass. To learn the uncertainty part during training, the Labelflip loss is employed. Writing x the binary ground truth label, $w = (1-x)*q+x*(1-q)$, and z the disagreement indicator between the predicted class and the ground truth, the labelflip loss is defined as :

$$\text{Focal}_{KL}(w\|p) + BCE(q,z) \tag{6}$$

$$\text{with Focal}_{KL}(w\|p) = (p-w)^2(wlog(w) - wlog(p)) \tag{7}$$

This function forces the model to output a high uncertainty in areas where the segmentation is not confident. The use of a Kullback–Leibler divergence term guarantees that uncertain voxels remain close to the decision boundary. We used the Dice loss in conjonction of the Labelflip loss to train the model.

Reminding that BEHT outputs a distinct prediction per member of the BatchEnsemble, we obtained a set of $\{q_n\}_{n=1}^{N}$ uncertainties for each input image. The final aleatoric uncertainty is formulated as the mean of this ensemble :

$$Al(x) = \frac{1}{N}\sum_{n=1}^{N} q_n \tag{8}$$

3.5 Predictive Uncertainty Quantification using the BEHT approach

The overall predictive uncertainty is obtained by summing the aleatoric and epistemic contributions. Yet, both quantities have distinct numeric ranges due to the different methods employed to compute them, which makes normalization step necessary. To do so, we first compute epistemic and aleatoric uncertainty maps for each test image, and extract the minimum and maximum voxel uncertainties across the test dataset. We then normalize each map in the range $[0, 1]$ by subtracting by the minimum and dividing by the range. The final predictive uncertainty $PU(x)$ for a given query image x is finally obtained with :

$$PU(x) = \frac{1}{2}[Ep_{[0,1]}(x) + Al_{[0,1]}(x)] \tag{9}$$

4 Experiments

We evaluated our proposed model with respect to 3 aspects: the quality of uncertainty estimates, the performance of the segmentation, and the inference time. To compare these results with state-of-the-art method, we also implemented a MC-Dropout Tiramisu 2.5D. Starting from the baseline model, a dropout rate of 20% is applied in the Dense-Blocks and Transition-Down layers, as proposed in (Jégou et al., 2017). The model is trained with a combination of the Dice and the Binary Cross-entropy losses. At inference, epistemic and aleatoric uncertainties are obtained using Equation 2 with $T = 20$ Monte Carlo samples. For each method, we compute segmentation masks and uncertainty maps. Representative examples of the obtained predictions are presented in Figure 2.

4.1 Evaluating Uncertainty Estimates

To evaluate the performance of the different uncertainty maps — epistemic, aleatoric, and predictive — generated by the 2 competing approaches, we implement a stratification approach. The desired uncertainty quantification highlights unconfident predictions, which are more likely to be incorrect. Thus, we expect uncertainty to be higher for incorrect predictions (IP) than for correct predictions (CP). By filtering predictions based on their certainty, we should remove more IP than CP.

To evaluate this property, we progressively filter out predicted voxels based on their certainty and monitor the variation of CP and IP. More precisely, at each step, we remove the $X\%$ most uncertain predicted voxels, where X is a threshold varying in the set $[0, 100]$. As a result, we obtain of couple (CP, IP) for each threshold X, used to draw a Stratification Curve.

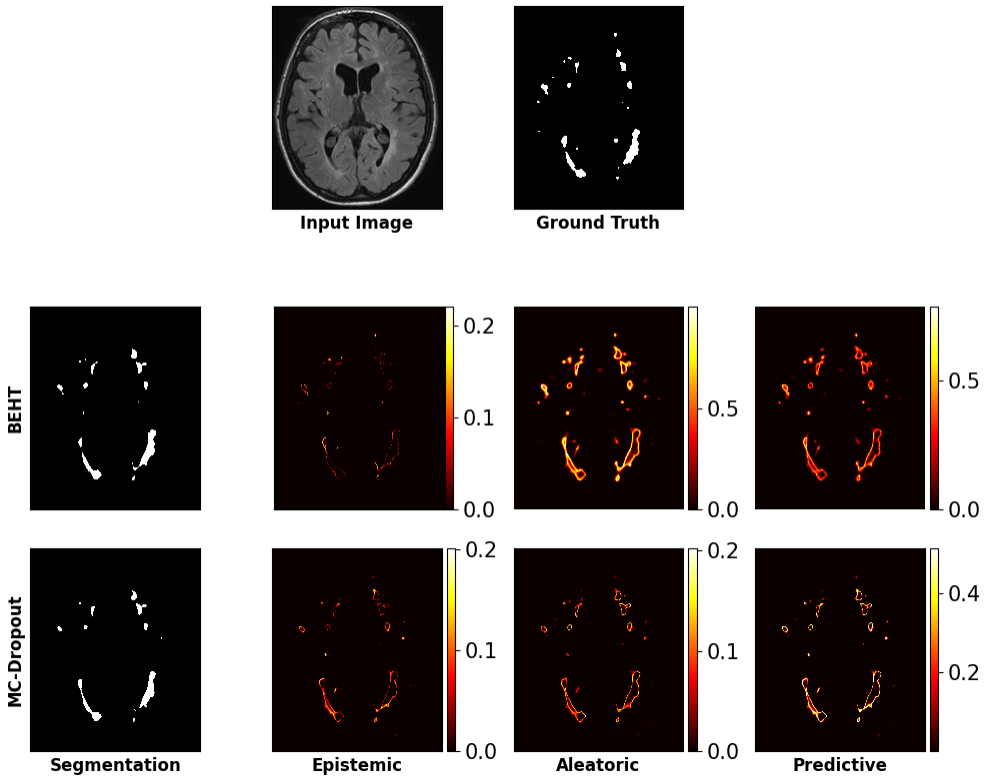

FIG. 2 – ***Representative examples of the segmentation masks and uncertainty maps obtained using each method.***

As the evaluated methods have varying segmentation performances, the CP and IP counts are different. Thus, we normalize the CP and IP counts in the range $[0, 1]$ so that the segmentation performance does not impact the evaluation of the uncertainty estimates. A CP of 1 corresponds to the maximum number of CP, obtained when no filtering is applied. A CP of 0.5 indicates that 50% of the CP are filtered out. To obtain a single quantitative score, we use the Area Under the Stratification Curve (AUSC).

4.2 Segmentation Performance

The desired UQ method should provide useful uncertainty estimates, while also preserving a satisfying segmentation performance. To evaluate this property, we assess the segmentation performance using Dice scores, a measure of overlap between the predicted segmentation and the ground truth. Segmentation quality improves as the Dice score increases.

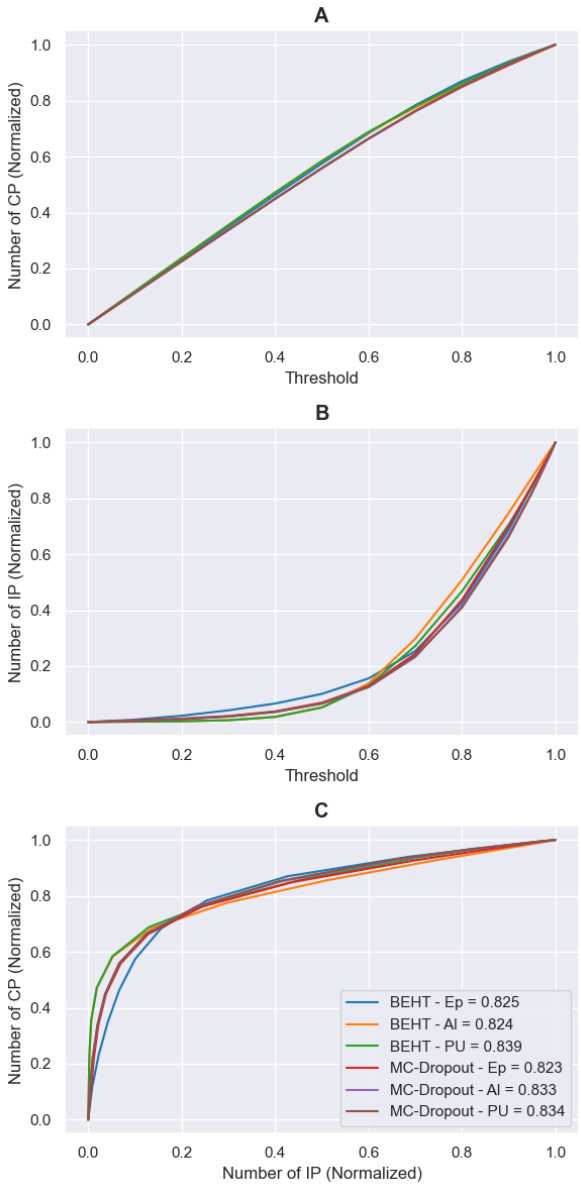

FIG. 3 – ***Evaluation of incertainty estimates.*** *A, B: Variation of Correct (CP) and Incorrect (IP) Predictions counts, respectively, with respect to the uncertainty threshold; C: Corresponding Stratification Curve and associated AUSC scores. Ep: Epistemic uncertainty; Al: Aleatoric uncertainty; PU: Predictive Uncertainty.*

	BEHT	MC-Dropout
Dice	0.785 ± 0.076	0.777 ± 0.083

TAB. 2 – ***Segmentation performance.*** *For each method, we provide the mean Dice and the standard deviation. Top performing method is highlighted in green.*

4.3 Inference time

Lastly, for smooth integration in clinical practice, the method should be as fast as possible. We compute the Mean Inference Time (MIT), representing the total duration in seconds of the computations steps for a single 3D volume. This represents the iterative segmentation of the volume using the 2.5D model, as well as the uncertainty quantification. The fastest method is the one that minimises the MIT.

	BEHT	MC-Dropout
MIT	5.434 ± 0.062	16.535 ± 0.485

TAB. 3 – ***Inference time performance.*** *For each method, we provide the MIT and the associated standard deviation. Top performing method is highlighted in green.*

5 Results and Discussion

Overall, BEHT outperforms the MC-Dropout approach on all 3 metrics. Regarding the UQ task, Figure 3 reveals that the AUSC score of the predictive uncertainty obtained with BEHT is slighty better than for MC-Dropout : 0.839 versus 0.834. Furthermore, the proposed approach also achieve the best segmentation quality as shown in Table 2. Finally, the most impressive improvement comes from the inference time, as the BEHT approach is 3-times faster than the MC-Dropout one, as presented in Table 3. In summary, our new BEHT method reaches — and even slightly overpasses — the UQ and segmentation performance of the state-of-the-art MC-Dropout approach, while drastically reducing the inference time.

Interestingly, for both approaches, combining the aleatoric and epistemic uncertainties results in improving the quality of the uncertainty estimates, demonstrated by the improvement of the AUSC score (see Figure 3). This suggests that taking into account both uncertainty sources is important and can actually provide more robust uncertainty estimates. Finally, our model provide robust uncertainty estimates without sacrificing segmentation performance, reaching a Dice score of 0.785 on the test dataset.

6 Future Directions

The proposed 2.5D approach is efficient regarding memory consumption, yet the 3D volume is processed sequentially, which prolonged the inference process and also limits the segmentation quality. Our future work will consist in the implementation of a fully 3D model which could reduce the inference time.

References

Abdar, M., F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. W. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion 76*, 243–297.

Arrieta, A. B., N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera (2018). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion 58*, 82–115.

Blundell, C., J. Cornebise, K. Kavukcuoglu, and D. Wierstra (2015). Weight uncertainty in neural network. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015 37*, 1613–1622.

Depeweg, S., J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft (2018). Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018 80*, 1192–1201.

Gal, Y. (2016). *Uncertainty in deep learning*. Ph. D. thesis.

Gal, Y. and Z. Ghahramani (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016 48*, 1050–1059.

Hendrycks, D. and K. Gimpel (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *5th International Conference on Learning Representations, ICLR 2017*.

Hüllermeier, E. and W. Waegeman (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning 110*, 457–506.

Jégou, S., M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017*, 11–19.

Kendall, A. and Y. Gal (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017 30*, 5574–5584.

Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015*.

Kwon, Y., J. Won, B. Kim, and M. C. Paik (2020). Uncertainty quantification using bayesian neural networks in classification : Application to biomedical image segmentation. *Comput. Stat. Data Anal. 142*.

Lakshminarayanan, B., A. Pritzel, and C. Blundell (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017 30*, 6402–6413.

McKinley, R., M. Rebsamen, K. Daetwyler, R. Meier, P. Radojewski, and R. Wiest (2020). Uncertainty-driven refinement of tumor-core segmentation using 3d-to-2d networks with

label uncertainty. *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries - 6th International Workshop, BrainLes 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Revised Selected Papers, Part I 12658*, 401–411.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, 8024–8035.

Senge, R., S. Bösner, K. Dembczynski, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier (2014). Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Inf. Sci. 255*, 16–29.

Tonekaboni, S., S. Joshi, M. D. McCradden, and A. Goldenberg (2019). What clinicians want: Contextualizing explainable machine learning for clinical end use. *Proceedings of Machine Learning Research 106*, 359–380.

Wen, Y., D. Tran, and J. Ba (2020). Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *8th International Conference on Learning Representations, ICLR 2020*.

Zhang, X., X. Xie, L. Ma, X. Du, Q. Hu, Y. Liu, J. Zhao, and M. Sun (2020). Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 739–751.

Résumé

Quantifier l'incertitude liée aux prédictions des modèles Deep Learning permet d'améliorer leur interprétation, favorisant ainsi l'acceptation de ces algorithmes. Cependant, les méthodes actuelles reposent sur des approches multi-étapes, ce qui augmente le temps d'inférence ainsi que les coûts en mémoire. En routine clinique, l'utilisation d'outils prédictifs nécessite une intégration dans le temps de la consultation clinique, ce qui motive le développement de nouvelles méthodes rapides et efficaces pour quantifier l'incertitude des modèles Deep Learning. Dans ce travail, nous proposons un nouveau modèle nommé BEHT, que nous évaluons sur une tâche de segmentation automatique d'hyper-intensités de la matière blanche, sur des séquences T2-FLAIR d'IRM de patients atteints de Sclérose en plaques. Nous démontrons que cette approche quantifie l'incertitude prédictive beaucoup plus rapidement que la méthode classique du Monte Carlo Dropout, avec une performance équivalente — et même légèrement supérieure. De façon intéressante, notre méthode permet de distinguer entre 2 sources d'incertitude différentes, à savoir les incertitudes aléatoire et épistémique.