GANEx: A complete pipeline of training, inference and benchmarking GAN experiments

Vajira Thambawita
SimulaMet
Norway
vajira@simula.no

Hugo Lewi Hammer
Oslo Metropolitan University
Norway
hugoh@oslomet.no

Michael Riegler
SimulaMet
Norway
michael@simula.no

Pål Halvorsen
SimulaMet, Norway
Oslo Metropolitan University, Norway
paalh@simula.no

***Abstract*—Deep learning (DL) is one of the standard methods in the field of multimedia research to perform data classification, detection, segmentation and generation. Within DL, generative adversarial networks (GANs) represents a new and highly popular branch of methods. GANs have the capability to generate, from random noise or conditional input, new data realizations within the dataset population. While generation is popular and highly useful in itself, GANs can also be useful to improve supervised DL. GAN-based approaches can, for example, perform segmentation or create synthetic data for training other DL models. The latter one is especially interesting in domains where not much training data exists such as medical multimedia. In this respect, performing a series of experiments involving GANs can be very time consuming due to the lack of tools that support the whole pipeline such as structured training, testing and tracking of different architectures and configurations. Moreover, the success of generative models is highly dependent on hyper-parameter optimization and statistical analysis in the design and fine-tuning stages.**

In this paper, we present a new tool called GANEx for making the whole pipeline of training, inference and benchmarking GANs faster, more efficient and more structured. The tool consists of a special library called FastGAN which allows designing generative models very fast. Moreover, GANEx has a graphical user interface to support structured experimenting, quick hyper-parameter configurations and output analysis. The presented tool is not limited to a specific DL framework and can be therefore even used to compare the performance of cross frameworks.

***Index Terms*—GANs, Neural Networks, Graphical User Interface, GAN Experiments, GAN Library, GAN Statistics**

I. Introduction

Generative models have become an active research area in recent years as a result of the introduction of generative adversarial networks (GANs) [1], [2]. Research such as deep convolution GAN [3], conditional GAN [4], coupled GAN [5], cycle GAN [6] and many more generative models [7] based on the original GAN idea have been published in recent years. These generative models are actively used in multimedia research because of the capabilities for generating images, sounds, texts and videos from noise or conditional inputs. Most of the GAN architectures follow the same set of logical training procedures, generative and adversarial network architectures and closely related optimization procedures. Researchers are wasting valuable time to implement the same logical flow of GANs over and over again implementing already available GAN architectures from scratch. In addition, they are facing problems in organizing deep learning (DL) experiments and experiment data.

In this context, our GANEx tool is a solution to perform GAN based research more effectively and efficiently. This tool is a complete pipeline for training, inference and analysis of generative models for saving the time of researchers and saving valuable data of experiments. The GANEx tool is enriched with real-time training analysing tools to support researchers to get early-stage decisions such as stopping the unstable training processes, detecting the unstable hyperparameters and other decisions which are more important to take early before starting the long training process of DL. Moreover, this tool is capable to handle GAN experiments in structured way and perform advanced analysis in the inference stage of GAN experiments.

As depicted in Figure 1, our main GANEx tool consists of three components; 1) a graphical user interface (GUI), 2) a library called FastGAN and 3) a DL library (Pytorch). In this paper, we discuss only the first two sections (our contributions) because the last section is implemented using the well known DL library Pytorch where details are found in [8]. Based on this, the main contributions of the presented tools are:

- The FastGAN library, which is introduced to develop, fine-tune, perform experiments and analyse generative models or GANs efficiently and effectively.
- The GANEx GUI, which is introduced to perform GAN experiments in a structured way and benchmarking them quickly for saving valuable time and experiment results such as parameters of GAN models, output data and other analysed statistical data.

In the next section (section 2), we discuss the available GUI based tools for running DL experiments. Section 3 covers the

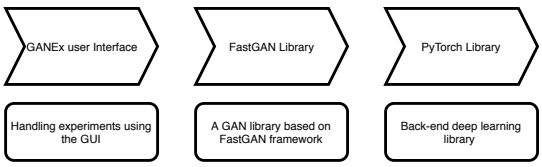


Fig. 1: Overview of the GANEx execution flow

978-1-7281-4673-7/19/$31.00 ©2019 IEEE

concept and architecture of the FastGAN library before we discuss the GANEx GUI in section 4. We present some ideas of how GANEx can be expanded in section 5 and finally, in section 6, we give a description of the proposed demo.

II. Related work

NVIDIA DIGITS [9] is one of the popular tools among researchers for running and analysing DL experiments in computer vision problems such as classification, detection and segmentation. Recently they have experimented how to run basic GAN experiments using this tool and they have added basic GAN training capabilities. However, NVIDIA DIGITS capabilities of the current version are not enough for analysing and performing advanced GAN experiments because it has general purpose DL capabilities and it does not have mechanisms to mange experiments and doing advanced statistical analysis. Moreover, this kind of web-based solutions are slower than stand-alone applications because it has limitations to work directly with OS functionalities. Therefore, our solution is designed using stand-alone application development concepts to keep good performance and reliability.

GAN lab [10] is another web-based tool for understanding the main GAN architecture. Users can play with simple data generation problems using this tool. It visualizes how the GAN changes the input noise to the target distribution. They clearly emphasize gradient changes of model parameters using visualization to give a deeper understanding of the GAN architecture. This web-page was designed using the Tensorflow Javascript [11] library, and they have not targeted any researchers who are doing new advanced generative model experiments.

Weka [12] is a popular tool among machine learning researchers from beginners to advanced users. However, handling DL experiments using the Weka tool is limited. Because it mainly targets statistical machine learning algorithms, it has restrictions for doing advanced DL experiments using the popular DL frameworks such as Pytorch and TensorFlow. But, the concepts for managing experiments underpinning the Weka tool are helpful for developers of GUI-based experiment tools. They use standalone application development methods with Java for maintaining reliability of the tool with OS interactions.

The TensorBoard [13] visualization tool comes with the Tensorflow [14] DL library, and it is rich with more visualization tools for model parameters and training process. This tool is also powered by web technologies. Moreover, this tool can be used to visualize the main components of any DL model. However, GAN-specific statistical analysis is more difficult with this tool as a result of a generalized tool with Tesorflow. Structured experiment handling and data handling capabilities have not been designed in this solution.

Deep learning studio [15] is another tool which is closer to the concept of NVIDIA DIGITS. This tool has more capabilities in designing deep neural networks compared to the designing capabilities of NVIDIA DIGITS tool. In contrast to these benefits, Deep learning studio suffers from a lack of capabilities for visualization of model learning patterns and reasoning of inference. However, this tool can be identified as a tool including all the steps like designing a model, training a model and inference from a pre-trained model. For example, an autoencoder [16] which is one of the generative models is demonstrated using this tool. The autoencoder model can be designed easily using this tool because the input of this type of model is an image while the output is also an image. In contrast to autoencoder experiments, GAN experiments' input data depend on latent spaces, images as well as labels while they generate several output data such as images and labels. Some generative models generate indirect inference parameters like standard deviation and mean of probability distributions. Therefore, statistical data analysing and parameter handling are more important for GAN experiments.

Deep learning studio and all other tools discussed above are lacking of GAN specific modifications, statistical analysis tools and developments procedures. Therefore, we address these issues in our proposed solution, the GANEx tool which is a complete GAN specific pipeline for training, inference and doing advanced statistical analysis.

III. The FastGAN library

The FastGAN implementation can be categorized as a core library because it opens paths for developers to define there own GAN experimental tools using other DL frameworks. The FastGAN library implementation is the core of the GANEx GUI tool. This library consists of high-end abstract logical flow which can be used to implement GAN based experiments very easily and quickly in a few steps. The main structure of the library is depicted in Figure 2.

The FastGAN library can be defined on top of available DL libraries such as Tensorflow, Pytorch, Microsoft Cognitive Toolkit or any other DL libraries available today. However, our first FastGAN library implementation is accomplished using the Pytorch library. This back-end dependency is depicted at the bottom of Figure 2. Visualization and 2D/3D plotting tools have to be provided alongside the main DL library for advanced statistical analysis of GAN experiments. The first FastGAN library uses "pyqtgraphs" [17] for this purpose because of the enrichment of plotting tools which are based on statistical and engineering applications.

The second level of this library consists of three main sections; FastGAN Nets, FastGAN Trainer and FastGAN Analyser. The FastGAN Nets should be packed with state of the art generative networks, discriminative networks, decoder and encoder networks which are used in generative model implementations. The FastGAN Trainer defines the logical training flows of generative models. It should have all possible training mechanisms of generative model training and adversarial generative model training. These mechanisms can be implemented as methods which are applicable for all generative models. For example, a training discriminator with real labels, a training discriminator with fake labels and a training generator with fake labels as real labels, can be identified as sub-components of the main training of basic

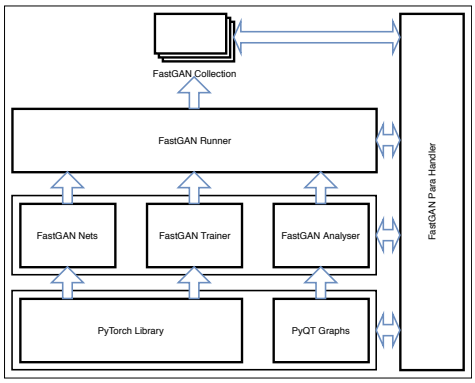


Fig. 2: The FastGAN library structure

GAN architectures. The FastGAN Analyzer is the next main module at the same level. This analyzer consists of various analysis tools for generative models. These analysis tools can include metrics for generative model comparison, plots of training and inference stages, parameter representations, input-output analysis, probability distribution representations and many more engineering and statistical analysis algorithms that are commonly used for generative models. The FastGAN analyzer can be implemented as an independent module which can be used for statistical analysis of any DL architecture.

The third level of the FastGAN library is named FastGAN Runner which defines how to execute training, re-training and inference of generative models based on the components of the bottom levels. This level includes data pre-processing mechanisms, parameters initialization methods for network parameters and noises, initializing analysis mechanisms of a specific GAN and interconnecting routings of all the components of the bottom levels.

The top level of the FastGAN library can be implemented by collecting implementations of all the state of the art generative models using the components of the bottom levels. Then, we can allow researchers to define their own parameters and input data without considering developments of generative models. If researchers are interested in more advanced modifications, then they can go through the levels of FastGAN from top to bottom as they required. This allows researchers to do GAN experiments from simple modifications to more advanced modifications.

The last component, the FastGAN Parahandler or parameter flow, is defined using the JSON library and dictionary data structures of Python in our test implementation. However, researchers or developers can use any mechanism such as a relational database to handle parameters and store parameters via all the levels.

IV. The GANEx GUI tool

The GANEx GUI implementation is developed on top of the FastGAN library, and it enables a structured way for researchers to train, re-train, save, analyse and manage experiments and experiment data. The GUI of this tool is designed using PyQT5 library which is based on the well known Qt [18] project. In this GANEx tool, we designed a mechanism to

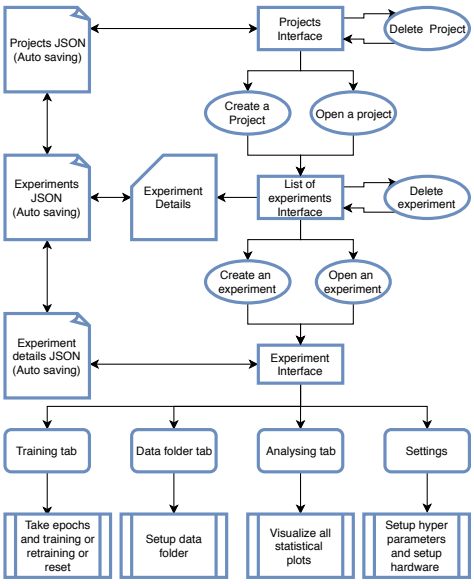
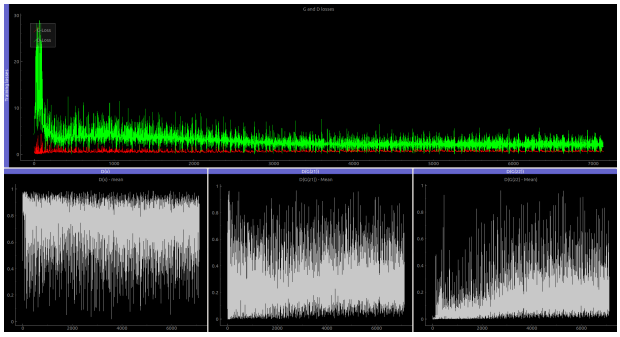


Fig. 3: GUI and JSON flow of GANEx

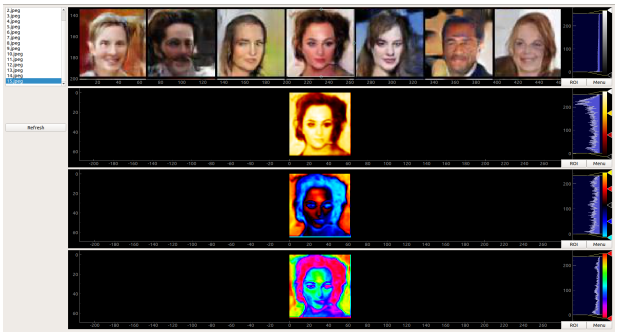
save all experimental details and data automatically to avoid losing them as a result of forgetting to save. In contrast to this, if the researcher wants to delete experiments or projects, they can use the delete option of our tool compared to the saving options of general GUI tools. The main motivation for this concept is that every experimental details may have valuable point in future and therefore it should be a reference or logged records to easily track previous experiments. The main GUI flow of our GANEx is depicted in Figure 3. This flow is defined from the top to the bottom; the top level is the starting point of the tool while the bottom level represents endpoints of the GUI flow. Four boxes in the bottom layer shows different user interfaces for configuring and visualizing experiments.

The top level of GANEx creates projects which can be consist of several GAN experiments. These experiments may have any type of FastGAN implementations. Then, GANEx enables users to organize their experiments in a structured way. As depicted in the left side of Figure 3, the GANEx tool uses its own JSON recorder to record project details.

The next main window is “list of experiments interface” which allows users to create different experiments based on different type of generative models. This window summarizes all the details about previous experiments and if researchers want it is possible to continue to the previous experiments. The user can create a new experiment by selecting a generative model type within the wizard window. Then, the GANEx GUI enables the main experiment window which has all the functionalities to control a specific GAN experiment. This experiments window is capable of handling several experiments of different types of generative models at the same time. Therefore, doing comparative generative model-based experiments are straightforward. The experiments JSON file organizes all details related to experiments of the current project.



(a) Real-time plotting



(b) Advanced analysis of generated images

Fig. 4: Sample screenshots of the GANEx tool; (a) - this screen shows real-time loss value plots of a GAN architecture to monitor the failures of GAN training process, (b) - in this window, the user can use different heat maps and color histograms to understand the generated outputs

The current implementation of the main experiment window has capabilities to train a generative model, handle input data sources, set or tune hyperparameters of generative models and analyze generative models by visualizing input data, generated data, training and re-training behaviour and many more statistical and engineering analysing mechanisms. Example screen shots of real application windows and plots are presented in Figure 4.

V. Expandable GANEx

This GANEx implementation opens doors to a wide range of directions for expansions. In the future, the GUI-based GANEx tool can be improved to design complex GAN architectures from scratch using drag and drop components while implementing training and inference via GUI-based flow diagrams. In adddition, the analysis functionalities can be expanded based on the state of the art findings without affecting them with the base implementation because we keep the analysing part as an independent section. Moreover, our tool can be upgraded with hardware resources monitoring for researchers who are dealing with performance improvements. Furthermore, using the concepts behind this tool, GANEx shows direction to implement advanced tools for other DL mechanisms also.

VI. DEMO

In this demo, participants can get hands-on knowledge of GANEx, and they will experience the power of the tool. They will be able to get an idea about how GANEx organize experiments and experimental details. In this session, users can train a pre-designed simple GAN model from scratch using simple datasets like MNIST (handwritten digits) [19] and CelebA (low resolution celebrity images) [20]. Then, they can analyse the training process in real-time and generated images using the analysis window of GANEx.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[2] I. J. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *Computing Research Repository (CoRR)*, vol. abs/1701.00160, 2016.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[4] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *Computing Research Repository (CoRR)*, vol. abs/1411.1784, 2014.

[5] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 469–477.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2017.

[7] A. Hindupur. (2017) Deep hunt: The gan zoo. [Online]. Available: https://deephunt.in/the-gan-zoo-79597dc8c347

[8] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[9] L. Yeager, J. Bernauer, A. Gray, and M. Houston, "Digits: the deep learning gpu training system," in *Proceeding of Automation of Machine learning (AutoML)*, 2015.

[10] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg, "Gan lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 310–320, 2019.

[11] A. Paler, "Surfbraid: A concept tool for preparing and resource estimating quantum circuits protected by the surface code," *arXiv preprint arXiv:1902.02417*, 2019.

[12] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, 4th ed. Morgan Kaufmann, 2016.

[13] G. LLC. (2019) Tensorboard. [Online]. Available: https://www.tensorflow.org/guide/summaries_and_tensorboard

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[15] I. Deep Cognition. (2018) Deep learning studio. [Online]. Available: https://deepcognition.ai/features/deep-learning-studio/

[16] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proceedings of the International Conference on Unsupervised and Transfer Learning Workshop (UTL)*. JMLR.org, 2011, pp. 37–50.

[17] L. Campagnola. (2011) Pyqt graphs. [Online]. Available: http://www.pyqtgraph.org/

[18] Q. G. N. H. QTCOM). (2019) Qt. [Online]. Available: https://www.qt.io/

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[20] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.