TOWARDS A FRAMEWORK FOR THE DEMOCRATISATION OF DEEP SEMANTIC SEGMENTATION MODELS

Rubén Escobedo and Jónathan Heras
Department of Mathematics and Computer Science, University of La Rioja, Spain
e-mail: {ruescog, joheras}@unirioja.es

Abstract

Semantic segmentation models based on deep learning techniques have been successfully applied in several contexts. However, non-expert users might find challenging the use of those techniques due to several reasons, including the necessity of trying different algorithms implemented in heterogeneous libraries, the configuration of hyperparameters, the lack of support of many state-of-the-art algorithms for training them on custom datasets, or the variety of metrics employed to evaluate semantic segmentation models. In this work, we present the first steps towards the development of a framework that facilitates the construction and usage of deep segmentation models.

Keywords: Semantic Segmentation; Deep Learning; Democratisation.

1 INTRODUCTION

Semantic segmentation is a computer vision task that aims to classify every pixel of an image in a fixed set of classes. This task has received a lot of attention in recent years due to its multiple applications in contexts such as agriculture, manufacturing, robotics or medicine [6]. The interest in semantic segmentation is partially due to the development of deep learning architectures that provide accurate segmentation models [5]. In spite of its success, the adoption of deep learning techniques for semantic segmentation by users outside the machine learning community is a slow process due to several factors [3]. First of all, the constant flow of new deep learning architectures for semantic segmentation makes it difficult to keep track of the best methods. In addition, even if there is a trend of publishing the source code of the algorithms associated with research papers, most algorithms are not prepared to train models with custom datasets, and it might be challenging to use such a code in new projects. This issue has been faced with the development of frameworks, such as MMSegmentation [14] or FastAI [7], that provide several deep segmentation algorithms ready to be trained with the users' datasets.

The implementation of several algorithms in the same framework is an important feature since there is not a silver bullet solution to solve all the semantic segmentation problems [15]; and, hence, it is necessary to search for the most suitable algorithm and configuration of hyperparameters for each particular task. However, there is not a single framework that provides all the existing semantic segmentation algorithms, and each library uses its own format for encoding datasets, and has its own training protocol, performance measures, and preprocessing steps. Therefore, it is difficult to train and compare different methods implemented in different frameworks, and conclude which one is the best for a particular problem. In this work, we present an ongoing project to tackle those problems by developing a high-level API that facilitates the construction and usage of deep semantic segmentation models.

The rest of this paper is organised as follows. In the next section, we provide the related work of this project, and, subsequently, in Section 3, we present the challenges that a non-expert user faces when using existing semantic segmentation libraries. After that, in Section 4, we present the design of our framework to tackle those challenges, and in Section 5, we show an example of the usage of the preliminary version of our framework. Finally, the paper ends with some conclusions and further work. Our framework is open-source and can be found at `https://github.com/ruescog/SegmentationManager`.

2 RELATED WORK

This work can be framed in the field of Automated Machine Learning (AutoML) [8]. In the particular context of semantic segmentation, AutoML techniques are mainly focused on automatically designing deep architectures; this is known as Neural Architecture Search (NAS) [16]. Even if this approach has outperformed manually designed architectures, these techniques are data and computationally intensive; and, therefore, they are not suitable to be applied for custom datasets, or by users that do not have access to large amounts of GPUs. On the contrary, we propose a framework

 https://doi.org/10.17979/spudc.9788497498418.0980

Table 1: Features of existing semantic segmentation libraries

Library	Base library	♯ architectures	Last update
FastAI	Pytorch	2	2022
MIScnn	Tensorflow	1	2022
MMSegmentation	Pytorch	33	2022
PaddlePaddleSeg	PaddlePaddle	45	2022
SegmenTron	Pytorch	28	2020
Semtorch	Pytorch	5	2021
SM	Tensorflow	4	2020
SMP	Pytorch	9	2022

aligned with systems such as Auto-Sklearn [4] or Auto-Weka [9] that are tools that automatically search through a space of machine learning algorithms and hyperparameters defined by the user. Namely, we aim to design a framework that helps non-expert users to automatically search through the space of semantic segmentation algorithms (available in several frameworks and libraries) to maximise their performance for a given problem.

Currently, we can find several libraries that provide deep semantic segmentation algorithms. For our work, we have focused on semantic segmentation libraries that provide several algorithms, can be trained on custom datasets, and are implemented in Python (the dominant programming language language for deep learning models), see Table 1. As can be seen in Table 1, this field is in constant evolution (all the tools have released new version in the last two years) and there is not a single underlying deep learning library — several libraries like Tensorflow, Pytorch and PaddlePaddle are employed. This variety of libraries hinders the comparison of models across tools; in addition, there are other challenges related to the construction and usage of semantic segmentation models.

3 CURRENT CHALLENGES

In the pipeline to create a semantic segmentation model, we can distinguish four stages that are common to any semantic segmentation library: data and model preparation, training, evaluation, and usage.

For the data and model preparation stage, one of the most important aspects is the annotation format (usually mask images or COCO files) taken as input by the frameworks — this is relevant since there is not a standard annotation format, and annotation tools, like LabelMe[1] or Computer Vision Annotation tool[2] produce different kinds of files. Another aspect in the data preparation step is related to how the data is split into training and testing sets. Most frameworks require a specific folder structure, and in some cases it is also necessary to explicitly provide files with the lists of training and testing images. Finally, in this first stage, it remains the question of how to configure the model (that is, fixing hyperparameters such as the network architecture, the batch size, the learning rate and so on). Such a configuration is usually provided by means of either a configuration file or by modifying the source code that launches the training process — usually, the latter requires some programming experience. Such a heterogeneity of annotation formats, file structures, and configuration files hinders the use of several frameworks and libraries.

For the training stage, the analysed frameworks of Table 1 provide similar features. Deep segmentation algorithms require large datasets to be trained from scratch (for instance, the COCO dataset includes 80 K images [11]); however, most custom datasets are small. A successful approach to deal with this problem is fine-tuning [1], a transfer learning technique that re-uses a model trained in a source task, where a lot of data is available, in a new target task, with usually scarce data. This functionality is supported by all the analysed frameworks, that, in addition, provide a model zoo to apply fine-tuning from different source models.

The last two stages of the pipeline, evaluation and usage, are closely related and pose similar challenges. Semantic segmentation algorithms are usually evaluated using metrics such as Dice coefficient or Jaccard index [6]. However, each framework has its own implementation of those metrics. Therefore, it might not be sound to directly compare the results used with different systems. Similarly, the usage of the models greatly varies from tool to tool since each system loads models, performs predictions, and outputs the results in a particular way. This makes difficult to compare models generated with different systems. An approach to solve these problems is based on the definition of converters among libraries. Some of these solutions are collected in a project called Deep learning model converter [18]. Another similar project is the Open Neural Network Exchange (ONNX) [13] that aims to build models with the ONNX representation and then use the framework that better adapts to the users' goals to execute it. The main problem with the approach based on converters is that some frameworks do not implement all types of layers of deep learning models; so, certain models cannot be converted to all the frameworks.

Finally, in all the stages of the construction of deep segmentation models might appear errors, that in some cases are difficult to decipher for both expert

[1] `http://labelme.csail.mit.edu/`
[2] https://github.com/openvinotoolkit/cvat

and non-expert users.

In our work, we aim to deal with all the challenges explained throughout this section by defining a high-level system that allows the integration of deep learning frameworks and libraries.

4 FRAMEWORK DESIGN

We have designed, and implemented, an open-source library in Python that simplifies and automatises the process of training multiple semantic segmentation models using different libraries (currently, it supports the tools of Table 1 that have Pytorch as underlying library), and select the best of them. To this aim, we have design a workflow that captures all the necessary steps to train several semantic segmentation models and select the best one. Currently, the workflow can be mainly used by means of a Jupyter notebook [10], and has been implemented using the facade design pattern so it can be easily extended in the future.

The workflow of our library can be summarised as follows. First of all, the user selects the image dataset to be studied. Such a dataset must contain two folders one with the images and one with the masks associated to such images. When loading the dataset, our library automatically performs some validity checks to avoid further errors (for instance, it checks that each image of the dataset has an associated mask, and that the mask has the correct format).

Subsequently, the user indicates how the dataset will be split for evaluation. Currently, our library provides two modes: train-val-split, and k-fold. In the former, the dataset is split into three groups: training (for optimising the weights of the models), validation (for choosing the best set of hyperparameters), and test (for finally, evaluating the model). In the latter mode, a cross validation procedure is employed for evaluating the different models.

After that, the user can apply several data augmentation methods [17]. Data augmentation is a regularisation technique that generates new training samples from the original dataset by applying colour or geometric transformations. This functionality is supported by the Albumentations library [2].

Finally, the user selects the architectures to train and their parameters. By default all the networks will be trained by using fine-tuning, but they can also be trained from scratch. After this step, the models are trained and evaluated automatically without any user intervention. As a result, the library produces a report where the different trained models are evaluated and compared, and also saves the best model for further usage.

5 A RUNNING EXAMPLE

As a running example, we have trained several segmentation models by using the vineyard dataset presented in [12]. In particular, we have trained a Unet model with the ResNet18 backbone, and a Unet model with the Resnet34 backbone, and a DeepLabv3+ model with a MobileNet backbone. The first model is trained thanks to the SemTorch library, the second using the SMP library, and the third model is trained thanks to the functionality of SegmenTron.

All the code that is required to train and compare different models with our library is provided in Figure 1. As can be seen from such an image, the user has to define a segmentation manager that consists of three objects: a dataset manager (in charge of loading the dataset), a validation manager (in charge of defining how the dataset is split for training and evaluation) and a transformation manager (that provides the data augmentation techniques that will be applied to the dataset). After that, the user can invoke the method `multiple_train` of the segmentation manager to train multiple models (the user can either employ the by-default hyperparameters of those models or fix them manually). After the training process, the user can visualise a summary of the models when evaluated on the test set, see Figure 2; and also generate a boxplot with those results, see Figure 3.

6 CONCLUSIONS AND FURTHER WORK

In this paper, we have presented an on-going work to facilitate the use and construction of semantic segmentation models using deep learning techniques. This is achieved by providing a high-level API that provides access to several semantic segmentation libraries.

As further work, there are several remaining tasks to tackle. First of all, we aim to include in our framework other libraries, and provide a mechanism to easily extend it in the future. Moreover, it would be interesting to incorporate techniques like Bayesian optimisation methods to select the best hyperparameter configuration for each semantic segmentation algorithm. Finally, another important challenge that will be faced in the future is how to reduce the time required to train semantic segmentation models. This issue can be alleviated by the usage of multiple GPUs or distributed

https://doi.org/10.17979/spudc.9788497498418.0980

```
[ ] dataset_manager = SegmentationManager.build_default_dataset("dataset", img_prefix = "color_", mask_prefix = "gt_")
    transform_manager = SegmentationManager.build_default_transformation(["default"])
    validation_manager = SegmentationManager.build_default_validation(mode = "kfold")
    sm = SegmentationManager(dataset_manager, transform_manager, validation_manager)
```

```
[ ] sm.multiple_train([
      ("model1", ARCHITECTURE.UNET, BACKBONE.RESNET18, WEIGHTS.NONE, 1e-3, "semtorch"),
      ("model2", ARCHITECTURE.UNET, BACKBONE.RESNET34),
      ("model3", ARCHITECTURE.DEEPLABV3_PLUS, BACKBONE.MOBILENET_V2)
      ], batch_size = 4, mode = "fine_tune", n_epochs = 5, n_freeze_epochs = 1)
```

```
[ ] sm.summary()
```

```
[ ] sm.plot_train_valid()
```

Figure 1: Code to train several models using our library

	model_name	fold	valid_loss	dice_multi
0	model1	test	0.540679	0.352730
1	model2	test	0.520797	0.417553
2	model3	test	0.412336	0.478929

Figure 2: Metrics obtained by the trained models

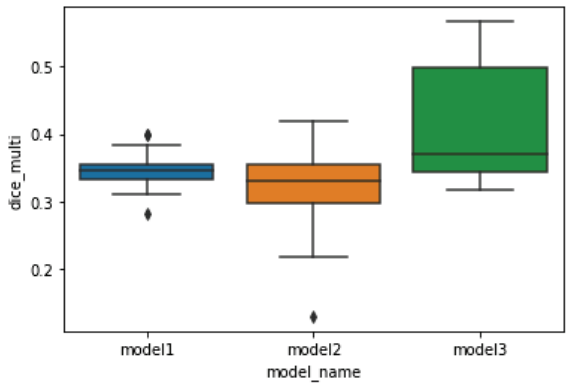

Figure 3: Boxplot of the results obtained by the trained models

training in a cluster of computers, but this feature should be provided to users in a transparent manner.

Acknowledgement

This work was partially supported by Ministerio de Ciencia e Innovación [PID2020-115225RB-I00 / AEI / 10.13039/501100011033].

References

[1] Azizpour, H., Sullivan, J., & Carlsson, S. (2014). "Cnn features off-the-shelf: An astounding baseline for recognition". In *CVPRW* (pp. 512-519).

[2] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). "Albumentations: fast and flexible image augmentations". *Information*, 11(2), 125.

[3] Dacrema, M. F., Cremonesi, P., & Jannach, D. (2019). "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In *Proceedings of the 13th ACM conference on recommender systems* (pp. 101-109).

[4] Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2020). "Auto-sklearn 2.0: The next generation". arXiv preprint arXiv:2007.04074, 24.

[5] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., & Garcia-Rodriguez, J. (2018). "A survey on deep learning techniques for image and video semantic segmentation". *Applied Soft Computing*, 70, 41-65.

[6] Hao, S., Zhou, Y., & Guo, Y. (2020). "A brief survey on semantic segmentation with deep learning". *Neurocomputing*, 406, 302-321.

[7] Howard, J., & Gugger, S. (2020). "Fastai: a layered API for deep learning". *Information*, 11(2), 108.

[8] Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). "Automated machine learning: methods, systems, challenges" (p. 219). Springer Nature.

[9] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2019). "Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA". In Automated machine learning (pp. 81-95). Springer, Cham.

[10] Kluyver, T., Ragan-Kelley, B., PÃ©rez, F., Granger, B. E., Bussonnier, M., Frederic, J., ... & Willing, C. (2016). "Jupyter Notebooks — a publishing format for reproducible computational workflows" (Vol. 2016, pp. 87-90).

[11] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). "Microsoft coco: Common objects in context". In *European conference on computer vision* (pp. 740-755). Springer, Cham.

[12] Marani, R., Milella, A., Petitti, A., & Reina, G. (2021). "Deep neural networks for grape bunch segmentation in natural images from a consumer-grade camera". *Precision Agriculture*, 22(2), 387-413.

[13] Microsoft, Facebook open source & AWS (2018). "ONNX: Open Neural Network Exchange".

[14] MMSegmentation Contributors (2020). "MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark". `https://github.com/open-mmlab/mmsegmentation`.

[15] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). "Foundations of machine learning". MIT press.

[16] Nekrasov, V., Chen, H., Shen, C., & Reid, I. (2019). "Fast neural architecture search of compact semantic segmentation models via auxiliary cells". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9126-9135).

[17] Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). "Best practices for convolutional neural networks applied to visual document analysis". In *Icdar* (Vol. 3, No. 2003).

[18] Yuan, S. (2018) "Deep learning model convertors".

 https://doi.org/10.17979/spudc.9788497498418.0980