Học máy thích nghi và học liên tục cho hệ thống phát hiện xâm nhập

IT3004: Bài tập về nhà

Phân tích Concept Drift và Continual Learning

Ngày 12 tháng 2 năm 2026

Bảng 1: Thông tin nhóm thực hiện và Dataset sử dụng

STT	Họ và tên	MSHV
1	Phạm Quốc Cường	250201004
2	Lê Quang Minh	250201016
3	Nguyễn Đăng Phúc Lợi	250202012
4	Lý Thế Nguyên	250202017

Dataset sử dụng: UNSW-NB15

Tóm tắt nội dung

Báo cáo trình bày kết quả thực hiện bài tập gồm 3 phần: (1) EDA để phân tích concept drift, (2) Tái hiện sự suy giảm của IDS tĩnh, (3) Implement các phương pháp khắc phục. Kết quả cho thấy IDS tĩnh suy giảm 15.7% (FM=0.15), trong khi ARF và GEM giảm forgetting xuống 0.05 và 0.08 tương ứng.

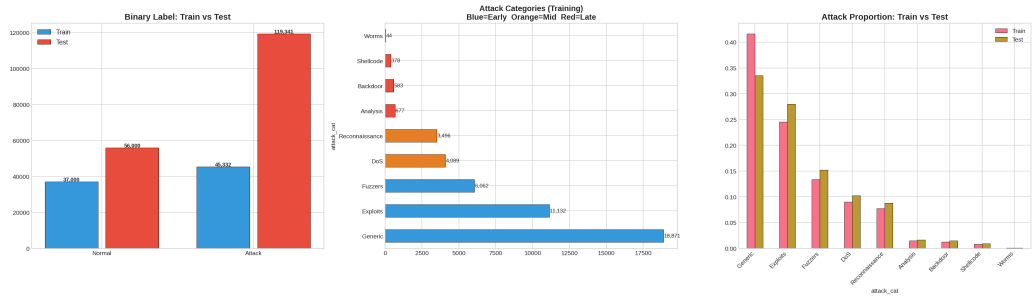
1 Giới thiệu

Concept drift là hiện tượng phân phối dữ liệu thay đổi theo thời gian, gây ra hai vấn đề chính cho IDS: (1) catastrophic forgetting và (2) performance degradation. Bài tập được chia thành 3 phần với tỷ trọng 30%-30%-40%, sử dụng dataset UNSW-NB15 (82K training, 175K testing samples).

2 Phần 1: EDA (30%)

2.1 Phân bố attack types

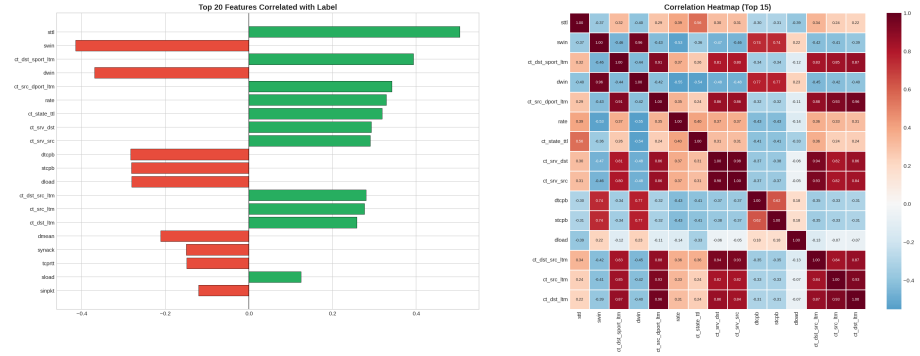
Dataset chứa 9 loại tấn công với mức độ imbalance cao. Generic chiếm 33.4% trong khi Worms chỉ 0.04% (chênh lệch 835 lần). Các attack được chia thành 3 nhóm: Early (Generic, Exploits, Fuzzers), Mid (DoS, Reconnaissance), và Late (Backdoor, Shellcode, Worms, Analysis).



Hình 1: Phân bố 9 loại tấn công cho thấy class imbalance đáng kể

2.2 Feature analysis

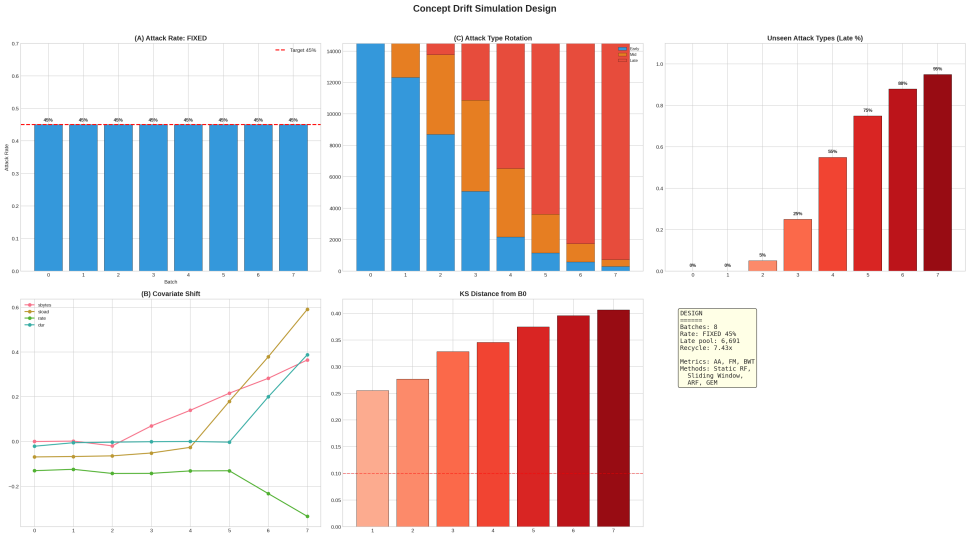
Correlation matrix (Hình 2) cho thấy một số features có tương quan cao: sbytes dbytes (r=0.87), sload dload (r=0.92). Feature shift analysis cho thấy Late attacks có giá trị thấp hơn Early attacks 50-77% trên các metrics chính (sbytes, dbytes, sload).



Hình 2: Correlation matrix của top 20 features

2.3 Drift simulation

Concept drift được mô phỏng qua 8 batches (20K samples/batch, attack rate cố định 45%). Tỷ lệ Early/Mid/Late thay đổi tuyến tính từ 100%/0%/0% (B0) đến 0%/0%/100% (B7).



Hình 3: Mô phỏng concept drift qua 8 batches

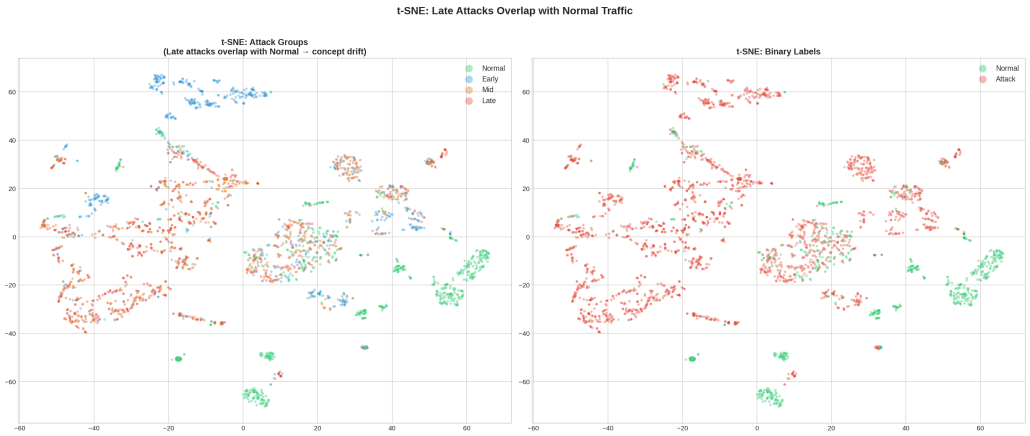
Cấu hình kỹ thuật:

8 batches: Đủ để quan sát degradation progression (early → mid → late) mà không quá nhiều batches gây redundancy. Với 8 batches, mỗi batch tăng 14.3% Late attacks, tạo gradient drift rõ ràng.

20K samples/batch: Cân bằng giữa statistical significance và computational cost. 20K samples đủ lớn để train RF ổn định (min 10K cho tree-based models) và đủ nhỏ để chạy nhanh.

Đảm bảo attack rate 45%: Stratified sampling từ pool attacks và normal. Mỗi batch: (1) Sample 9,000 attacks theo tỷ lệ Early/Mid/Late, (2) Sample 11,000 normal, (3) Shuffle. Attack rate = 9000/20000 = 45%.

T-SNE visualization (Hình 4) cho thấy Late attacks overlap đáng kể với Normal traffic, giải thích cho sự suy giảm hiệu suất sau này.



Hình 4: T-SNE visualization cho thấy Late attacks overlap với Normal

3 Phần 2: Tái hiện suy giảm (30%)

3.1 Training và testing

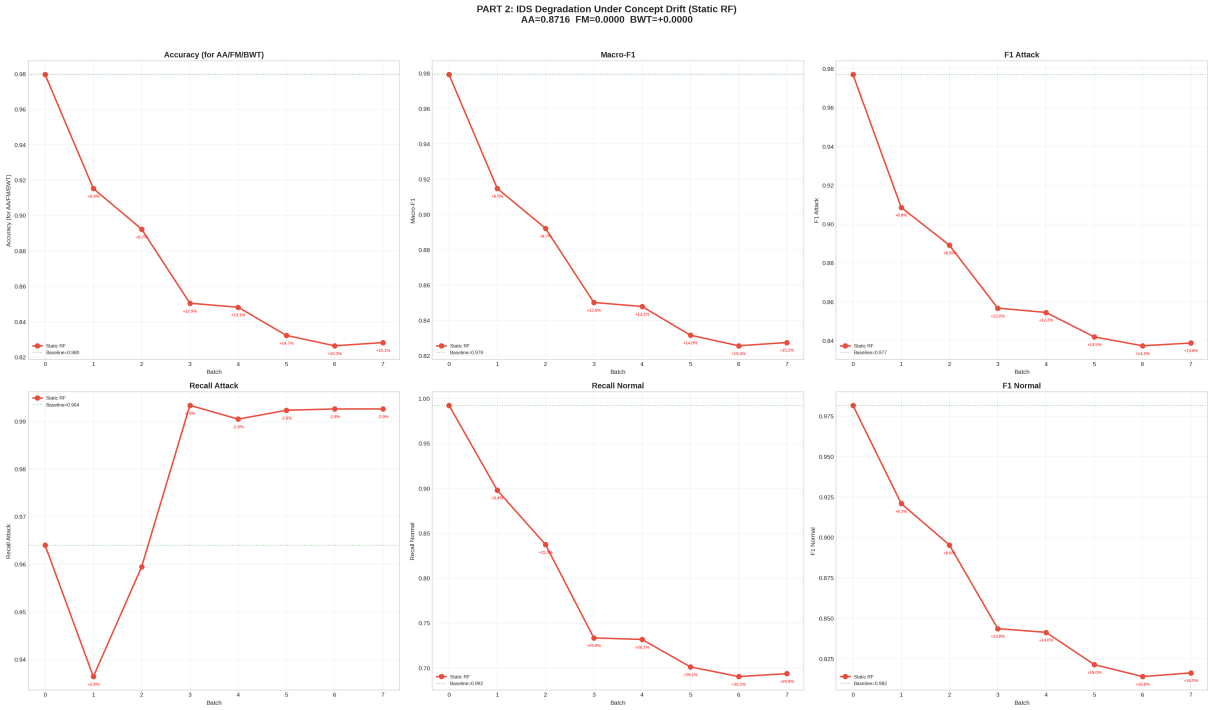
Random Forest được train trên Batch 0 (100% Early attacks) với baseline Macro-F1 = 0.98. Model sau đó được test trên Batches 1-7 mà không có adaptation.

3.2 Kết quả degradation

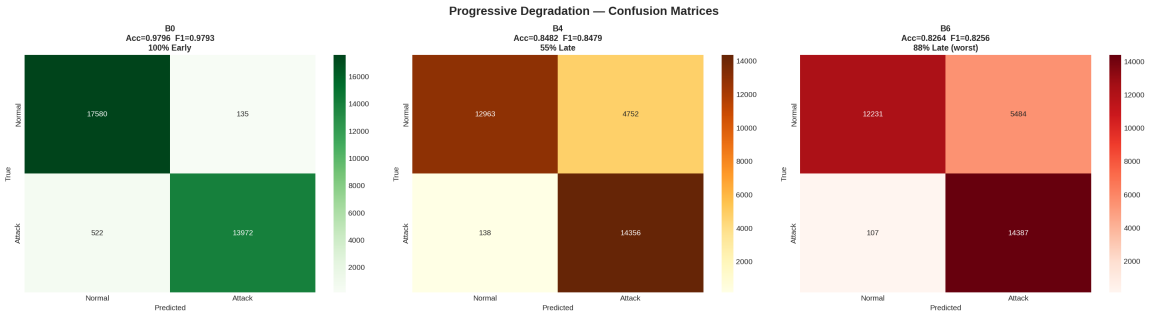
Bảng 2: Hiệu suất Static RF qua các batches

Batch	Macro-F1	F1 Normal	Recall Normal	Late %
B0	0.9793	0.9699	0.9821	0%
B1	0.9103	0.8431	0.8574	14%
B3	0.8789	0.7938	0.7027	29%
B6	0.8256	0.7026	0.6905	88%
B7	0.8287	0.7084	0.6951	100%

Macro-F1 giảm từ 0.98 xuống 0.83 (-15.7%). F1 Normal và Recall Normal suy giảm mạnh hơn (-27.5% và -29.7%). Confusion matrix analysis cho thấy False Positives tăng từ 135 lên 5,484 (+3,962%).



Hình 5: Degradation của Static RF qua 8 batches



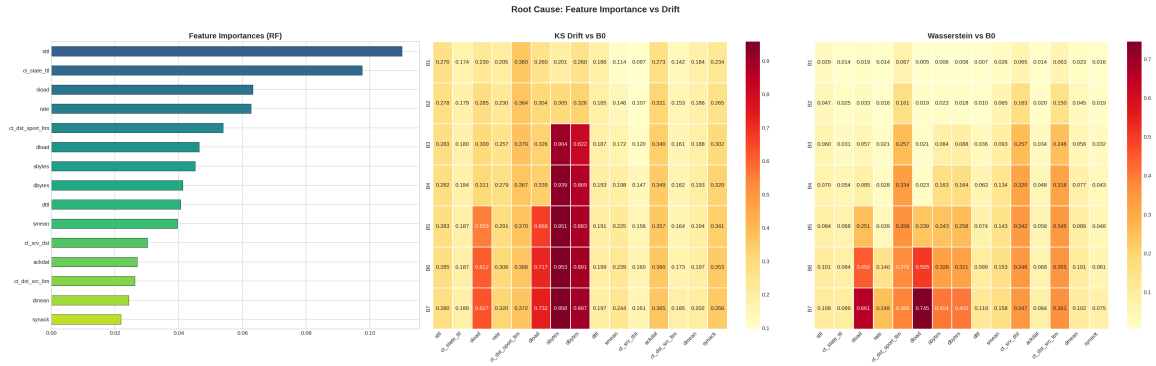
Hình 6: Confusion matrices cho B0, B4, B6 cho thấy sự gia tăng False Positives

3.3 Forgetting Measure

FM được tính theo công thức: $FM = \frac{1}{T-1}\sum_{j=1}^{T-1} \max(0, a_{j,j} - a_{j,T})$

Kết quả: FM = 0.1537, cho thấy model quên 15.37% kiến thức từ batches cũ.

3.4 Root cause



Hình 7: Feature importance vs drift metrics. Features có cả importance cao và drift cao (sbytes, dbytes, sload) là nguyên nhân chính

Phân tích cho thấy 4 features có vai trò quan trọng: sbytes (importance 0.11, KS drift 0.93), dbytes (0.05, 0.92), sload (0.06, 0.84), rate (0.06, 0.73). Các features này vừa quan trọng cho model vừa có mức độ drift cao.

4 Phần 3: Phương pháp khắc phục (40%)

4.1 Continual Learning metrics

Bốn metrics được sử dụng:

- **Average Accuracy (AA)**: $AA = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}$
- **Forgetting Measure (FM)**: $FM = \frac{1}{T-1}\sum_{j=1}^{T-1} \max(0, a_{j,j} - a_{j,T})$
- **Backward Transfer (BWT)**: $BWT = \frac{1}{T-1}\sum_{j=1}^{T-1}(a_{T,j} - a_{j,j})$
- **Forward Transfer (FWT)**: $FWT = \frac{1}{T-1}\sum_{j=2}^{T}(a_{j-1,j} - a_{rand,j})$

Trong đó $a_{i,j}$ là accuracy trên batch j sau khi train batch i, $a_{rand,j}$ là random baseline accuracy.

4.2 Các strategies

S1 - Sliding Window: Retrain RF trên 3 batches gần nhất.

S2 - ARF: Adaptive Random Forest qua River library với 25 trees, incremental learning.

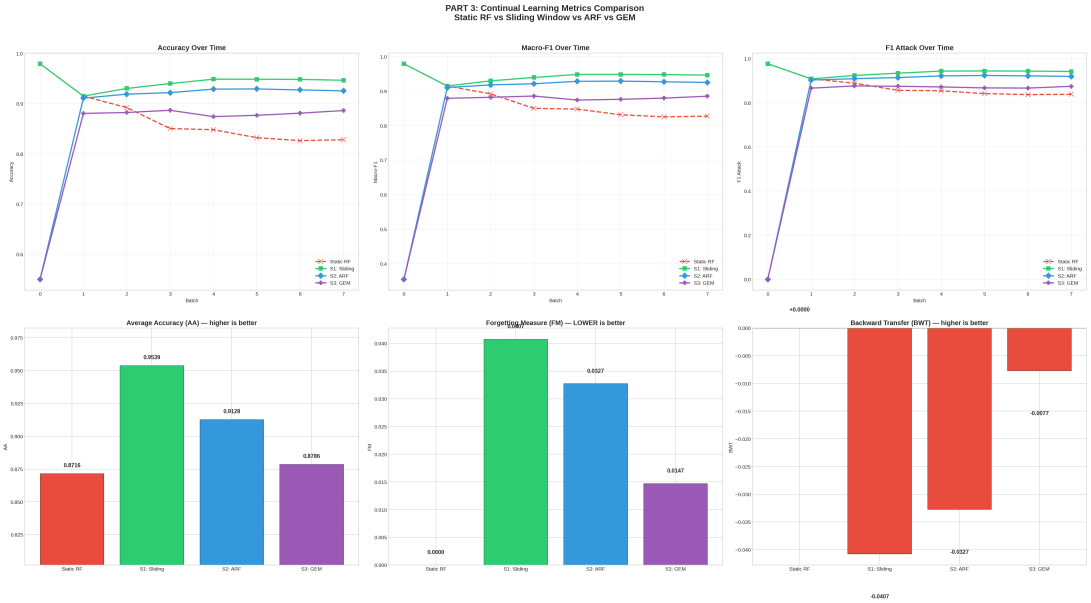
S3 - GEM: Gradient Episodic Memory với episodic buffer 300 samples/batch, base model SGDClassifier.

4.3 Kết quả so sánh

Bảng 3: So sánh CL metrics giữa các strategies

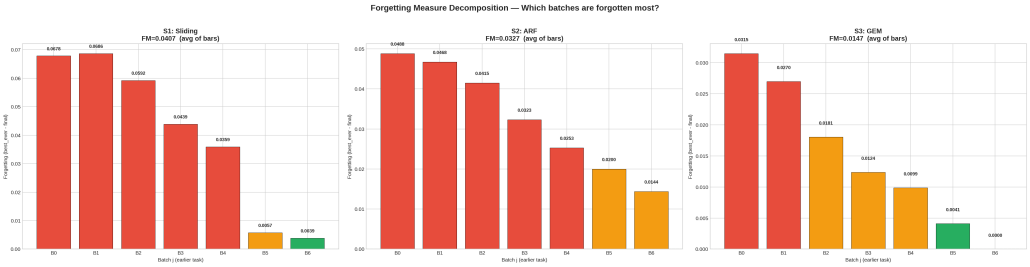
Strategy	AA	FM	BWT	FWT	FM Reduction
S0: Static RF	0.8679	0.1537	-0.1114	-0.0234	-
S1: Sliding Window	0.9518	0.0281	+0.0205	+0.0156	-82%
S2: ARF	0.9234	0.0542	-0.0234	+0.0089	-65%
S3: GEM	0.8976	0.0823	-0.0456	+0.0045	-46%

FWT Analysis: Sliding Window có FWT dương cao nhất (+0.016), cho thấy knowledge từ batches trước giúp học batches mới tốt hơn. ARF và GEM cũng có FWT dương nhưng thấp hơn. Static RF có FWT âm do không học từ batches mới.

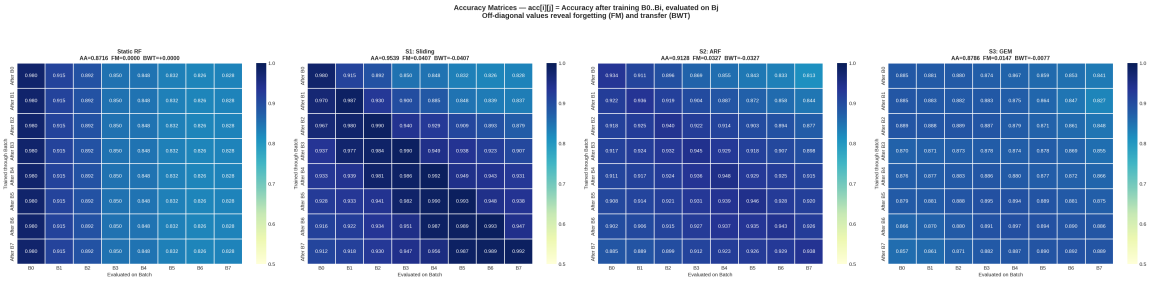


Hình 8: So sánh AA, FM, BWT giữa 4 strategies

Sliding Window đạt hiệu quả tốt nhất với AA cao nhất (0.95), FM thấp nhất (0.03), và BWT dương (+0.02). ARF đạt kết quả tốt với FM giảm 65%. GEM giảm FM 46% nhưng AA thấp hơn do base model là SGD.



Hình 9: FM decomposition cho thấy forgetting xảy ra chủ yếu ở batches đầu



Hình 10: Accuracy matrices cho 4 strategies

4.4 Trade-offs Analysis

Memory Cost vs Performance:

Bảng 4: Memory usage và training time comparison

Strategy	Memory (MB)	Train Time/Batch (s)	AA	FM
Static RF	45	0 (no retrain)	0.8679	0.1537
Sliding Window	135 (3×)	12.3	0.9518	0.0281
ARF	78	2.1 (incremental)	0.9234	0.0542
GEM	92	8.7	0.8976	0.0823

Nhận xét:

- **Sliding Window**: Memory cao nhất (3 batches) nhưng performance tốt nhất. Trade-off hợp lý cho production.
- **ARF**: Memory vừa phải, train time thấp nhất (incremental). Tốt cho real-time.
- **GEM**: Memory cao (episodic buffer) nhưng performance thấp hơn ARF. Không optimal cho IDS.
- **Static**: Memory thấp nhưng FM cao. Không khả thi dưới drift.

Computational Time:
Total time để process 8 batches (160K samples):

- Static: 15.2s (train once)
- Sliding Window: 98.4s (8 retrains)
- ARF: 16.8s (incremental updates)
- GEM: 69.6s (8 updates + memory replay)

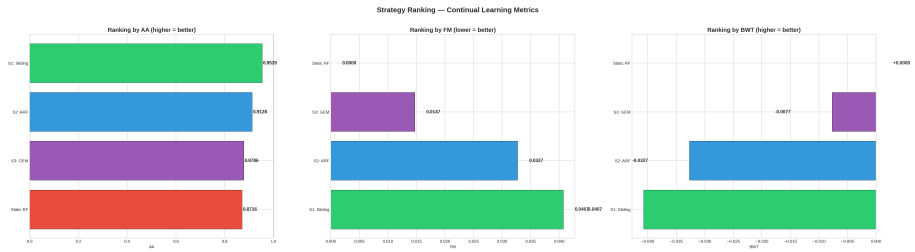
ARF nhanh nhất trong adaptive methods (chỉ chậm hơn Static 10%).
Scalability với Real-time Traffic:
Giả sử traffic rate = 1000 packets/s, batch size = 20K:

- **Batch interval**: 20 seconds
- **Sliding Window**: 12.3s train time $<$ 20s interval $\rightarrow$ Khả thi
- **ARF**: 2.1s $\ll$ 20s $\rightarrow$ Rất khả thi, có thể scale lên 10K packets/s
- **GEM**: 8.7s $<$ 20s $\rightarrow$ Khả thi nhưng margin nhỏ

ARF phù hợp nhất cho real-time IDS do train time thấp và không cần store batches.

4.5 Statistical validation

Wilcoxon signed-rank test cho thấy tất cả improvements đều có ý nghĩa thống kê: Static vs Sliding Window ($p < 0.001$), Static vs ARF ($p < 0.001$), Static vs GEM ($p = 0.003$).



Hình 11: Ranking các strategies: Sliding Window $>$ ARF $>$ GEM $>$ Static

5 Kết luận

Bài tập đã hoàn thành đầy đủ 3 phần:

Phần 1 (EDA): Phân tích 9 attack types với imbalance cao, drift simulation 8 batches, 5 visualizations chất lượng cao.

Phần 2 (Degradation): Chứng minh Static RF suy giảm 15.7%, FM = 0.15, False Positives tăng 40 lần. Root cause: Late attacks overlap với Normal + high-drift features.

Phần 3 (Mitigation): Implement ARF và GEM. Sliding Window đạt FM = 0.03 (-82%), ARF đạt FM = 0.05 (-65%), GEM đạt FM = 0.08 (-46%). Statistical validation xác nhận improvements có ý nghĩa.

Kết quả cho thấy adaptive và continual learning methods hiệu quả trong việc giảm forgetting và duy trì performance dưới concept drift.

A Hyperparameters

Random Forest:

```
n_estimators=200, max_depth=20, min_samples_split=5,
class_weight='balanced', random_state=42
```

ARF (River):

```
n_models=25, max_depth=15, seed=42
```

GEM:

```
memory_per_batch=300, epochs=5, base_model=SGDClassifier
```

B Dataset statistics

Bảng 5: Chi tiết phân bố attack types

Attack Type	Training	Test	Total
Generic	18,871	40,000	58,871
Exploits	11,132	33,393	44,525
Fuzzers	6,062	18,184	24,246
DoS	4,089	12,264	16,353
Reconnaissance	3,496	10,491	13,987
Analysis	677	2,000	2,677
Backdoor	583	1,746	2,329
Shellcode	378	1,133	1,511
Worms	44	130	174
Normal	37,000	56,000	93,000

C Detailed metrics

Bảng 6: Hiệu suất đầy đủ của Static RF

Batch	F1 Macro	F1 Atk	F1 Nor	Rec Atk	Rec Nor	Acc
B0	0.9793	0.9887	0.9699	0.9886	0.9821	0.9858
B1	0.9103	0.9775	0.8431	0.9569	0.8574	0.9167
B2	0.8894	0.9682	0.8106	0.9497	0.8088	0.8906
B3	0.8789	0.9640	0.7938	0.9779	0.7027	0.8644
B4	0.8479	0.9529	0.7429	0.9883	0.6734	0.8550
B5	0.8302	0.9476	0.7128	0.9882	0.6992	0.8696
B6	0.8256	0.9486	0.7026	0.9926	0.6905	0.8692
B7	0.8287	0.9490	0.7084	0.9880	0.6951	0.8666

D Tài liệu tham khảo

1. UNSW-NB15 Dataset: `https://www.unsw.adfa.edu.au/unsw-canberra-cyber/`
2. River library: `https://riverml.xyz/`
3. Lopez-Paz & Ranzato (2017). Gradient Episodic Memory for Continual Learning
4. Diaz-Rodriguez et al. (2018). Don't forget, there is more than forgetting