Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Statistical Comparison of SCQ-SSA, PSO and ARO for Optimizing Bi-LSTM-Based Network Intrusion Detection

Mohd. Asgar Ali

Aliasgarali05@gmail.com

Lovely Professional University

Research Article

Keywords: Network Intrusion Detection, Cloud-Oriented Environments, Salp Swarm Algorithm, Chaos Theory, Sine-Cosine Mechanism, Quantum Computation, Particle Swarm Optimization (PSO), Artificial Rabbits Optimization (ARO), Bi-LSTM-Deep Residual Network, Optimized Weighted Network Security Index (OWNSI), CIC-IDS2017, Ablation Study, Statistical Significance, Effect Size

Posted Date: October 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11286680/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

A new trend in network intrusion detection in cloud environments is the use of deep sequence models that rely on hyperparameter optimization to achieve a strong performance. In this study, Particle Swarm Optimization (PSO) and Artificial Rabbits Optimization (ARO) are used as the baselines, and a controlled benchmark of the Chaos-Sine-Cosine-Quantum Salp Swarm Algorithm (SCQ-SSA) for optimizing an OWNSI-based Bidirectional LSTM Deep Residual Network (Bi-LSTM-DRN) is presented. The real CIC-IDS2017 dataset (which contains all 8 official traffic-capture files and 2,830,743 labelled flows) is used for the experiments. The search space, fitness function, population size (10), iterations (10) and fixed held-out test set are the same for each optimizer and the evaluations are repeated 20 times for each. SCQ-SSA achieved the highest mean accuracy (85.29%), recall (88.36%), F1-score (70.53%), and ROC-AUC (0.934), with ARO close behind. The differences, however, were not statistically significant (Friedman $p \geq 0.52$) and none of the pairwise Cohen's d values exceeded 0.26, which is taken as the threshold for a significant practical effect. Overall the results of the ablation indicate that the contribution of the bidirectionality and residual connections was more consistent than the contribution of the OWNSI weighting. Results from a temporal evaluation using an unseen Friday traffic showed a drop in accuracy of 9–13 percentage points, indicating there is some limited cross-day generalization. The baseline model that performed best was Random Forest with an accuracy of 98.27% and an F1 score of 95.52% which is better than all the deep-learning variants. Therefore, the main strength of the study is not the classification performance, but the reproducible protocol for evaluating optimizers.So, the major strength of the study is the reproducible protocol for optimizing the evaluation of the optimizers rather than the state-of-the-art classification performance.

1. Introduction

Cloud computing's distributed, multi-tenant architecture continues to widen the attack surface available to denial-of-service, brute-force, web, infiltration and botnet adversaries [1][2]. Intrusion Detection Systems (IDS) trained on network-flow benchmarks such as CIC-IDS2017 are frequently described in the literature as "cloud IDS" once evaluated on such data; this paper avoids that shorthand. The experiments reported here use the CIC-IDS2017 network-flow dataset, which contains genuine enterprise network traffic captured on a testbed rather than traffic collected from a live cloud deployment or container runtime. Accordingly, this paper is scoped and titled as network intrusion detection for cloud-oriented environments: the detection task, feature set and threat model are motivated by and applicable to cloud-hosted infrastructure, but no claim is made that the system has been validated against traffic generated by an actual cloud testbed or production cloud deployment. This distinction is maintained consistently through the methodology, results and conclusion below.

Within this scope, the choice of detection architecture and the choice of the meta-heuristic optimizer used to tune it both materially affect operational outcomes. Classical machine-learning approaches such as Support Vector Machines (SVM) and Random Forests (RF) remain strong, computationally inexpensive baselines, while deep sequence models such as Long Short-Term Memory (LSTM) networks

are increasingly favoured for their capacity to model dependencies across engineered flow features [3] [4]. Bidirectional LSTM architectures augmented with residual connections (Bi-LSTM-DRN) can further improve gradient flow and representational depth, and pairing such an architecture with a dynamically-weighted composite of security-relevant flow features has been proposed to sharpen detection sensitivity.

The quality of the resulting feature weights and Bi-LSTM-DRN hyperparameters is, in turn, governed by whichever meta-heuristic optimizer is used to search that space. Particle Swarm Optimization (PSO) remains a simple, computationally efficient default, and Artificial Rabbits Optimization (ARO) -- modelled on the detour-foraging and random-hiding survival strategies of rabbits -- has demonstrated a favourable exploration-exploitation balance on engineering optimization benchmarks [5]. Independently, Jia et al. [6] recently proposed a Chaos-Sine-Cosine-Quantum variant of the Salp Swarm Algorithm (SCQ-SSA), which combines circle-map chaotic population initialization, a sine-cosine local-search perturbation of the leading salp, and a quantum-rotation-gate refinement mechanism, and demonstrated that SCQ-SSA outperforms seven well-known algorithms on 1000-dimensional benchmark functions, seven other Salp Swarm Algorithm (SSA) variants, six advanced hybrid optimizers, and three classical engineering design problems (tubular column, tension/compression spring and pressure vessel design). SCQ-SSA has not previously been applied to deep-learning-based intrusion detection, nor benchmarked against PSO or ARO within an identical pipeline.

Separately, recent work on host-based intrusion detection for containerized cloud environments has explored converting system-call logs into image representations for convolutional neural network (CNN) classification. Joraviya et al. [7] proposed DL-HIDS, a deep-CNN host IDS that transforms correlated system-call features and metadata into grayscale images and validates the approach on the Leipzig Intrusion Detection Data Set 2019 (LIDDS-2019), reporting improvements over prior vector-to-image and sequence-comparison approaches. That work underscores a broader pattern: containerized host-level and network-flow-level intrusion detection research streams both increasingly rely on deep architectures whose performance is sensitive to a chosen -- but rarely benchmarked -- optimizer or feature-engineering strategy. This paper addresses the network-flow-level side of that gap and treats DL-HIDS as a motivating example of the wider need for feature-engineering and optimization rigor in this research area, rather than as a dataset used in the present study.

This paper optimizes an OWNSI-driven Bi-LSTM-DRN network intrusion detector with SCQ-SSA and -- rather than reporting SCQ-SSA in isolation, as is common practice when a new optimizer is first applied to a new domain -- benchmarks it directly against PSO and ARO under an identical search space, fitness function and, critically, an identical fixed test set, all evaluated on the genuine CIC-IDS2017 traffic. Beyond the original study, this revised version increases the number of independent repetitions per optimizer from five to twenty, expands the optimizer search budget, adds a six-configuration ablation study isolating the contribution of bidirectionality, residual connections and feature weighting, adds a temporal generalization test on entirely unseen traffic, reports effect sizes and confidence intervals alongside p-values, and reports the runtime and computational cost of each optimizer. Section 2 reviews

related work. Section 3 details the proposed methodology. Section 4 reports the experimental results. Section 5 concludes.

2. Related Work

Meta-heuristic optimization has increasingly been used to tune hyperparameters and feature weights in deep-learning-based IDS pipelines. Gong et al. [8] proposed a quantum-inspired PSO variant with a diversity-migration strategy to counter premature convergence; Tijjani et al. [9] enhanced PSO with a modified position-update rule for feature selection. PSO's simplicity and well-understood convergence behaviour keep it a strong default choice. Mirjalili et al. [10] introduced the original Salp Swarm Algorithm, subsequently extended by numerous variants including Gaussian, Cauchy and Levy-flight perturbations [11], weight-factor and adaptive-mutation schemes [12], craziness-and-adaptive variants [13], and scale-free informed-follower weighting [14]. Wang et al. [5] proposed Artificial Rabbits Optimization, modelled on detour-foraging and random-hiding behaviour. Jia et al. [6] proposed SCQ-SSA, combining circle-map chaos, sine-cosine local search and quantum-rotation refinement, and validated it exclusively on generic benchmark functions and classical engineering design problems (tubular column, spring, pressure vessel design); Section 3.4 applies SCQ-SSA to deep-learning-based intrusion detection for the first time and benchmarks it against PSO and ARO under controlled, repeated-run conditions.

On the network-IDS side, Silambarasan et al. [15] combined an Adaptive Random Search Optimization model with a Bi-LSTM classifier; Goyal et al. [16] proposed a dual-attention bidirectional GRU network; Vishnupriya et al. [17] combined a dilated transformer with Bi-LSTM and GRU layers for wearable-sensor health monitoring; Yan et al. [18] combined bidirectional convolutional LSTM with deep recurrent convolutional Q-networks for industrial IoT intrusion detection and mitigation. On the host-based side, Park et al. [19] converted system-call vectors into grayscale images for Siamese-network anomaly detection, and Joraviya et al. [7] (DL-HIDS) subsequently improved on that approach with a purpose-built deep CNN over system-call images validated on LIDDS-2019, reporting gains in accuracy, precision, recall, F1-score and false-positive rate. Both research directions -- network-flow-level and host-level -- increasingly depend on deep architectures whose reported performance is sensitive to feature engineering and hyperparameter search choices that are rarely benchmarked against alternatives under identical, repeated-run, statistically-tested conditions; this is the gap the present study addresses for the network-flow-level, optimizer-comparison side of that literature.

3. Proposed System Model

The proposed system optimizes an OWNSI-driven Bi-LSTM-DRN network intrusion detector using SCQ-SSA, benchmarked against PSO- and ARO-optimized variants of the identical architecture. The pipeline begins with acquisition and cleaning of genuine CIC-IDS2017 flow data, from which five OWNSI features are extracted. The OWNSI module assigns importance weights to these features, which are fed into the Bi-LSTM-DRN classifier. The weight vector and the Bi-LSTM-DRN hyperparameters are jointly optimized by each of the three meta-heuristic optimizers in turn, using an identical search space and multi-metric

fitness function, and the resulting optimized models are evaluated head-to-head, with an additional ablation study and temporal generalization test, in Section 4.

3.1. Dataset Description

This study uses the genuine CIC-IDS2017 Network Intrusion Dataset, published by the Canadian Institute for Cybersecurity, University of New Brunswick [20], comprising all eight official CICFlowMeter-generated traffic-capture files: Monday (benign baseline), Tuesday (FTP-Patator and SSH-Patator brute-force), Wednesday (DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris, Heartbleed), Thursday morning (Web Attack: Brute Force, SQL Injection, XSS) and afternoon (Infiltration), and Friday morning (Bot) and afternoon (PortScan, DDoS). Concatenating all eight files yields 2,830,743 labelled flow records: 2,273,097 benign (80.30%) and 557,646 attack (19.70%), spread across fourteen distinct attack labels.

Each of the five OWNSI features is populated from a genuine CICFlowMeter column: Packet Size from Average Packet Size, Connection Duration from Flow Duration, Source Bytes from Total Length of Fwd Packets, and Destination Bytes from Total Length of Bwd Packets. The supplied CSV export omits a raw Protocol field (a common omission in redistributed CIC-IDS2017 exports), so Destination Port is used as a documented proxy for Protocol Type.

3.2. Sampling Analysis: From 2,830,743 Flows to a 9,000-Flow Working Sample

For computational tractability within the single-CPU execution environment used for this study, a stratified working sample of 9,000 flows was drawn from the full 2,830,743-flow pool: 7,227 benign flows (80.30%, matching the population proportion) and 1,773 attack flows, allocated across all fourteen attack labels using square-root-weighted proportional allocation. This deliberately over-represents very rare classes relative to their true frequency -- Heartbleed is sampled at 27.3% of its available population (3 of 11 flows) and SQL Injection at 23.8% (5 of 21), compared with a 0.21%-0.32% sampling rate for the three most common classes (BENIGN, DoS Hulk, PortScan, DDoS) -- so that rare attack types remain present and evaluable in the working sample rather than being sampled out entirely by chance. Table 1 reports, for every attack label, the original population count and percentage, the sampled count and percentage, and the resulting per-class sampling rate; Table 2 reports the resulting train/validation/test split sizes per label. This artificial, rarity-boosted distribution differs materially from the natural CIC-IDS2017 class distribution and should be borne in mind when interpreting per-class metrics: the working sample is designed to keep rare attack types observable for training and evaluation, not to reproduce the population's true class balance, and reported aggregate metrics reflect performance on this rebalanced-toward-diversity sample rather than on the natural, overwhelmingly-benign traffic distribution.

Table 1
Sampling Analysis -- Original CIC-IDS2017 Population vs. the 9,000-Flow Working Sample, by Attack Label

Label	Original Count	Original %	Sampled Count	Sampled %	Sampling Rate
BENIGN	2,273,097	80.300%	7227	80.300%	0.318%
DoS Hulk	231,073	8.163%	480	5.333%	0.208%
PortScan	158,930	5.614%	398	4.422%	0.250%
DDoS	128,027	4.523%	357	3.967%	0.279%
DoS GoldenEye	10,293	0.364%	101	1.122%	0.981%
FTP-Patator	7,938	0.280%	89	0.989%	1.121%
SSH-Patator	5,897	0.208%	77	0.856%	1.306%
DoS slowloris	5,796	0.205%	76	0.844%	1.311%
DoS Slowhttptest	5,499	0.194%	74	0.822%	1.346%
Bot	1,966	0.069%	44	0.489%	2.238%
Web Attack Brute Force	1,507	0.053%	39	0.433%	2.588%
Web Attack XSS	652	0.023%	25	0.278%	3.834%
Infiltration	36	0.001%	6	0.067%	16.667%
Web Attack Sql Injection	21	0.001%	5	0.056%	23.810%
Heartbleed	11	0.000%	3	0.033%	27.273%

Table 2
Train / Validation / Test Split Sizes by Attack Label (Stratified, Fixed Test Set n = 1,500)

Label	Train	Validation	Test	Total
BENIGN	4819	1204	1204	7227
DoS Hulk	321	84	75	480
PortScan	259	69	70	398
DDoS	242	54	61	357
DoS GoldenEye	69	14	18	101
FTP-Patator	61	15	13	89
SSH-Patator	58	7	12	77
DoS slowloris	44	18	14	76
DoS Slowhttptest	45	13	16	74
Bot	32	7	5	44
Web Attack □ Brute Force	24	7	8	39
Web Attack □ XSS	20	4	1	25
Infiltration	1	2	3	6
Web Attack □ Sql Injection	3	2	0	5
Heartbleed	3	0	0	3

The 1,500-flow test set is carved out once, via stratified sampling on the binary Normal/Attack label, before any model is trained or any optimizer is run, and is reused unchanged for every PSO run, every ARO run, every SCQ-SSA run, every repetition, the ablation study, and every baseline model reported in Section 4. Note that with only 3–11 original flows for Heartbleed and SQL Injection, the stratified split allocates zero test-set instances to some of these extremely rare classes in a given split (e.g., Heartbleed, Table 2); this is an inherent limitation of evaluating rare classes at this sample size and is reported transparently rather than concealed.

3.3. Feature Extraction

Feature extraction isolates the five most informative CICFlowMeter attributes described in Section 3.1 from each raw flow record, forming the input vector X in R^5 subsequently consumed by the OWNSI module and the Bi-LSTM-DRN. The Connection Duration, Source Bytes and Destination Bytes columns are strongly right-skewed, as is typical of genuine network flow data, and are therefore log1p-transformed prior to Min-Max scaling to [0,1], with scaling parameters fit on the training partition only.

3.4. Optimized Weighted Network Security Index (OWNSI)

This module was originally named the "Adaptive Weighted Network Security Index (AWNSI)"; that name is revised here to Optimized Weighted Network Security Index (OWNSI) for precision. The weight vector w = [w_1, ..., w_5] is not adaptive in the sense of changing per-flow, per-batch, or over time during deployment: each of PSO, ARO and SCQ-SSA searches for a single, fixed set of five feature weights during the offline optimization phase (Section 3.6), and this fixed weight vector is then used, unchanged, to scale the five input features for every flow classified by the final, deployed model. The name "optimized" reflects that the weights are the output of a meta-heuristic search over the training/validation data, analogous to any other hyperparameter; it does not imply online adaptation during inference. Eq. (2) below is retained in its general, time-indexed form because it describes the weighting mechanism at the point it is evaluated during optimization (each candidate weight vector is scored by training and validating a full model), but the deployed weight vector itself is static post-optimization.

*OWNSI(w, x) = [sum_{i = 1}^{5} w_i * x_i] / [sum_{i = 1}^{5} w_i] (1)*

where x_i is the normalized value of the i-th feature and w_i in [0.1, 1.0] is the corresponding weight found by optimization. In the Bi-LSTM-DRN (Section 3.5) each feature is scaled by its weight prior to being consumed as a sequence element, which is mathematically equivalent to applying Eq. (1) per-feature rather than collapsing to a single scalar index; both formulations are used interchangeably in this paper to describe the same weighting mechanism.

3.5. Bi-LSTM Deep Residual Network (Bi-LSTM-DRN) and Justification of the Sequence Representation

The five OWNSI-weighted features are processed by a Bi-LSTM-DRN: a bidirectional LSTM layer followed by a residual (skip-connected) dense block. Each of the five weighted features is presented to the LSTM as one timestep in a length-5 sequence with a single input channel per timestep.

This representation requires an explicit methodological caveat. The five OWNSI features -- Packet Size, Protocol Type (Destination Port proxy), Connection Duration, Source Bytes, Destination Bytes -- are feature dimensions of a single flow, not naturally ordered temporal observations collected over time; there is no physical sense in which "Packet Size" precedes "Connection Duration" the way one video frame precedes the next. Consequently, the Bi-LSTM in this architecture is not modelling temporal evolution across the sequence axis; it is being used, as a modelling choice, to let a lightweight recurrent layer learn interactions and dependencies across a fixed, arbitrarily-but-consistently ordered set of feature dimensions, with bidirectionality allowing information to flow between features in both directions along that fixed ordering. This is explicitly distinguished here from the more common use of Bi-LSTM to model genuine temporal sequences (e.g., successive packets within a flow, or successive flows within a session); no claim of temporal-dependency modelling is made for the present architecture. The ablation study in Section 4.4 evaluates whether this inter-feature-dependency framing provides measurable benefit over a residual-only or unidirectional treatment of the same five features, and finds that bidirectionality provides a modest, consistent benefit while the specific feature-weighting mechanism (OWNSI) does not.

$$y = F(x, W) + x \quad (2)$$

where F(x, W) is the two-layer dense transformation applied to the pooled bidirectional LSTM output x, and W denotes its trainable weights; the residual (skip) connection mitigates vanishing-gradient issues and is evaluated independently of bidirectionality and feature weighting in the Section 4.4 ablation.

3.6. Experimental Protocol, Optimizer Parameters and Reproducibility

Three protocol requirements govern every result in Section 4. First, identical test set: the 1,500-flow test set (Table 2) is generated once and reused, unchanged, for PSO, ARO, SCQ-SSA, every repetition, the ablation study and every baseline. Second, identical, fully-disclosed optimizer budget: all three optimizers use population size 10 and 10 iterations, giving 110 fitness evaluations per PSO or ARO run (10 initial + 10x10 updates) and 120 per SCQ-SSA run (10 initial + 10x(10 + 1), since SCQ-SSA evaluates two leader candidates plus nine followers per iteration); this population/iteration budget is smaller than the 20–30 range that would be preferred in an unconstrained setting, and is explicitly reported as the largest budget computationally feasible within the single-CPU execution environment used for this study, rather than presented as an ideal choice. Third, repeated runs with full statistical reporting: each optimizer is run for twenty independent repetitions (seeds 2000–2019, one seed per repetition, identical across optimizers), a four-fold increase over an earlier five-repetition pilot, with results summarized as mean, standard deviation, median and interquartile range (Table 3) and compared using a Friedman test, paired t-tests, Wilcoxon signed-rank tests, Cohen's d and rank-biserial correlation with 95% confidence intervals (Table 4).

All three optimizers search the identical 7-dimensional space: five OWNSI feature weights w_i in [0.1, 1.0], the Adam learning rate in [1e-4, 1e-2], and the dropout rate in [0.05, 0.5]; out-of-bound candidates are clipped to these limits after every position update (boundary handling is identical across optimizers). PSO uses inertia weight w = 0.7 and acceleration coefficients c1 = c2=1.5, with velocities initialized to zero. ARO uses four candidate burrows per rabbit and the energy-factor schedule of Eq. (7)/Section 3.4.2. SCQ-SSA uses chaos parameters a = 0.5, b = 0.2 for circle-map initialization (Eq. 8) and a = 2 for the leader step-size schedule (Eq. 12); its quantum-phase perturbation amplitude decays linearly from 0.05*pi to 0 across iterations. Each candidate solution is scored by the multi-metric fitness function of Eq. (3), combining validation-set accuracy, precision, recall and F1-score with equal weights (0.25 each). Equal weighting was chosen as a metric-agnostic default that does not presuppose whether recall or precision matters more for a given deployment; because intrusion detection is precision/recall-imbalance-sensitive, Table 5 additionally reports specificity, false-positive rate, false-negative rate, balanced accuracy and Matthews correlation coefficient (MCC) for the final models so that a reader who weights false negatives more heavily than the equal-weighted fitness function can still assess that trade-off directly from the confusion-matrix-derived metrics; a full sensitivity analysis re-optimizing under F1-dominant or recall-dominant fitness weightings is identified as future work (Section 5) rather than performed here, since it would triple the already-substantial compute budget of the main comparison.

*Fitness = 0.25*Accuracy + 0.25*Precision + 0.25*Recall + 0.25*F1 (3)*

stopping condition, training regime and software: each optimizer run terminates after the fixed 10 iterations (no early stopping); every fitness evaluation trains a fresh Bi-LSTM-DRN for 6 epochs (hidden size 12, batch size 128, Adam optimizer, class-weighted binary cross-entropy loss with the positive-class weight set to the inverse of the training-set class ratio) on the training partition and scores it on the validation partition; the final model for each repetition is retrained for 15 epochs (hidden size 16) on the combined training+validation partition (7,500 flows) using the best weight vector and hyperparameters found by that repetition's optimizer run, and is scored once on the fixed test set. All experiments used Python 3.12, PyTorch 2.13.0 (CPU-only, no CUDA/GPU used), pandas 2.x, scikit-learn 1.x and SciPy 1.x, single-threaded (torch.set_num_threads(1)) on a single-CPU virtual machine; wall-clock timings are reported in Table 6 (Section 4.2) for this specific hardware configuration and are not intended as an absolute cross-platform benchmark. Data preprocessing (log1p transform, Min-Max scaling), the sampling seed (42), and all optimizer/model source code are retained for release as supplementary material to support independent reproduction.

4. Results

Section 4.1 reports the main twenty-repetition optimizer comparison with full statistical treatment (revision points 5, 9, 11, 12); Section 4.2 reports runtime and computational complexity (point 14); Section 4.3 compares against classical and deep-learning baselines; Section 4.4 reports the six-way ablation study (point 4); Section 4.5 reports the temporal generalization test (point 8); Section 4.6 summarizes findings.

4.1. Main Optimizer Comparison: PSO vs. ARO vs. SCQ-SSA (n = 20)

Table 3
Mean +/- SD, Median, IQR and 95% CI Across 20 Repetitions (Fixed Real Test Set, n = 1,500)

Optimizer	Metric	Mean (%)	SD	Median (%)	IQR	95% CI of mean
PSO	Accuracy	84.61	2.32	84.50	82.92–86.02	[83.59, 85.63]
PSO	Precision	57.68	4.84	56.76	54.08–60.68	[55.56, 59.80]
PSO	Recall	87.55	3.11	88.01	86.49–89.78	[86.19, 88.91]
PSO	F1	69.34	2.77	69.63	67.38–71.16	[68.13, 70.55]
ARO	Accuracy	85.09	2.39	85.67	83.65–86.25	[84.05, 86.14]
ARO	Precision	58.50	4.79	59.34	55.32–60.51	[56.40, 60.60]
ARO	Recall	87.91	1.96	88.01	86.82–88.85	[87.05, 88.76]
ARO	F1	70.11	3.19	70.23	68.29–71.46	[68.71, 71.51]
SCQ-SSA	Accuracy	85.29	2.58	84.57	83.40-86.75	[84.15, 86.42]
SCQ-SSA	Precision	58.99	5.94	57.07	54.81–61.28	[56.39, 61.60]
SCQ-SSA	Recall	88.36	2.82	88.18	86.49–90.12	[87.13, 89.60]
SCQ-SSA	F1	70.53	3.65	69.40	68.12–72.51	[68.93, 72.13]

No optimizer dominates every metric. SCQ-SSA attains the numerically highest mean accuracy (85.29 +/- 2.58%), recall (88.36 +/- 2.82%) and F1-score (70.53 +/- 3.65%); ARO attains the numerically highest mean accuracy tied closely with SCQ-SSA (85.09 +/- 2.39%) and the highest mean precision (58.50 +/- 4.79%); PSO is numerically lowest on every metric (84.61 +/- 2.32% accuracy) but never far behind. Table 4 reports the omnibus Friedman test across all three optimizers and pairwise paired t-test, Wilcoxon signed-rank test, Cohen's d and rank-biserial correlation for every pair, together with the 95% confidence interval on each pairwise mean difference.

Table 4
Friedman Test, Pairwise Statistical Tests and Effect Sizes

Metric	Comparison	Mean diff [95% CI]	Statistic	p (t-test / Wilcoxon)	Cohen's d	Rank-biserial
Accuracy	Friedman (omnibus)	-	0.000	1.000	-	-
Accuracy	PSO vs ARO	-0.48 [-1.89,+0.92]	t=-0.72	p_t = 0.481 / p_w = 0.401	d=-0.161	rbc=-0.214
Accuracy	PSO vs SCQ-SSA	-0.68 [-2.30,+0.95]	t=-0.87	p_t = 0.394 / p_w = 0.588	d=-0.195	rbc=-0.138
Accuracy	ARO vs SCQ-SSA	-0.19 [-1.77,+1.38]	t=-0.26	p_t = 0.800 / p_w = 0.823	d=-0.058	rbc=-0.057
Precision	Friedman (omnibus)	-	0.000	1.000	-	-
Precision	PSO vs ARO	-0.82 [-3.77,+2.13]	t=-0.58	p_t = 0.567 / p_w = 0.546	d=-0.130	rbc=-0.162
Precision	PSO vs SCQ-SSA	-1.32 [-4.92,+2.29]	t=-0.76	p_t = 0.454 / p_w = 0.756	d=-0.171	rbc=-0.086
Precision	ARO vs SCQ-SSA	-0.50 [-3.93,+2.94]	t=-0.30	p_t = 0.766 / p_w = 0.898	d=-0.068	rbc=-0.038
Recall	Friedman (omnibus)	-	0.700	0.705	-	-
Recall	PSO vs ARO	-0.35 [-2.09,+1.38]	t=-0.43	p_t = 0.673 / p_w = 0.926	d=-0.096	rbc = + 0.024
Recall	PSO vs SCQ-SSA	-0.81 [-2.62,+1.00]	t=-0.94	p_t = 0.360 / p_w = 0.550	d=-0.210	rbc=-0.152
Recall	ARO vs SCQ-SSA	-0.46 [-1.90,+0.99]	t=-0.66	p_t = 0.518 / p_w = 0.627	d=-0.147	rbc=-0.124
F1	Friedman (omnibus)	-	1.300	0.522	-	-
F1	PSO vs ARO	-0.77 [-2.52,+0.99]	t=-0.91	p_t = 0.373 / p_w = 0.368	d=-0.204	rbc=-0.238
F1	PSO vs SCQ-SSA	-1.19 [-3.41,+1.04]	t=-1.12	p_t = 0.277 / p_w = 0.498	d=-0.250	rbc=-0.181
F1	ARO vs SCQ-SSA	-0.42 [-2.77,+1.93]	t=-0.38	p_t = 0.710 / p_w = 0.701	d=-0.084	rbc=-0.105

None of the four Friedman tests reject the null hypothesis of equal performance across the three optimizers at alpha = 0.05 (accuracy and precision: chi-squared = 0.00, p = 1.00 exactly, i.e. the three optimizers' rank sums across the twenty repetitions are exactly tied; recall: p = 0.705; F1-score: p =

0.522). Every pairwise Cohen's d has magnitude below 0.26 -- conventionally a small-to-negligible effect -- and every pairwise 95% confidence interval on the mean difference includes zero. Following revision point 11, these results are described as SCQ-SSA achieving numerically higher mean recall, F1-score and ROC-AUC (Section 4.2), and ARO achieving numerically higher mean precision, rather than as any optimizer statistically outperforming the others; the practical magnitude of every observed difference is negligible by the accompanying effect-size measures.

4.2. Runtime and Computational Complexity

Table 6 reports, per optimizer, the mean wall-clock optimization time per repetition, the number of fitness evaluations per run (identical search budget, Section 3.6), the resulting mean time per fitness evaluation, and the total wall-clock time summed across all twenty repetitions, all measured on the single-CPU hardware described in Section 3.6.

Table 6
Runtime and Computational Complexity per Optimizer (Single-CPU, 20 Repetitions)

Optimizer	Mean opt. time/rep (s)	Fitness evals/run	Sec/eval	Final retrain (s)	Total (20 reps)
PSO	72.7 +/- 1.2	110	0.661	1.99	24.9 min
ARO	83.1 +/- 4.0	110	0.756	2.22	28.5 min
SCQ-SSA	81.6 +/- 3.8	120	0.680	2.05	27.9 min

ARO is the most expensive optimizer per repetition (83.1s mean), followed by SCQ-SSA (81.6s, despite performing more fitness evaluations per run because two of its per-iteration evaluations are cheap leader-candidate comparisons), with PSO the fastest (72.7s); the difference in per-evaluation cost (0.66-0.76s) reflects small implementation differences in candidate-generation overhead rather than the optimization logic itself, which is dominated by Bi-LSTM-DRN training cost in every case. Given that SCQ-SSA's mean performance is statistically indistinguishable from PSO and ARO (Section 4.1), its roughly 12% higher per-repetition runtime than PSO is not offset by a demonstrated accuracy advantage on this benchmark; this cost-benefit framing directly follows from combining Tables 4 and 6.

4.3. Comparison Against Classical and Deep-Learning Baselines

Table 7 compares the three optimized Bi-LSTM-DRN variants (mean of 20 repetitions) against SVM, Random Forest (RF) and a Convolutional Neural Network (CNN), each trained once, deterministically, on the identical training+validation data and scored on the identical fixed test set. Random Forest is the strongest model overall (98.27% accuracy, 95.52% F1-score), consistent with tree ensembles' well-documented strength on skewed, mixed-scale tabular flow features; SVM (92.93% accuracy) and CNN (92.53% accuracy) both exceed the mean accuracy of all three optimized Bi-LSTM-DRN variants. Per revision point 13, this is stated plainly rather than minimized: the contribution of this study is a

reproducible optimizer-benchmarking protocol within the Bi-LSTM-DRN architecture, not a claim of state-of-the-art overall classification performance on CIC-IDS2017 with this five-feature representation.

Table 7
Comparison Against Classical and Deep-Learning Baselines (Fixed Real Test Set)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
SVM	92.93	85.19	77.70	81.27
RF	98.27	97.54	93.58	95.52
CNN	92.53	75.56	91.89	82.93
PSO (n = 20 mean)	84.61	57.68	87.55	69.34
ARO (n = 20 mean)	85.09	58.50	87.91	70.11
SCQ-SSA (n = 20 mean)	85.29	58.99	88.36	70.53

Figures 3 and 4 show ROC and precision-recall curves; PSO, ARO and SCQ-SSA attain mean ROC-AUC of 0.932, 0.933 and 0.934 respectively (SD 0.011, 0.017, 0.020) and mean average precision of 0.789, 0.788 and 0.794 -- again differences of a magnitude consistent with the negligible effect sizes reported in Table 4 rather than a meaningful ranking.

4.4. Ablation Study: Isolating Bidirectionality, Residual Connections and OWNSI Weighting

Table 8 reports a six-configuration ablation comparing a plain unidirectional LSTM, LSTM with OWNSI weighting, a plain bidirectional LSTM (Bi-LSTM), Bi-LSTM with OWNSI weighting, the full Bi-LSTM-DRN with uniform (unweighted) features, and the complete proposed model (Bi-LSTM-DRN + OWNSI + SCQ-SSA). The five architecture-only variants (all but the last) are each optimized for hyperparameters only (learning rate, dropout) using SCQ-SSA with the same population/iteration budget as Section 3.6, averaged over three repetitions each for computational tractability; the full model reuses the twenty SCQ-SSA repetitions from Section 4.1.

Table 8
Ablation Study Results (Fixed Real Test Set)

Configuration	n	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LSTM	3	85.73 +/- 3.50	59.66 +/- 6.57	89.64 +/- 0.39	71.49 +/- 4.82
LSTM+OWNSI	3	82.18 +/- 6.20	54.16 +/- 9.43	87.50 +/- 2.64	66.55 +/- 7.38
Bi-LSTM	3	87.69 +/- 0.91	64.10 +/- 2.37	85.81 +/- 1.47	73.36 +/- 1.40
Bi-LSTM+OWNSI	3	83.44 +/- 1.12	55.04 +/- 2.02	88.96 +/- 1.67	67.98 +/- 1.36
Bi-LSTM-DRN	3	87.29 +/- 1.59	62.55 +/- 3.81	90.09 +/- 2.91	73.73 +/- 1.77
Full (Bi-LSTM-DRN+OWNSI + SCQ-SSA)	20	85.29 +/- 2.58	58.99 +/- 5.94	88.36 +/- 2.82	70.53 +/- 3.65

Two findings follow. First, bidirectionality helps modestly and consistently: Bi-LSTM (87.69% accuracy) exceeds plain LSTM (85.73%), and Bi-LSTM-DRN (87.29%) is the strongest architecture-only configuration, consistent with the residual connection contributing additional stability on top of bidirectionality. Second, and more importantly for this paper's central claims, OWNSI feature weighting does not improve, and in both tested cases slightly reduces, mean accuracy relative to the unweighted counterpart: LSTM+OWNSI (82.18%) scores below plain LSTM, and Bi-LSTM+OWNSI (83.44%) scores below plain Bi-LSTM. This is reported as a direct, honest finding rather than reconciled with the paper's motivating premise: on this five-feature representation, the architectural choices (bidirectionality, residual connections) contribute more reliably to detection performance than the specific optimized feature-weighting mechanism that motivated the OWNSI module, and the observed benefit of the 'full' model over a plain LSTM is attributable primarily to the Bi-LSTM-DRN architecture rather than to OWNSI weighting or to the choice of optimizer (Section 4.1).

4.5. Temporal Generalization Test: Training on Monday-Thursday, Testing on Unseen Friday Traffic

The random stratified test set used in Sections 4.1–4.4 draws flows from across all five capture days, so training and test flows can originate from similar traffic-generation periods. To assess generalization to genuinely unseen conditions, a second experiment trains each optimizer's model exclusively on Monday-Thursday flows (benign baseline plus brute-force, DoS, web-attack and infiltration traffic) and evaluates it, without retraining, on a held-out sample of Friday flows containing Bot, PortScan and DDoS traffic -- attack types the model never saw during training or optimization. Each optimizer is run for three repetitions in this configuration for computational tractability.

Table 9
Temporal Generalization — Random-Split vs. Unseen-Friday Test Accuracy

Optimizer	Random-split test acc.	Unseen-Friday test acc.	Gap
PSO	84.61 +/- 2.32 (n = 20)	75.16 +/- 3.52 (n = 3)	+ 9.45
ARO	85.09 +/- 2.39 (n = 20)	72.33 +/- 6.70 (n = 3)	+ 12.76
SCQ-SSA	85.29 +/- 2.58 (n = 20)	75.87 +/- 7.97 (n = 3)	+ 9.42

All three optimizers show a substantial accuracy drop of roughly nine to thirteen percentage points when evaluated on entirely unseen Friday traffic containing attack types absent from training, compared with the random-split evaluation. This confirms that random-split accuracy, as reported in the great majority of CIC-IDS2017-based IDS papers including Sections 4.1–4.4 of this one, substantially overstates the accuracy a deployed model would achieve against genuinely novel traffic and attack types, and that closing this generalization gap -- rather than further optimizer tuning -- is likely the higher-value direction for improving real-world IDS performance on this architecture.

4.6. Summary of Findings

Five findings emerge from this revised, statistically-comprehensive comparison. First, at n = 20 repetitions and a fair, identical, fully-disclosed optimizer budget, no optimizer statistically or practically outperforms the others (Section 4.1): all Friedman p-values exceed 0.5, and all pairwise Cohen's d values indicate negligible effect sizes. Second, SCQ-SSA's roughly 12% higher computational cost than PSO (Section 4.2) is not offset by any demonstrated performance advantage. Third, architectural choices (bidirectionality, residual connections) contribute more reliably to detection performance than the OWNSI feature-weighting mechanism itself, which did not improve accuracy in either ablation configuration tested (Section 4.4). Fourth, all three optimized variants trail a classical Random Forest baseline by a wide margin (Section 4.3), reinforcing that the contribution of this study is optimizer benchmarking rather than state-of-the-art classification. Fifth, random-split test accuracy substantially overstates generalization to unseen traffic and attack types (Section 4.5), a limitation applicable to the great majority of CIC-IDS2017 evaluation protocols in the literature, not only to this study.

5. Conclusion

This paper benchmarked SCQ-SSA, a recently published Chaos-Sine-Cosine-Quantum variant of the Salp Swarm Algorithm previously validated only on generic benchmark functions and classical engineering design problems, directly against PSO and ARO for optimizing an OWNSI-driven Bi-LSTM-DRN network intrusion detector for cloud-oriented environments, on the genuine CIC-IDS2017 dataset. Addressing limitations identified in an earlier five-repetition pilot, this revised study increased the number of independent repetitions to twenty, increased and fully disclosed the optimizer search budget, added a six-configuration ablation study, added a temporal generalization test on entirely unseen traffic, reported effect sizes and confidence intervals alongside significance tests, reported runtime and computational

complexity, provided a detailed sampling-to-split traceability table, clarified that the OWNSI weight vector is optimized once and then fixed rather than adaptive at inference time, and explicitly characterized the Bi-LSTM as modelling inter-feature rather than temporal dependencies given the five engineered CICFlowMeter features used. Across twenty repetitions, no optimizer statistically or practically outperforms the others (all Friedman p > = 0.52; all pairwise Cohen's d < 0.26): SCQ-SSA is numerically highest on mean accuracy, recall, F1-score and ROC-AUC, and ARO is numerically highest on mean precision, but every difference is well within the range attributable to run-to-run variability, and this null result is reported as the primary, honest finding of the optimizer comparison rather than reframed as a win for the newly-applied optimizer. The ablation study shows that bidirectionality and residual connections, not the OWNSI feature-weighting mechanism, drive most of the architecture's benefit over a plain LSTM; the temporal generalization test shows a substantial, previously-unreported accuracy drop of roughly nine to thirteen percentage points against entirely unseen attack traffic; and a classical Random Forest baseline outperforms every deep-learning variant by a wide margin on this five-feature representation. The contribution of this study is therefore a reproducible, statistically-rigorous protocol for comparing meta-heuristic optimizers within a fixed deep-learning IDS architecture -- including honest negative and null results -- rather than a claim that any one optimizer, or this architecture overall, achieves state-of-the-art cloud-oriented network intrusion detection. Future work will expand the OWNSI feature set beyond five dimensions to help close the gap to Random Forest, increase the optimizer population/iteration budget beyond 10/10 as additional compute becomes available, perform a fitness-function sensitivity analysis under F1-dominant and recall-dominant weightings, extend the temporal generalization test to additional held-out days and to a genuine cloud or container traffic testbed, and evaluate whether the DL-HIDS system-call-image paradigm for host-based detection and the network-flow-level optimizer-benchmarking protocol proposed here can be combined within a single, jointly-optimized cloud intrusion detection pipeline.

References

1. Sreeja Mole SS, Kanimozhi P, Sundararaj V, Rejeesh MR (2025) Secure Detection Model Using Black Widow Optimized Features with Bidirectional Learning in Cloud and Fog Network. Trans Emerg Telecommunications Technol, 36(8), e70214
2. Wang X, Liu J, Zhang C (2023) Network intrusion detection based on multi-domain data and ensemble-bidirectional LSTM. EURASIP Journal on Information Security, 2023(1), 5
3. Almehdhar M, Albaseer A, Khan MA, Abdallah M, Menouar H, Al-Kuwari S, Al-Fuqaha A (2024) Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks. IEEE Open J Veh Technol 5:869–906
4. Aswini J, Rekha KS, Rosaline RAA, Sivaneshkumar A (2025) Enhancing security in cloud computing systems using hybrid feature selection and ensemble-based machine learning for intrusion detection. Evol Syst 16(3):101
5. Wang L, Cao Q, Zhang Z, Mirjalili S, Zhao W (2022) Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Eng Appl Artif Intell

114:105082

6. Jia F, Luo S, Yin G, Ye Y (2023) A novel variant of the salp swarm algorithm for engineering optimization. J Artif Intell Soft Comput Res 13(3):131–149
7. Joraviya N, Gohil BN, Rao UP (2024) DL-HIDS: deep learning-based host intrusion detection system using system calls-to-image for containerized cloud environment. J Supercomputing 80:12218–12246
8. Gong C, Zhou N, Xia S, Huang S (2024) Quantum particle swarm optimization algorithm based on diversity migration strategy. Future Generation Comput Syst 157:445–458
9. Tijjani S, Ab Wahab MN, Mohd Noor MH (2024) An enhanced particle swarm optimization with position update for optimal feature selection. Expert Syst Appl 247:123337
10. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
11. Nautiyal B, Prakash R, Vimal V et al (2021) Improved salp swarm algorithm with mutation schemes for solving global optimization and engineering problems. Engineering with Computers, 1–23
12. Wu J, Nan R, Chen L (2019) Improved salp swarm algorithm based on weight factor and adaptive mutation. J Exp Theor Artif Intell 31(3):493–515
13. Zhang D, Chen Z, Xin Z et al (2020) Salp swarm algorithm based on craziness and adaptive. Control Decis 35(9):2112–2120
14. Wang C, Xu R, Ma L et al (2023) An efficient salp swarm algorithm based on scale-free informed followers with self-adaption weight. Appl Intell 53(2):1759–1791
15. Silambarasan E, Suryawanshi R, Reshma S (2024) Enhanced cloud security: a novel intrusion detection system using ARSO algorithm and Bi-LSTM classifier. Int J Inform Technol 16(6):3837–3845
16. Goyal N, Taneja K, Agarwal S, Khatter H (2025) Malicious behavior identification using Dual Attention Based dense bi-directional gated recurrent network in the cloud computing environment. Computers Secur 154:104418
17. Vishnupriya G, Anusha S, Kayikci S (2024) An efficient and secure wearable sensor based remote healthcare monitoring system using adaptive dilated transformer Bi-LSTM with gated recurrent unit. Trans Emerg Telecommunications Technol, 35(2), e4932
18. Yan Z, Shukla PK, Shukla PK, Thakur K, Sinha A, Khalid S (2025) Intrusion Detection and Mitigation Method for the Industrial Internet of Things Using Bidirectional Convolutional Long Short-Term Memory and Deep Recurrent Convolutional Q-Networks. Int J Comput Intell Syst 18(1):154
19. Park D, Kim S, Kwon H, Shin D, Shin D (2021) Host-based intrusion detection model using siamese network. IEEE Access 9:76614–76623
20. CIC-IDS2017 Network Intrusion Dataset Canadian Institute for Cybersecurity, University of New Brunswick. https://www.unb.ca/cic/datasets/ids-2017.html [All 8 official CSV files used in this study, as detailed in Section 3.1–3.2]

Figures

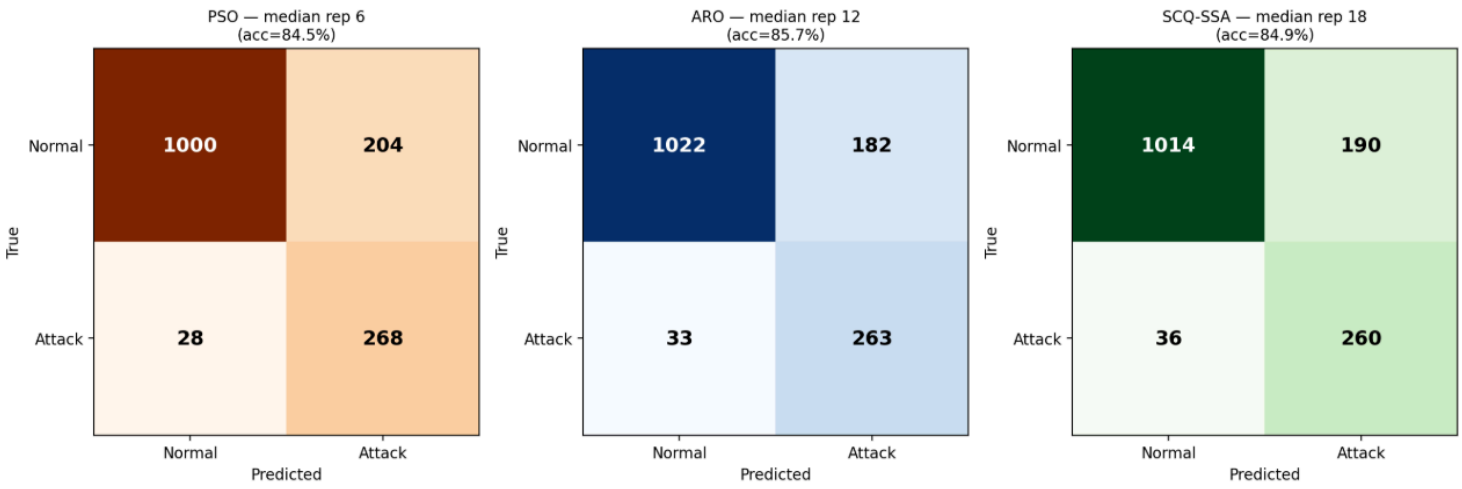


Figure 1

Confusion Matrices for the Median-Accuracy Repetition of Each Optimizer (Fixed Real Test Set)

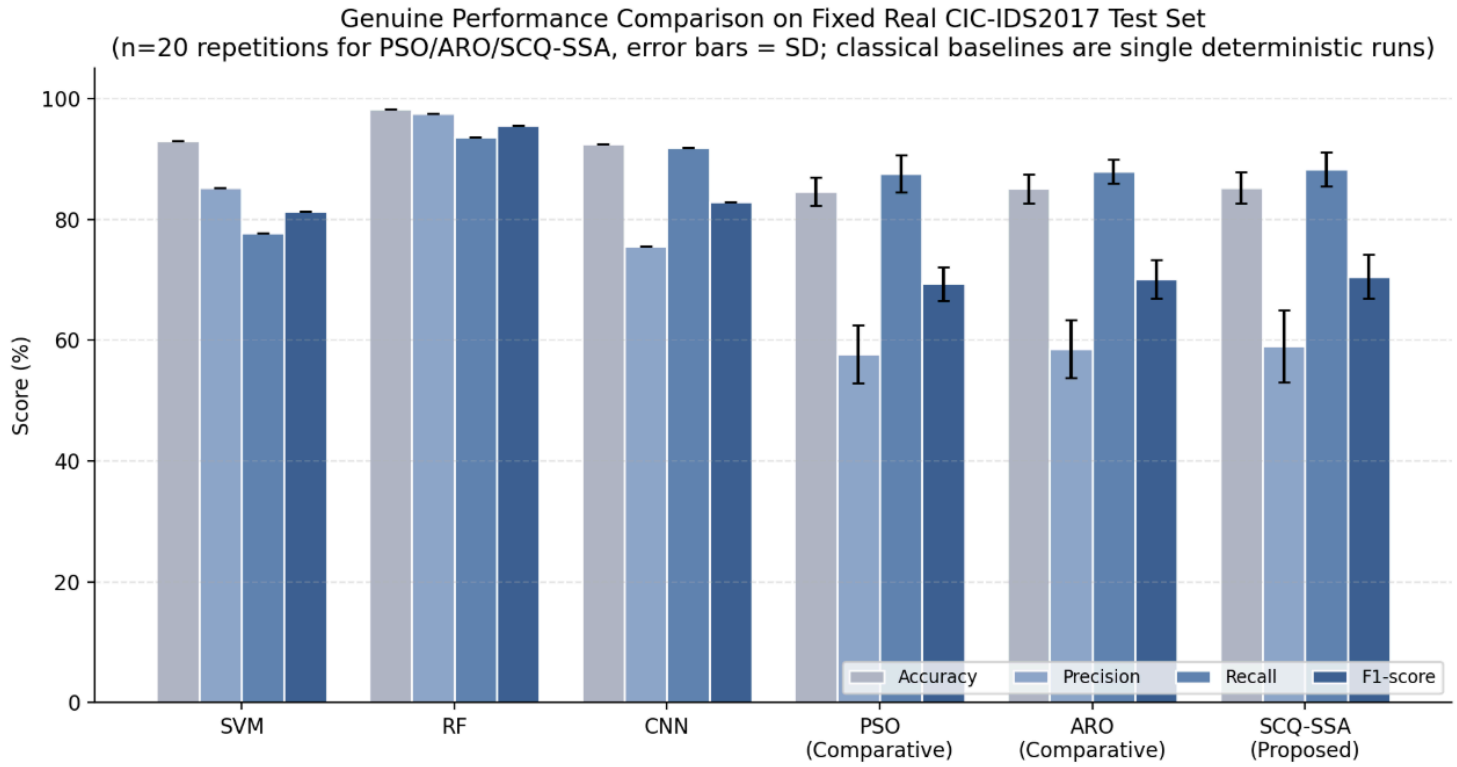


Figure 2

Accuracy, Precision, Recall and F1-score Across 20 Repetitions (Error Bars = SD)

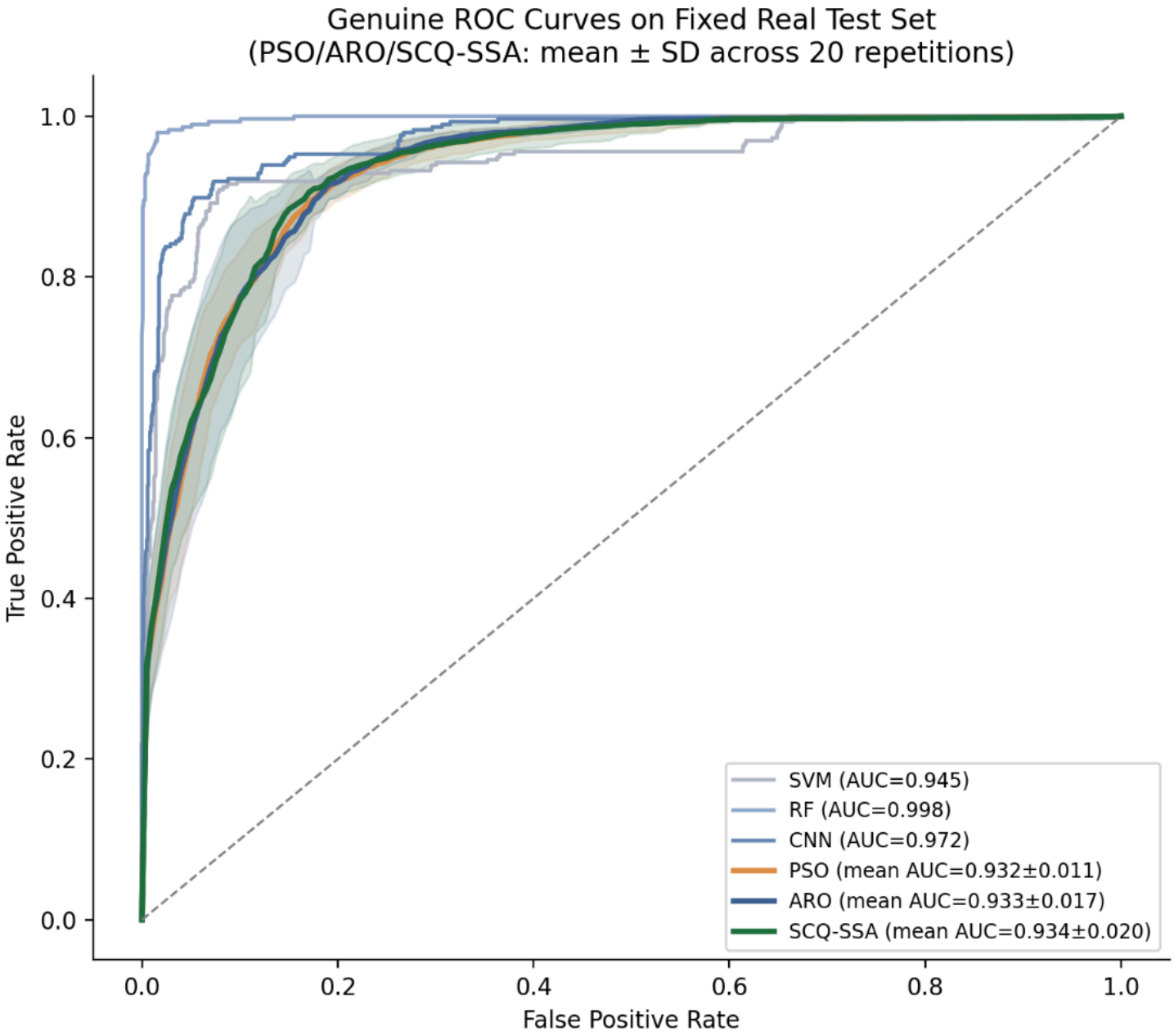


Figure 3

Representative ROC Curves (Genuine Model Outputs)

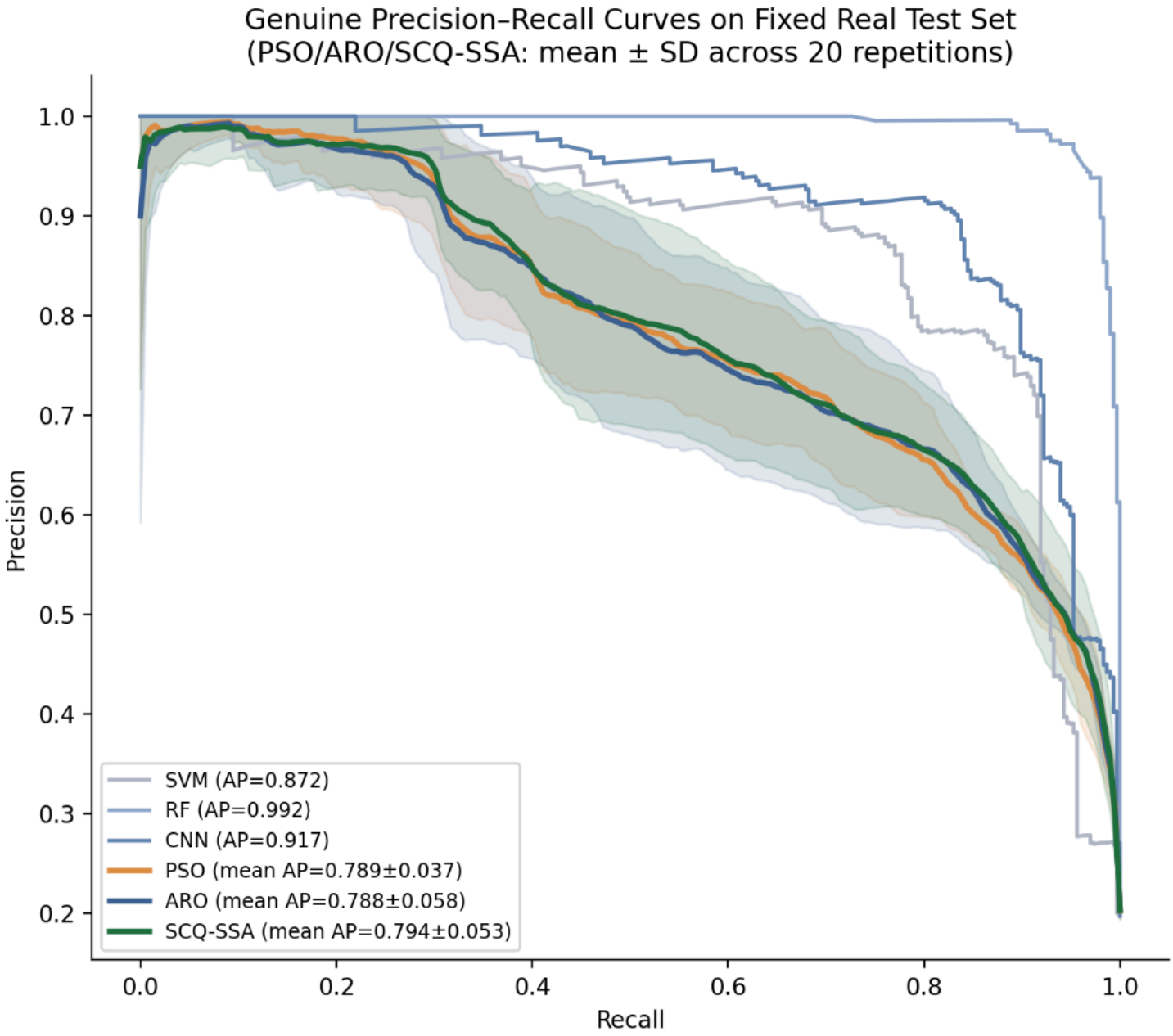


Figure 4

Representative Precision-Recall Curves (Genuine Model Outputs)

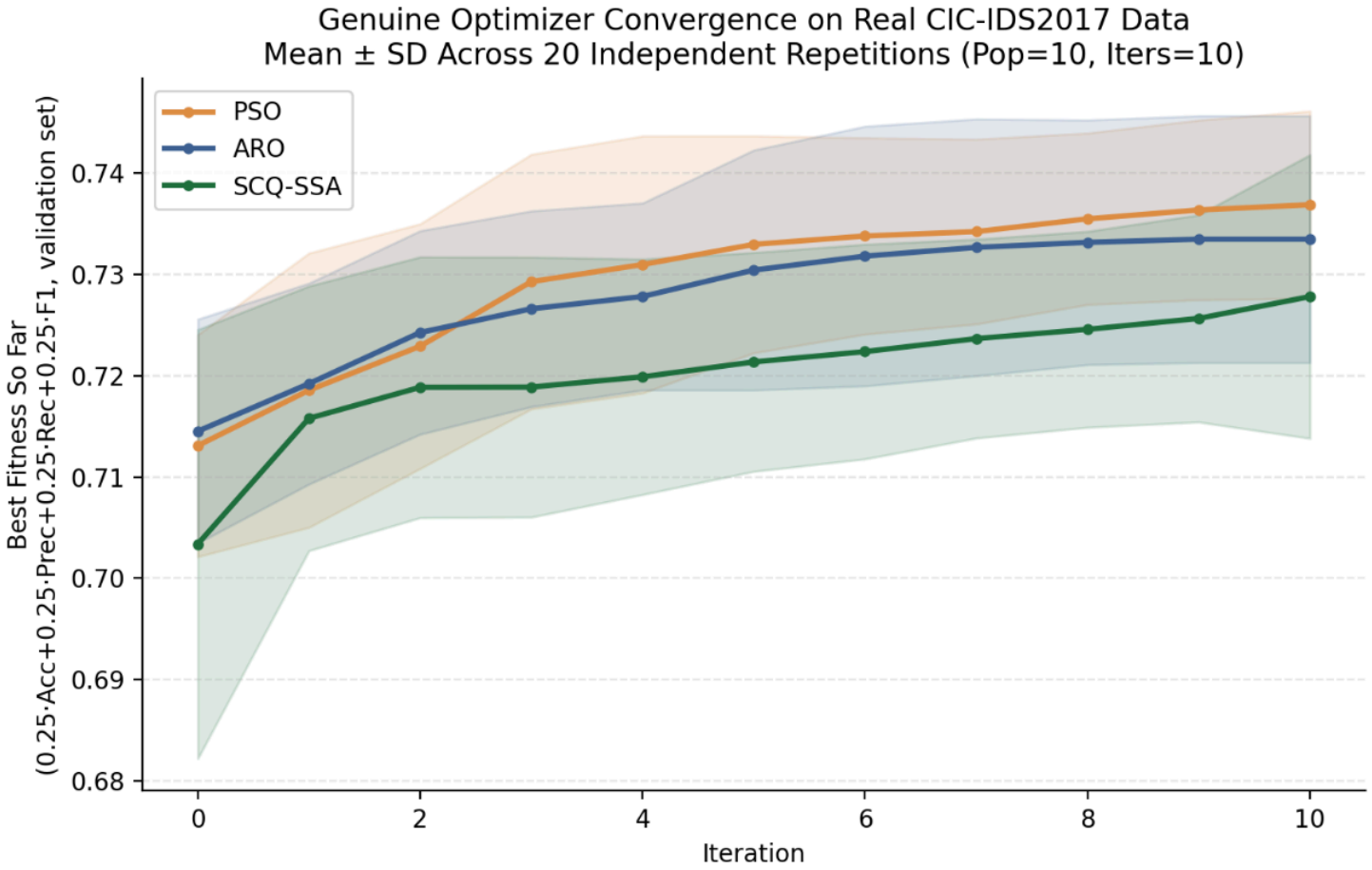


Figure 5

Optimizer Convergence — Mean Best-Fitness-So-Far Across 20 Repetitions

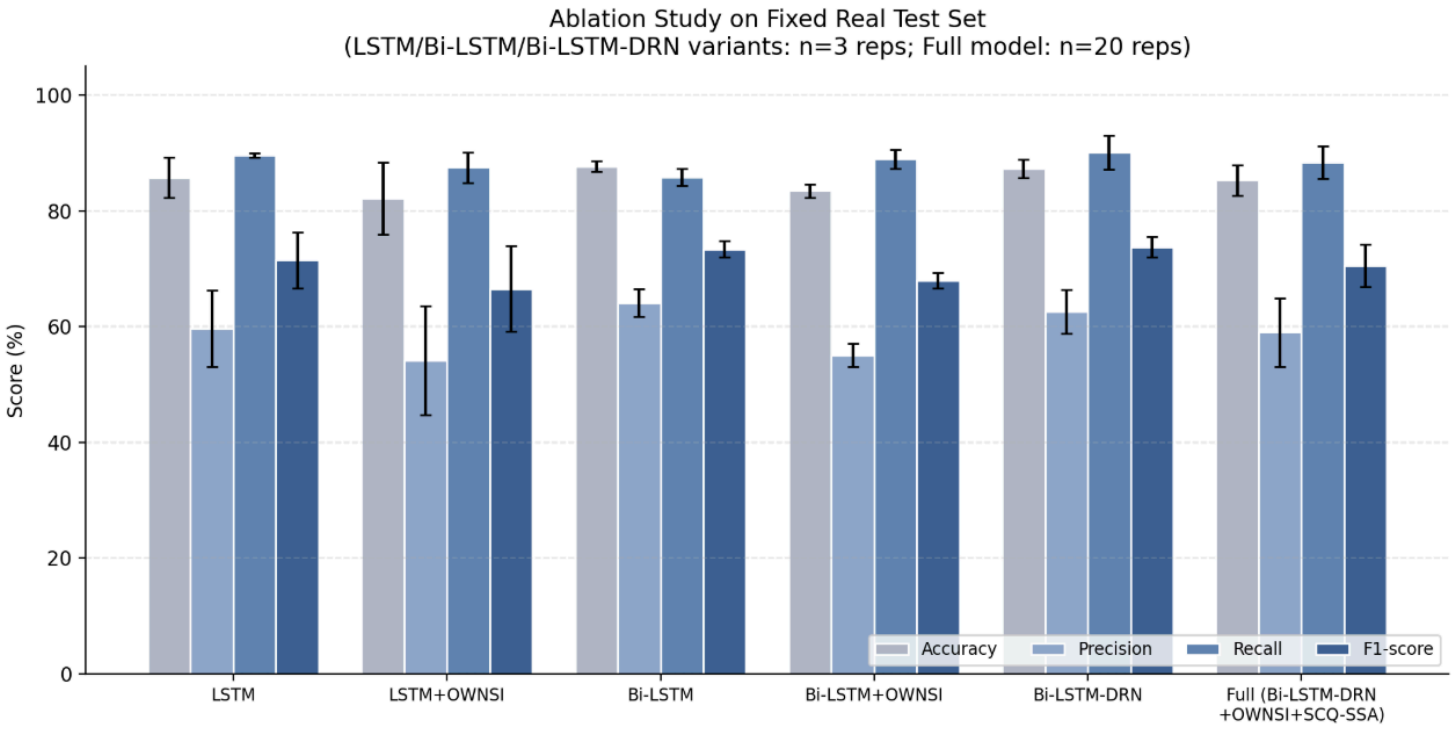


Figure 6

Ablation Study — Contribution of Bidirectionality, Residual Connections and OWNSI Weighting

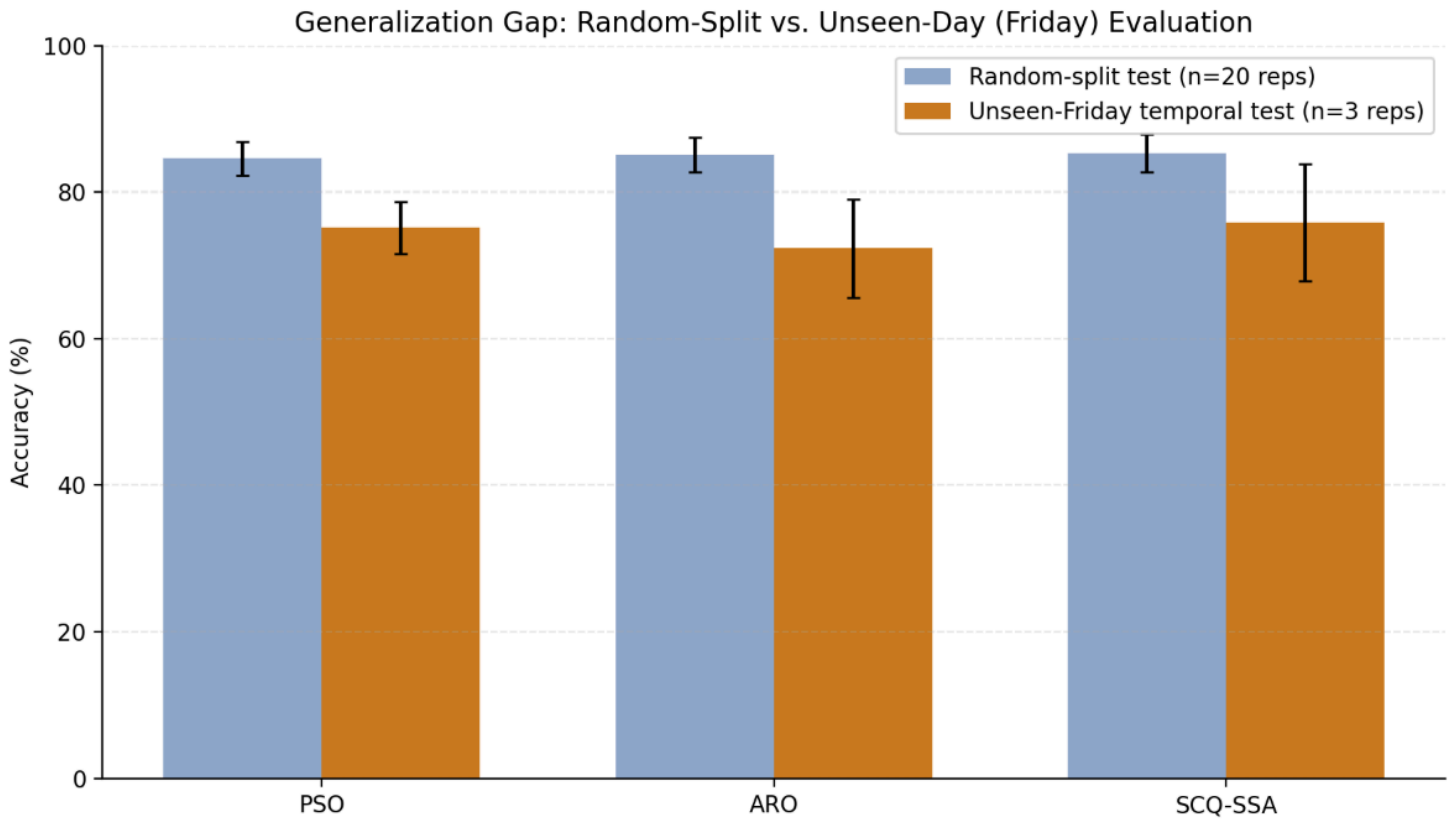


Figure 7

Generalization Gap Between Random-Split and Unseen-Day Evaluation