ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

SEMESTER PROJECT

MASTER IN COMPUTATIONAL SCIENCE ENGINEERING

Deep Learning for the measurement of the Dynamic Aperture at CERN

Author:
Arthur TABARY

Supervisors (from LPAP):
Davide DI CROCE
Tatiana PIELONI
Mike SEIDEL

Abstract

The dynamic aperture (DA) is an important concept in the study of nonlinear beam dynamics in Accelerator Physics. The DA is the extent of the simply-connected region of phase space in which the motion of a particle remains bounded over a finite number of turns. Such a volume is shaped by the nonlinear imperfections in the magnetic fields, beam-beam effects, electron lens, electron clouds and others non-linear effects. In this study, we want to investigate whether Deep Learning methods could estimate the DA volume more efficiently than the techniques currently used at CERN. For this purpose, 3 Graphs Neurals Networks using GCN or GraphSAGE layers are implemented to estimate the volume of DAs in 2D ($p_x = p_y = 0$). The results show that for an almost equivalent accuracy (99.1 % versus 98.1 %), the SAGE model estimates the DA volumes 3 to 4 times faster than classical numerical integration methods such as *contourArea()* from the OpenCV library.

I. Introduction

The dynamic aperture (DA) is an important concept in the study of nonlinear beam dynamics in Accelerator Physics. The DA is the extent of the simply-connected region of phase space in which the motion of a particle remains bounded over a finite number of turns. Such a volume is shaped by the nonlinear imperfections in the magnetic fields, beam-beam effects, electron lens, electron clouds and others non-linear effects. Therefore, the study of the DA allows the understanding the time evolution of beam losses in a circular particle accelerator under the influence of non-linear effects, which is important for the operation of accelerators as the LHC at CERN and also for the design of the future ones.

So far at CERN, the 2D volume of the DA is approximated by calculating the area of the mean circle; that is, by calculating the area of the circle formed by the mean radius of the 11 bins. Although this technique is fast, it is not very accurate due to the non-linear effects shaping the DA as an irregular ellipse with holes and islands rather than a regular circle.

Some have tried to overcome this problem by using numerical integration techniques resulting in an error of the order of 2% on a space-phase volume estimate. However, the computational CPU time is far too large for this to be applicable at CERN [1].

The latest progress in deep learning could offer a promising solution to accurately and efficiently estimate the 4D volume of the DA in phase space. Therefore, the idea of this project is to establish a proof of concept by developing a neural network able to accurately measuring the volume of the DA in 2D. If the results are convincing, this model will be upgraded to work in space phase domain.

II. Neural Networks Models

What is challenging in the estimation of the DA volume is to be able to distinguish the simply-connected region of phase space from unstable disconnected particles (Fig. 1). In the x, y space, it is then possible to use Machine Learning (ML) techniques like Support Vector Machine (SVM) to first exclude the unconnected particles [2] to then apply algorithms from the OpenCV library [3] and obtain an accuracy of about 99.1% on the 2D volume estimation. However, these algorithms are time-consuming and are not scalable in 4 dimensions. That's why this project explores different models of Graph Neural Networks (GNNs) which are successful in the last years.

Graphs are an efficient way to represent a complex system of nodes (particles in the case of DA) and their relations with other nodes (here the position of the nearest neighbors). Such a representation emphasizes the distribution of particles with respect to other particles which is an appropriate way to train a neural network able to distinguish simply-connected regions of DA space phase.

In this project, two popular GNNs architectures are explored: Graph Convolutional Networks (GCN) and GraphSAGE. While GraphSAGE uses "sampling and aggregating" process to generate node embeddings, which allows it to generalize to new nodes that were not present in the training set [4], GCN uses the entire graph to make predictions and therefore is able to model more complex relationships in the data and achieve good generalization as a result [5].

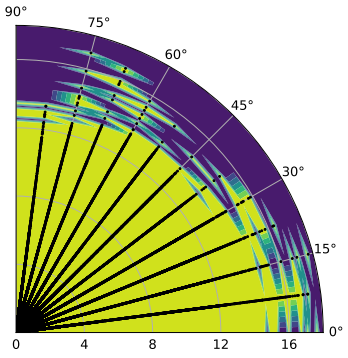

Fig. 1: Dynamic Aperture of particles having survived more than 10000 turns in the LHC. The contour draws in yellow shows the irregularity of the shape and the different connected regions of the phase space projected in 2D ($p_x = p_y = 0$).

Therefore, 3 different models involving these two architectures are implemented in this project:

- A model with 3 GraphSAGE hidden layers and 64 channels per layer, named **SAGE**.
- An other model with 3 GCN hidden layers and 64 channels per layers, named **GCN**.
- A last model with 1 GCN hidden layer and 64 channels per layer, named **SIMPLE**.

The accuracy and speed of the models are compared to each other and to the prediction make by OpenCV numerical methods [3] in the Section IV.

III. Creation of the Dataset

To train such Deep Learning models, especially complex models like SAGE, it is necessary to use hundreds of thousands or even millions of samples of labeled dynamic apertures. Since there is no such dataset available, one must be generated.

This is a complex task because the DAs observed in the LHC have complex, irregular and very varied shapes whose area cannot be calculated analytically. Moreover, this step highly influence the results of the model because the closer the samples are from the real data, the better and more robust the predictions will be. Deep learning models are highly sensitive to training set.

The parameters below, chosen to generate the training and test set samples, are the result of a compromise between generating samples as varied and complex as CERN DAs while keeping a manageable size dataset (< 200 Go).

- All samples are in polar coordinates and represent only the positive x, y quarter of the total shape.
- Although this is not the case in the LHC, the particle distribution is uniform.
- The maximum radius of the generated shape is between 4 and 15.
- each sample is made of 10 000 to 100 000 points.

Furthermore, to produce complex shapes whose 2D volume can be analytically computed, samples are produced by building a main ellipse to which, one or two circles are added or truncated. This results in complex shapes, with connected particles and islands, whose volume is computed by summing or subtracting the area of the intersections between the different circles and ellipses. To make it clearer, 4 samples from the generated dataset are shown in Fig. 2.

Finally, to speed up the training of the neural network, the coordinates and labels are normalized between 0 and 1.

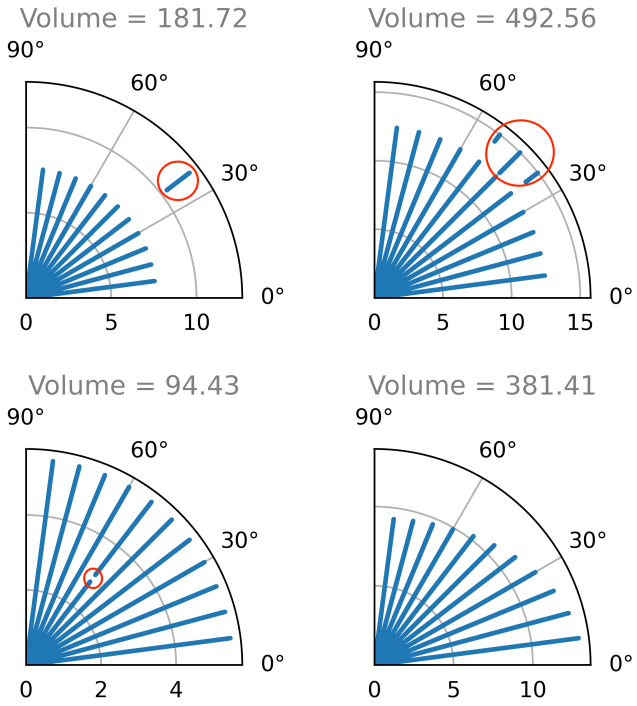

Fig. 2: Samples from the generated dataset. The top shapes are formed by adding a circle, while the bottom shape is formed by truncating a circle into the main circle. The bottom right sample is a regular ellipse.

IV. Results

The dataset 275 000 samples (250 000 for the training set and 25 000 for the test set) was generated with the EPFL supercomputer SCITAS in 30 core-hours. The loading time of a single sample is about 6 ms.

The SIMPLE and GCN models were trained over 3 days, or 72 gpu-hours, while SAGE was trained over 6 days, or 144 GPU-hours. This last model having a more complex architecture requires a longer training.

All these GNNs were trained with the Adam optimizer [6], with a scheduled learning rate and a batch size of 128 samples. The latter could not be larger due to the limitation of 144GB of RAM on SCITAS GPU clusters. The loss function used is Huber [7] and its evolution over epochs is shown Fig. 3.

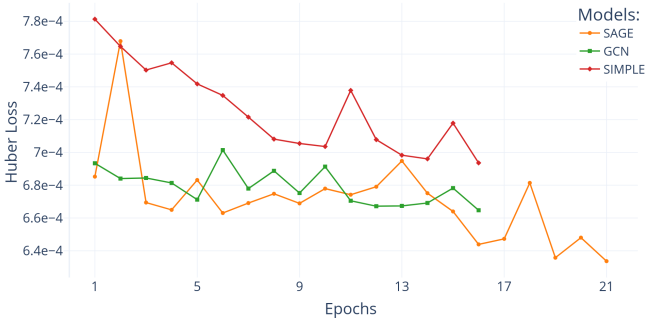

Fig. 3: Validation loss for the 3 models using the loss function Huber. SAGE has been trained 6 days while SIMPLE and GCN has been trained 3 days.

	OpenCV	SAGE	GCN	SIMPLE
Testset Accuracy [%]	99.1 ± 2.2	98.1 ± 3.4	98.1 ± 3.4	98.0 ± 3.5
Real DAs Accuracy [%]	/	91.1 ± 6.8	87 ± 11	84 ± 11
Runtime to estimate 10'000 DAs	90 min	20 min	20 min	20 min

TABLE I: Summary table of the performance of the models presented in this report. As the real volume of these DAs is not known, the accuracy of the models is computed from the results given by the *contourArea* method of OpenCV.

Despite differences in the loss functions and a longer training for SAGE, the average accuracy on the testset is similar between the different GNN models (about 98% accuracy). Tab. I, Fig. 4. With an accuracy of $99.1 \pm 2.2\%$, the numerical integration algorithm *contourArea()* is the one with the best estimates. Tab. I. However, the accuracy boost of OpenCV's numerical integrator comes with a significant speed cost with a run time 3 to 4 times longer than Deep Learning algorithms. Tab. I.

These models are then evaluated on 100 real DAs formed by particles which have survived 10 000 turns in the LHC. Since the 2D volume is not known, the accuracy of the GNN models is evaluated with respect to the estimate given by the OpenCV algorithm.

Contrary to the previous results, the performances of the neural networks are different with a domination of the SAGE model with $91.1 \pm 6.8\%$ of accuracy, followed by GCN ($87 \pm 11\%$) then SIMPLE ($84 \pm 11\%$). Tab. I, Fig. 5.

Finally, to understand where the weaknesses of each model come from, the average accuracy according to the sample categories of the generated dataset is shown in Fig.6. Although the differences are not large, the samples of altered circles are surprisingly the best estimated.

V. Discussion

The performance of the GNN models on the test set is really good with an accuracy of 98%. Although with different architecture, the three models are performing equally well on generated DAs. The difference grows when these models are evaluated on real DAs. (Tab. I, Fig. 4, 5). Moreover, the real DAs of CERN being varied and complex have sometimes very different structures from the training data making the predictions for GCN models more complicated especially if it has a single layer like SIMPLE. However, GraphSAGE architecture handles unseen nodes better than GCN, because it uses a "sampling and aggregating" process to generate node embeddings, which allows it to generalize to new nodes that were not present in the training set. This can make GraphSAGE more suitable for tasks where the graph structure is changing over the DAs. [4], [5]

It is however difficult to draw definitive conclusions from this higher performance of GraphSAGE on the real DAs because the estimates of the GNNs are compared to the prediction of OpenCV's *contourArea()* to compute their accuracy. The better performance of GraphSAGE for CERN DAs could also be due to the fact that the predictions of OpenCV and SAGE are close and make the same errors.

Overall, the standard deviations on the model GNN estimates are high emphasizing that the models are not sufficiently robust. This behavior is even more visible when estimating real DAs. This reveals the fact that the training dataset is probably not varied and complex enough to correctly reflect the DAs produced at CERN. To overcome this, it would be necessary to build a more exhaustive dataset and therefore have storage and RAM resources that are all the more important for managing this new dataset.

Other changes requiring higher computational resources could improve the performance of the GNN modelss. For example, the batch size could be increased to 256 samples in clusters having more than 144GB of RAM to reduce the outliers sensitivity if the loss function during training. Also, increasing the number of edges (Nearest Neighbors considered per particle) considered when constructing the graphs could improve the predictions by the fact that connected and disconnected regions of the DAs would be more easily detected. Again, increasing the number of egdes per node, would require more RAM memory.

Finally, other cutting-egde Deep Learning algorithms such as Vision Transformer are to be explored as they seem promising for such problems.

VI. Conclusion

To summarize, despite a slightly lower accuracy than classical numerical integration algorithms such as *contourArea()*, the runtime speed of the Graph Neural Networks algorithms make them more valuable tools in the CERN context.
Among the models implemented in this project, SAGE is the most promising because of its robustness and its great generalization to real data. To extend DAs volume estimates to the 4-dimensional phase space, it is currently the most relevant choice.

References

[1] M. Giovannozzi and E. Todesco. Numerical methods to estimate the dynamic aperture. 01 1996.

[2] Frederik Van der Veken, M. Giovannozzi, Ewen Maclean, Carlo Montanari, and Gianluca Valentino. Using machine learning to improve dynamic aperture estimates. 09 2021.

[3] Documentation opencv contourarea. https://go.epfl.ch/OpenCV_ContourArea.

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. 2017.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2016.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[7] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

Appendix

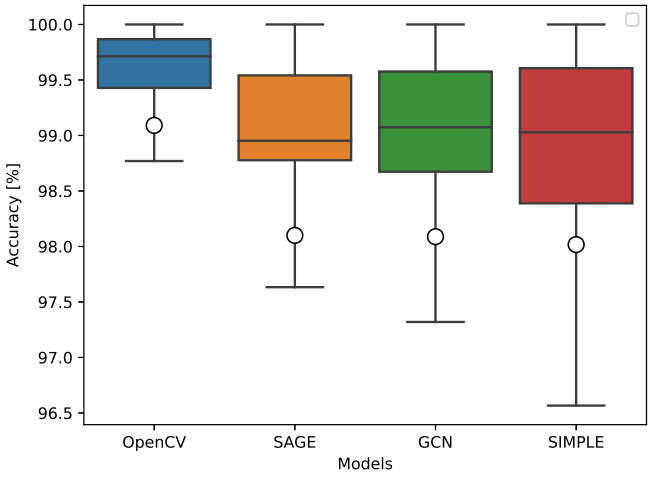

Fig. 4: Evaluation of the accuracy of the GNN models on the test set including 25 000 samples, i.e. 10% of the training set size. Their accuracy is compared to the *contourArea()* method from OpenCV. The white circles show the average accuracy.

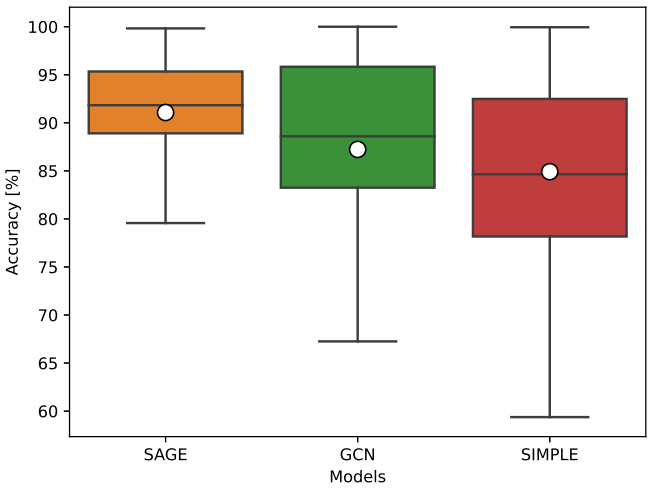

Fig. 5: Evaluation of the accuracy of GNN models on 100 CERN DAs produced in the LHC for different configurations. As the real volume of these DAs is not known, the accuracy of the models is computed from the results given by the *contourArea()* method from OpenCV. The white circles show the average accuracy.

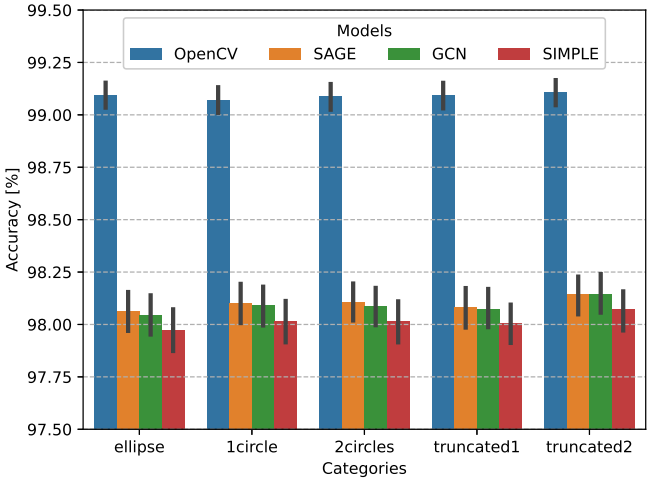

Fig. 6: Evaluate the accuracy of the different models and the OpenCV algorithm on the test set by categorizing the samples by the way they were generated. See Section III for more details about these categories.