Yanez Multimodal Inorganic Identities Dataset (MIID) Subnet (ث)

Phase 3

V.3.1 – Dec 2025

Phase 3 — Threat Scenario Expansion and Continuous Improvement

The reward system in MIID's subnet is designed to incentivize real quality, not just participation. It fuses data quality, diversity, and contextual alignment with YANEZ's roadmap and customer-driven testing needs — rewarding miners who strengthen the data foundation used to build bulletproof financial crime detection systems. Our focus remains on signal over noise — ensuring emissions are directed only to miners who contribute measurable value.

Phase 3 marks the transition from static evaluation to continuous self-learning cycles, directly supporting our roadmap goal to turn the Subnet into a Self-Learning Loop.

> *"Expand the Threat Scenario Query System to allow proposed (unknown) threat scenarios from miners."*

Each new cycle in Phase 3 becomes a **closed-loop iteration of improvement** — a feedback system where miners don't just execute queries but actively **teach the subnet new behaviors and threat patterns.**

Through this loop:

- Miners propose and submit **new or unknown attack vectors (UAVs)**, enriching our testing corpus beyond known patterns.
- Validators measure performance, validate scenarios, and refine reward scoring logic based on novelty, precision, and impact.
- Penalty enforcement ensures **repetitive or low-value submissions** are filtered out, maintaining a strong signal-to-noise ratio.

This transforms MIID's subnet into a **self-adaptive learning system**, where every contribution strengthens the model, the rule base, and the reward logic for the next iteration.

Roadmap Transition and Cycle Overview

Phase 3 introduces a structured roadmap built on iterative learning and feedback:

- t_0 – **Baseline (current stage):**
    - The rank–quality fused system establishes stable scoring and emission behavior.
- t_1 – **Sandbox (Cycle 1 Initialization):**
    - A controlled environment for testing reward distribution, penalty calibration, and anti-cheat logic before live execution.
- t_2 – **Execution (Cycle 1 Live):**
    - Miners generate threat scenarios, validators assess quality and novelty, and the system collects real-time data for refinement.
    - Outputs from t2 are analyzed, post-validation is performed, and insights are integrated into LDS vNext, completing the loop.
- t_3 – **Sandbox (Next Cycle Preparation):**
    - The refined **LDS version** and **updated reward model** are deployed in a controlled test cycle.
    - This stage finalizes system adjustments, verifies consistency across miners, and prepares the subnet for the next live execution cycle.
    - Miners begin receiving post-validation UAV rewards based on the formula (novelty × impact × quality × reputation × penalty), recognizing those who contributed validated, unknown attack vectors that improved the system.

In essence, Phase 3 marks MIID's transition from static evaluation to dynamic learning — a subnet that improves through miner behavior, validator feedback, and detector updates. Each iteration strengthens the loop between quality, transparency, and reward logic.

The latest update to our reward mechanism, the rank-quality fused system, on the mainnet since October 28th, 2025, reflects this shift: a system that not only evaluates what miners deliver, but *how distinct, consistent, and valuable their contributions are over time*.

Rank–Quality Fused Reward System

The rank–quality fused reward system (live on mainnet) aligns rewards with **true miner performance**. It blends **rank-based reward scaling** with **reward scoring (as shown in Phase 2 above in this document),** so top contributors earn proportionally higher rewards

while keeping paths open for all qualified miners. The aim isn't to just "pass" — it's to **excel** through originality and precision.
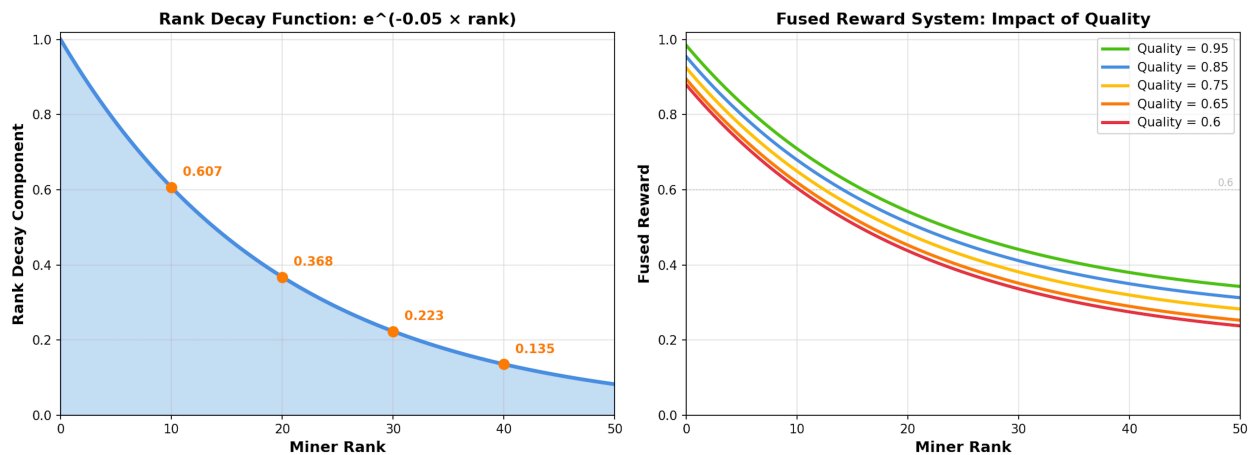
Miners are ordered by performance. Rewards follow an **exponential decay** so top ranks earn noticeably more.

Fused formula (simplified):

$$r_{(fused)} = 0.7 \times e^{-0.05 \times \text{rank}} + 0.3 \times \text{Quality}$$

rank: rank of top 50 miners based on their final reward as calculated in Phase 2, and must be >= 0.6

Quality: Final Reward as calculated in the Phase 2 section

This ensures that **position**, **data quality, and penalty avoidance** are all important.

Why Rank Alone Wasn't Enough

Rank-only decay made rewards too flat at the top, underweighting genuine quality differences.

Rank-only: $r = e^{-0.05 \times \text{rank}}$

Example (rank-only):

Miner Rank Quality Reward

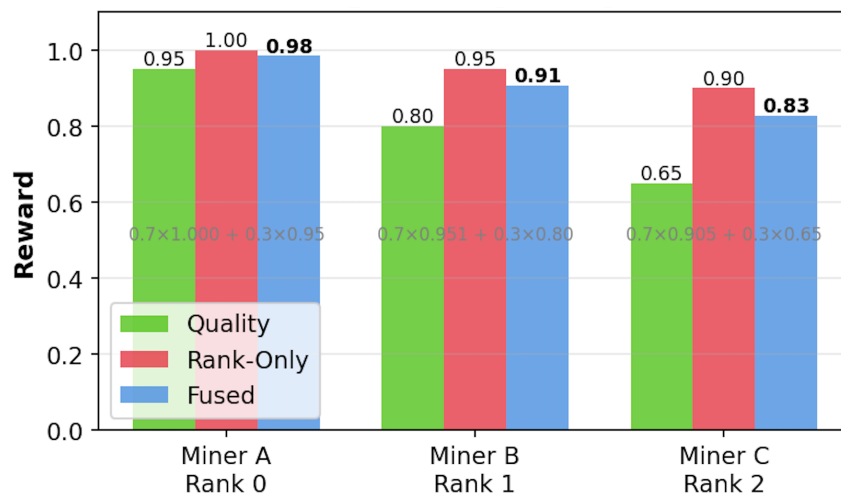| Miner	Rank	Quality	Reward
A	0	0.95	1.00
B	1	0.70	0.95
C	2	0.65	0.90

The top ranks were too close. By **fusing quality**, we restore meaningful separation:

Fused: $r = 0.7 \cdot e^{-0.05 \cdot \text{rank}} + 0.3 \cdot \text{Quality}$

Example (rank + quality fused):

Miner Rank Quality Reward

Miner	Rank	Quality	Reward
A	0	0.95	**0.97**
B	1	0.70	**0.86**
C	2	0.65	**0.83**

Another Example in Practice

The **final fused reward** is tied to each round's **incentive score**. To receive emissions, a miner must:

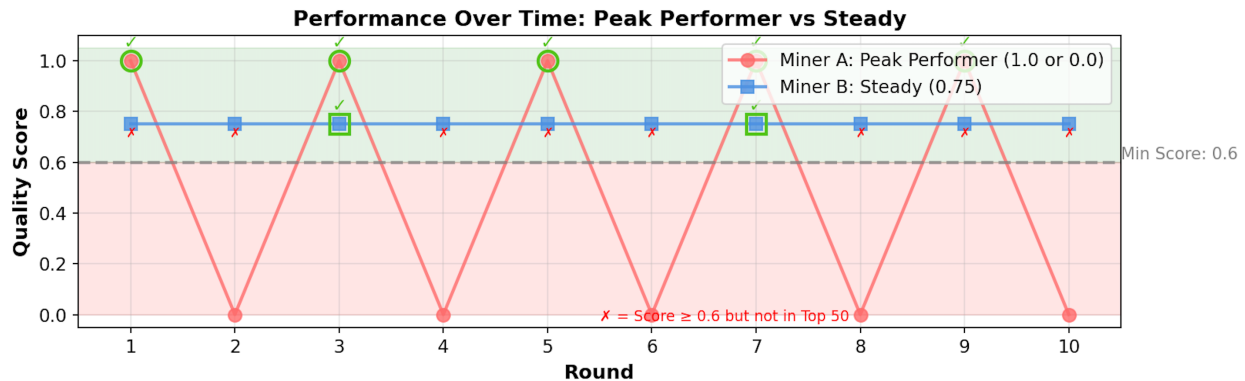- Score ≥ **0.6**, and
- Be in the **top 50** for that round.

Only qualified miners receive the **scaled reward** from the fused function. This emphasizes **per-query performance** over long-term averages.

Example:

- **Miner A:** 1.0 half the time, 0.0 half the time
- **Miner B:** steady 0.75 every time

Miner B's **mean** (0.75) may look higher on Yanez miners' dashboard, which present average hourly, but if 0.75 often **misses the top-50 threshold**, emissions stay low. Miner A, despite volatility, **crosses the threshold** more frequently and receives higher incentives.

Net effect: the system rewards **peaks of excellence**, not just steady averages.

Performance Over Time: Peak Performer vs Steady

Why We Designed It This Way

- Discourages pipeline cloning across multiple keys
- Rewards originality in data, models, and tuning
- Keeps incentives **adaptive**, tied to real quality and diversity
- Creates a **self-optimizing curve** where better data yields exponential value

Reputation and Transparency System

As MIID's subnet matures, the next layer of evolution focuses on **Reputation and Transparency** — creating a system where every miner's contribution is traceable, explainable, and valued beyond a single cycle.

We are still finalizing the **exact method of calculating and quantifying reputation**, and those details will be shared once we complete further design and validation.

What's clear, however, is the direction: a fair, data-driven system that carries miner intelligence forward and rewards genuine contribution over time.

With each cycle, our goal is for the subnet to get **smarter**, the miner base to get **stronger**, and the reward system to get **fairer** and more **data-driven**.

- Miners will be rewarded for **novel, high-impact scenarios**, not repetition.

- **Penalty enforcement** will ensure low-value or templated submissions don't dilute the overall quality signal.
- Reputation will act as a **trust multiplier**, reflecting both performance consistency and contribution quality across cycles.

Reputation isn't a static score — it's an evolving signal that grows with the miner's body of work and how well their contributions enhance subnet learning and validation outcomes. Over time, reputation becomes both a **trust signal** and a **performance multiplier** — recognizing sustained excellence, not one-time wins.

How Reputation Will Work

Reputation will be earned and updated through a combination of **performance validation** and **behavior analysis** carried out after each execution phase (t_1):

Post-Cycle Validation and Classification

After every execution round, all submissions undergo offline validation to establish *Ground Truth (GT)*. Submissions are classified into:

- **KAV (Known Attack Vector):** Expected, requested scenarios.

- **UAV (Unknown Attack Vector):** Novel or unanticipated attack patterns.

- **Invalid or Cheat:** Duplicated, spam, or collusive content.

Confirmed UAVs are added to the subnet's known_uav_corpus and contribute to the next version of the Detection System based on the task at hand. This process ensures reputation is built on *verified contribution quality*, not volume or luck.

Reputation Growth and Decay

- Every miner starts from a **neutral baseline (1.0)**.
- Over time, reputation increases or decreases based on behavior and data impact:

Behavior Type	Effect on Reputation	Example Impact
Consistent top performance	Gradual increase above 1.0	Rewards sustained accuracy
Verified UAV discovery	Sharp positive boost	Credits novel, high-impact work
Repeated or low-value submissions	Gradual decay below 1.0	Penalizes redundancy

Behavior Type	Effect on Reputation	Example Impact
Collusion or spam	Significant penalty	Protects data integrity

Over multiple cycles:

- **Longstanding, high-reputation miners** will gradually gain more influence in subnet learning and validation.
- **New miners** or those with a decayed reputation can still participate and earn through base checks (e.g., KAV-based validation). Still, they must demonstrate consistent quality to recover toward or above the baseline.
- This creates a natural **separation between sustained contributors and short-term opportunists**, balancing opportunity with merit.

Why It Matters

Reputation connects effort to longevity; it ensures that:

- The subnet retains and amplifies high-quality contributors.
- Long-term trust and originality matter more than short-term activity.
- Every miner can see a clear path to growth through consistent, meaningful work.

This transforms mining from a single-round competition into a **continuous learning ecosystem** — one where integrity, innovation, and sustained contribution define success.

Transparency & Miner Feedback

Transparency is at the core of how we are designing MIID's reputation and reward systems. Reputation must not feel like a black box — miners should understand *how* their scores are determined, *why* penalties occur, and *what* they can do to improve.

Visibility and Breakdown of Scores

Miners will be able to see not only their position on the leaderboard but also a detailed breakdown of how their score was constructed.

Each score component will be traceable back to measurable signals such as:

- **Quality:** Accuracy and completeness of submissions.
- **Novelty:** Contribution of unique or previously unseen patterns (UAVs).
- **Penalties:** Deductions for duplication, collusion, or spam detection.

This breakdown will help miners correlate their actions with their resulting scores — turning feedback into learning.

Advanced Miner Dashboard

We are developing a new **interactive miner dashboard** that extends far beyond current leaderboard metrics.

It will allow miners to:

- View performance trends across cycles — how quality and reputation evolve over time.
- Access anonymized examples of *successful* and *penalized* submissions, showing what worked and what triggered deductions.

The goal is to turn scoring into a **two-way transparency channel** — one where miners can *see their progress* and *learn from the system* itself.

Sample-Based Feedback Mechanism

To enhance interpretability, we are designing a mechanism that links reputation changes to specific submission samples.

For example:

- A miner can review a selection of their own entries from a cycle.
- Each example will include contextual notes, such as *"passed quality gate," "flagged for duplication,"* or *"classified as novel UAV."*
- This lets miners verify results and understand the reasoning behind reward or penalty adjustments.

We are testing approaches to securely share this feedback — preserving miner privacy while ensuring the verifiability of the post-validation process. And beyond the dashboard, our door remains open: we continue to meet directly with miners who want to share feedback, ideas, or concerns. This is a system we're shaping together.

Burning Mechanism & Emission Evolution

Reputation helps the subnet identify consistent, high-quality miners.

Transparency ensures every participant understands *how and why* they earned what they did.

And **burning** — it's the stability regulator.

The burn mechanism exists not to penalize, but to **stabilize** the subnet's growth. It keeps reward cycles, reputation-based incentives, and token emissions sustainable as the system evolves.

By dynamically adjusting emissions based on *network maturity* and *data quality*, it ensures that every phase of the roadmap — from early compliance testing to advanced KYC scenario modeling — operates on a stable and value-aligned foundation.

Why Burning Matters

Our burn mechanism serves three key purposes:

1. **Ease sell pressure** during the first few cycles while miner quality and diversity stabilize.
2. **Sustain YANEZ operations** — supporting post-validation, offline audits, and reputation-building work without overextending emissions.
3. **Preserve long-term network value** — ensuring each emission cycle reflects meaningful, verifiable contributions.

Each new roadmap milestone introduces new and more complex tasks — from name variation for sanction and address validation to a KYC threat simulation.

The burn system ensures these transitions happen under **controlled, value-aligned economic conditions**, balancing incentive growth with ecosystem health.

Long-Term Vision

- **Gradual Burn Reduction:** As subnet maturity, miner diversity, and the Detection System's reliability, based on the task at hand, increase, the burn percentage will decrease.
- **Stabilized Emissions:** Over time, rewards will be distributed more heavily among high-reputation miners and validated contributors of novel data.
- **Self-Sustaining Growth:** Once miner quality becomes consistent and self-reinforcing, emission burns will serve primarily as *fine-tuning controls* rather than safeguards.

The following section outlines the details of **Cycle 1**, including its timeline, objectives, and emission plan. It explains how this first execution stage sets the foundation for the subnet's long-term evolution.

Cycle 1 — Building the Location Detection System (LDS v0)

Cycle 1 marks the first live phase within YANEZ's strategic plan to develop a **Location Detection System (LDS)**.

This system is engineered to assist sanction-screening mechanisms in identifying high-risk or sanctioned addresses, along with their potential variations.

During this phase, the subnet begins transitioning from merely generating name variations and addresses to **incorporating geolocation intelligence**, thereby establishing the foundational framework for **LDS vNext**.

To establish a baseline dataset and evaluation loop for LDS v0, real addresses are combined with synthetic variations and geocoding validation utilizing open-source mapping APIs.

The outputs generated from this phase will underpin the first models for location-risk detection and will delineate the evaluation metrics to be employed in subsequent cycles.

LDS v0 Components

- **Map API:** Nominatim OpenStreetMap Search API— used for address normalization, geocoding, and verification of miner submissions.
- **Validation Engine:** Existing MIID validator framework (rank–quality fused system) extended with geocoding checks.
- **Model Version: LDS v0** — rule-based and API-assisted; future versions will integrate learning components as validated data accumulates.

Miner Tasks

Miners contribute data enabling LDS v0 to learn how to distinguish **valid** from **ambiguous or false** addresses.

Each miner submits two categories of data:

1. Known Attack Vectors (KAVs)

- Valid, well-structured addresses that can be geocoded successfully by the API.
- Serve as benchmarks for consistency, baseline scoring, and geographic coverage testing.

2. Unknown Attack Vectors (UAVs)

Per seed country or region, miners will:

- Generate a **look-alike or variant** of a real address that appears legitimate but fails API validation (false negative).
- Provide a **short description or label** explaining why it could represent a true address (e.g., local abbreviation, transliteration, or postal variant).
- Include the **longitude and latitude** of the source address used to create this variant.
- Submit through the validator pipeline for logging.

These UAVs form the first layer of **challenging test cases** that probe LDS v0's detection limits.

Validator Role

Validators enforce the scoring and emission logic and monitor subnet health.

- **Ranking:** Rank–quality fused scoring system applied per query.
- **Reward Distribution:** 60 % of emissions allocated through KAV quality checks.
- **Burn:** 40 % of emissions are burned to stabilize early cycles
- **Evaluation:** Check address validity, geocode accuracy, and explanation coherence.
- **Freeze Window:** At the end of t_1 (Execution Stage), UAV submissions will be paused — no new unknown attack vectors will be accepted beyond this point. This ensures that all received UAVs are processed consistently for post-validation, knowledge base updates, and LDS v1 retraining in the following cycle.
- **Preview of Cycle 2:** The validated UAVs and ranking data from Cycle 1 will seed the next round of calibration. Cycle 2 will introduce a **reputation-weighted blended reward system** and refine emission scaling (30% burn, 30% allocated to KAVs during Cycle 2, and 40 % reserved for top performers from Cycle 1).

YANEZ Role

- **Receive and log** all UAVs sent by miners for offline post-validation and inclusion in the known_uav_corpus for **LDS v1 (beta)**.
- *Manually* review UAVs to confirm novelty and real-world plausibility.
- **Retrain** and **release LDS v1 (beta)** after Cycle 1 concludes, after the freeze Window.
- **Publish** miner leaderboard, reward summaries, and performance metrics for transparency.

Outcome

By the end of Cycle 1:

- A **verified seed dataset** of real and variant addresses will be established.
- A **new detector (LDS v1)** will be released — upgraded from the baseline LDS v0 with integrated geocoding validation, normalization logic, and miner-informed address variation mappings.
- **Update knowledge base:** add confirmed UAVs to the known_uav_corpus.
- **Retrain & release LDS v1 (beta)** to the sandbox for Cycle 2.
- The subnet will have a **practiced, high-signal miner community** prepared for Cycle 2's learning-driven refinement and reputation weighting phase.

Timeline

Sandbox Start: *November 4*

Execution Phase: *November 21 – December 16*

Cycle 1 is not just a test of emissions and scoring — it's the foundation for a **self-regulating ecosystem** that learns and improves with every iteration.

This phase sets the tone for how YANEZ evolves:

- **Quality, originality, and diversity** of miner data drive measurable impact.
- Miners **evolve with the YANEZ roadmap**, contributing directly to each new detector release.
- **Reputation compounds** across cycles, rewarding consistency and genuine innovation.
- Poor or repetitive work **phases out naturally**, allowing stronger contributions to lead.

We're not rewarding activity — we're rewarding intelligence.

The subnet learns who contributes meaningfully and aligns incentives around that.

As Cycle 1 progresses, details of **Cycle 2** — including its objectives, expanded UAV validation process, and reputation-weighted reward updates — will be shared with the community during the execution phase.

Phase 2: Threat Scenario Query Execution & Initial Deployment

Phase 2 introduces the Threat Scenario Query System with an expanded scope, enabling validators to reward miners for generating execution vectors that target not only name-based evasion tactics but also those involving date of birth and address information.

In addition to phonetic and orthographic similarity constraints (now extended to include non-Latin names), queries may specify a set of explicit transformation rules that a subset of the generated variations must comply with. These rules could include, for example, character doubling, specific typos, homoglyph substitutions, language-specific patterns, or manipulations of date and address formats.

How to query a subnet miner?

Query Structure

1- Seed Identities Are Predefined

   a. The validator generates and provides a list of seed Identities using internal logic (e.g., via faker, US sanction list).
   b. Miners are **not** responsible for creating or selecting seed identities.
   c. Identities are currently made of 3 different things: (Name, address, DOB)

2- Dynamic Name Variation Generation (M Variations per Name)

   Each seed name requires M variations, which miners must generate based on linguistic transformation rules rather than producing random outputs.

   o Phonetic (Sound Similarity) Transformations
      ▪ The request specifies what percentage of M variations should exhibit phonetic similarity to the original seed name.
      ▪ Miners must create phonetic similar variations while following predefined similarity levels: Light, Medium, Far
   o Orthographic Transformations

- The request specifies what percentage of M variations should introduce orthographic modifications (e.g., script changes or spelling variations).
- Miners must apply script-based transformations based on similarity level such as edit-based metrics with predefined similarity levels, Light, Medium, Far
  - Rule-based:
    - The query may contain a rule-based section specifying required transformation rules and a minimum coverage percentage for these rules.
  - Non-Latin Names:
    - When seed names are in non-Latin scripts, miners must first transliterate the name to Latin script before applying phonetic and orthographic transformations.
    - Validators will also use an LLM to generate and evaluate transformed names, grading miners based on the Latin-transliterated form.

3- Dynamic Address Variation Generation (M Variations per Name). Each seed address requires M variations, which miners must generate unique, realistic addresses within that country/city for each variation. Each address must be distinct, plausible, and match the requested geographic context.
4- Dynamic Date of Birth (DOB) Variation Generation (M Variations per Name). Each seed DOB requires M variations, which miners must generate multiple DOB variations covering at least one example in each of the following categories:
  - ± 1 day
  - ± 3 days
  - ± 30 days
  - ± 90 days
  - ± 365 days
  - Year + Month only (no day specified)

How This Prevents Simple Lookups and Functions Like a Hash

Our approach transforms straightforward name generation queries into complex linguistic and computational challenges akin to a "hash function" in cryptography. This ensures that resolving a query requires advanced NLP models rather than simple database lookups, preventing brute-force retrieval and ensuring proper generative synthesis.

1- Dynamic Name Seed Encoding
  a. Instead of directly providing names, Yanez applies attribute-driven transformations, ensuring that each query is unique, computationally complex, and resistant to lookup-based retrieval.
  b. Encoded Query Structure: Rather than explicitly listing names, queries define abstract constraints (e.g., language distribution, variation ratios).

c. Contextual Name Synthesis: Miners must generate names dynamically while preserving linguistic characteristics, ensuring the results are not retrievable from a pre-existing dataset.

2- Layered Variation Generation
a. Each query defines M variations per name, but instead of following direct mappings, miners must derive phonetic and orthographic transformations algorithmically based on similarity constraints.
b. Non-Linear Similarity Scoring: Variations are generated across multiple levels of similarity (Light, Medium, Far), requiring miners to compute variations dynamically rather than applying simple letter substitutions.
c. Adaptive NLP Models: The system forces miners to apply transformation rules rather than relying on pre-stored lists.

3- Attribute-Based Constraint Encoding
a. Instead of explicitly stating how many phonetic or orthographic variations must be generated, the system encodes constraints into abstract feature representations that miners must decode before producing variations.

Why This Matters

By enforcing complex, NLP-driven transformations, Yanez ensures that subnet miners cannot rely on existing name databases, making the system:

- More Secure – Prevents direct retrieval from known datasets, reducing exploitation risks.
- More Adaptable – Enables context-aware name generation, supporting multiple languages and cultural variations.
- More Scalable – Works across different identity frameworks (AML testing, AI research, gaming, VR, decentralized ID verification).

Query Examples:

- "Query: Generate 15 variations of margot noël (latin), ensuring phonetic similarity (100% Medium) and orthographic similarity (10% Light, 30% Medium, 60% Far). Approximately 58% of the total 15 variations should follow these rule-based transformations: Additionally, generate variations that: Swap random adjacent letters. The following address is the seed country/city to generate address variations for: Saint Pierre et Miquelon. Generate unique real addresses within the specified country/city for each variation. The following date of birth is the seed DOB to generate variations for: 1977-04-23. [ADDITIONAL CONTEXT]: - Address variations should be realistic addresses within the specified country/city - DOB variations ATLEAST one in each category (±1 day, ±3 days, ±30 days, ±90 days, ±365 days, year+month only) - For year+month, generate the exact DOB without day - Each variation must have a different, realistic address and DOB

- "Query: Generate 11 variations of maxi maestre (latin), ensuring phonetic similarity (10% Light, 30% Medium, 60% Far) and orthographic similarity (30% Light, 40% Medium, 30% Far). Approximately 41% of the total 11 variations should follow these rule-based transformations: Additionally, generate variations that: Replace double letters with a single letter. The following address is the seed country/city to generate address variations for: Venezuela. Generate unique real addresses within the specified country/city for each variation. The following date of birth is the seed DOB to generate variations for: 1940-04-12. [ADDITIONAL CONTEXT]: - Address variations should be realistic addresses within the specified country/city - DOB variations ATLEAST one in each category (±1 day, ±3 days, ±30 days, ±90 days, ±365 days, year+month only) - For year+month, generate the exact DOB without day - Each variation must have a different, realistic address and DOB"

How should a subnet validator evaluate the subnet miner response?

- Validate Seed Names & Query Compliance:
    - Ensure that the seed names in the miner's response exactly match those provided in the query, or, if names are generated, verify that they conform to the requested language/script distribution and adhere to script constraints (e.g., Arabic, Cyrillic, Latin) as specified.
- Evaluate Attribute-Based and Rule-Based Constraints:
    - Compute the phonetic distance and edit distance (e.g., Levenshtein) for each variation relative to its seed name.
    - Confirm that the actual distribution of similarity levels (Light, Medium, Far) in the generated variations matches the requested breakdown in the query.
    - If the query includes explicit rule-based requirements, verify that at least the required percentage of variations match one of the requested transformation rules (e.g., double consonant, homoglyph substitution), and assess coverage across all specified rules.
- Score the Synthetic Names:
    - Assess relevance to the query, based on requested attribute and rule-based percentages and transformation rules.
    - Evaluate novelty: check if generated variations are sufficiently different from previously generated names for the same seed (future phase).
    - Identify potential issues: determine if any generated variation closely resembles a name in the exclusion database.
- Enforce Diversity and Constraint Compliance:
    - Reject responses that fail to meet phonetic, orthographic, or rule-based constraints, overuse trivial modifications, or lack required diversity in transformations.
- Data Storage and Logging:
    - Save all validator results, including queries, responses, and computed rewards, locally as structured JSON files for traceability and auditability.

- o After validation and signing, upload results as JSON payloads to a centralized, authenticated backend API endpoint for aggregation, benchmarking, and further processing.
  - o Log detailed metrics and rewards to Weights & Biases (wandb) for experiment tracking, analysis, and reproducibility.
- Date of Birth (DOB) Evaluation:
  - o Ensure that each miner response includes at least one DOB variation in each of the required categories:
    - ±1 day
    - ±3 days
    - ±30 days
    - ±90 days
    - ±365 days
    - Year + Month only (no day specified)

  - o Verify that the generated DOBs are valid calendar dates and correctly follow the specified offsets relative to the seed DOB.

- Address Evaluation:
  - o Ensure that all generated address variations look like syntactically valid addresses.
  - o Verify that the generated addresses share the same country and city context as the seed address provided in the query.
  - o Check that each address is unique and realistically formatted.
  - o Perform a country/city containment check: ensure that each generated city actually exists inside the specified country.
  - o Randomly select one or more generated addresses and validate them via an external address verification API to confirm plausibility.

How should a subnet miner be rewarded for its response?

Miners in the Yanez MIID subnet are rewarded based on how well their generated **execution vectors** (e.g., name variations) match the specified threat scenario constraints. The reward mechanism evaluates each response across multiple quality dimensions and penalizes incomplete or misaligned outputs.

Reward Calculation Pipeline

For each miner, the final reward is calculated as:

$$\text{Final Reward} = \text{Average Quality Score Across Identities } (Q) \times \text{Completeness Penalty } (P)$$

* Completeness Penalty (P) a multiplier, not a penalty in the strict sense; 1 means no penalty, lower values apply a penalty.

Quality Score per Identity (Q_i)

$$Q = w_{\text{name}} \times \left(\frac{1}{N} \sum_{i=1}^{N} Q_i \right) + w_{\text{dob}} \times Q_{\text{dob}} + w_{\text{address}} \times Q_{\text{address}}$$

$$w_{name} = 0.7, \ w_{dob} = 0.1, \ w_{address} = 0.2$$

Where:

- w_{name}: name compliance weight
- Q_{dob}: Date of birth score
- w_{dob}: Date of birth weight
- $Q_{address}$: Address score
- $w_{address}$: Address weight

For each **seed name**, a "variation quality" is computed:

$$Q_i = (1 - w_{\text{rule}}) \times Q_{\text{base}} + w_{\text{rule}} \times Q_{\text{rule}}$$

Where:

- w_{rule} :Rule compliance weight
- Q_{base}: Base quality score (from similarity, count, uniqueness, length)
- Q_{rule} : Rule compliance score

Base Quality Score per Name (Q_{base}):

The quality of variations generated for each seed name is computed using a weighted sum of four factors (Latin and non-Latin):

$$Q_{base} = w_s \cdot S + w_c \cdot C + w_u \cdot U + w_l \cdot L$$

Where:

- S: Combined **similarity score** (phonetic + orthographic)
- C: **Count score**, rewarding the correct number of variations
- U: **Uniqueness score**, penalizing duplicate outputs
- L: **Length score**, rewarding reasonable variation in lengths

With default weights:

$$w_s = 0.6, w_c = 0.15, w_u = 0.1, w_l = 0.15$$

Quality Score Components

A) Similarity Score (S)

This measures how closely the generated variations align with the phonetic and orthographic similarity constraints specified in the query.

$$S = \frac{S_{\text{phonetic}} + S_{\text{orthographic}}}{2}$$

Similarity is computed using:

• **Phonetic**: via Soundex + Levenshtein

• **Orthographic**: via normalized Levenshtein distance

Each sub-score is calculated using **level-based binning**, based on how well the distribution of generated variations matches the target distribution over similarity levels (Light, Medium, Far).

$$S_t = \sum_{l \in \{\text{Light, Medium, Far}\}} w_l \cdot \min\left(\frac{\text{actual}_l}{\text{target}_l}, 1.0\right)$$

Where:

- w_l: Target weight for level l
- actual_l: Number of variations in level l
- $\text{target}_l = w_l \cdot |V|$: Expected count in level l

Phonetic Ranges:
Light: 0.8–1.0, Medium: 0.6–0.8, Far: 0.3–0.6

Orthographic Ranges:
Light: 0.7–1.0, Medium: 0.5–0.7, Far: 0.2–0.5

The similarity distribution from the query (e.g., 50% Medium, 30% Far, 20% Light) is compared against actual distribution from the miner's output.

B) Count Score (C)

Measures whether the number of returned variations V is close to the expected count E.

A tolerance margin $\varepsilon = 0.2 (20\%)$ is used.

$$C = \begin{cases} 1.0, & if \ |V - E| \leq 0.2 \cdot E \\ 1.0 - \min\left(1.0, \dfrac{|V - E|}{E}\right), & Otherwise \end{cases}$$

C) Uniqueness Score (U)

Rewards miners for returning distinct variations and penalizes duplication.

$$U = \frac{\text{Number of Unique Variations}}{\text{Total Variations}}$$

D) Length Score (L)

Measures how proportionally long each variation is compared to its seed. Extremely short or long variations reduce the score.

$$L_v = \min\left(\frac{|v|}{|o|}, \frac{|o|}{|v|}\right)$$

$$L = \frac{1}{|V|}\sum_{v \in V} L_v$$

Where:

• $|v|$: Length of the variation

• $|o|$: Length of the original seed name

A high Length Score encourages **natural-looking, proportionally adjusted** variations that are realistic and useful for testing real-world screening systems. It helps filter out edge cases and improves dataset quality.

Combining First and Last Name Parts

When a name has multiple parts (e.g., first and last), each part gets its own base score; then they are combined using weights:

$$Q_{\text{base}} = w_{\text{first}} \cdot Q_{\text{first}} + w_{\text{last}} \cdot Q_{\text{last}}$$

Each name part's weight is set proportional to its character length within the full name, then adjusted by up to ±20% random variation (with deterministic seeding) and finally normalized so all weights sum to one. In this version, we will utilize only two weights for the first name and last name.

Rule Compliance Score (Q_{rule}):

The Rule Compliance Score, Q_{rule}, is the product of a quantity score, q , measuring how many unique variations meet the rule-based requirements compared to the target percentage, and a diversity factor, d, the proportion of requested rules that are covered by at least one variation. This structure ensures that the miner's output not only meets the minimum requested compliance but also distributes compliance across the full set of target rules, maximizing both coverage and diversity.

$$Q_{\text{rule}} = q \times d$$

$$q = \begin{cases} 0 & if\ V_{comp}\ =\ 0 \\ \dfrac{V_{comp}}{E} & if\ V_{comp}\ \le\ E \\ 1.5\ -\ 0.5\ \dfrac{V_{comp}}{E} & if\ V_{comp}\ >\ E \end{cases}$$

Where:

- V_{comp}: unique variations matching any target rule
- E : expected compliant count

$$d = \frac{R_{\text{met}}}{R}$$

Where:

- R_{met} : number of distinct target rules satisfied by at least one variation
- R : total number of target rules

Completeness Penalty (P$_{\text{pre}}$)

$$\text{Penalty} = \min\big(0.9, P_{\text{extra}} + P_{\text{missing}}\big)$$

The penalty is **additive penalty** logic, capped at 0.9 total.

Completeness Penalty (P) = max(0.1, 1.0- Penalty)

A **minimum floor of 10%** of the total reward is maintained to avoid zeroing out valuable partial submissions.

Miners are penalized for:

- **Missing Seed Names**:

$$P_{\text{missing}} = \min\left(0.9, 0.2 \cdot N_{\text{missing}}\right)$$

  - 20% penalty per missing name
  - Capped at 90% total penalty
- **Extra Names:**

$$P_{\text{extra-name}} = \min(0.7, 0.1 \cdot N_{\text{extra}})$$

$$N_{\text{extra-varitaions}} = \left(N_{\exp ected} \cdot 1.2\right) - \left(N_{amount-send}\right)$$

$$P_{\text{extra-variation}} = 0.05 \cdot (N_{\text{extra-varitaions}})$$

$$P_{\text{extra-dup}} = 0.05 \cdot N_{\text{extra-dup}}$$

$$P_{\text{extra = min}} = \min\left(1.0, P_{extra-name} + P_{extra-variation} + P_{exrta-dup}\right)$$

  - 10% penalty per unexpected name
  - 5% per extra variation with a grace of 20%
  - 5% per duplicate name
  - Capped at 100% total penalty

Non-Latin names:

They will have the same scoring as the above. The only difference is that there will be an LLM that transliterates the names and then uses the transliteration as the seed name for the scoring. It will be a part of the "variation quality" average.

Date of birth (Q_{dob}):

- Makes sure there is generated birthday that fails in each category (1,3,30,90,365)
- One more category where its just month and year no date
- S_{DOB} = Found ranges / total ranges

Address ($Q_{address}$):

- Make sure it looks like an address by have some key features:
  - Has letters
  - Has numbers
  - 10 < length < 200
- Make sure the address is in the right location based on seed address
- Making sure that the city in the generated address is in the country mentioned
- Api randomly chooses one generated address and makes sure its valid

- $S_{address}$ = 1
- If Api times out: $S_{address}$ = 0.3
- If any fail: $S_{address}$ = 0.0

Response Duplication Penalties ($P_{response\_duplication}$)

$$P_{response\_duplication} = min(1.0, P_{collusion} + P_{duplication} + P_{signature\_copy} + P_{special\_chars})$$

The $P_{response\_duplication}$ is made up of Penalty collusion, Penalty duplication, Penalty signature_copy and Penalty special_chars and is cap at 1.

Miners are penalized for:

- **Collusion:**
  - At least 5 miners in the same reward bucket (same score).
  - Score < 0.95 (not a perfect score, so collusion is suspected rather than legitimate).
  - 75% penalty per collusion detected
  - Capped at 100% total penalty
- **Duplication**:
  - Based on miners responses

- o   Overlap coefficient > 0.95 or Jaccard coefficient > 0.90
- o   Capped at 50% total penalty
- o   Also checks for address that are the same
- **Signature Copy**:
  - o   Make a hash out of miners entire response if hash is the same then penalize both miners
  - o   Flat rate 80% total penalty

- **Special Character Abuse**:

$$Var\_Ratio = (\# \text{ variations with special chars} / \text{total } \# \text{ of variations})$$

$$special\_char\_penalty = min(1.0, (Var\_Ratio - 0.5)/0.5)$$

- o   too many variations containing special characters (punctuation, symbols)
- o   Only penalize if more than 50% of the variations have this abuse
- o   Ratio of variations w/ special characters vs total variations divide by 0.5
- o   Capped at 1.0

Overall total penalty ($P_{overall}$)

(Note: this is not used in calculating the reward this is made so that we can see what the total penalties are. In the reward calculation we keep the sperate. They are added at 2 different times)

$$P_{overall} = (1 - P_{Pre}) * (1 - P_{response\_duplication})$$

Post Response Duplication Penalty ($P_{response\_duplication}$):

<u>Final Reward</u> = Current reward * $(1 - P_{response\_duplication})$

- This is done after all the rewards of all the miners' rewards are calculated
- We do a post penalty check to see if there are any response duplication penalties