Masalah dataset yang statis dan kurang variatif memang menjadi musuh utama bagi model *Reinforcement Learning* (RL) maupun klasifikasi, karena model akan mengalami *overfitting* pada kondisi normal dan gagal saat menghadapi anomali. Keinginan Anda menggunakan GAN/WGAN sudah sangat tepat, namun seperti yang Anda khawatirkan, tantangan utamanya adalah bagaimana memastikan data sintetis tersebut **realistis (tidak "ngasal")** dan **memiliki makna fisik** yang sesuai dengan protokol jaringan.
Berdasarkan sumber-sumber terbaru yang Anda berikan, berikut adalah strategi teknis untuk menjawab kebutuhan Anda:

1. Agar Data Tidak "Ngasal": Gunakan *Graph-Conditioned WGAN* dengan Panduan LLM

Untuk memastikan data buatan mematuhi aturan jaringan (misal: jika throughput tinggi, maka packet_rate juga harus tinggi), Anda tidak bisa hanya menggunakan WGAN biasa. Anda perlu "memaksa" generator untuk memahami hubungan antar-fitur.

- **Graph-Conditioned Generation:** Sumber terbaru dari **Alyoubi et al. (2025)** mengusulkan penggunaan **Graph-Conditioned WGAN**. Caranya adalah dengan membuat *Feature Relationship Graph* menggunakan korelasi Pearson (linear), Spearman (non-linear), dan *Mutual Information* dari data asli Anda. Graf ini menjadi input "kondisi" bagi generator agar ia tahu fitur mana yang saling bergantung. Ini menjamin data sintetis menjaga struktur statistik data asli 1, 2, 3.
- **Semantic Constraints via LLM:** Anda bisa menggunakan LLM (seperti GPT-4) untuk mengekstrak aturan semantik (constraints). Contohnya, Anda meminta LLM mendefinisikan aturan: *"Jika status adalah 'Burst', maka packet rate harus melonjak > X pps dalam waktu singkat"*. Aturan ini dimasukkan sebagai *penalty term* dalam *loss function* WGAN. Jika generator membuat data yang melanggar aturan ini, ia akan dihukum berat saat training 1, 4.

2. Strategi Skenario Kombinasi (Stable, Overload, Burst)

Ide Anda untuk membuat kombinasi kondisi (misal: Port 1 Stable, Port 2 Overload) sangat valid dan dikenal sebagai **Conditional Generation** atau **Curriculum Generation**.

1. **Conditional WGAN-GP (CWGAN-GP):** Alih-alih membiarkan GAN men-generate data acak, Anda menggunakan **Conditional GAN**. Anda menyuplai label kondisi (misal: vektor [5] untuk kondisi "Port 1 Overload") ke dalam Generator dan Discriminator.
2. Sesuai referensi 6 dan 7, **Conditional GAN (CGAN)** memungkinkan Anda mengontrol output spesifik. Anda bisa mensintesis data untuk kelas minoritas (seperti "Burst Traffic") secara tertarget.
3. Anda bisa merancang matriks skenario seperti yang Anda bayangkan:
4. Skenario A: P1 (Stable), P2 (Overload), P3 (Stable)
5. Skenario B: P1 (Burst), P2 (Stable), P3 (Overload)
6. Dan seterusnya. Latih WGAN untuk mengenali label-label ini.
7. **EO-WGAN (Hybrid SMOTE + WGAN):** Jika data kondisi ekstrem (seperti "Overload") sangat sedikit di dataset asli, WGAN mungkin sulit belajar dari awal. Referensi **Riaz et al. (2025)** menyarankan metode **EO-WGAN**.
8. Gunakan **SMOTE** dulu untuk memperbanyak sampel kondisi ekstrem secara kasar (interpolasi).

9. Gunakan data hasil SMOTE tersebut untuk melatih **WGAN-GP**. WGAN akan "memperhalus" data kasar tadi agar distribusinya lebih realistis dan menghilangkan *noise* buatan SMOTE 8, 9, 10.

3. Menjaga Aspek Waktu (Time-Series)

Karena data trafik jaringan adalah *time-series*, data "burst" tidak terjadi dalam satu baris data saja, melainkan urutan waktu.

- **LSTM-GAN:** Mengacu pada **König et al. (2023)**, untuk data *condition monitoring* atau trafik yang bergantung waktu, gabungkan arsitektur **LSTM (Long Short-Term Memory)** dengan WGAN. Generatornya bukan *Dense Layer* biasa, tapi LSTM yang bisa menghasilkan *sequence* data 11. Ini penting agar RL agent Anda bisa melihat pola "kenaikan trafik" sebelum overload terjadi.

4. Penerapan pada Pelatihan Model (Curriculum Learning)

Setelah Anda punya dataset buatan dengan berbagai kombinasi kondisi tadi, jangan langsung suapkan semuanya ke model RL. Gunakan pendekatan **Genet** (Automatic Curriculum Generation):

- Mulailah melatih agen RL dengan data kondisi "Stable" (mudah).
- Secara bertahap, masukkan data sintetis "Overload" dan "Burst" (sulit) berdasarkan performa agen.
- Referensi **Xia et al.** menunjukkan bahwa pendekatan kurikulum ini membuat agen belajar lebih *robust* dibanding langsung dilatih dengan data acak yang sulit 12, 13.

Rangkuman Langkah Implementasi:

- **Analisis Fitur:** Buat matriks korelasi dari dataset asli untuk memetakan hubungan antar fitur (misal: util vs drop).
- **Define Conditions:** Labeli data Anda (atau buat aturan manual) untuk 3 kondisi: *Stable*, *Overload*, *Burst*.
- **Train Conditional WGAN-GP:**
- Input: Noise + Label Kondisi (misal: "Port 2 Burst").
- Guidance: Gunakan *Graph Correlation* agar fitur-fitur tetap logis secara fisik.
- **Generate Data:** Minta model membuat ribuan sampel untuk setiap kombinasi skenario yang jarang terjadi di data asli.
- **Validasi:** Cek apakah data buatan mematuhi logika (misal: tidak ada *negative latency*).
- **Train RL:** Latih agen dengan campuran data asli (untuk kondisi normal) dan data sintetis (untuk kondisi ekstrem/corner cases).