Comparative Review of Deep Learning Models for Malware Detection Under Adversarial Concept Drift

Idayana Alabere [1], Mirin Henry Anthony [2]
Department of Computer Science and Informatics
Federal University Otuoke, Otuoke, Bayelsa State
alabereii@fuotuoke.edu.ng, mirinhenry299@gmail.com
https://doi.org/10.56201/ijcsmt.vol.12.no10.2026.pg138.153

Abstract
Malware writers keep changing their programs so that security tools cannot catch them. Deep-learning detectors may work very well in the laboratory but become less reliable when they meet new and changed threats in real life. This study presents a comparative review of deep-learning models for malware detection under adversarial concept drift. Adversarial concept drift means deliberate changes in the nature or behaviour of malware that slowly reduce the reliability of a detector. Using a structured review method, the study searched digital libraries, conference archives and a preprint server, coded 20 studies published between 2018 and 2025, and checked their quality with six questions. It then compared nine families of deep-learning models and seven adaptation strategies using an ordinal rating on six criteria. Eighteen studies passed the quality threshold and formed the core set. Fifteen of the 20 studies tested their models over time, but only three described an attacker, and none combined deliberate attack with testing over time. Evidence on drift for deep models exists only for feed-forward networks, one autoencoder or contrastive encoder, and one graph neural network, so five of the nine families have no such evidence. Feed-forward networks ranked first under equal weighting (3.33), while graph neural networks ranked first when robustness against attack was counted twice (3.29). Drift detection (3.67) and feature stabilisation (3.50) were the best adaptation strategies. How well a model recovers after a long deliberate attack has not been tested. The findings favour methods that need few labels and that are tested over time, and this is useful for organisations in Nigeria.

Keywords—*Adversarial concept drift, Concept drift, Cybersecurity, Deep learning, Malware detection.*

I. Introduction

Malware attacks are now more common and more advanced. They threaten information security, digital infrastructure and the world economy. Malware is harmful software made to damage the confidentiality, integrity and availability of computer systems, networks and information. It can be used to steal private data, disturb operations and give attackers access without permission (Saracino et al., 2018). Detection has become more difficult because attackers use code obfuscation, encryption, dynamic execution and repackaging of applications to hide from security tools (Liu et al., 2020). It is therefore not enough to judge a detection system by its first accuracy. We must also ask whether it stays reliable as threats change with time (Deldar & Abadi, 2023; Saad et al., 2019).

Signature-based detection compares a suspicious file with signatures of known malware. It works well for known threats, but it often fails when the malware is new or has been modified, because its signature is not yet recorded (Bensaoud et al., 2024). Classical machine-learning methods, such as decision trees, random forests and support vector machines, depend on good training data and on a stable link between features and classes (Arp et al., 2022). For this reason, deep learning has become attractive, since it can learn directly from executable files, API-call sequences, network traffic and runtime behaviour (Gibert et al., 2020). However, deep-learning models can still be fooled by deliberate manipulation, and they may perform poorly when new inputs are very different from the training data (Bensaoud et al., 2024).

One major threat to reliability is concept drift. This happens when the data, or the link between input features and the target classes, changes with time (Lu et al., 2019). In malware detection, drift occurs when attackers bring new families, change how programs run or use new ways of evading detection. It increases both false negatives and false positives. Adversarial concept drift should not be confused with ordinary adversarial examples. An adversarial example is an input that has been changed on purpose to cause a wrong prediction, while drift is about how data or feature–class relationships change over time (Biggio & Roli, 2018; Lu et al., 2019). In this study, adversarial concept drift means deliberate changes in malware behaviour that slowly reduce the reliability of a detector. A model may resist single adversarial examples and still decline when the change continues for long, so we must check both resistance to attack and performance over time.

Ways of adapting to change include periodic retraining, online learning and drift detection. Lu et al. (2019) note that adaptive learning is an important response to changing data. Still, each method has its limits. Retraining is costly when labels are few, online learning suffers when labels come late or are misleading, and drift detectors may give false alarms or react too late. A fair comparison must therefore look at drift detection, adaptation and recovery, using tests over time and measures beyond accuracy (Pendlebury et al., 2019). Unfortunately, standard benchmarks are hard to find, and this makes comparison across datasets and platforms difficult (Bensaoud et al., 2024).

This paper compares deep-learning models for malware detection under adversarial concept drift. It explains how nine families of models read a program, how they age, and how an attacker can disturb them. It compares seven adaptation strategies in terms of detection, adaptation and recovery, and it points out the gaps that make fair comparison difficult. The problem is very important in Nigeria, because Kshetri (2019) reports that cybercrime is growing in Africa while the capacity to respond remains small. Figure 1 shows how detection approaches have developed, and this is the reason for the review.

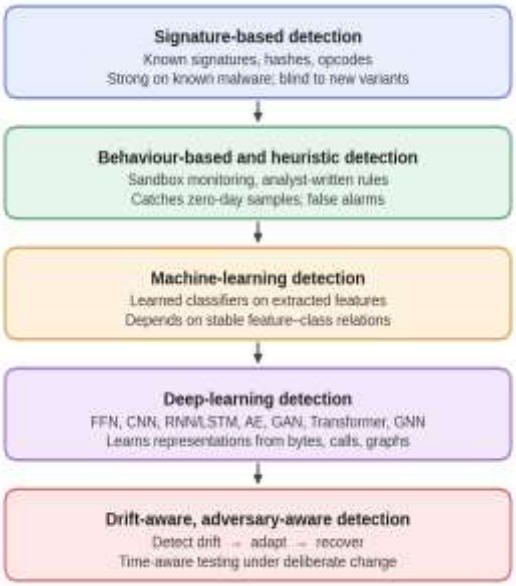


Figure 1. Evolution of Malware Detection Approaches

II. Methodology

A. Review design

This study is a qualitative comparative review. It follows structured review steps and the reporting principles of the PRISMA 2020 statement (Page et al., 2021). A review was chosen because the aim is to compare existing models and strategies, not to build a new detector. Also, the published studies differ in datasets, time span, attacker assumptions and metrics, so combining their numbers statistically would not be sound. Integrative reviews are suitable for such mixed evidence (Snyder, 2019). No experiment was run and no malware sample was collected. Three questions guided the work. RQ1 asks how the model families compare in reliability over time, in noticing change in behaviour, and in exposure to manipulation. RQ2 asks how adaptation strategies compare in keeping or restoring performance. RQ3 asks which gaps make fair comparison difficult.

B. Search and selection

Evidence was collected from IEEE Xplore, the ACM Digital Library, ScienceDirect, SpringerLink, USENIX proceedings and arXiv (marked as not peer-reviewed). Google Scholar and backward and forward snowballing were also used. The Boolean search strings joined four groups of ideas (malware detection, deep learning, drift or change over time, and adversarial manipulation or adaptation). The search covered 2018 to 2026, except for foundational sources. A source was kept if it dealt with malware detection, or a method that can be transferred to it, under distribution shift; if it concerned deep-learning models or techniques that apply to them; if it was peer-reviewed or an archival report from a recognised body; if it was written in English; and if it described its data and evaluation well enough to be coded. Opinion pieces, non-malware studies, duplicates and sources without described data or results were removed. Selection was done in three passes (Figure 2).

C. Appraisal, coding and rating

Each study was checked with six quality questions adapted from common review guidelines (Page et al., 2021). These ask whether the dataset and collection period are described (Q1), whether the split is by time (Q2), whether metrics beyond accuracy are reported (Q3), whether an attacker model is stated (Q4), whether adaptation or drift handling is tested (Q5), and whether the work can be reproduced (Q6). A score of 1, 0.5 or 0 was given, and sources below 2.5 out of 6 were kept only as background. Thematic coding was then used (Snyder, 2019) and gave five themes: model family, drift-detection method, adaptation method, attacker assumption and evaluation design. Each model family and strategy was then rated from 1 to 5 on six criteria fixed before coding: drift sensitivity (C1), label economy (C2), adversarial robustness (C3), knowledge retention (C4), practical feasibility (C5) and malware-specific evidence (C6).

Numbers from the original studies are reported as the authors gave them. They were never recalculated or ranked across studies. The ordinal ratings are the reviewer's judgement about design features and reported evidence, and should not be taken as measured performance. Evidence was read using common measures (accuracy, precision, recall, F1-score and false-positive rate). Precision and recall were treated as more informative than accuracy when classes are unequal. Drift-related measures such as detection delay, performance drop, adaptation speed and recovery time were also considered (Bayram et al., 2022). To protect validity, the criteria were fixed before coding, the coding was repeated after some time, and a sensitivity check counted adversarial robustness twice. Only published material was used, so no ethical clearance was needed.

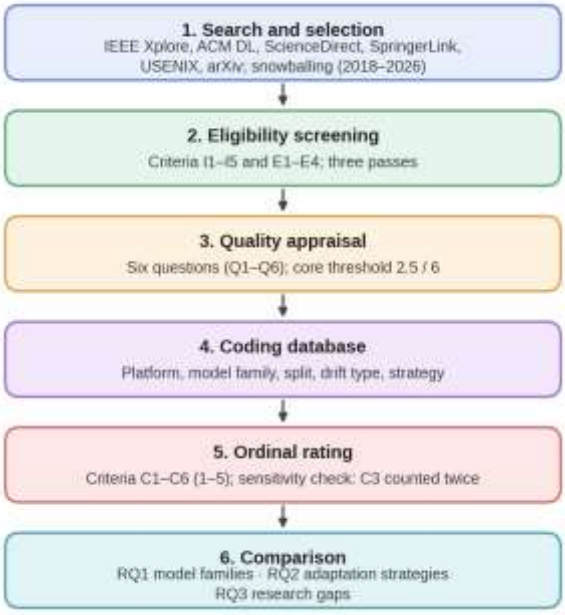


Figure 2. Comparative Framework for Evaluating Deep-Learning Malware Detectors under Adversarial Concept Drift

III. Comparative Review of Existing Approaches

A. From signatures to deep learning

Static signature-based detection matches files against a database of known signatures. It finds known malware with few false alarms, but it fails when no matching signature exists (Aslan & Samet, 2020). Polymorphism, packing and obfuscation allow authors to change how a program looks and pass through signature matching (Bacci et al., 2018; Or-Meir et al., 2019). Behaviour-

based detection watches what a program does, so it can catch zero-day malware (Or-Meir et al., 2019). However, it gives more false alarms and needs more training time (Apruzzese et al., 2023). Heuristic detection depends on rules written by analysts, which take time and skill to confirm (Aslan & Samet, 2020), and hybrid methods that join static and dynamic evidence can be heavy on computing resources. Machine learning removes much of this manual work by learning from labelled malware and benign samples (Ucci et al., 2019), and deep learning goes further by learning layered representations with little manual feature design (Gibert et al., 2020).

B. Deep-learning model families

Nine families of models are commonly discussed, and each one reads a program in its own way. Feed-forward networks need a fixed-length vector, such as the byte histogram, string statistics and header fields in the EMBER feature set (Anderson & Roth, 2018). Al-Dujaili et al. (2018) used adversarial training to make such a network stronger. Convolutional networks read the binary as a greyscale image (Vasan et al., 2020) or as raw bytes (Raff et al., 2018). Yet Kreuk et al. (2018) showed that a few well-chosen bytes can change the decision without changing how the file behaves. Recurrent networks read events in order, and Rhode et al. (2018) reported classifying an executable within its first five seconds with 94% accuracy. LSTM and GRU networks solve the vanishing-gradient problem of simple recurrent networks and have been used on API-call sequences (Catak et al., 2020). Bidirectional types, as in DeepCatra (Wu et al., 2022), need the whole sequence, so they suit offline analysis. All these sequence readers can be fooled by padding with harmless calls.

Autoencoders learn mostly from benign files and flag any sample that is rebuilt poorly (Pang et al., 2021). A carefully made sample may still be rebuilt well, and the benign class also drifts. Generative adversarial networks are used mainly as the attacker's tool. They can create evasive malware against black-box detectors, although Kim et al. (2018) used transferred GANs to detect zero-day malware. Transformers replace recurrence with self-attention and did better than an LSTM baseline on API-call sequences (Demirkıran et al., 2022), but evidence under deliberate change is mostly missing. Graph neural networks read control-flow or call graphs (Yan et al., 2019). It is harder to rewrite program structure than to add bytes, and Li et al. (2024) joined graph networks with adversarial domain adaptation to learn features that do not change with drift.

C. Adversarial concept drift and manipulation

Ordinary drift comes from natural changes in data, while adversarial drift comes from deliberate manipulation (Lu et al., 2019). Attackers make changes that keep the program's function but alter what the classifier sees, so the harmful purpose remains (McCarthy et al., 2022). Malware also grows like a data stream whose distribution keeps shifting (Ceschin et al., 2023). Drift can be gradual, sudden or recurring (Lu et al., 2019), and Guerra-Manzanares et al. (2022) traced how the importance of Android system-call features changed over seven years.

Four ways of manipulation appear again and again in the literature. First, static file features can be edited, for example through reinforcement-learning-based evasion (Anderson et al., 2018) or black-box attacks that keep functionality (Demetrio et al., 2021; Ling et al., 2023). Second, polymorphic and metamorphic variation changes the code while keeping the harm (Aslan & Samet, 2020; Gibert et al., 2020). Third, behavioural mimicry makes runtime activity look like normal operations (Or-Meir et al., 2019). Fourth, the training and feedback process can be poisoned (Cinà et al., 2023; Severi et al., 2021). Successive releases may combine these tricks, so that a classifier grows old as newer variants appear (Chen et al., 2023). Random train–test splits

can hide this fall in performance, but testing in time order reveals it (Barbero et al., 2022; Pendlebury et al., 2019).

D. Representative studies

Table 1 summarises eight representative studies. Detector papers such as MalConv were tested on static splits that ignore time. Evaluation papers such as TESSERACT showed that published detectors lose performance under realistic time conditions. Drift or adaptation papers study natural drift but not an attacker who knows the system.

Table 1. Comparative Summary of Representative Studies

Study	Approach	Key contribution	Limitation
Raff et al. (2018)	Raw-byte CNN (MalConv)	End-to-end learning from executables without feature engineering	Static splits; no adversary
Anderson & Roth (2018)	EMBER dataset; boosted-tree baseline	About 1.1 million PE files; baseline beat MalConv	Not built for adversarial drift
Pendlebury et al. (2019)	TESSERACT evaluation framework	Time-aware testing; detector performance dropped sharply	Natural drift only; no defence
Xu et al. (2019)	Online model with pseudo-labels (DroidEvolver)	Updates itself without true labels	Later found to degrade quickly
Zhang et al. (2020)	API-semantics features	Slowed ageing of Android classifiers	Android only; natural drift
Kolosnjaji et al. (2018)	Byte-appending attack on MalConv	Few appended bytes lowered detection	White-box; one architecture
Yang et al. (2021)	Contrastive encoder (CADE)	Detects and explains drifting samples	Natural drift only
Chen et al. (2023)	Continuous active learning	Sustained performance with few labels	Assumes trusted labels; Android only

IV. Comparative Analysis of Models and Adaptation Strategies

A. Model families

Table 2 shows what each family reads and what an attacker must change to escape it. Reliability over time can be judged only where studies used splits by time. Among deep-learning families, such evidence exists only for feed-forward networks, an autoencoder or contrastive encoder, and a graph network. When published Android detectors, some of them deep, were tested again under realistic time conditions, their performance dropped sharply (Pendlebury et al., 2019). In contrast, the F1-scores of 96 to 99 per cent reported by Karbab et al. (2018) show performance when training and test samples come from the same period. They cannot tell us how long a detector stays reliable.

Table 2. What Each Model Family Reads and What an Attacker Must Change

Family	What it reads	Attacker must change	n
Feed-forward	Fixed feature vector	Feature values; add features	4
CNN	Image or raw bytes	Appended bytes; packer layout	4
RNN	Ordered API calls	Insert or reorder calls	0
BRNN	Sequence, both directions	Pad both sides	0
LSTM / GRU	Long gated sequences	Benign padding; mimicry	0
Autoencoder	Reconstruction error	Craft samples that reconstruct well	1
GAN	Generator and discriminator	Mostly the attacker's tool	0
Transformer	Attention over sequence	Untested under deliberate shift	0
Graph NN	Control-flow or call graph	Rewrite structure	1

n = number of corpus studies centred on the family.

Five studies focused on detecting drift (Barbero et al., 2022; Ceschin et al., 2023; Guerra-Manzanares et al., 2022; Herzog et al., 2025; Yang et al., 2021). Only Yang et al. used a deep model, all of them dealt with natural drift, and none considered an attacker who knows the detector. Label-free methods are attractive to organisations that cannot label many files, but Barbero et al. showed that their success depends on the classifier and the nonconformity measure. An attacker can also study methods that judge credibility or distance, and then shape samples to look reliable. Three attack studies (Al-Dujaili et al., 2018; Kolosnjaji et al., 2018; Kreuk et al., 2018) show that small changes that keep behaviour can alter the decision of a deep detector. The only defence in the corpus, adversarial training, reduced evasion only against the attacks used in training (Al-

Dujaili et al., 2018). Wider work agrees: where bytes are placed affects attack success (Suciu et al., 2019), neural and some commercial detectors were misled by added content (Fleshman et al., 2018), and black-box evasion of Android detectors is possible (Bostani & Moonsamy, 2024). Many published defences fail once the attacker adapts to them (Tramèr et al., 2020). A high robustness rating is therefore fair only where a defence has been tested against an adaptive attacker (Carlini et al., 2019).

B. Adaptation strategies

Table 3 compares seven strategies. Periodic retraining is the oldest remedy, but it needs full relabelling and may forget older families. Active learning chooses which samples analysts should label, and Chen et al. (2023) kept detection stronger over a multi-year stream than ordinary retraining, at much lower labelling cost. Pseudo-label learning needs no new ground truth. After Xu et al. (2019) proposed it, however, Kan et al. (2021) found that it degraded quickly because the model learned from its own mistakes. Feature stabilisation looks for features that remain useful. Zhang et al. (2020) added API semantics to slow ageing, Li et al. (2024) used graph-based domain adaptation, and Fernando and Komninos (2022) applied feature selection to ransomware detection under drift.

Table 3. Adaptation Strategies Compared

Strategy	New labels	Main risk	Score
Drift detection	None to partial	Detects but does not repair; may be gamed	3.67
Feature stabilisation	None to partial	Does not react once change is large	3.50
Active learning	Partial	Assumes labels are trustworthy	3.33
Pseudo-label learning	None	Self-poisoning feedback loop	3.17
Continual learning	Partial	Vision methods transfer poorly	3.00
Adversarial training	Full	Protects mainly against training attacks	3.00
Periodic retraining	Full	Costly; forgetting; poisonable labels	2.67

Score = composite ordinal rating (1–5), equal weights.

Continual learning protects old knowledge, but methods developed for computer vision do not transfer directly to malware classification (Rahman et al., 2022). Malware drift is mostly a domain change, in which labels stay the same but inputs move (van de Ven et al., 2022). Adversarial training is the only strategy that directly targets evasion, but it protects mainly against the attacks used in training and does not respond to natural change. Drift detection gives the signal to act, but it does not repair the model by itself.

V. Results

A. Profile of the corpus

The coded corpus has 20 studies published between 2018 and 2025. Six appeared in 2018, the first wave of work on byte-level detectors and attacks, and 11 (55%) appeared from 2021 to 2025, a period led by drift detection and adaptation. Android is the most studied platform (11 studies, 55%), followed by Windows executables (7, 35%) and multi-platform studies (2). Only 10 studies centre on a named deep-learning family: four on CNNs, four on feed-forward networks, one on an autoencoder or contrastive encoder and one on a graph network. One study evaluates several algorithms, and nine use non-neural classifiers that supply methods for drift detection and adaptation that can be transferred.

Splits by time were used in 15 studies (75%), random splits in four and k-fold validation in one. All 14 studies of natural drift used splits by time, but all three studies of deliberate manipulation used random splits. Four studies are preprints (Herzog et al., 2025; Kreuk et al., 2018; Li et al., 2024; Sabbah et al., 2025) and are marked as not peer-reviewed.

B. Quality appraisal

The mean quality score was 3.98 out of 6. Two studies (Karbab et al., 2018; Raff et al., 2018) scored 2.0, below the 2.5 threshold, and were kept as background only. The remaining 18 form the core corpus, with a mean of 4.19. Their low scores come from their purpose and not from poor work, because they were written to show that networks can learn from raw bytes or API sequences, not to answer questions about time or attackers. Table 4 shows that reporting of datasets and metrics is strong, but an attacker model is stated in only three of the 20 studies. This agrees with the concern of Cavallaro et al. (2023) about whether machine-learning detectors are ready for real use.

Table 4. Mean Appraisal Score by Question

Question	Mean (0–1)	Reading
Q1 Dataset and period described	0.88	Strong
Q2 Time-aware split	0.72	Fairly strong
Q3 Metrics beyond accuracy	0.88	Strong
Q4 Adversary model stated	0.15	Very weak
Q5 Adaptation evaluated	0.62	Moderate
Q6 Reproducible	0.72	Fairly strong

C. Ordinal comparison

Figure 3 shows the composite ratings. Under equal weighting, feed-forward networks rank first (3.33), followed by autoencoders and graph networks (3.17 each). CNNs and LSTM or GRU networks share fourth place (2.50), RNNs, GANs and transformers follow (2.17), and bidirectional networks are last (1.67). Feed-forward networks gain from low cost and from the amount of drift

evidence available. When adversarial robustness is counted twice, graph networks move to first place (3.29), ahead of feed-forward networks (3.14) and autoencoders (3.00), because program structure is harder to rewrite than bytes or feature values. Transformers rise from sixth to joint fifth (2.29), but this only reflects the hope that pre-training improves robustness (Hendrycks et al., 2019), which is yet to be shown for malware. The low ratings of the recurrent families come from missing drift evidence and heavier cost, and not from proven poor performance.

For adaptation strategies, drift detection ranks first (3.67), then feature stabilisation (3.50), active learning (3.33) and pseudo-label learning (3.17). Continual learning and adversarial training tie at 3.00, and periodic retraining is last (2.67). When robustness is counted twice, drift detection and feature stabilisation tie for first (3.43), adversarial training joins active learning in third place (3.14), and pseudo-label learning falls to joint fifth (2.86). No study reported a full recovery curve for a deep-learning model after a deliberate attack, so recovery time could not be compared in numbers.

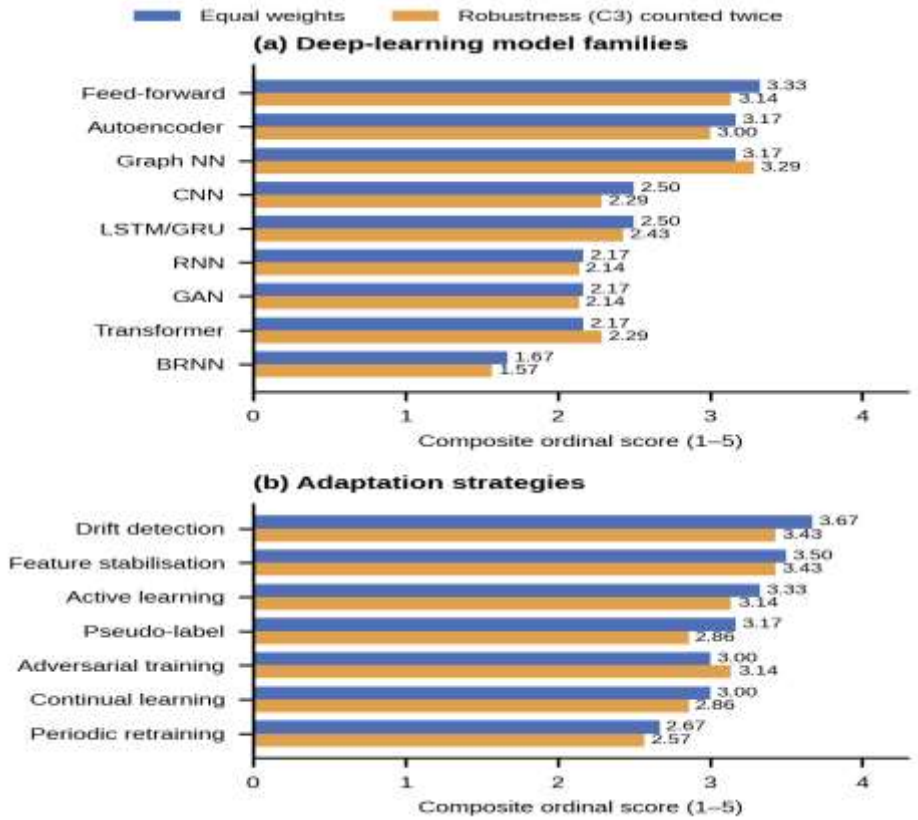


Figure 3. Composite Ordinal Scores of (a) Model Families and (b) Adaptation Strategies

D. Research gaps

Six gaps stand out. First, no study tested a deep-learning model under deliberate manipulation and testing over time together. Second, recurrent, bidirectional, LSTM or GRU, GAN and transformer families have no drift evidence. Third, none of the 14 natural-drift studies described an attacker who adapts to the detector or its update process. Fourth, the three attack studies used fixed attacks at one point in time. Fifth, Android makes up 55% of the corpus and no study used Nigerian data.

Sixth, datasets, split rules, drift simulation and metrics differ, so only ordinal comparison is possible and not a pooled ranking.

VI. Discussion

The findings can be understood through three theories. Concept drift theory says that the link between data and labels changes over time, and that drift may be sudden, gradual or recurring (Lu et al., 2019). The evidence supports this and adds that the speed of ageing depends on what a model reads. Families that read weak surface features, such as appended bytes or raw feature values, are more exposed than those that read structure. Adversarial machine-learning theory describes a continuing contest between attacker and defender (Biggio & Roli, 2018), and it builds on the finding that small input changes can mislead neural networks. The reviewed evidence has studied this contest in one direction only, since attack studies fix the defender in time while drift studies fix the attacker. This is the central gap in the field. Catastrophic forgetting theory warns that learning new data may erase old knowledge (van de Ven et al., 2022). Few studies report performance on older families after an update, so knowledge retention was rated mainly from design features. From the ratings, a sensible combination is a label-free drift detector as the trigger, active learning to choose samples for analysts, feature stabilisation to slow decay between updates, and adversarial training used selectively for high-risk detectors. This is a reasoned reading of the evidence, and it has not been tested as a package.

In Nigeria, cybercrime is rising across Africa while the capacity to respond is limited (Kshetri, 2019). No study in the corpus used Nigerian data, so the published figures may not describe local traffic and software. The benign class on local mobile banking applications and unlicensed software may also differ from that of foreign datasets. Because analysts are few and labelling is costly, strategies that need few labels (drift detection, active learning and feature stabilisation) suit local organisations better than full periodic retraining. Every label that enters retraining should also be checked, because the update path is itself a target. Large banks and telecommunication companies may combine a feed-forward or autoencoder core with a graph model for high-risk files. Mid-sized firms may use a feed-forward network on stable features with drift alerts. Smaller institutions are better served by managed detection services that supply drift alerts and update schedules. This review has some limitations. Few studies examine adversarially induced drift in deep detectors, datasets and metrics differ, most evidence comes from outside Nigeria, and the ordinal ratings are judgements and not measurements.

VII. Conclusion and Future Research Directions

This comparative review shows that deep-learning malware detectors are not safe from ageing. Models that look excellent under random testing lose strength on later samples, so laboratory accuracy alone does not prove that a detector will stay reliable after deployment. What a model reads decides how its ages and how it can be attacked, and graph-based models look promising although their evidence is still small. The evidence is uneven, since five of the nine families have no drift evidence, and attackers and time have been studied separately, so adversarial concept drift has hardly been tested on deep detectors. Strategies that need few labels give the best balance of benefit, cost and evidence. No single strategy is enough, and recovery after a long deliberate attack remains an open question. Future research should (i) report results on splits by time, with stated dates and with metrics suited to unequal classes; (ii) state the attacker model, including what the attacker knows about the detector and its update process; (iii) test defences against adaptive attackers and study sustained manipulation across successive releases; (iv) examine recurrent,

bidirectional, LSTM or GRU, GAN and transformer families under drift; (v) develop continual-learning methods made for malware and report how well older families are retained; (vi) report recovery time and labelling effort; and (vii) build time-stamped Nigerian datasets with proper ethical and legal approval, and extend coverage to Internet of Things and mobile-money malware.

References

Al-Dujaili, A., Huang, A., Hemberg, E., & O'Reilly, U.-M. (2018). Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)* (pp. 76–82). IEEE. https://arxiv.org/abs/1801.02950

Anderson, H. S., Kharkar, A., Filar, B., Evans, D., & Roth, P. (2018). *Learning to evade static PE machine learning malware models via reinforcement learning* (arXiv:1801.08917). arXiv. https://arxiv.org/abs/1801.08917

Anderson, H. S., & Roth, P. (2018). *EMBER: An open dataset for training static PE malware machine learning models* (arXiv:1804.04637). arXiv. https://arxiv.org/abs/1804.04637

Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A. V., & Di Franco, F. (2023). The role of machine learning in cybersecurity. *Digital Threats: Research and Practice, 4*(1), Article 8. https://doi.org/10.1145/3545574

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., & Rieck, K. (2022). Dos and don'ts of machine learning in computer security. In *Proceedings of the 31st USENIX Security Symposium* (pp. 3971–3988). USENIX Association.

Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access, 8*, 6249–6271.

Bacci, A., Bartoli, A., Martinelli, F., Medvet, E., Mercaldo, F., & Visaggio, C. A. (2018). Impact of code obfuscation on Android malware detection based on static and dynamic analysis. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)* (pp. 379–385). SciTePress. https://doi.org/10.5220/0006642503790385

Barbero, F., Pendlebury, F., Pierazzi, F., & Cavallaro, L. (2022). Transcending Transcend: Revisiting malware classification in the presence of concept drift. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy* (pp. 805–823). IEEE.

Bayram, F., Ahmed, B. S., & Kassler, A. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems, 245*, Article 108632. https://doi.org/10.1016/j.knosys.2022.108632

Bensaoud, A., Kalita, J., & Bensaoud, M. (2024). A survey of malware detection using deep learning. *Machine Learning with Applications, 16*, Article 100546.

Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition, 84*, 317–331. https://doi.org/10.1016/j.patcog.2018.07.023

Bostani, H., & Moonsamy, V. (2024). EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection. *Computers & Security, 139*, Article 103676.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., & Kurakin, A. (2019). *On evaluating adversarial robustness* (arXiv:1902.06705). arXiv. https://arxiv.org/abs/1902.06705

Catak, F. O., Yazı, A. F., Elezaj, O., & Ahmed, J. (2020). Deep learning based sequential model for malware analysis using Windows exe API calls. *PeerJ Computer Science, 6*, Article e285. https://doi.org/10.7717/peerj-cs.285

Cavallaro, L., Kinder, J., Pendlebury, F., & Pierazzi, F. (2023). Are machine learning models for malware detection ready for prime time? *IEEE Security & Privacy, 21*(2), 53–56.

Ceschin, F., Botacin, M., Gomes, H. M., Pinagé, F., Oliveira, L. S., & Grégio, A. (2023). Fast & Furious: On the modelling of malware detection as an evolving data stream. *Expert Systems with Applications, 212*, Article 118590. https://doi.org/10.1016/j.eswa.2022.118590

Chen, Y., Ding, Z., & Wagner, D. (2023). Continuous learning for Android malware detection. In *Proceedings of the 32nd USENIX Security Symposium* (pp. 1127–1144). USENIX Association.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., & Roli, F. (2023). Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys, 55*(13s), Article 294. https://arxiv.org/abs/2205.01992

Deldar, F., & Abadi, M. (2023). Deep learning for zero-day malware detection and classification: A survey. *ACM Computing Surveys, 56*(2). https://doi.org/10.1145/3605775

Demetrio, L., Biggio, B., Lagorio, G., Roli, F., & Armando, A. (2021). Functionality-preserving black-box optimization of adversarial Windows malware. *IEEE Transactions on Information Forensics and Security, 16*, 3469–3478.

Demirkıran, F., Çayır, A., Ünal, U., & Dağ, H. (2022). An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. *Computers & Security, 121*, Article 102846. https://doi.org/10.1016/j.cose.2022.102846

Fernando, D. W., & Komninos, N. (2022). FeSA: Feature selection architecture for ransomware detection under concept drift. *Computers & Security, 116*, Article 102659.

Fleshman, W., Raff, E., Zak, R., McLean, M., & Nicholas, C. (2018). Static malware detection & subterfuge: Quantifying the robustness of machine learning and current anti-virus. In *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)* (pp. 1–10). IEEE.

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications, 153*, Article 102526. https://doi.org/10.1016/j.jnca.2019.102526

Guerra-Manzanares, A., Luckner, M., & Bahsi, H. (2022). Android malware concept drift using system calls: Detection, characterization and challenges. *Expert Systems with Applications, 206*, Article 117200. https://doi.org/10.1016/j.eswa.2022.117200

Hendrycks, D., Lee, K., & Mazeika, M. (2019). Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 97, pp. 2712–2721). PMLR.

Herzog, A., Eusebi, A., & Cavallaro, L. (2025). *On the reliability and stability of selective methods in malware classification tasks* (arXiv:2505.22843). arXiv. https://arxiv.org/abs/2505.22843

Kan, Z., Pendlebury, F., Pierazzi, F., & Cavallaro, L. (2021). Investigating labelless drift adaptation for malware detection. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec '21)* (pp. 123–134). ACM. https://doi.org/10.1145/3474369.3486873

Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2018). MalDozer: Automatic framework for Android malware detection using deep learning. *Digital Investigation, 24*, S48–S59. https://doi.org/10.1016/j.diin.2018.01.007

Kim, J.-Y., Bu, S.-J., & Cho, S.-B. (2018). Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. *Information Sciences, 460–461*, 83–102.

Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., & Roli, F. (2018). Adversarial malware binaries: Evading deep learning for malware detection in executables. In *Proceedings of the 26th European Signal Processing Conference* (pp. 533–537). IEEE.

Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., & Keshet, J. (2018). *Deceiving end-to-end deep learning malware detectors using adversarial examples* (arXiv:1802.04528). arXiv. https://arxiv.org/abs/1802.04528

Kshetri, N. (2019). Cybercrime and cybersecurity in Africa. *Journal of Global Information Technology Management, 22*(2), 77–81. https://doi.org/10.1080/1097198X.2019.1603527

Li, A. S., Iyengar, A., Kundu, A., & Bertino, E. (2024). *Revisiting concept drift in Windows malware detection: Adaptation to real drifted malware with minimal samples* (arXiv:2407.13918). arXiv. https://arxiv.org/abs/2407.13918

Ling, X., Wu, L., Zhang, J., Qu, Z., Deng, W., Chen, X., Qian, Y., Wu, C., Ji, S., Luo, T., Wu, J., & Wu, Y. (2023). Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art. *Computers & Security, 128*, Article 103134.

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of Android malware detection approaches based on machine learning. *IEEE Access, 8*, 124579–124607. https://doi.org/10.1109/ACCESS.2020.3006143

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering, 31*(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

McCarthy, A., Ghadafi, E., Andriotis, P., & Legg, P. (2022). Functionality-preserving adversarial machine learning for robust classification in cybersecurity and intrusion detection domains: A survey. *Journal of Cybersecurity and Privacy, 2*(1), 154–190.

Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys, 52*(5), Article 88.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., . . . Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ, 372*, Article n71. https://doi.org/10.1136/bmj.n71

Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys, 54*(2), Article 38.

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2019). TESSERACT: Eliminating experimental bias in malware classification across space and time. In *Proceedings of the 28th USENIX Security Symposium* (pp. 729–746). USENIX Association.

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. K. (2018). Malware detection by eating a whole EXE. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press. https://arxiv.org/abs/1710.09435

Rahman, M. S., Coull, S., & Wright, M. (2022). On the limitations of continual learning for malware classification. In *Proceedings of the First Conference on Lifelong Learning Agents* (Proceedings of Machine Learning Research, Vol. 199, pp. 564–582). PMLR.

Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *Computers & Security, 77*, 578–594. https://doi.org/10.1016/j.cose.2018.05.010

Saad, S., Briguglio, W., & Elmiligi, H. (2019). The curious case of machine learning in malware detection. In *Proceedings of the 5th International Conference on Information Systems Security and Privacy* (pp. 528–535). SciTePress.

Sabbah, A., Jarrar, R., Zein, S., & Mohaisen, D. (2025). *Empirical evaluation of concept drift in ML-based Android malware detection* (arXiv:2507.22772). arXiv. https://arxiv.org/abs/2507.22772

Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2018). MADAM: Effective and efficient behavior-based Android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing, 15*(1), 83–97.

Severi, G., Meyer, J., Coull, S., & Oprea, A. (2021). Explanation-guided backdoor poisoning attacks against malware classifiers. In *Proceedings of the 30th USENIX Security Symposium* (pp. 1487–1504). USENIX Association.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research, 104*, 333–339. https://doi.org/10.1016/j.jbusres.2019.07.039

Suciu, O., Coull, S. E., & Johns, J. (2019). Exploring adversarial examples in malware detection. In *2019 IEEE Security and Privacy Workshops (SPW)* (pp. 8–14). IEEE.

Tramèr, F., Carlini, N., Brendel, W., & Madry, A. (2020). On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1633–1645). Curran Associates.

Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security, 81*, 123–147. https://doi.org/10.1016/j.cose.2018.11.001

van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence, 4*, 1185–1197. https://doi.org/10.1038/s42256-022-00568-3

Vasan, D., Alazab, M., Wassan, S., Safaei, B., & Zheng, Q. (2020). Image-based malware classification using ensemble of CNN architectures (IMCEC). *Computers & Security, 92*, Article 101748.

Wu, Y., Shi, J., Wang, P., Zeng, D., & Sun, C. (2022). *DeepCatra: Learning flow- and graph-based behaviors for Android malware detection* (arXiv:2201.12876). arXiv. https://arxiv.org/abs/2201.12876

Xu, K., Li, Y., Deng, R. H., Chen, K., & Xu, J. (2019). DroidEvolver: Self-evolving Android malware detection system. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 47–62). IEEE. https://doi.org/10.1109/EuroSP.2019.00014

Yan, J., Yan, G., & Jin, D. (2019). Classifying malware represented as control flow graphs using deep graph convolutional neural network. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 52–63). IEEE.

Yang, L., Guo, W., Hao, Q., Ciptadi, A., Ahmadzadeh, A., Xing, X., & Wang, G. (2021). CADE: Detecting and explaining concept drift samples for security applications. In *Proceedings of the 30th USENIX Security Symposium* (pp. 2327–2344). USENIX Association.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M., & Yang, M. (2020). Enhancing state-of-the-art classifiers with API semantics to detect evolved Android malware. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (pp. 757–770). ACM. https://doi.org/10.1145/3372297.3417291