Ultrasound Nerve Segmentation Using Deep Probabilistic Programming

Iresha Rubasinghe & Dulani Meedeniya*

Department of Computer Science and Engineering, Faculty of Engineering, University of Moratuwa, Moratuwa 10400, Sri Lanka
*E-mail: dulanim@cse.mrt.ac.lk

Abstract. Deep probabilistic programming concatenates the strengths of deep learning to the context of probabilistic modeling for efficient and flexible computation in practice. Being an evolving field, there exist only a few expressive programming languages for uncertainty management. This paper discusses an application for analysis of ultrasound nerve segmentation-based biomedical images. Our method uses the probabilistic programming language Edward with the U-Net model and generative adversarial networks under different optimizers. The segmentation process showed the least Dice loss (-0.54) and the highest accuracy (0.99) with the Adam optimizer in the U-Net model with the least time consumption compared to other optimizers. The smallest amount of generative network loss in the generative adversarial network model gained was 0.69 for the Adam optimizer. The Dice loss, accuracy, time consumption and output image quality in the results show the applicability of deep probabilistic programming in the long run. Thus, we further propose a neuroscience decision support system based on the proposed approach.

Keywords: *deep learning; deep probabilistic programming; generative adversarial network; nerve segmentation; neuroscience decision support.*

1 Introduction

Deep learning (DL) has become a leading technique in data science research with its rapid algorithmic improvements, advances in hardware technology and support for exponentially growing data. DL is advantageous over traditional machine learning (ML) mainly due to the automatic feature learning capability on larger datasets, higher accuracy and faster inference [1,2]. The basis of DL architectures is similar to that of the human nervous system, where large sets of neurons are linked and pass data [3].

Deep probabilistic programming (DPP) is a DL technique that is a blend of a deep neural network and a probabilistic model that is able to perform computations efficiently and flexibly. A deep neural network (DNN) is an automatic hierarchical representation of a learning model that consists of several layers between the input and the output layer. DNN uses mathematical concepts

Received July 4th, 2019, Revised December 31st, 2019, Accepted for publication December 31st, 2019.
 ISSN: 2337-5787, DOI: 10.5614/ itbj.ict.res.appl.2019.13.3.5

and languages for complex data processing. Probabilistic programming describes probabilistic models, performs inferencing and supports decision making under uncertainty. DNN exemplifies the models in deep learning using its feed-forward neural networks with hidden layers. A generative adversarial network (GAN) is a type of DNN that trains two models in parallel [1]. The convolutional neural network (CNN) is a DNN architecture for spatial data that is widely used, for example in image classification with minimal pre-processing, and is inspired by the human visual system [1,4]. The need for probabilistic programming for DL models has eventually led to the growth of deep probabilistic programming languages (DPPL) [5,6]. The expressiveness and powerful inferencing capabilities of DPPLs facilitate the uncertainty management of DL models. However, the use of DPPLs in DL model solutions has not been evaluated well due to the novelty of the leading DPPLs.

The contribution of this study is to propose an application of a DL GAN architecture with probabilistic modeling using the novel DPPL Edward, which addresses the software engineering perspective of DL and DPPL [5]. Nerve segmentation on ultrasound image data was used in an experiment to test the proposed methodology [7]. The accurate identification of nerve structures is important during surgery when catheters are inserted to reduce pain. Manually differentiating nerves in an ultrasound scan is a complex process, which motivated the authors to research ultrasound nerve image segmentation.

The main aim was to see the possibility of applying both DL and DPPL to decide nerve points in support of surgical catheter insertion. First, the principles of GAN were embedded into the popular biomedical image segmentation model U-Net. The performance was analyzed using a set of ultrasound images with prominent optimizers [3]. Then, GAN with inferencing capabilities was applied in DPPL Edward on the same dataset. The performance was analyzed using the same set of optimizers [5]. The novelty of this study is the exploration of the applicability of DPPL for medical image segmentation. Furthermore, we also propose a generic model for a decision support system based on healthcare data.

2 Background

2.1 Overview of DPPLs

Probabilistic programming provides an abstract way to specify probabilistic models as programs and compile them into inference procedures without exposing the underlying complex inference algorithms. Thus, probabilistic programming languages (PPLs) eliminate the representational gap between general purpose programming and probabilistic modeling [5]. There exist PPLs that extend general languages. For example, Dimple extends Java, PRISM

extends Prolog, Infer.NET extends the .NET framework, and Anglican extends Clojure, while Stan is an example of a self-contained state-of-the-art PPL written in C++ [5,8]. Stan is a mature, expressive and imperative PPL that supports conditional statements and variable declarations, where all expressions are statically typed, including variables. It is based on Hamiltonian Monte Carlo (HMC) with built-in inference engines, including Markov Chain Monte Carlo (MCMC) samplers and optimizers, which perform Bayesian inference [8].

According to related studies on PPLs, Birch is an imperative, Turing complete, universal PPL [9]. A unique option in Birch is delayed sampling, which delays the execution of a reached checkpoint to provide optimization for inference problems using partial analytical solutions. The work in [10] extended the delayed sampling concept by focusing on Sequential Monte Carlo to reduce the variance in estimations. Although it has a better user experience for PPLs, it has an additional computational cost. Another domain-specific PPL is TerpreT [11], which is based on gradient descent and linear program relaxation on graphical models. TerpreT compiles different inference algorithms by separating the model specification from the inference algorithm. Thus, hierarchical modeling becomes a powerful method for probabilistic programming. Most of the PPLs adhere to general MCMC methods for inference is because of the inability to specialize the inference techniques over language expressiveness. Thus, PPLs trade off the expressiveness of the language against the computational efficiency of the inference [5,8].

DPPLs have started to evolve into a combination of DL and PPL with the introduction of variational inference (VI), which converts an inference problem to an optimization problem and outperforms the sampling-based Monte Carlo methods in Bayesian on larger datasets [5,6,12]. Edward and Pyro are two of the leading state-of-the-art, Turing complete DPPLs that are actively used in current research. Edward [5] is based on TensorFlow [13] and defines two compositional representations as random variables and inference. It is capable of a variety of inference methods, from point estimation to VI, to MCMC, which makes it a deep PPL. Edward shows that reusing the modeling representation within inference can improve variational models and GANs and is more than 35 times faster than Stan, not having any runtime overhead. Pyro [6] released by Uber AI Labs after Edward, is based on the PyTorch framework [14]. Pyro scales to larger datasets by adapting a gradient-based stochastic VI algorithm that uses SGD along with other probabilistic inference algorithms. Further, Pyro has better language expressiveness in terms of dynamic control in comparison to both Stan PPL and Edward, which have static control flow. In contrast, Edward is more concise than Pyro. Otherwise, Edward and Pyro have similar scalability and flexibility characteristics. Due to DPPLs' success, Stan has also been extending towards DeepStan using VI on PyTorch like Pyro [12].

In this approach, the Stan programs were primarily translated into Pyro. Another novelty was added to the DPPL context in related work [15], which presented a modular DPPL library called MXFusion. It introduced reusable building blocks, called probabilistic modules, consisting of random variables with probabilistic distribution and inference methods to reduce the performance gap. Hence, DPPL has significant promise as a new active research field.

2.2 Machine-Learning Frameworks

Several machine-learning frameworks that provide built-in functions for pre-processing are available, such as TensorFlow [13], PyTorch [14], Microsoft Cognitive Toolkit (CNTK) [16], Apache MXNet [17], etc. Table 1 summarizes the applicability and limitations of these machine-learning frameworks.

Table 1 Comparison of DL frameworks.

Features	TensorFlow	PyTorch	CNTK	Apache MXNet
ML/ DL Applicability	High	High	High	High
Scalability	High	Low	High	High
In-built visualization	TensorBoard	Low	Low	Low
Programming language support	Python	Python	Python, C++, C# and Java	Python, JavaScript, Julia, C++, Matlab, Scala, R, Perl, Go and Wolfram
API support	High	Low	High	High
Debugging	Low	High	High	High

TensorFlow [13] is an open-source machine-learning framework, comprising a set of ML and DL algorithms that provides functions to obtain data, train the model, predict and refine the results on a large scale. The front-end application is based on Python and the backend execution is based on C++. TensorFlow is platform-independent and several studies have used it for image classification. The developer can describe the abstract structure of the data flow using a set of nodes, where each node consists of a computation. Each link among the nodes is a tensor or a data array. Thus, the user can get an overall view of the application without knowledge of the internal functional handling.

Similarly, PyTorch [14] is a high-performance and interactive development model. This enables efficient development, but it may have scalability issues and limitations in complex workflows compared to TensorFlow. CNTK is also an open-source framework, which mainly considers the efficient creation of DL neural networks. It uses a node-link structure with a set of computations to represent the dataflow. This can be included as a library in Python, Java, C++

and C# or as a standalone tool using its model description language. Apache MXNet [17] is another leading DL framework, with a scalability feature among many GPUs and machines. It supports a range of programming languages such as Python, JavaScript, Julia, C++, Matlab, Scala, R, Perl, Go and Wolfram.

2.3 Related Studies

Various imaging modalities are used in practice for advanced diagnosis purposes, such as MRI, fMRI, EEG and CT scans [4,18,19]. Manual analysis of biomedical imaging data is challenging due to its complexity. Thus, computational models for diagnosis of biomedical imaging data using deep learning are actively investigated [20,21]. Table 1 summarizes related work that has incorporated DL and the use of DPPLs related to probabilistic aspects, such as VI, VAE and GAN for neuroimaging.

Table 2 Related work on medical imaging analysis using learning models.

Related work	Dataset	Advantages	Limitations
A variational Bayes method for MRI image segmentation [18].	Synthetic data from Brainweb and real MRIs from OASIS, IXI.	Increases computational stability and robustness.	Low accuracy in real data due to the complexity in MRI. DPPL not included.
Handling uncertainties in CNN-based MRI image segmentation [4].	T2-weighted MRIs of 60 fetuses.	Better uncertainty estimation and minimization of incorrect predictions.	Support small training dataset due to performance issues. DPPL not incorporated.
Visualizing thoughts from brain EEGs (ThoughtViz) [19].	EEG data from 3 public datasets.	Suitability for the domain- and scale-independent datasets.	Suited for small datasets. DPPL not incorporated.
Synthesizing medical images from a source image to a target without a scan [22].	MRIs and CT scan from 3 real datasets including ADNI database.	Reduce medical imaging costs for patients. Removes the blurry effect in the output.	DPPL not incorporated.
Pelvic organ segmentation from MRI (STRAINet) [20].	MRIs of 50 prostate cancer patients.	Resolves limitations in normal FCNs. Robust and high performance.	DPPL not incorporated.
Decode neuroimaging data in EEG (Brain2Image) [21].	EEG signal dataset with 2000 images.	Higher performance and shows that GAN is ahead of VAE.	Noise removal needs to be improved. DPPL not incorporated.

3 Design and Implementation

The methodology consisted of two phases. In Phase 1, nerve segmentation was performed using the U-Net DL model by embedding GAN features to identify the use of GAN and an ideal optimizer for U-Net model other than its originally specified SGD optimizer [23]. In Phase 2, a GAN model was implemented in

DPPL Edward and applied to the same dataset to determine the usefulness of an individual GAN model together with DPPL towards segmentation by improving image quality. The variations of applying different optimizers on a single GAN model were also analyzed in Phase 2. A publicly available dataset of ultrasound images published for a nerve segmentation competition held by Kaggle was selected for the application proposed in this work. It consists of 5635 ultrasound images of the neck in Tagged Image File Format (TIFF), each with dimensions of 580 x 420 [7]. Implementation was carried out in an environment consisting of Intel® Core (TM) i7-4790 CPU @ 3.600GHz and 8GB RAM.

3.1 Phase 1: Nerve Segmentation by U-Net CNN

Generally, segmentation groups similar pixels in an unsupervised learning approach, where parameter estimation is essential. Nerve segmentation determines the nerves in an ultrasound image [24]. Manually, it is challenging to differentiate the nerve structures accurately, which is crucial during surgical activities. Phase 1 of our methodology determined the segmentation performance of GAN adapted to U-Net and identified the differences in the performance of prominent optimizers, as shown in Figure 1.

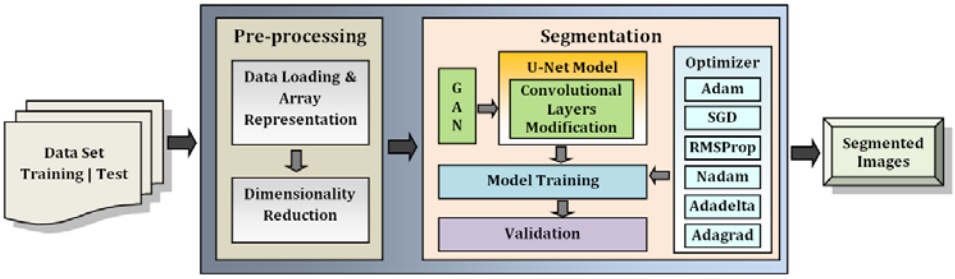

Figure 1 Phase 1 workflow.

U-Net is a fully convolutional network (FCN) specific for medical image segmentation by predicting classes of pixels [1]. The original U-Net model has convolutional layers combined with Max Pooling, ReLu activation functions and SGD optimizer [23]. We adjusted this U-Net model architecture, as shown in Figure 2, by convolution with 2x2 strides adapted from the GAN structure to download, instead of the originally specified Max Pooling. Further, we replaced the dense layers by 1x1 convolutional layers. The Keras neural network API in Python with TensorFlow as the backend was used for the implementation of the modified U-Net model [25]. Algorithm 1 summarizes the overall process.

First, the ultrasound images were pre-processed, resized to 96x96 and saved in NumPy binary file (.npy) format using NumPy Python library. Then, the dataset

was trained and validated for distinct optimizers, i.e. Adam, RMSProp, Nadam, Adadelta and Adagrad in addition to U-Net's initial SGD optimizer. Further, TensorBoard graph visualization was integrated to monitor the training results.

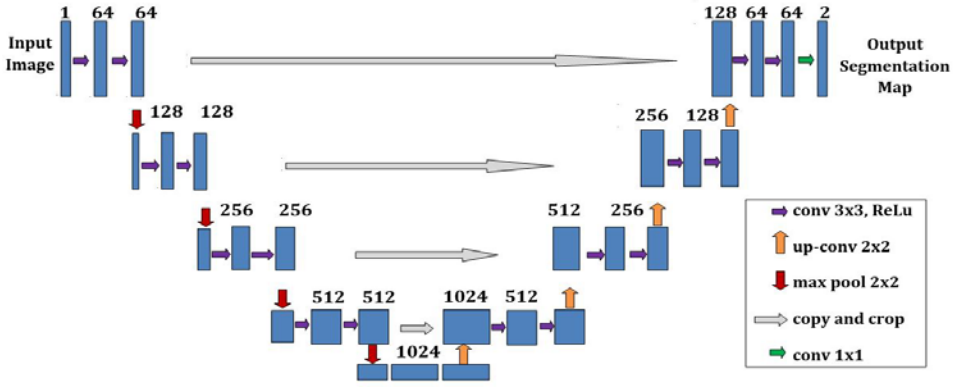

Figure 2 Modified U-Net model.

Algorithm: Phase 1

```
Pre-process (images)
Save in NumPy
Split dataset (images)
Training set = (0.8)*images
Validation set = (0.2)*images
TensorBoard log ()
Optimizers = [Adam, SGD, RMSProp, Nadam, Adadelta, Adagrad]
For optimizer in Optimizers:
    Training (Training set, optimizer)
    Validation (Validation set, optimizer)
        Save images
```

3.2 Phase 2: GAN model in DPPL

In the second phase, we implemented a GAN model in one of the leading DPPLs, Edward, on the same dataset with the ideal optimizer results obtained from phase 1 [5]. GAN has a generator that creates noise data and a discriminator that determines the real data. Thus, both the model and the loss functions are trained in a GAN, where the generator and discriminator sub-networks compete. On the other hand, DPPL Edward is powered with support for GAN using only a nominal amount of code as the GAN model structure is highly applicable for probabilistic modeling [5]. Figure 3 illustrates the abstract design combining GAN with inferencing capabilities in Edward.

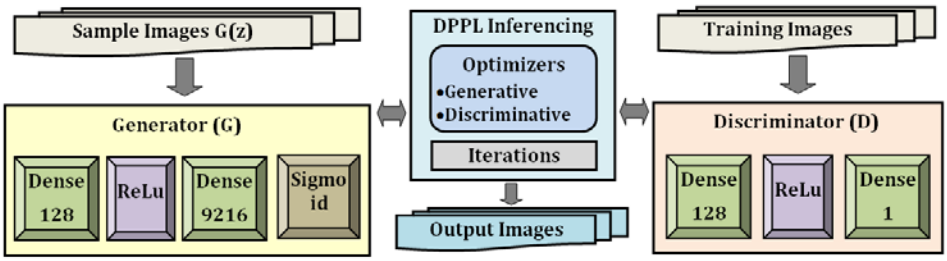

Figure 3 GAN model with DPPL.

Our goal in this phase was to infer a model that transforms ultrasound images for successful segmentation to mitigate performance and quality preservation issues in individual DL models and to increase the accuracy with probabilistic uncertainty handling. The same pre-processed dataset in NumPy file format in Phase 1 was used in Phase 2 to maintain the uniformity between both phases for comparison. As shown in Figure 3, we first implemented a generator subnetwork that feeds data in batches of size 128 into GAN.

The generative process can be formally specified as in Eq. (1), where G (.; θ) denotes a generator subnetwork that takes in ε samples and *p(ε)* represents injected random noisy data. Accordingly, we performed parameter estimation from the generative sub-network by adapting likelihood-free algorithms that encourage samples only from the model to learn by comparison [26].

$$\varepsilon \sim p(\varepsilon), \quad x = G(\varepsilon; \theta) ; \tag{1}$$

Secondly, we implemented the discriminator sub-network *D (.; θ)* using TensorFlow layers with ReLu activation function to compute the probability that a given data *(x)* obtained by the discriminator is from the real data. Consequently, the optimization problem we used in GAN is as shown in Eq. (2), where p*(x) denotes the real data distribution.

In the DPPL Edward, we implemented this using the existing function GANInference, which takes in a density model of the input data with a parameterized method called discriminator. We applied optimizers to the generator and the discriminator sub-networks by considering the different optimizer results gained from Phase 1.

$$\min_{\theta} \max_{\phi} \; \mathbb{E}_{p^*(\mathbf{x})}[\log D(\mathbf{x}; \phi)] + \mathbb{E}_{p(\mathbf{x};\theta)}[\log(1 - D(\mathbf{x}; \phi))] \tag{2}$$

Thirdly, GAN training was conducted with parameter learning. However, the criticism capabilities of the model for evaluation, which is a major portion of Edward, is currently an open challenge for GAN. Thus, the results were

analyzed using TensorBoard graph visualizations and produced sample outcomes of ultrasound images. Algorithm 2 summarizes the process.

Algorithm: Phase 2

```
Batch = Generator (batch size)
Inference = Edward GANInference (data, Discriminative network ())
Initialize inference (Generative network Optimizer, Discriminative network Optimizer,
Iterations)
For iteration in Iterations:
    Update inference (Batch)
    Save images
```

4 Evaluation of the Results

The GAN embedded U-Net network considered in Phase 1 consisted of a total of 7,759,521 parameters and was trained for 20 epochs under each optimizer based on the computational capabilities of the used hardware. A constant learning rate of 1.0×10^{-5}, batch size of 32, the Dice-coefficient, which is a popular loss function in image segmentation, training and testing metrics for measuring accuracy were configured. The dataset was split so that 20% was used for validation of the trained model following the characteristics in related works [19,22].

Table 3 presents the training and validation summary of our GAN embedded U-Net model, including Dice loss, accuracy and time taken for the leading optimizers. The results show that Adam exceeded both the accuracy and the performance of the other optimizers, including U-Net's originally specified optimizer, SGD. The lowest accuracy was obtained by the SGD and Adagrad optimizers. Figure 4 shows an instance of nerve segmented image outcomes for the Adam optimizer applied in the training and testing session.

Table 3 Summary of GAN embedded U-Net model results.

Optimizer	Training Dice loss	Validation Dice loss	Training accuracy	Validation accuracy	Time hr: min
Adam	-0.54	-0.30	0.99	0.99	4:58
SGD	-0.02	-0.02	0.72	0.76	5:07
RMSProp	-0.29	-0.04	0.98	0.99	5:18
Nadam	-0.38	-0.18	0.98	0.98	4:58
Adadelta	-0.02	-0.02	0.86	0.86	5:10
Adagrad	-0.03	-0.03	0.19	0.24	5:12

Next, the GAN model implemented in DPPL Edward was applied to the same dataset with a batch size of 128 and 1.0×10^4 epochs since it requires a large set

of training data to learn parameter estimation. It performed better compared to the GAN embedded U-Net model training where the epoch count was 20. We performed training for the same set of optimizers to see whether the ideal optimizer in Phase 1 was the same in Phase 2.

In this complete GAN model in Edward, two optimizers were required, one for the generator sub-network and one for the discriminator sub-network. The TensorBoard-monitored scalar loss graphs thus obtained for all the optimizers in the generator and the discriminator subnetwork are shown in Figures 5 and 6, respectively.

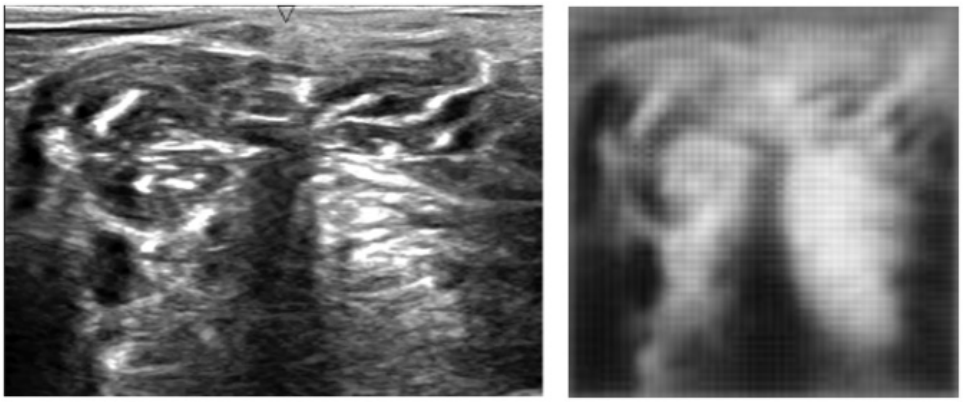

Figure 4 Nerve segmentation using GAN, U-Net and Adam.

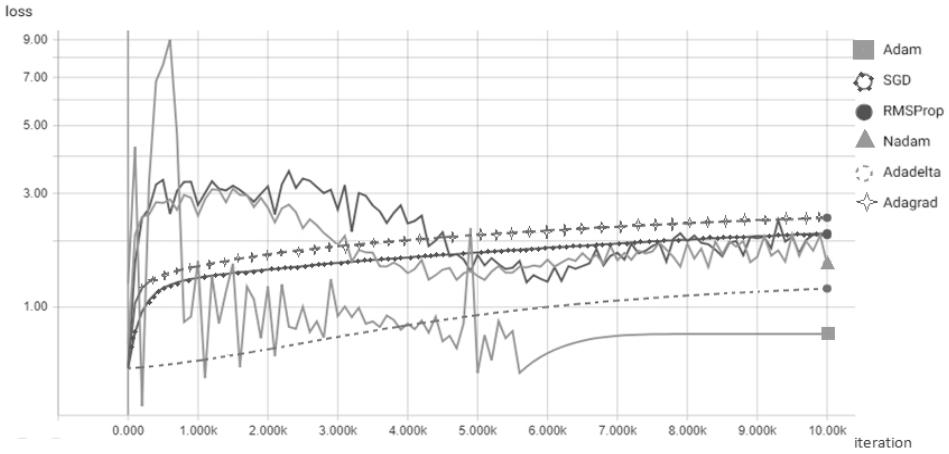

Figure 5 The generative loss in the DPPL-based GAN model.

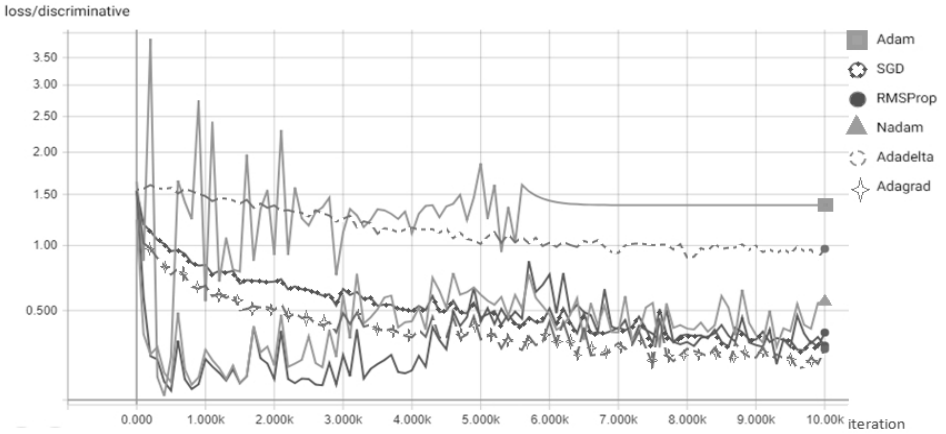

Figure 6 The discriminative loss for the DPPL-based GAN model.

The competitive behavior between the generator and the discriminator sub-networks in GAN can be clearly observed in the generative loss and the discriminative loss graphs. Hence, if the generative loss of an optimizer is low, it eventually means a higher discriminative loss. For instance, the ideal discriminative performance was shown by the Adam optimizer, which performed the best with Phase 1 U-Net too. However, Adam performed the worst in terms of generative loss. These results highlight the impact of the selected optimizer both in the DL model and the DPPL probabilistic model and suggest that the Adam optimizer is superior with or without DPPL. Table 4 summarizes the results. As shown in Figure 7, the GAN model with DPPL performed well towards nerve segmentation, with less computational cost and comparatively better performance compared to the non-DPPL U-Net model.

Table 4 Summary of GAN model results.

Optimizer	Generative network loss	Discriminative network loss	Time min:sec
Adam	0.693	1.386	25:25
SGD	2.142	0.359	25:17
RMSProp	2.110	0.281	28:13
Nadam	1.608	0.560	23:51
Adadelta	1.243	0.971	26:31
Adagrad	2.452	0.261	26:38

Thus, it is possible to improve the DL model with DPPL inferencing, criticism capabilities to handle uncertainty with the use of language expressiveness, scalability and speed in DPPLs. Adapting GAN with DPPL as a pre-segmentation task is one possibility, so that the segmentation can be performed

on more noise-removed quality images. Also, adapting it as a post-segmentation task is advantageous for the preparation of quality-segmented images. The Adam optimizer can be incorporated into either case as it showed comparatively the best results in both Phases 1 and 2.

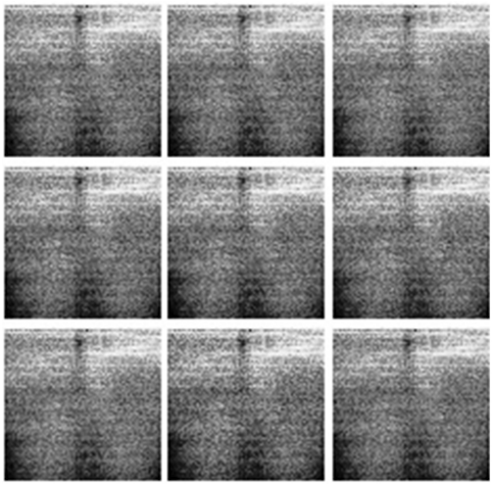

Figure 7 The output of the GAN probabilistic model for Adam.

5 Discussion and Future Work

DPPL benefits from its underlying DL framework, such as Edward from TensorFlow and Pyro from PyTorch [5,6]. DPPL is a challenging and novel research field due to the existing trade-offs between their characteristics. For instance, Edward lacks the ability to provide complex control flows with flexible inference, which is still an open challenge. Pyro has maintained a balance between these design aspects with higher language expressiveness than Edward. Table 5 gives a comparison with existing studies. Compared to the discussed related works [4,19], who also applied the Adam optimizer and achieved a dice coefficient of 85+% and 80+% accuracy respectively. According to the table all these related works obtained 80+% for the Dice score and accuracy without experimenting with DPPL aspects [4,18-22].

The DPPL-based approach presented in this study can be extended to other domains as well. For instance, DPPL can be used in the diagnosis of psychophysiological disorders using neuroimaging data [27,28]. Figure 8 illustrates a possible abstract design of such a generalized neuroscience decision support system model that can identify different diseases and disorders using neuroimaging data. The DPPL- and GAN-based approach together with the

Adam optimizer addressed in this work can be used to increase the performance of the segmentation process, as indicated by the highlighted area in Figure 8.

Generally, neuroimaging processing has a high computational cost. A deep probabilistic approach can be used to minimize this associated cost and increase the performance. Accordingly, the combination of deep learning and probabilistic programming is a research field with strong potential, encouraging future contributions to achieve advances in DL outcomes.

Table 5 Comparison with related works.

Related work	Variational inference	Test-time Augmentation	Adversarial learning inference	FCN	Residual learning	Variational Autoencoders (VAE)	GAN	DPPL	Results
A variational Bayes method for MRI image segmentation [18].	√								Dice 90+%
Uncertainty handling in CNN-based MRI segmentation [4].		√							Dice 85+%
Visualizing thoughts from brain EEG (ThoughtViz) [19].			√				√		Accuracy 80+%
Synthesizing medical images from a source image to a target without a practical scan [22].			√	√					Dice 80+%
Pelvic organs segmentation from MRI (STRAINet) [20].			√	√	√				Dice 86+%
Decode neuroimaging data in EEG (Brain2Image) [21].						√	√		Accuracy 80+%
Presented in this study.	√						√	√	Accuracy 99%

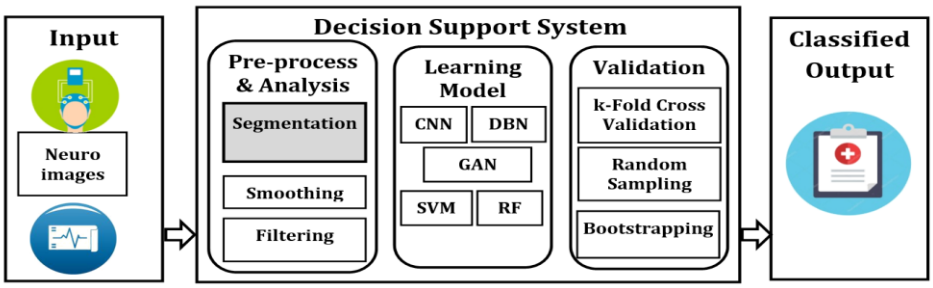

Figure 8 Proposed neuroscience decision support system.

6 Conclusion

The requirement of powerful Deep Probabilistic Programming Languages (DPPL) to cope with deep learning (DL) models has arisen due to the associated model uncertainties and excessive computational costs. This paper addressed the possibility of the adaptation of GAN using DPPL and the Adam optimizer to improve the image quality during the segmentation process. This approach was applied in the domain of biomedical images to segment ultrasound images of nerves. The obtained noise-removed quality images can be passed on to the learning model. Hence, they support classification accuracy. As a novel contribution, we used an application of Edward, which is a DPPL adapted to a GAN DL model in the context of neuroimaging, by showing positive results and research directions. For instance, U-Net model-based segmentation with the Adam optimizer showed the lowest Dice loss (-0.54) and the highest accuracy (0.99). With the GAN model, the Adam optimizer showed the lowest network loss (0.69). Thus, DPP used to generate quality segmentation with least dice loss, high accuracy and less time consumption.

Acknowledgement

The authors acknowledge the support received from the Conference & Publishing grant, University of Moratuwa, Sri Lanka for publishing this paper.

References

[1] Deng, L. & Yu, D., *Deep Learning: Methods and Applications*, Found. Trends® Signal Process., **7**, pp. 197-387, 2014.

[2] Ghahramani, Z., *Probabilistic Machine Learning and Artificial Intelligence*, Nature, **521**, pp. 452-459, 2015.

[3] Bottou L., *Large-Scale Machine Learning with Stochastic Gradient Descent*, in Proceedings of COMPSTAT, pp. 177-186, 2010.

[4] Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S. & Vercauteren, T., *Aleatoric Uncertainty Estimation with Test-time Augmentation for Medical Image Segmentation with Convolutional Neural Networks*, Neurocomputing, **338**, pp. 34-45, 2019.

[5] Baudart, G., Hirzel, M. & Mandel, L., *Deep Probabilistic Programming Languages: A Qualitative Study*, arXiv:**1804.06458**, 2018.

[6] Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P. & Goodman, N.D., Pyro: Deep Universal Probabilistic Programming, Journal of Machine Learning Research, **20**, pp. 1-6, 2019.

[7] Kaggle Inc., *Ultrasound Nerve Segmentation*, 2019. Available: https://www.kaggle.com/c/ultrasound-nerve-segmentation. (16-Jan-2019).

[8] Carpenter B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P. & Riddell, A., *Stan: A Probabilistic Programming Language*, J. Stat. Softw., **76**, pp. 1-32, 2017.

[9] Murray, L.M. & Schön, T.B., *Automated Learning with a Probabilistic Programming Language: Birch*, Annu. Rev. Control, **46**, pp. 29-43, 2018.

[10] Murray, L.M., Lundén, D., Kudlicka, J., Broman, D. & Schön T.B., *Delayed Sampling and Automatic Rao-Blackwellization of Probabilistic Programs*, Proc. 21st Int. Conf. Artif. Intell. Stat., 2017.

[11] Gaunt, A.L., Brockschmidt, M., Singh, R., Kushman, N., Kohli, P., Taylor, J. & Tarlow, D., *TerpreT: A Probabilistic Programming Language for Program Induction*, CoRR, arXiv:**1608.04428**, 2016.

[12] Burroni, J., Baudart, G., Mandel, L., Hirzel, M. & Shinnar, A., *Extending Stan for Deep Probabilistic Programming*, CoRR, arXiv:**1810.00873**, 2018.

[13] Rampasek, L. & Goldenberg, A., *TensorFlow: biology's gateway to deeplearning?*, Cell Syst, **2**, pp. 12–14, 2016.

[14] Ketkar, N., *Introduction to PyTorch*, Deep Learning with Python, Apress, pp. 195-208, 2017.

[15] Dai, Z., Meissner, E., & Lawrence, N.D., *MXFusion: A Modular Deep Probabilistic Programming Library*, NIPS Workshop MLOSS, 2018.

[16] Docs.microsoft.com, *The Microsoft Cognitive Toolkit*, available: https://docs.microsoft.com/en-us/cognitive-toolkit/. (03-Jul-2019)

[17] Mxnet.apache.org, *MXNet*, available: https://mxnet.apache.org/. (03-Jul-2019).

[18] Blaiotta, C., Cardoso, M.J. & Ashburner, J., *Variational Inference for Medical Image Segmentation*, Comput. Vis. Image Underst., **151**, pp. 14-28, 2016.

[19] Tirupattur, P., Rawat, Y.S., Spampinato, C., & Shah M., *ThoughtViz: Visualizing Human Thoughts Using Generative Adversarial Network*, in Multimedia Conference on Multimedia Conference, pp. 950–958, 2018.

[20] Nie, D., Wang, L., Gao, Y., Lian, J. & Shen, D., *STRAINet: Spatially Varying Stochastic Residual Adversarial Networks for MRI Pelvic Organ Segmentation*, IEEE Trans. Neural Networks Learn. Syst., pp. 1-13, 2018.

[21] Kavasidis, I., Palazzo, S., Spampinato, C., Giordano, D. & Shah, M., *Brain2Image: Converting Brain Signals into Images*, in Proceedings of the 2017 ACM on Multimedia Conference, pp. 1809-1817, 2017.

[22] Nie, D., Trullo, R., Lian, J., Wang, L., Petitjean, C., Ruan, S., Wang, Q. & Shen, D., *Medical Image Synthesis with Deep Convolutional Adversarial Networks*, IEEE Trans. Biomed. Eng., **65**, pp. 2720-2730, 2018.

[23] Ronneberger, O., Fischer, P. & Brox, T., *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Springer, Cham, pp. 234-241, 2015.

[24] Klette, R., *Concise Computer Vision*, London: Springer London, 2014.

[25] Chollet, F., *Keras*, 2015. Available: https://keras.io/. (03-Jan-2019).

[26] Marin, J.M., Pudlo, P., Robert, C.P. & Ryder, R.J., *Approximate Bayesian Computational Methods*, Stat. Comput., **22**(6), pp. 1167-1180, 2012.

[27] De Silva, S., Dayarathna, S., Ariyarathne, G., Meedeniya, D., & Jayarathna, S., *A Survey of Attention Deficit Hyperactivity Disorder Identification Using Psychophysiological Data*, International Journal of Online and Biomedical Engineering, **15**(13), pp. 61-76, 2019.

[28] Brihadiswaran, G., Haputhanthri, D., Gunathilaka, S., Meedeniya, D., & Jayarathna, S., *EEG-based Processing and classification methodologies for Autism Spectrum Disorder: A Review*, Journal of Computer Science, **15**(8), pp. 1161.1183, 2019.