Thesis Research Addendum:
A ”Robust Model” (v2) for PESV Traffic Classification

Undergraduate Thesis - Research Log

November 2, 2025

1 Problem Analysis and v2 Strategy

The initial analysis detailed in `dokumentasi.pdf` established the Path-Embedding Signature Vector, $\Sigma = (\alpha, \beta, \gamma)$. The subsequent classification in `PESV.ipynb` revealed a significant performance discrepancy:

- **Binary (VPN/Non-VPN):** High accuracy ($\sim$86%).
- **Category (6-class):** Moderate accuracy ($\sim$70%).
- **Application (16-class):** Low accuracy ($\sim$62%).

A detailed review identified three primary root causes for the low multi-class performance:

1. **Severe Class Imbalance:** The full 13,450-flow dataset was dominated by applications like `Skype` (4704) and `Email` (1901), while others like `AIM Chat` (46) were statistically insignificant, biasing the model.
2. **”One-to-Many” Mapping:** Ambigious applications (e.g., `Skype`) were labeled in three different categories (`Chat`, `VoIP`, `File Transfer`). This forced the classifier to learn that three fundamentally different traffic patterns belonged to the *same* application class, creating a ”confused” model.
3. **Weak Feature Components:** The ablation study in `PESV.ipynb` showed that adding the β component (IAT Wasserstein Distance) *decreased* accuracy. This suggests the β and γ components, which rely on comparison to a ”blurry average” prototype, were acting as noise rather than a clear signal.

Based on this analysis, a new ”Robust Model” (v2) workflow was designed to address all three issues.

2 Methodology for ”Robust Model” (v2)

The v2 workflow consists of two main parts: (1) refining the dataset to create a balanced, logical classification problem, and (2) engineering a more powerful feature vector.

2.1 Dataset Refinement (v2-final_flows)

Instead of using all 13,450 flows, a new dataset was curated to satisfy two constraints: simplify the problem to 6 applications and ensure all 6 categories were still represented. This provides a balanced and comparable dataset for all three classification tasks.

The selected applications were:

- `Skype` (as the multi-category example)

- `Email` (for the `Email` category)
- `SCP` (as a strong `File Transfer` example)
- `VOIPBuster` (for the `VoIP` category)
- `YouTube` (as the largest `Streaming` example)
- `BitTorrent` (for the `P2P` category)

A filtering script (`filter_flows.py`) was used to copy these flows, resulting in the `v2-final_flows` directory, containing 10,284 flows. The new dataset composition is summarized in Table 1.

Table 1: Composition of the `v2-final_flows` Dataset (10,284 Flows)

Application	Category	Count	% of Total
Skype	(Chat, VoIP, File Transfer)	4704	45.7%
Email	Email	1901	18.5%
SCP	File Transfer	1741	16.9%
VOIPBuster	VoIP	1196	11.6%
YouTube	Streaming	379	3.7%
BitTorrent	P2P	363	3.5%

2.2 Redefining the Feature Vector ($\Sigma_{v2} = (\alpha, \delta, \gamma')$)

The β and γ components were deprecated and replaced with more descriptive statistical features, resulting in a new "Robust Model" vector.

2.2.1 Component α (Alpha): Learned Sequence Representation

This component is **unchanged** from the original methodology. It provides a 32-dimensional latent vector from an LSTM Autoencoder trained on the first 128 packet sizes (with direction) of a flow.

- **Script:** `generate_alpha_v2_fixed.py`
- **Note:** The `_fixed` version corrected a `NaN` bug by clipping inputs and using a `tanh` activation, stabilizing model training.

2.2.2 Component δ (Delta): Total Flow Statistical Profile

This component **replaces** β. Instead of comparing a flow's IAT histogram to a prototype, δ is a direct statistical summary of the *entire* flow, calculated bidirectionally. This provides a raw, objective profile of the flow's behavior.

- **Script:** `generate_delta_v2.py`
- **Key Features (39 total):**
  - `flow_duration`, `total_packets`, `total_volume`
  - Bidirectional stats (c2s and s2c) for Packet Sizes (mean, std, min, max, etc.)
  - Bidirectional stats (c2s and s2c) for Inter-Arrival Times (mean, std, min, max, etc.)

2.2.3 Component γ' (Gamma-Prime): Expanded Burst Profile

This component **replaces** γ. The original γ only used 4 averages. The new γ' captures a full statistical profile of *all bursts* in a flow (using the original 1.0s idle time definition).

- **Script:** `generate_gamma_prime_v2.py`
- **Key Features (37 total):**
  - `total_bursts`
  - Full stats (mean, std, min, max, median, etc.) for:
    - Packets per burst
    - Volume per burst (bytes)
    - Duration per burst (seconds)
    - Idle time between bursts (seconds)

3 Final Assembly and Classification Pipeline

3.1 Assembly Script (`assemble_pesv_v2.py`)

A major improvement in the v2 workflow is that all three component scripts (`alpha`, `delta`, `gamma_prime`) ran on the same 10,284 files and used identical skipping/labeling logic.

This resulted in three component CSVs with perfectly aligned rows. The final assembly script no longer needs a complex, key-based merge (as in `dokumentasi.pdf`). Instead, it performs a simple and fast `pd.concat` to combine the columns. The final output is `final_PESV_dataset_v2.csv`.

3.2 Classification Script (`PESV_Classifier_v2.py`)

The final classification script was redesigned to incorporate machine learning best practices to handle the new, high-dimensionality feature set and known class imbalance.

Three key improvements were implemented:

1. **StandardScaler:** The δ and γ' components introduce features with vastly different scales (e.g., `total_volume` vs. `c2s_iat_mean`). A `StandardScaler` is used to normalize all features, ensuring they contribute equally to the model.
2. **Pipeline:** The `StandardScaler` and the classifier are combined into a `sklearn.pipeline.Pipeline` to prevent data leakage from the test set.
3. **Class Weighting:** To combat the known class imbalance (see Table 1), the `RandomForestClassifier` is instantiated with `class_weight='balanced'`. This automatically calculates and applies weights inversely proportional to class frequency, forcing the model to pay more attention to minority classes like `YouTube` and `BitTorrent`.

4 Conclusion

The "Robust Model" (v2) workflow addresses the key weaknesses of the initial approach. By creating a more balanced and logically sound dataset, replacing weak prototype-based features (β, γ) with strong statistical profiles (δ, γ'), and using a classification pipeline that handles feature scaling and class imbalance, this new methodology is expected to yield significantly higher and more reliable accuracy for all three classification tasks.