An Interactive Framework for Pausable and Switchable GAN Training

Bilal Azdad
ENSEEIHT / ALTEN
bilal.azdad.pro@gmail.com

Abstract

Objectives: This extended abstract presents an interactive framework enabling users to pause, resume, and switch the training focus between a GAN's generator and discriminator in real time. The project aims to provide fine-grained control over adversarial training, including mid-training adjustments to epoch count and learning rates, while offering immediate feedback on metrics. Such flexibility facilitates experimental workflows in deep learning research.

Methods: The approach relies on a modular system that separates the user interface from the model-training logic. The user interface (UI) is built in Python's Tkinter, through which users load configuration files specifying the architectures (layers, activation functions) for both generator and discriminator. Training loops run in separate threads for each network, employing pause/resume events. At any moment, the user may modify hyperparameters (learning rates, total epochs) and then resume the training with updated values.

Results: Empirical tests on the MNIST dataset have shown that toggling the active network mid-training can affect loss curves in characteristic ways: when the discriminator is trained alone, it becomes more proficient at distinguishing fakes; when the generator is resumed, it faces a stronger discriminator and exhibits a spike in loss before adapting. Meanwhile, the ability to pause, switch, and dynamically alter epoch or learning rate values allows new experimental routines, including extended training and real-time fine-tuning of hyperparameters.

Conclusions: By decoupling user interface, training controller, and persistent threads for each GAN component, the system provides a robust solution for interactive adversarial training. Researchers and practitioners can exploit dynamic training control, hyperparameter edits, and event-based pausing to explore and refine the behavior of GANs in ways that surpass conventional approaches.

Keywords: Generative Adversarial Networks, Pausable Training, Interactive Framework, Deep Learning, Threaded Architecture, Real-Time Control

1 Introduction

Generative Adversarial Networks (GANs) [1] have emerged as a powerful approach to generative modeling. In a standard GAN training scenario, the generator attempts to produce realistic data samples, while the discriminator seeks to distinguish real from generated samples. Typically, this competition proceeds via small alternating steps, with each network updated briefly in turn. Yet certain research contexts call for more flexible strategies—such as training one network for extended durations, or only training a single network and freezing the other. Furthermore, real-time modifications of hyperparameters, such as learning rates and epoch counts, can be extremely helpful in diagnosing instabilities (e.g., mode collapse [2]).

Implementing such flexibility is non-trivial. One must be able to pause or resume a network's gradient updates without re-initializing its weights, handle user interactions to switch between generator and discriminator, and preserve continuity of each network's training state. Additionally, adjusting the learning rate mid-training poses its own challenges in ensuring both loops remain consistent. This extended abstract proposes a solution: an *interactive framework* that runs each network in its own threaded loop, controlled by user-driven pause/resume and switch events. The user interface provides real-time feedback on performance metrics and the option to modify epochs or learning rates, enabling quick iteration and deeper insights into adversarial training dynamics.

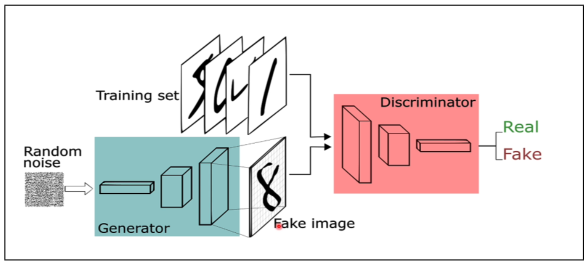


Figure 1: GAN Architecture image from https://medium.com/@danushidk507 .

2 Related Work

While many libraries implement GAN training, most focus on conventional step-by-step updates and lack interactive controls. Some advanced frameworks allow partial freezing of layers or custom schedules [3], but few deliver a complete UI with on-demand pausing, switching, or hyperparameter updates (e.g., adjusting the learning rate or number of epochs). Although the notion of flexible adversarial training has been explored theoretically (e.g., extended training on one network [4]), a user-facing interface for dynamic, real-time control remains uncommon. Our project addresses this gap by offering event-driven training with immediate logs, metrics, and image previews.

3 Details of the Research

3.1 System Architecture

Modular Design: The proposed framework is divided into three primary modules:

i) **UI (Tkinter):** Allows users to load a JSON file specifying network architectures (e.g., layer types, activations, noise dimension). Users can also edit training hyperparameters (learning rates, batch sizes, *and* total epochs) mid-training. A dedicated tab displays logs for real-time observation of losses and any updates to epoch or learning rate.

ii) **Controller (GANController):** Builds the generator and discriminator from the loaded file, initializes a DataLoader (MNIST or another dataset), and orchestrates threads. It reacts to pause/resume/switch commands from the UI, setting or clearing the appropriate events. If a user modifies the epoch limit or learning rate, the controller updates these parameters in the trainer accordingly before resuming.

iii) **Trainer (Trainer Class):** Contains the loops for generator and discriminator. Each loop runs persistently in its own thread, blocking on a pause event if it is not active. Once unpaused (or resumed), the loop continues from the same epoch index and the newly assigned hyperparameters (if changed).

3.2 Threaded Approach

In typical GAN code, alternating between generator and discriminator is done via something like `train_generator_step()` followed by `train_discriminator_step()`. By contrast, this framework employs two persistent loops:

- A **generator loop** that trains if unpaused for an entire epoch (or step).
- A **discriminator loop** that likewise operates when marked active.

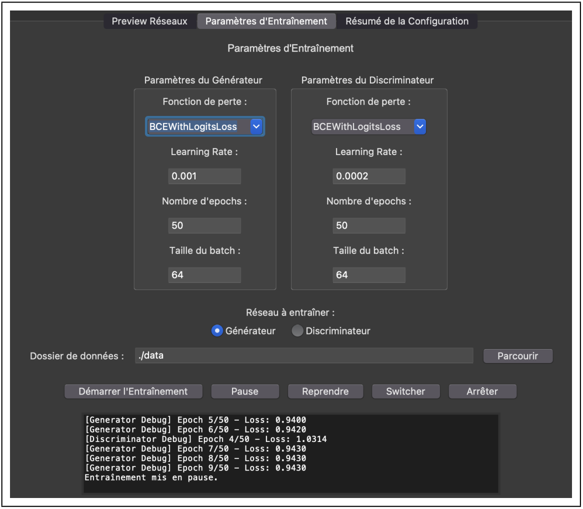
When the user presses "Switch," the system pauses the currently active network (clearing its event) and unpauses the other (setting its event). In addition, if the user typed a new epoch limit or changed the learning rate, the system updates these in the trainer before continuing. This yields a flexible user experience that surpasses the rigid, short-step alternation typical in many GAN scripts and allows real-time hyperparameter tuning.

3.3 Data Handling

Though tested primarily on MNIST, the code can adapt to any dataset. The DataLoader is rebuilt or re-invoked each epoch, returning fresh minibatches. If a user changes the epoch limit during training, the loop simply checks the new limit at the epoch boundary. If the user modifies the learning rate, the trainer applies that updated rate on subsequent training steps.

3.4 Logging and Visualization

Each training loop logs loss values to a text widget. Observers see how generator loss spikes if the discriminator has grown stronger, or how the discriminator might begin struggling if the generator significantly improved while it was paused. Additional logs or previews of generated images can be incorporated to visualize how the generator's outputs evolve. Changes to epoch or learning rate are also noted in the logs to keep track of user interventions.



Paramètres d'Entraînement

	Paramètres du Générateur	Paramètres du Discriminateur
Fonction de perte :	BCEWithLogitsLoss	BCEWithLogitsLoss
Learning Rate :	0.001	0.0002
Nombre d'epochs :	50	50
Taille du batch :	64	64

Réseau à entraîner : [✓ Générateur] Discriminateur

Dossier de données : ./data

```
[Generator Debug] Epoch 5/50 - Loss: 0.9400
[Generator Debug] Epoch 6/50 - Loss: 0.9420
[Discriminator Debug] Epoch 4/50 - Loss: 1.0314
[Generator Debug] Epoch 7/50 - Loss: 0.9430
[Generator Debug] Epoch 8/50 - Loss: 0.9430
[Generator Debug] Epoch 9/50 - Loss: 0.9430
Entraînement mis en pause.
```

Figure 2: This is the UI displaying live logs, a pause button, a switch button, and hyperparameter entry fields.

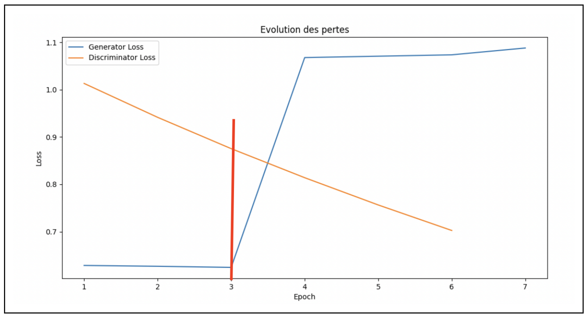


Figure 3: loss curves demonstrating how the generator's loss may spike after discriminator-only training.

4 Results and Conclusions

4.1 Empirical Observations

A common scenario involves launching a DCGAN [2], letting the generator train uninterrupted for several epochs, switching entirely to the discriminator, then updating the generator's maximum epoch count to a larger value while paused. Observations include:

- **Loss Spikes**: If the discriminator trains alone, it becomes more adept at classification, so resuming the generator—even at higher epochs—initially shows higher loss before readapting.
- **Mid-training Parameter Shifts**: Changing learning rates on the fly can show immediate improvements or destabilizations in adversarial training, highlighting the delicate balance between these networks.

4.2 Conclusion

Our interactive system demonstrates a **pausable, switchable** training approach that affords considerable experimental freedom, including on-the-fly hyperparameter updates like epoch count and learning rate. By empowering users to dynamically guide the adversarial process, it unveils new ways to explore GAN behavior beyond standard fixed scheduling. Future possibilities include multiple discriminators, specialized critics, or advanced data augmentation, all under the same interactive, threaded paradigm that supports real-time parameter changes.

Acknowledgements

We would like to express our sincere appreciation to **Alexandre Dore**, whose guidance and mentorship were invaluable throughout this project. We also extend our gratitude to the open-source community behind PyTorch, Tkinter, and related libraries. Finally, we thank our colleagues for their insightful suggestions on dynamic scheduling, UI refinements, and mid-training hyperparameter updates, all of which played a pivotal role in shaping the final design.

References

[1] I. Goodfellow, *et al.* Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 27, 2014.

[2] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*, 2016.

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. In *NIPS*, 2017.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *ICML*, 2017.