Ground Truth: The Invisible Eggshells of the Simulation Floor

Executive Summary: The Fragility of the Artificial

The trajectory of artificial intelligence has historically been defined by the pursuit of data. For decades, the "Ground Truth"—the objective reality against which models are measured—was a scarce resource, mined laboriously from the physical world. We built pipelines to ingest, clean, and label this precious commodity, treating every sample as a nugget of gold. However, we have now crossed a threshold into a new era: the era of synthetic abundance. Generative models, from Large Language Models (LLMs) to time-series synthesizers, offer an infinite spring of data. They promise to solve the "cold start" problem, democratize access to training resources, and allow us to simulate scenarios that are too dangerous or rare to capture in reality. But this abundance comes with a hidden, structural cost.

This report, commissioned for the *Ground Truth* newsletter, investigates the treacherous terrain of this new landscape. We argue that while synthetic data is a powerful fuel, it builds a "simulation floor" covered in invisible eggshells—hidden zones of brittleness, mode collapse, and underspecification that threaten the reliability of enterprise AI systems.

Our analysis focuses on the specific domain of **Behavioral Biometrics**, using the **SnapLock** application as a primary case study. SnapLock, a keystroke dynamics authentication system, represents the quintessential challenge of the synthetic age: how to verify *real* human identity using models trained on *simulated* human behavior.

We structure this investigation around three core pillars, supported by ten scientifically rigorous, peer-reviewed citations:

1. **The Physics of Model Collapse**: An analysis of the entropic decay that occurs when models recursively train on their own output (Shumailov et al., 2023).
2. **The Sim-to-Real Gap**: A technical dissection of why simulation fails to capture the stochasticity of the physical world, and how Domain Randomization (Tobin et al., 2017) and Adversarial Adaptation (Ganin et al., 2016) can bridge this divide.
3. **The SnapLock Paradigm**: A deep dive into the generation of synthetic keystroke dynamics using TimeGANs (Yoon et al., 2019) and statistical modeling (Migdal & Rosenberger, 2019), and the adversarial threats posed by generative attacks (Roy & Rumee, 2025).

This document serves not just as a report, but as a blueprint for the "chassis" of the synthetic data pipeline—the infrastructure required to navigate the simulation floor without breaking the eggshells.

Part I: The Entropy of Recursion – The Physics of Model Collapse

To understand the risks of synthetic data, we must first understand the mathematical inevitability of its degradation. In the early days of generative AI, there was a naive optimism that synthetic data could create a virtuous cycle: a model generates data, a second model trains on it and becomes better, generating even better data, *ad infinitum*.

Recent theoretical work has shattered this illusion. We now understand that without a constant injection of external entropy (real-world data), recursive generative loops act as contraction mappings, leading to a state of **Model Collapse**.

1.1 The Mathematical Mechanics of Variance Reduction

The primary mechanism of model collapse is variance reduction. A generative model is, fundamentally, a probability density estimator. Given a finite set of training data drawn from a true distribution $\mathcal{D}_{real}$, the model attempts to learn parameters θ such that the model distribution P_θ approximates $\mathcal{D}_{real}$.

However, this approximation is never perfect. To minimize loss (typically cross-entropy or Mean Squared Error), models tend to focus on the "modes" of the distribution—the areas of highest probability density. In doing so, they inevitably smooth out the "tails"—the rare events, the outliers, the nuances.

When a subsequent model (Generation $N+1$) is trained on data sampled from Generation N, it is training on an approximation of an approximation. The tails that were truncated in Generation N are not just ignored in Generation $N+1$; they are mathematically non-existent.

Shumailov et al. (2023) [1] provide the definitive theoretical treatment of this phenomenon in *The Curse of Recursion: Training on Generated Data Makes Models Forget*. Their work demonstrates that this is not merely a practical issue of poor hyperparameter tuning, but a fundamental property of statistical learning.

The Gaussian Mixture Proof

Consider a dataset modeled by a Gaussian Mixture Model (GMM). A GMM represents data as a combination of several normal distributions (bells curves). Shumailov et al. demonstrate that when a model estimates the parameters of a GMM based on samples from a previous GMM, the variance of the components shrinks with each generation.

If we denote the variance of the n-th generation model as σ_n^2, the relationship

can be expressed as a recursive function where $\sigma_{n+1}^2 < \sigma_n^2$. As $n \to \infty$, the variance $\sigma_n^2 \to 0$.

The implications of this are profound. A distribution with zero variance is a Dirac delta function—a single point.

- In **Image Generation**, this looks like the model producing the exact same face, regardless of the prompt.
- In **Language Modeling**, it manifests as the model repeating a small set of generic phrases, losing all creativity and nuance.
- In **SnapLock (Keystroke Dynamics)**, this means the synthetic data generator would eventually produce only the "average" typing rhythm (e.g., exactly 120ms dwell time for every key), completely losing the messy, jittery variance that characterizes distinct human individuals.

The "invisible eggshell" here is the illusion of convergence. As the model collapses, its performance on its *own* synthetic validation set actually improves. It becomes extremely confident because it is predicting a simplified, low-variance reality. But the moment it encounters the high-variance chaos of a real human user, it fails catastrophically.

1.2 The Erasure of the Tails and Minority Classes

Model collapse is not a uniform degradation. It is a targeted erasure of the statistical tails. In almost every real-world dataset, the "tails" contain the most information. They represent the edge cases, the anomalies, and often, the minority groups.

Seddik et al. (2024) [4] expand on this in *How Bad is Training on Synthetic Data?*, analyzing the phenomenon through the lens of statistical scaling laws. They argue that while mixing real data with synthetic data can delay the onset of collapse, it does not prevent the specific erosion of low-probability events unless those events are explicitly over-sampled.

For a product like **SnapLock**, this is a critical vulnerability.

- **The "Average" User**: Types with two hands, uses a QWERTY layout, has a standard flight time of 100-200ms.
- **The "Tail" User**: Types with one hand (due to disability or holding a coffee), uses a Dvorak layout, or has extremely fast "bursty" typing speeds.

If SnapLock is trained on recursively generated synthetic data, the "Tail User" is the first to disappear from the distribution. The model effectively "forgets" that one-handed typists exist. When a real one-handed typist attempts to authenticate, the model views their behavior not as a valid variation, but as an out-of-distribution anomaly—an attack.

This creates a hidden fairness issue. The "invisible eggshells" are scattered disproportionately in the areas of the simulation floor occupied by non-standard users. The system works

perfectly for the majority, hiding the brittleness that affects the minority.

1.3 The Thermodynamics of Data: An Irreversible Process

We can conceptualize Model Collapse through the lens of thermodynamics. Real-world data has high entropy—it is full of noise, surprise, and unstructured information. Generative models act as filters that reduce this entropy, organizing the data into structured patterns.

When we feed the output of the filter back into the input, we are creating a closed system. The Second Law of Thermodynamics dictates that in a closed system, entropy must increase, but in the context of *information content* (useful signal), the recursive loop essentially "burns off" the signal, leaving only the low-entropy residue (the mean).

Shumailov et al. (2023) describe this state as an "absorbing state." Once the model has collapsed, it cannot recover the lost information. You cannot "un-collapse" a model by training it longer on the same synthetic data. The only cure is the injection of fresh, high-entropy data from the real world.

Strategic Implication for Ground Truth:
This physics dictates the architecture of the data pipeline. A pipeline that relies 100% on synthetic data is doomed to thermodynamic death (collapse). A sustainable pipeline must be an open system, continuously ingesting "Ground Truth" from the physical world to counteract the entropy reduction of the models.
The "chassis" of the AI system must therefore include a "Real-World Reservoir"—an immutable, growing dataset of verified human actions that serves as the anchor for the simulation.

Part II: The Reality Gap – Engineering Across the Divide

If Model Collapse is the long-term strategic risk, the **Sim-to-Real Gap** is the immediate tactical hurdle. This concept, borrowed from robotics, describes the performance drop-off that occurs when a model trained in a simulated environment is deployed in the physical world.

In robotics, a simulator (like MuJoCo or Gazebo) uses physics equations to approximate gravity, friction, and collision. But equations are idealizations. They rarely capture the non-linear friction of a rusty gear, the latency of a USB controller, or the speck of dust on a camera lens.

For **SnapLock**, the "simulator" is the algorithm generating synthetic keystroke dynamics. The "real world" is the chaotic environment of a user's device: background CPU processes causing

lag, a sticky "E" key, the user's fatigue level, or the vibration of a train commute.

2.1 Domain Randomization: The Hammer of Robustness

The most effective technique for bridging this gap was formalized by **Tobin et al. (2017)** [6] in their seminal paper *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*.

Tobin's team faced a classic problem: they wanted to train a robot arm to grasp objects, but training on a real robot is slow and dangerous. Training in a realistic simulator is hard because matching reality perfectly is impossible. Their solution was counter-intuitive: instead of trying to make the simulation *more* realistic, they made it *less* realistic but *more* varied.

They randomized every parameter of the simulation:

- **Visuals**: The table was not brown wood; it was a static noise pattern or disco colors.
- **Lighting**: Lights moved randomly and changed intensity.
- **Physics**: The mass and friction of objects varied wildly.

The hypothesis was that if the neural network could learn to detect the object amidst this "psychedelic" chaos, the real world—with its boring wood tables and consistent lighting—would appear to the model as just another, simpler variation of the simulation.

> "With enough variability in the simulator, the real world may appear to the model as just another variation of simulation." — Tobin et al. (2017) [6]

Applying Domain Randomization to SnapLock:
For the SnapLock pipeline, this implies that generating "realistic" synthetic data is not enough. We must generate "Hyper-Synthetic" data that stresses the model beyond reality.

- **Latency Injection**: Instead of simulating a perfect USB poll rate (1000Hz), the generator should inject random latency spikes of 50ms, 100ms, or even 500ms.
- **Hardware Simulation**: Simulate key switches that are "sticky" (long dwell times) or "bouncy" (double presses).
- **Environmental Noise**: Superimpose sinusoidal noise patterns on the flight times to simulate a user typing on a moving bus.

By training on this randomized data, the SnapLock model learns features that are *invariant* to these disturbances. It learns the core "signature" of the user's motor cortex planning, ignoring the superficial noise of the hardware or environment.

2.2 Domain-Adversarial Training (DANN)

While Domain Randomization is a data-centric approach, **Domain-Adversarial Neural Networks (DANN)**, proposed by **Ganin et al. (2016)** [9], offer a model-centric architecture to

solve the same problem.

Ganin et al. recognized that deep neural networks are prone to learning "domain-specific" features—artifacts that exist in the source domain (synthetic) but not the target domain (real). To prevent this, they introduced a **Gradient Reversal Layer (GRL)**.

The architecture consists of three parts:

1. **Feature Extractor**: Maps input data to a latent vector.
2. **Label Predictor**: Predicts the class (e.g., "User ID").
3. **Domain Classifier**: Predicts the domain (Synthetic vs. Real).

The training objective is a minimax game: The Feature Extractor tries to *minimize* the Label Predictor's error but *maximize* the Domain Classifier's error.

> "We propose a new approach to domain adaptation... that ensures the feature distribution is invariant to the change of domain." — Ganin et al. (2016) [9]

Table 1: Comparison of Sim-to-Real Strategies

Strategy	Mechanism	Key Citation	Application to SnapLock
Domain Randomization	Increase variance of synthetic data to encompass reality.	Tobin et al. (2017)	Inject random latency, jitter, and hardware noise into synthetic keystrokes.
Domain Adaptation (DANN)	Force model to learn features indistinguishable between domains.	Ganin et al. (2016)	Use adversarial loss to strip "synthetic artifacts" from the learned embeddings.
Fine-Tuning	Pre-train on synthetic, fine-tune on small real dataset.	Standard Practice	Train base model on TimeGAN data, refine on 5-10 real logins.

2.3 Adaptation in Time: VRADA

SnapLock deals with time-series data, which introduces complexity beyond static images. User behavior is dynamic and stateful. The "Sim-to-Real" gap in time-series often manifests as a mismatch in temporal dependencies (e.g., a synthetic user might not capture the correlation between typing "th" and the subsequent pause).

Purushotham et al. (2017) [12] extended the DANN concept to the temporal domain with **Variational Recurrent Adversarial Deep Domain Adaptation (VRADA)**.

VRADA combines **Variational RNNs (VRNNs)** with adversarial training. VRNNs are capable of modeling the complex, non-linear variability of time-series data by introducing a latent random variable at each time step. By aligning the distributions of these latent variables between synthetic and real data, VRADA ensures that the model is robust not just to static noise, but to *dynamic* noise (like the changing rhythm of a fatigued user).

For SnapLock, implementing a VRADA-like architecture in the "chassis" ensures that the model adapts to the stochastic nature of human behavior, bridging the gap between the clean, algorithmic precision of the generator and the messy, fluid reality of human typing.

Part III: The SnapLock Paradigm – Synthetic Behavioral Biometrics

We now turn our focus to the specific application: **SnapLock**. As a tool for behavioral biometrics, SnapLock authenticates users based on *how* they type—their keystroke dynamics. This domain is uniquely sensitive to the issues of synthetic data because human behavior is both highly individual and highly variable.

3.1 The Anatomy of Keystroke Dynamics

Keystroke dynamics relies on the measurement of timing events during typing. The two primary atomic features are:

1. **Dwell Time (Hold Time)**: The duration a key is pressed down. $HT(k) = T_{release}(k) - T_{press}(k)$.
2. **Flight Time (Inter-Key Latency)**: The interval between releasing one key and pressing the next. $FT(k_1, k_2) = T_{press}(k_2) - T_{release}(k_1)$.

From these atomic units, we derive complex features:

- **Digraph Latency**: The time to type a specific pair (e.g., "t-h").
- **Trigraph Latency**: The time to type a triplet (e.g., "i-n-g").
- **Rhythm Consistence**: The variance of flight times over a session.

The "invisible eggshells" in this domain are the subtle correlations. For a real human, the flight

time of "t-h" is heavily dependent on the flight time of the preceding character. If the user is rushing, the entire sequence compresses. If they are hesitant, it expands. Synthetic generators often fail to capture this *covariance structure*, producing sequences that look statistically correct on a per-key basis but "feel" robotic to a sophisticated classifier.

3.2 The Migdal-Rosenberger Statistical Model

To generate synthetic users (e.g., to create a "background set" for anomaly detection training), we need a rigorous statistical framework. **Migdal & Rosenberger (2019)** [14] provide the gold standard in their paper *Statistical Modeling of Keystroke Dynamics Samples for the Generation of Synthetic Datasets*.

They conducted an exhaustive analysis of public datasets (like the GREYC dataset) to determine the probability distributions that best fit human typing patterns.

- **Dwell Times**: Best modeled by **Log-Normal** or **Gamma** distributions. Humans have a "minimum" physical time to press a key, and a long tail of "holding" the key while thinking.
- **Flight Times**: Best modeled by **Gaussian** distributions (for specific digraphs) but often exhibiting multimodal behavior (e.g., fast typing vs. pause-and-think).

  > "We propose... an original approach for the generation of synthetic keystroke data... enabling sharing of Keystroke Dynamics datasets without exposing biometric information." — Migdal & Rosenberger (2019) [14]

Application to SnapLock:
Using the Migdal-Rosenberger framework, SnapLock can define a "Synthetic User" as a set of parameters vector $\Psi = \{\mu_{dwell}, \sigma_{dwell}, \mu_{flight}, \sigma_{flight}, \dots\}$. By sampling these parameters from the global population distribution, SnapLock can generate thousands of unique "fake" users.
This allows for:

1. **Data Augmentation**: Expanding a small real dataset.
2. **Privacy Preservation**: Training on synthetic proxies of real users.
3. **Background Models**: Creating a "Universal Background Model" (UBM) that represents "general human typing" to contrast against a specific user.

However, the Migdal-Rosenberger model assumes a certain level of independence between features. It is a "Bag of Gaussian" approach. It breaks down when modeling long-range dependencies (e.g., fatigue setting in over a 10-minute session).

3.3 TimeGAN: Preserving Temporal Dynamics

To capture these long-range dependencies, we must move beyond statistical sampling to Deep Generative Models. **Time-series Generative Adversarial Networks (TimeGAN)**, introduced by **Yoon et al. (2019)** [17], represents the state-of-the-art for this task.

Standard GANs (Generative Adversarial Networks) struggle with time-series data because the discriminator gives feedback on the *whole sequence*, often missing stepwise errors. TimeGAN introduces a novel architecture that combines the flexibility of GANs with the control of autoregressive models.

The TimeGAN Architecture:

1. **Embedding Network**: Maps the high-dimensional static and temporal features into a lower-dimensional latent space.
2. **Recovery Network**: Reconstructs the data from the latent space (ensuring no information loss).
3. **Generator**: Generates sequences in the *latent space*.
4. **Discriminator**: Operates in the *latent space* to distinguish real vs. synthetic.

Crucially, TimeGAN includes a **Supervised Loss** term. The generator is not just penalized for being "unrealistic" (unsupervised loss); it is also penalized if it cannot predict the *next step* in the sequence (supervised loss).

$$\mathcal{L}_{total} = \mathcal{L}_{unsupervised} + \mathcal{L}_{supervised} + \mathcal{L}_{reconstruction}$$

> "A good generative model for time-series data should preserve temporal dynamics... TimeGAN... allows us to control the trade-off between the flexibility of the unsupervised GAN and the control of supervised training." — Yoon et al. (2019) [18]

SnapLock Implementation:
By training TimeGAN on real user logs, SnapLock can generate synthetic sessions that preserve the "micro-rhythms" of the user—the slight acceleration before a spacebar, the hesitation after a backspace. This data is far superior to statistical sampling for training Deep Learning classifiers (like LSTMs or Transformers) because it contains the complex temporal features that those classifiers rely on.

3.4 The Adversarial Frontier: Weaponized Synthesis

The power of TimeGAN highlights a dangerous duality. If we can use it to improve SnapLock, attackers can use it to break SnapLock.

Roy & Rumee (2025) [19] demonstrated this vulnerability in *Forging Keystrokes: Practical GAN-based Presentation Attacks on Typing Biometrics*. They trained GANs (LSTM-GAN, CNN-GAN) to mimic specific users' typing patterns.

The results were alarming. The GAN-generated keystrokes were able to bypass a standard Siamese Neural Network authenticator with a **95-96% success rate**. The synthetic data was "indistinguishable" from the real user's typing rhythm.

> "Synthetic keystrokes were indistinguishable from real patterns... underscores a serious flaw in keystroke-based authentication and the pressing need for strong defences." — Roy & Rumee (2025) [20]

The Invisible Eggshell:
This reveals the ultimate brittleness of the simulation floor. If SnapLock is trained primarily on synthetic data (even high-quality TimeGAN data), it may become blind to "synthetic artifacts." It learns to recognize the mathematical structure of the user, which the GAN can replicate perfectly. It fails to look for the "human imperfections" that the GAN might miss.
Defense Strategy:
To counter this, the SnapLock pipeline must include Adversarial Training.

1. **Red Teaming**: Use the Roy & Rumee method to generate "attack samples" against your own users.
2. **Spoof Class**: Add a "Spoof" class to the classifier. Train the model to explicitly distinguish between "Real User A" and "Synthetic User A."
3. **Liveness Detection**: Introduce challenge-response mechanisms (e.g., asking the user to type a specific, random sentence) that disrupts the pre-computed GAN sequence.

Part IV: The Brittleness of the Synthetic – Validation as the New Training

The central thesis of the *Ground Truth* newsletter is that infrastructure matters more than the model. In the context of synthetic data, this means **Validation Infrastructure** becomes the most critical component of the chassis.

If the training data is synthetic, we cannot trust the training loss. The model might just be memorizing the generator's biases (Model Collapse). Therefore, the "Ground Truth" must be preserved in the Validation set.

4.1 Underspecification: The Hidden Failure Mode

D'Amour et al. (2020) [21] introduced the concept of **Underspecification** in *Underspecification Presents Challenges for Credibility in Modern Machine Learning*.

They observed that machine learning pipelines often produce models that have identical performance on the standard validation set but behave radically differently in deployment. This happens because the validation set is not "specific" enough to constrain the model to a single solution.

Scenario for SnapLock:
Imagine training a SnapLock model.

- **Model A** learns to identify the user based on "average typing speed" (a fragile feature).

- **Model B** learns to identify the user based on "relative rhythm between keys" (a robust feature).

If the validation set consists only of standard typing samples, both models will achieve 99% accuracy. They are "underspecified."
However, on deployment day, the user is tired and types 20% slower.
- **Model A** fails (False Rejection).
- **Model B** succeeds (Robust).

The standard validation pipeline failed to detect the eggshell (brittleness to speed changes) because it didn't test for it.

> "Predictors that are equivalent according to standard held-out validation metrics can behave very differently in deployment... underspecification is a ubiquitous and serious problem." — D'Amour et al. (2020) [21]

4.2 Synthetic Data as Validation (Stress Testing)

Paradoxically, the solution to the risks of synthetic data is *more* synthetic data—but applied differently. **Hu et al. (2023)** [23] propose the paradigm of **Synthetic Data as Validation**.

Instead of trusting a small, potentially biased real-world validation set, we should use generative models to create massive, targeted "stress test" suites.

For SnapLock, we can programmatically generate "Edge Case" validation sets:

1. **The Fatigue Set**: Take real user samples and mathematically stretch the dwell times to simulate fatigue. Does the model hold?
2. **The Injury Set**: Simulate a user typing with one hand (altering specific flight times).
3. **The Network Lag Set**: Inject random Poisson noise into the timestamps.

If the model performs well on these synthetic stress tests, we have reduced underspecification. We have proven that the model is robust to specific, known variations.

> "Synthetic data... underlines a promising avenue to enhance AI robustness... [acting as] a proxy sim-to-real gap." — Hu et al. (2023) [23]

4.3 Measuring the Generator: Precision and Recall

Finally, how do we know if our synthetic data generator (TimeGAN or Migdal-Rosenberger) is good enough? Standard metrics like Log-Likelihood are misleading. A generator that memorizes the training data has perfect likelihood but zero utility (overfitting). A generator that produces only one perfect sample has high quality but zero diversity (mode collapse).

Sajjadi et al. (2018) [26] introduced a rigorous metric in *Assessing Generative Models via*

Precision and Recall.

- **Precision (Quality)**: Do the generated samples look like real data? (Are the keystroke sequences physically possible?)
- **Recall (Diversity)**: Do the generated samples cover the *entire* distribution of real data? (Does the generator produce fast typists, slow typists, and erratic typists?)

Table 2: Sajjadi Metrics for Synthetic Data Pipeline

Metric	Definition	Failure Mode	Impact on SnapLock
Precision	Proportion of synthetic samples that fall within the real data manifold.	**Hallucination**: Generator produces impossible typing patterns (e.g., negative flight times).	Model learns garbage features; high False Positive Rate.
Recall	Proportion of real data manifold covered by synthetic samples.	**Mode Collapse**: Generator produces only "average" users.	Model becomes biased; fails to recognize unique/outlier users (High False Rejection).

For the *Ground Truth* pipeline, monitoring **Recall** is the primary defense against Model Collapse. If the Recall score of the synthetic generator starts to drop over time (as it trains recursively), it is a red alert that the "invisible eggshells" are forming—the model is forgetting the tails.

Part V: Conclusion – Constructing the Chassis

The "simulation floor" is not a solid foundation; it is a dynamic, entropic surface. The "invisible eggshells" are the inevitable byproducts of recursive approximation: the loss of variance, the widening reality gap, and the hidden brittleness of underspecified models.

For the *Ground Truth* newsletter, and for the architecture of **SnapLock**, the research dictates a clear set of architectural principles:

1. **The Immutable Ground Truth**: Establish a "Gold Standard" reservoir of real-world data that is *never* effectively overwritten by synthetic loops. This prevents the thermodynamic death of Model Collapse (Shumailov et al.).
2. **Chaos is Consistency**: Use Domain Randomization (Tobin et al.) to inject noise, latency, and hardware jitter into the synthetic pipeline. Hardening the model against simulation chaos makes it robust to reality.
3. **Adversarial Alignment**: Implement Domain-Adversarial layers (Ganin et al., Purushotham et al.) to force the model to unlearn the "artifacts" of synthesis.
4. **Synthesize to Destroy**: Use synthetic data not just to train, but to attack and stress-test the model (Hu et al., Roy & Rumee). Use it to probe for underspecification.
5. **Metric Vigilance**: Move beyond accuracy. Monitor the **Recall** of the generator (Sajjadi et al.) to detect the early onset of mode collapse.

By adhering to these principles, we can build a chassis that is strong enough to carry the weight of generative AI. We can walk on the simulation floor without breaking the eggshells, ensuring that SnapLock—and the systems that follow it—remain robust, secure, and fundamentally grounded in reality.

Annotated Bibliography of Rigorous Citations

1. **** Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., & Anderson, R. "The Curse of Recursion: Training on Generated Data Makes Models Forget." *Nature / arXiv:2305.17493*. [1]
2. **** Seddik, M. E. A., et al. "How Bad is Training on Synthetic Data? A Statistical Analysis of Language Model Collapse." *arXiv:2404.05090*. [4]
3. **** Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [6]
4. **[Ganin et al., 2016]** Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. "Domain-Adversarial Training of Neural Networks." *Journal of Machine Learning Research (JMLR)*, 17(59):1-35. [9]
5. **[Purushotham et al., 2017]** Purushotham, S., Carvalho, W., Nilanon, T., & Liu, Y. "Variational Recurrent Adversarial Deep Domain Adaptation." *International Conference on Learning Representations (ICLR 2017)*. [12]
6. **** Yoon, J., Jarrett, D., & van der Schaar, M. "Time-series Generative Adversarial Networks." *Advances in Neural Information Processing Systems (NeurIPS 2019)*. [17]
7. **** Migdal, D., & Rosenberger, C. "Statistical Modeling of Keystroke Dynamics Samples for the Generation of Synthetic Datasets." *Future Generation Computer Systems*, Vol 100. [14]
8. **** Roy, I., & Rumee, S. T. A. "Forging Keystrokes: Practical GAN-based Presentation

Attacks on Typing Biometrics." *2025 IEEE International Conference on Signals and Systems (ICSigSys)*. [19]
9. **** D'Amour, A., et al. "Underspecification Presents Challenges for Credibility in Modern Machine Learning." *Journal of Machine Learning Research (JMLR)*. [21]
10. **** Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., & Gelly, S. "Assessing Generative Models via Precision and Recall." *Advances in Neural Information Processing Systems (NeurIPS 2018)*. [26]
11. **[Hu et al., 2023]** Hu, Q., Yuille, A., & Zhou, Z. "Synthetic Data as Validation." *arXiv:2310.16052 / IEEE/CVF*. [23]

Works cited

1. Addressing Concerns of Model Collapse from Synthetic Data in AI, accessed December 18, 2025, https://www.gretel.ai/blog/addressing-concerns-of-model-collapse-from-synthetic-data-in-ai
2. [PDF] AI models collapse when trained on recursively generated data - Semantic Scholar, accessed December 18, 2025, https://www.semanticscholar.org/paper/AI-models-collapse-when-trained-on-recursively-data-Shumailov-Shumaylov/603d3f90fc40f79ff51258f0295de3ec5107f73e
3. The Curse of Recursion: Training on Generated Data Makes Models Forget (2023) | Ilia Shumailov | 136 Citations - SciSpace, accessed December 18, 2025, https://scispace.com/papers/the-curse-of-recursion-training-on-generated-data-makes-34zgysyl
4. How bad is training on synthetic data? A statistical analysis of language model collapse, accessed December 18, 2025, https://openreview.net/forum?id=t3z6UlV09o
5. [2404.05090] How Bad is Training on Synthetic Data? A Statistical Analysis of Language Model Collapse - arXiv, accessed December 18, 2025, https://arxiv.org/abs/2404.05090
6. [1703.06907] Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World - ar5iv, accessed December 18, 2025, https://ar5iv.labs.arxiv.org/html/1703.06907
7. [1703.06907] Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World - arXiv, accessed December 18, 2025, https://arxiv.org/abs/1703.06907
8. [PDF] Domain randomization for transferring deep neural networks from simulation to the real world | Semantic Scholar, accessed December 18, 2025, https://www.semanticscholar.org/paper/Domain-randomization-for-transferring-deep-neural-Tobin-Fong/32ceb28e45a445df4d89df281bb0e3ab5aab1a2a
9. Domain-Adversarial Training of Neural Networks - Journal of Machine Learning Research, accessed December 18, 2025, https://jmlr.org/papers/volume17/15-239/15-239.pdf

10. Domain-Adversarial Training of Neural Networks, accessed December 18, 2025, https://jmlr.org/papers/v17/15-239.html
11. [1505.07818] Domain-Adversarial Training of Neural Networks - arXiv, accessed December 18, 2025, https://arxiv.org/abs/1505.07818
12. Variational Recurrent Adversarial Deep Domain Adaptation - OpenReview, accessed December 18, 2025, https://openreview.net/forum?id=rk9eAFcxg
13. AdaDiag: Adversarial Domain Adaptation of Diagnostic Prediction with Clinical Event Sequences - NIH, accessed December 18, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9580228/
14. Analysis of Keystroke Dynamics for the Generation of Synthetic Datasets - ResearchGate, accessed December 18, 2025, https://www.researchgate.net/publication/329958878_Analysis_of_Keystroke_Dynamics_for_the_Generation_of_Synthetic_Datasets
15. Statistical Modeling of Keystroke Dynamics Samples For the Generation of Synthetic Datasets - ResearchGate, accessed December 18, 2025, https://www.researchgate.net/publication/332317758_Statistical_Modeling_of_Keystroke_Dynamics_Samples_For_the_Generation_of_Synthetic_Datasets
16. Christophe Rosenberger - DBLP, accessed December 18, 2025, https://dblp.org/pid/r/ChristopheRosenberger
17. Codebase for Time-series Generative Adversarial Networks (TimeGAN) - NeurIPS 2019 - GitHub, accessed December 18, 2025, https://github.com/jsyoon0823/TimeGAN
18. Time-series Generative Adversarial Networks - NIPS papers, accessed December 18, 2025, http://papers.neurips.cc/paper/8789-time-series-generative-adversarial-networks.pdf
19. Forging Keystrokes: Practical GAN-based Presentation Attacks on Typing Biometrics, accessed December 18, 2025, https://www.semanticscholar.org/paper/Forging-Keystrokes%3A-Practical-GAN-based-Attacks-on-Roy-Rumee/b73eb72c37b3491aeadcc5397c2c4542bbc1ea9c
20. Forging Keystrokes: Practical GAN-based Presentation Attacks on Typing Biometrics - IEEE Xplore, accessed December 18, 2025, https://ieeexplore.ieee.org/iel8/11269102/11269119/11269164.pdf
21. Underspecification Presents Challenges for Credibility in Modern Machine Learning, accessed December 18, 2025, https://www.jmlr.org/papers/volume23/20-1335/20-1335.pdf
22. Underspecification Presents Challenges for Credibility in Modern Machine Learning - arXiv, accessed December 18, 2025, https://arxiv.org/abs/2011.03395
23. [2310.16052] Synthetic Data as Validation - arXiv, accessed December 18, 2025, https://arxiv.org/abs/2310.16052
24. SYNTHETIC DATA AS VALIDATION - OpenReview, accessed December 18, 2025, https://openreview.net/pdf/f0fa9bc5765813847bb89f384f89a8c579482211.pdf
25. Qixin Hu - Google Scholar, accessed December 18, 2025, https://scholar.google.com/citations?user=EqD5GP8AAAAJ&hl=en
26. ON SELF MODULATION FOR GENERATIVE ADVER- SARIAL NETWORKS -

OpenReview, accessed December 18, 2025, https://openreview.net/pdf?id=HkI5aoR5tm
27. Assessing Generative Models via Precision and Recall | Empirical Inference, accessed December 18, 2025, https://is.mpg.de/ei/publications/sajbahlucbougel18
28. [1806.00035] Assessing Generative Models via Precision and Recall - arXiv, accessed December 18, 2025, https://arxiv.org/abs/1806.00035
29. The Curse of Recursion: Training on Generated Data Makes Models Forget - arXiv, accessed December 18, 2025, https://arxiv.org/abs/2305.17493
30. Forging Keystrokes: Practical GAN-based Presentation Attacks on Typing Biometrics, accessed December 18, 2025, https://www.researchgate.net/publication/398394482_Forging_Keystrokes_Practical_GAN-based_Presentation_Attacks_on_Typing_Biometrics