IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2020, accepted July 16, 2020, date of publication July 22, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011293

Learning to Learn Sequential Network Attacks Using Hidden Markov Models

TIMOTHY CHADZA[1,2], (Member, IEEE), KONSTANTINOS G. KYRIAKOPOULOS[1], (Member, IEEE), AND SANGARAPILLAI LAMBOTHARAN[1], (Senior Member, IEEE)

[1]Wolfson School of Mechanical, Electrical, and Manufacturing Engineering, Loughborough University, Loughborough LE11 3TU, U.K.
[2]Department of Electrical Engineering, University of Malawi-Polytechnic, Blantyre, Malawi

Corresponding author: Konstantinos G. Kyriakopoulos (k.kyriakopoulos@lboro.ac.uk)

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), U.K., through the Communications Signal Processing-Based Solutions for Massive Machine-to-Machine Networks (M3NETs) Project, under Grant EP/R006385/1.

ABSTRACT The global surge of cyber-attacks in the form of sequential network attacks has propelled the need for robust intrusion detection and prediction systems. Such attacks are difficult to reveal using current intrusion detection systems, since each individual attack phase may appear benign when examined outside of its context. In addition, there are challenges in building supervised learning models for such attacks, since there are limited labelled datasets available. Hence, there is a need for updating already built models to specific operational environments and for addressing the concept drift. A hidden Markov model (HMM) is a popular framework for sequential modelling, however, in addition to the above challenges, the model parameters are difficult to optimise. This paper proposes a transfer learning (TL) approach that exploits already learned knowledge, gained from a labelled source dataset, and adapts it on a different, unlabelled target dataset. The datasets may be from a different but related domain. Five unsupervised HMM techniques are developed utilising a TL approach and evaluated against conventional machine learning approaches. Baum-Welch (BW), Viterbi training, gradient descent, differential evolution (DE) and simulated annealing, are deployed for the detection of attack stages in the network traffic, as well as, forecasting both the next most probable attack stage and its method of manifestation. Specifically, for the prediction of the three next most likely states and observations, TL with DE achieved a maximum accuracy improvement of 48.3%, and 27.4%, respectively. Finally, the actual detection prediction for the three next most probable states and methods of manifestation reaches 78.9% and 96.3% using TL with BW and DE, respectively.

INDEX TERMS Transfer learning, hidden Markov model, Viterbi decoding, forward-backward, sequential network attacks.

I. INTRODUCTION

Sequential network attacks (SNAs) may exist as advanced persistent threats (APTs) or multi-stage network attacks (MSAs) and are executed in a series of steps; however, the individual steps may either be benign or malicious. Intrusion detection systems (IDSs), a mandatory line of defence in a network, may be unable to detect these attacks due to time variation between attack stages, which may span a long time [1]. Generally, a cyber-attack can be represented as a diamond model [2], an attack graph or a kill chain [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

Advanced, sequential cyber threats have recently seen a resurgence, due to the emergence of the Internet of Things and the increase in the number of interconnected devices [4]. Cybersecurity Ventures [5] forecasts that cybercrime will incur a cost of over $6 trillion annually by 2021, hence the drive for robust detection and prediction systems is essential.

Essentially, conventional machine learning (CML) techniques can be utilised to improve the responsiveness of detection and prevention models in dealing with security threats. Efforts towards forecasting sequential network attacks are in progress and hidden Markov model (HMM) is a popular machine learning technique that is being adopted [6]. HMM has a tractable mathematical formalisation and it utilises state

transition and emission probability distributions to recognise SNAs.

Parameter learning in HMM context is a major problem due to the unavailability of labelled datasets in some domains, which is frequent problem in cyber-security. In other situations, the dataset may be insufficient, outdated or may require complex computations to train the model. In addition, there is a need for adapting models to specific environments, for learning alternative tasks (multi-task learning) and for addressing concept drift [7]. Transfer Learning (TL) addresses these challenges by reusing prior knowledge, acquired from a source domain, towards the development of a model in a target domain.

TL is currently experiencing significant interest and is expected to be the next value driver of machine learning (ML) after supervised learning (SL), as mentioned in [8]. TL methods have successfully been applied in various real-world domains, however, they have been less explored in sequential analysis [9]. Using TL, HMM parameters can be transferred from the source to the target domain in an effort to enhance the task of the target domain [10]. The source domain model can either be used in its entirety or reconstituted to suit the target domain.

This work addresses the challenge of detecting and forecasting SNAs in computer networks given observations from the captured network traffic. However, in contrast to CML approaches, TL has been deployed in order to leverage already learned knowledge from models built on the source domain and use them as a starting point to efficiently develop models in the target domain.

For classification purposes in the target domain, five popular HMM learning algorithms, namely, Baum Welch (BW), Viterbi training (VT), differential evolution (DE), gradient descent (GD), and simulated annealing (SA) are deployed and their performance is compared against the conventional HMM approaches.

To evaluate the efficacy of using TL over CML, the DARPA 2000 [11] dataset has been utilised, since it is well documented and illustrates a typical MSA scenario. In addition, a recent dataset, CICIDS2018 [12], has been analysed to further highlight the performance benefits of TL, when applied on modern cyber-attack sequences. Snort IDS [13], a popular, open-source signature-based solution, has been used in both the source and target domains to generate alerts, which are then mapped to HMM observations. Overall, the contributions of this paper are as follows:

- A comprehensive analysis for applying TL in HMMs for detection and prediction of SNAs. Algorithms adapted for TL purposes include both local (BW, VT and GD) and global (DE and SA) optima solutions. Evaluation results include: detecting all states (AS), current state (CS), next state (NS), and next observation (NO) using three levels of accuracy.
- A total of 48 distinct experimental scenarios, covering 15 HMM training combinations and five core algorithms have been analysed on both TL and CML. The five core algorithms are BW, VT, DE, GD and SA. Furthermore, 38 multiple runs were performed on individual stochastic training combinations.
- Analysis of data sampling techniques for splitting dataset for training and evaluation. In addition to the sequential sampling technique, uniform sampling has been introduced, as well as sample injection for both the sequential and uniform sampling. This will ensure consistency in the observation symbols between the training and evaluation datasets.

The rest of the sections of this paper are consecutively arranged as follows: Section II reviews the TL related work in general followed by closely related work on TL and HMM. In Section III both the SL and unsupervised learning (UL) HMM approaches have been outlined. The experimental setup is presented in Section V followed by the methodology in Section VI and then the results and discussions in Section VII. The paper concludes in Section VIII.

II. RELATED WORK

Despite the popularity of TL, to the best of our knowledge, no previous work has investigated sequence-based analysis of network attacks using TL in HMM. For this reason, general prior work on TL and HMM is presented to appreciate the advancements in those generic fields. Authors in [14] attest to the fact that TL has been investigated in various applications, particularly, in deep learning, where pre-trained models are reused for visual recognition and natural language processing. They presented a transductive TL approach for classifying unseen attacks utilising prior knowledge from known attacks and reinforced the necessity for a vigorous exploration of TL. In contrast, this article addresses detection and forecasting of states and observations from sequence analysis.

The fundamentals of TL have been well articulated in a survey by authors in [10] and this served as the building block in identifying the practicability of TL for SNA detection and prediction. Three different settings of TL that were identified included the inductive, transductive and unsupervised approaches categorised with respect to the variations in domains and tasks for both source and target. The paper reveals that TL has, generally, been limited to small scale applications excluding network traffic analysis and the fact that the unsupervised TL approach has not been explored in depth.

Authors in [9] presented a new perspective regarding the mismatch that exists between the source and target domains. They proposed a sub-structural regularisation TL model aimed at preserving target domain features. The work integrated HMM and regularisation theory and was validated in part-of-speech tagging using textual corpora.

The work by authors in [15] proposes a new approach to effectively evaluate the amount of knowledge that should be transferred from source to target domain. They demonstrated the efficacy of their approach in clustering, co-transfer

TABLE 1. The standard representation of the HMM parameters where N is number of states, M is number of unique observations, s_i is the unique state i, q_t is the state at time t, v_k is the probability of observing symbol k. Additionally, the limiting conditions for the parameters have been provided.

Parameter	Symbol	Probability notation	Matrix notation	Condition
Initial state vector	$\pi = \{\pi_i\}$	$P(q_1 = s_i)$	$\begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_N \end{bmatrix}$	$\sum_{i=1}^{N} \pi_i = 1$
State transition matrix	$A = \{a_{ij}\}$	$P(q_{t+1} = s_j \mid q_t = s_i)$	$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_N & \dots & a_{NN} \end{bmatrix}$	$\sum_{j=1}^{N} a_{ij} = 1\ , i \in [1, N]$
Observation matrix	$B = \{b_j(v_k)\}$	$P(o_t = v_k \mid s_t = q_j)$	$\begin{bmatrix} b_1(v_1) & b_1(v_2) & \dots & b_1(v_M) \\ b_2(v_1) & b_1(v_2) & \dots & b_2(v_M) \\ \dots & \dots & \dots & \dots \\ b_N(v_1) & b_N(v_2) & \dots & b_N(v_M) \end{bmatrix}$	$\sum_{k=1}^{M} b_j(v_k) = 1, j \in [1, N]$

learning and classification tasks. The results highlighted the performance enhancement of heterogeneous TL and future research direction includes the statistical modelling of source domain features and determining the optimum features.

Authors in [16] proposed using TL in HMM for capturing anomalous states of nodes at the physical substrate in a network slicing scenarios, based on observations from the virtual nodes. The BW algorithm, was modified to transfer knowledge between two physical nodes.

With respect to research work in conventional HMMs, there are numerous efforts that have been developed to address MSA and advanced persistent threats (APTs). Notable work includes [17] where the authors developed a framework for real-time MSA prediction. They conducted HMM parameter-estimation using both UL and SL approaches. Even though the SL was able to detect three out of the five DARPA 2000 attack phases, the UL was unable to detect any attack phase. This highlights the drive for vigorous approaches to improve the prediction mechanisms for UL, as labelled datasets, may not always be available.

Furthermore, the work in [4] develops an alert correlation framework for predicting APTs using BW for forecasting the next step of an attack campaign based on a synthetic dataset. The work in [1] attempts to address both the state and observations detection and prediction of a MSA based on DARPA 2000 dataset. The work highlighted that the performance of BW and VT may significantly be improved following the proposed hybrid parameter learning approaches.

Overall, the reviewed work in this section has demonstrated the significant impact that TL can yield when introduced in learning models. These works highlight the need for vigorous efforts towards overhauling CML techniques. Presumably, it is anticipated that HMM learning can be improved beyond the current state-of-the-art of BW and expectation maximisation. Furthermore, research should not only focus on state, but also on observation forecasting.

III. HIDDEN MARKOV MODEL

A HMM has been defined in [1] as two level stochastic process, where the first level represents the unobserved states of a modelled system and the second level represents the observations or emissions obtained from this system. Primarily, the HMM parameters are normally represented as a 3-tuple, $\lambda = (A, B, \pi)$, where A, B and π denote the initial state vector, state transition matrix and the observation matrix, respectively. These parameters as described in [18], have been illustrated in Table 1, while considering a set, $S = \{s_1, s_2, \dots, s_N\}$, of N states, a set, $V = \{v_1, v_2, \dots, v_M\}$, of M distinct number of observation symbols, and a sequence, $O = \{o_1, o_2, \dots, o_T\}$, of T observations, where $o_i \in V$ and $i \in 1 \dots T$.

There are three fundamental problems in HMM [18]: i) estimation of the observation sequence probability, ii) determination of the model that represents a given observation sequence, and iii) the decoding of the hidden states that best represent an observation sequence, given the model. Of these three problems, representing an observation sequence with the appropriate HMM model, which is referred to as learning or training or parameter estimation, remains a major HMM challenge. In the next subsection, both supervised and unsupervised HMM learning techniques are described.

A. SUPERVISED LEARNING

In SL approaches, the training data is labelled, and in the case of HMM, this means that the prior knowledge of both the observation and hidden states is utilised to estimate the HMM parameters, $\lambda = (A, B, \pi)$. In most SL HMM implementations, it is a common practice to assume the initial state to be state of the first observation symbol, s_1, hence estimation usually focuses only on A and B matrices. However, the computation of π is also discussed, to avoid any assumptions about the initial state. In [17], the authors illustrated how the HMM parameters are estimated based on the maximum likelihood estimate, which is practically obtained by simply counting the number of transitions and observations in each HMM state.

Specifically, each element, π_i, of the initial state probability vector, π, is calculated as the number of times, $\pi(i)$, that the state is s_i, divided by the total number of elements in the state sequence, T, as shown in (1).

$$\pi_i = \frac{\pi(i)}{T} \tag{1}$$

Equation (2), shows how each element of A, a_{ij}, is computed as the number of transitions from state i to j, $a(i, j)$,

divided by the total number of transitions from state i.

$$a_{ij} = \frac{a(i,j)}{\sum_{j=1}^{N} a(i,j)} \tag{2}$$

On the other hand, each element of B, $b_j(v_k)$, is obtained by counting the number of occurrences of the observation v_k in state j, $b(j, v_k)$, divided by the total number of observations in state j as in (3).

$$b_j(v_k) = \frac{b(j,v_k)}{\sum_{k=1}^{M} b(j,v_k)} \tag{3}$$

B. UNSUPERVISED LEARNING

In unsupervised learning (UL) approach, the training data is unlabelled, and for HMM, this implies that only the observation sequence is known, without any state knowledge. Computation of π, A and B can be achieved through two possible ways. First way is by obtaining the most probable state sequence of a given observation sequence using Viterbi decoding and then finding the maximum likelihood estimate of both A and B. Second way is random initialisation and re-estimation of A and B using parameter learning algorithms, such as BW, VT, DE, GD and SA. These algorithms have been described in the remaining sections.

1) BAUM-WELCH ALGORITHM

The BW algorithm is an expectation maximisation approach customised for the estimation of HMM parameters. It uses forward-backward algorithm to estimate π, A and B following a two-step expectation maximisation process, firstly in the expectation step or E-step, the likelihood is obtained using the current parameters, and secondly, in the maximisation step or M-step new parameters are re-estimated while optimising the expected likelihood [19].

The BW algorithm has been described in details in [18], including how HMM parameters are re-estimated, furthermore, the pseudocode has been presented in [1]. The backward and forward variables are the two most pertinent parameters in the proposed work and are briefly discussed as follows:

a) *Forward variable*, $\alpha_t(i)$: This is the probability of encountering the partial observations, $o_1, o_2, ..o_t$ at time t, while in state i, and for initialisation ($t = 1$), induction ($1 < t \leq T$) and termination ($t = T$), $\alpha_t(i)$ is obtained as in (4), (5) and (6), respectively.

$$\alpha_1(i) = \pi_i b_j(o_1) \tag{4}$$

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij} \tag{5}$$

$$P(O_T|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{6}$$

b) *Backward variable*, $\beta_t(i)$: This is the probability of the remaining partial observation sequence, $o_{t+1}, o_{t+2}, ..o_T$, while in state i at time T, given the HMM parameters, λ [20]. For initialisation and induction, $\beta_t(i)$ is obtained as in (7) and (8), respectively.

$$\beta_T(i) = 1 \tag{7}$$

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) b_j(o_{t+1}) a_{ij} \tag{8}$$

2) VITERBI TRAINING

The VT algorithm, also called segmental K-means, uses the Viterbi decoding algorithm to obtain the estimates of states which are then used alongside the observation sequence to determine the HMM parameters. It is therefore necessary to first describe the Viterbi algorithm process which is key to the VT.

The Viterbi algorithm is similar to the forward-backward algorithm except that at each node, instead of computing all the available paths, only the best or Viterbi path is selected and used in the next computation. Two parameters are introduced, first, $\alpha_t(i)$ which represents the Viterbi path and second, the $\psi_t(i)$ parameter to track the prior state that maximises the likelihood towards all states and at each time instance [4].

The VT pseudocode is illustrated in Algorithm 1 and in particular, both the Viterbi algorithm and SL are used to determine the state sequence Q and λ, respectively. This process is repeated until a convergence threshold is met or the number of maximum iterations is exceeded. Lines 3-11 shows the recursive process for VT.

Algorithm 1 Viterbi Training Pseudocode

Input: O, λ, maximum iterations G, convergence threshold Th
Output: $\hat{\lambda}$

1: Obtain the likelihood $P(O|\lambda)$ from λ and O
2: $Iter = 1$
3: **repeat**
4: Obtain Q using O and λ through Viterbi algorithm
5: Use Q and O to obtain $\hat{\lambda}$ through SL
6: Obtain new likelihood $P(O|\hat{\lambda})$ from $\hat{\lambda}$ and O
7: $\epsilon = |P(O|\hat{\lambda}) - P(O|\lambda)|$
8: $\lambda = \hat{\lambda}$
9: $P(O|\lambda) = P(O|\hat{\lambda})$
10: $Iter = Iter + 1$
11: **until** $\epsilon > Th \vee Iter > G$

3) DIFFERENTIAL EVOLUTION

The DE is a global optimisation algorithm that undergoes a three-step process of genetic mutation, crossover, and selection [21]. Authors in [1] describe the DE process as follows:

a) *Transformation of initial* λ: This corresponds to the transformation of the initial λ into single-dimensional vectors.

b) *Differential mutation*: This involves the computation of mutants or new individuals through the utilisation of three randomly selected vectors.

c) *Differential crossover*: This uses a predetermined crossover rate to generate a trial vector by recombining mutant and parent vector [22].
d) *Termination*: The maximum likelihood can be used to replace the parent vectors with new vectors that are closer to the global optima solution [21].

4) GRADIENT DESCENT

The GD algorithm uses a differential function to compute the training parameter of a model based on a defined cost function. In HMM lingua, the cost function, J, is the cross-entropy, defined as the negative log-likelihood, $logP(O|\lambda)$, that an observation sequence O was generated by a model, λ [23]. This is shown in (9).

$$J = -logP(O|\lambda) \tag{9}$$

To begin with, $P(O|\lambda)$ can be represented as a function of forward and backward variables, and HMM parameters as shown in (10).

$$P(O|\lambda) = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_jO_{t+1}\beta_{t+1}(j) \tag{10}$$

In [24], it has been shown that each HMM parameters require the partial differentials of $P(O|\lambda)$ and these have been derived as shown in (11), (12), and (13), for A, B and π, respectively.

$$\frac{\partial P(O|\lambda)}{\partial a_{ij}} = \sum_{t=1}^{T-1}\alpha_t(i)b_jO_{t+1}\beta_{t+1}(j) \tag{11}$$

$$\frac{\partial P(O|\lambda)}{\partial b_j(v_k)} = \sum_{t\ni O_t=v_k}\sum_{i=1}^{N}\alpha_t(i)a_{ij}\beta_{t+1}(j) + \delta(O_1, v_k)\pi_j + \beta_1(j) \tag{12}$$

$$\frac{\partial P(O|\lambda)}{\partial \pi_i} = b_iO_1\beta_1(i) \tag{13}$$

In (12), the symbol δ represents the Kronecker delta function, which generates a zero when the two elements being compared are different and a one if otherwise. Therefore, $\delta(O_1, v_k)$ is represented as shown in (14):

$$\delta(O_1, v_k) = \begin{cases} 0, & O_1 \neq v_k \\ 1, & O_1 = v_k \end{cases} \tag{14}$$

Authors in [23] further highlight how the λ can be optimised using the cost function. For instance, each element of re-estimated A, $\hat{a}_{ij}$ is determined by (15). Both B and π are obtained similarly. Parameter η is the step size or learning rate, and was set to 0.001. The transition parameter is updated as in (15):

$$\hat{a}_{ij} = a_{ij} - \eta\frac{\partial J}{\partial a_{ij}} \tag{15}$$

GD's pseudocode is illustrated in Algorithm 2 where line 1 obtains the cost function. Lines 3-12 recursively re-estimate the new HMM parameters using partial derivatives.

Algorithm 2 Gradient Descent Pseudocode

Input: O, λ, step size η, maximum iterations G, convergence threshold Th
Output: $\hat{\lambda}$

1: Compute J_0 from $P(O|\lambda)$ as in (9) using O, λ, α, β
2: $Limit$ = FALSE
3: **repeat**
4: Compute the partial derivatives $\frac{\partial J_o}{\partial a_{ij}}$, $\frac{\partial J_o}{\partial b_j v_k}$, and $\frac{\partial J_o}{\partial \pi_i}$ using (11), (12) and (13)
5: Compute $\hat{a}_{ij}$ based on (15), and similarly for $\hat{b}_j v_k$ and $\hat{\pi}$ to obtain $\hat{\lambda}$
6: Compute J_1 from $P(O|\hat{\lambda})$ as in (9) using O and $\hat{\lambda}$
7: **if** $|J0 - J1| <= Th$ **then**
8: $Limit$ = TRUE
9: $Iter = Iter + 1$
10: $\lambda = \hat{\lambda}$
11: $J_0 = J_1$
12: **until** $Limit$ = TRUE $\vee$ $Iter > G$

5) SIMULATED ANNEALING

The SA algorithm is a stochastic global optimisation search algorithm that analogises a metal cooling process to the search for a minimum in a general system [25]. Specifically, the SA utilises a temperature control parameter for searching the global solution and the adaptation of the cooling schedule, thereof, determines its efficiency [26]. Several works have proposed the integration of SA and HMM to provide a self-tuning capability of SA and commonly, SA starts from a random solution, however, the initialisation has been made flexible to permit different types of initialisation which are in turn used for setting up of the cost function. The following steps are used to train an HMM based on SA:

a) *Initialisation*: This involves the generation of prior estimates of the model commonly using random solutions.
b) *Specifying the temperature*: Two control parameters are required: the temperature (control) parameter T_c and temperature reduction rate T_r.
c) *Obtaining the objective function*: The log-likelihood of initial solutions can be adopted as an objective function. Alternatively, various benchmark cost functions have been proposed in [27].
d) *Creation of new neighbourhood structure*: This is also referred to as the perturbation process and begins with the selection of random elements from the old model. In [28], one of the suggested perturbation processes is by adding a random value to the randomly selected elements, and since the rows' sum may exceed unity, normalisation is required at this stage.
e) *Selection of solution*: When the threshold is not met, new solutions are created by selecting a random index in the current solution and adding an exponent parameter as in (16), followed by re-normalisation.

$$\lambda \leftarrow \lambda + \frac{\exp(-\epsilon)}{T_c} \tag{16}$$

f) *Annealing*: The annealing process simply reduces the temperature or control parameter based on the specified reduction rate and is obtained by: (17).

$$T_c \leftarrow T_c \times T_r \tag{17}$$

g) *Termination*: The maximum likelihood can be used to replace the old model parameters with the new parent parameters when a threshold is met, or, the maximum number of iterations is exceeded.

Algorithm 3 outlines the SA pseudocode where the initialisation and the temperature specification are considered as inputs. Line 2 obtains the objective function, which is commonly the negative log-likelihood. The new neighbourhood structure is defined in Lines 4-6. The selection of a solution is presented in lines 9-11 and lastly, the annealing process is shown in line 13.

Algorithm 3 Simulated Annealing Pseudocode

Input: O, λ, temperature parameter T_c, reduction rate T_r, maximum iterations G, convergence threshold Th

Output: $\hat{\lambda}$

1: **for** $Iter \leftarrow 1$ to G **do**
2: Assign the likelihood $P(O|\lambda)$ to F_1
3: $\hat{\lambda} = \lambda$
4: Select random elements in $\hat{\lambda}$
5: Add a random value to the randomly selected elements of $\hat{\lambda}$
6: Normalise rows of $\hat{\lambda}$ so that they sum to unity
7: Assign the likelihood $P(O|\hat{\lambda})$ to F_2
8: $\epsilon = |F_1 - F_2|$
9: **if** $\epsilon > Th$ **then**
10: Select random index in $\hat{\lambda}$ and add $\frac{\exp(-\epsilon)}{T_c}$
11: Normalise rows of $\hat{\lambda}$ such that they sum to unity
12: $\lambda = \hat{\lambda}$
13: $T_c = T_c \times T_r$

IV. TRANSFER LEARNING

A. DOMAINS AND TASKS

Domains and tasks are two fundamental TL concepts that need to be defined before describing the proposed approach. A domain, $\mathcal{D}$, is composed of two elements: a feature space X, and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$ [10]. $\mathcal{X}$ is the space of all possible features, X is a specific learning sample, and x_i is an individual feature. A domain represents the space where the data is defined in and is denoted as $\mathcal{D} = \{\mathcal{X}, P(X)\}$. A source domain $\mathcal{D}_\mathcal{S}$ encompasses training instances, whereas a target domain $\mathcal{D}_\mathcal{T}$ contains testing instances [29].

Given a specific domain $\mathcal{D}$, a task, $\mathcal{T}$, is also represented by two elements: a set of all possible labels $\mathcal{Y}$ and a model or predictive function $f(.)$ that determines a corresponding label based on training data, x, [30]. A task is mathematically represented as a pair $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, where $P(Y|X)$ is the predictive function $f(x)$. Task examples, in the source, $\mathcal{T}_\mathcal{S}$, or target, $\mathcal{T}_\mathcal{T}$, domains include regression, classification, clustering, and dimensionality reduction.

TL is defined as the process of improving the learning of the target predictive function, $f_T(x)$, in $\mathcal{D}_\mathcal{T}$ using the knowledge in $\mathcal{D}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{S}$, or $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{S}$ [10]. If the domains are different, this implies that either $X_S \neq X_T$ or $P_S(X) \neq P_T(X)$.

B. TRANSFER LEARNING TYPES

In a TL scheme, development of a detection or prediction system can be achieved using two common approaches, namely the develop model and pre-trained model [31]. Typically, depending on the relationship between the source and target domains or tasks, there are three TL types. These types are inductive, transductive and unsupervised. Unlike the CML, which shares the same domains and tasks, TL types do not necessarily need to be similar.

The inductive TL setting, as defined in [10], is when the target and source tasks are different, regardless as to whether the source and target domains are similar. Inductive TL has labelled data in $\mathcal{D}_\mathcal{T}$ and depending on the availability of labels in $\mathcal{D}_\mathcal{S}$, it is further divided into self-taught or multi-task learning. Specifically, self-taught inductive TL posses no labelled data in $\mathcal{D}_\mathcal{S}$, whereas multi-task inductive TL has labelled data in $\mathcal{D}_\mathcal{S}$, which permits simultaneous learning.

The transductive TL type is dissimilar to the inductive TL, in that both the source and target tasks are similar, $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$, whereas the domains are different, though related, $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$. Furthermore, the labelled data is available only in $\mathcal{D}_\mathcal{S}$. There exist two specific cases of transductive TL as highlighted in [10]. The first case is when both the source and target domains have different feature spaces, $\mathcal{X}_S \neq \mathcal{X}_D$. The second case occurs when the feature spaces between the source and target domains are similar, $\mathcal{X}_S = \mathcal{X}_D$ though the marginal probability distributions are different, $P_S(X) \neq P_S(X)$.

In the unsupervised TL, there are different, though related, $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$, as well as, $\mathcal{T}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$, and labelled data is unavailable in both domains. Figure 1 summarises the TL settings or configurations using a simple flowchart based on data labelling in both $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$.

C. KNOWLEDGE EXCHANGE CASES

To understand how TL can be implemented in HMMs, it is necessary to briefly describe the four cases of knowledge exchange that may take place between a source and a target domain:

a) *Instance-based*: certain parts of data in the source domain may be reused in the target domain by re-weighting [32].
b) *Feature representation*: learns the suitable features of source domain that may be adapted in the target domain [33].

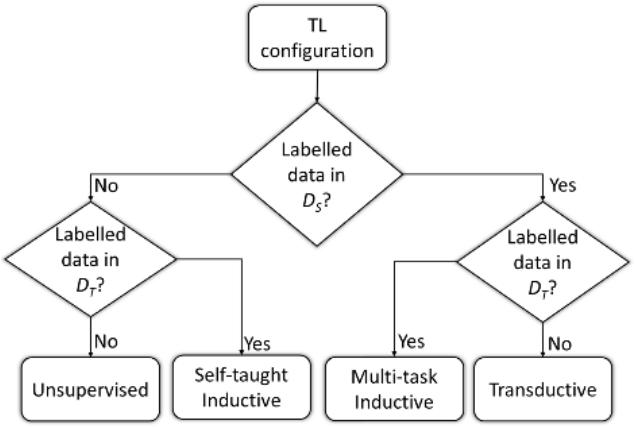

FIGURE 1. An illustration of the transfer learning (TL) configurations with respect to data labelling in both the source and target domain.

c) *Parameter transfer*: common model parameters or prior distributions, such as HMM's λ, may be transferred from the source to the target domain [9].
d) *Knowledge transfer*: considers the existence of some similarities between source and target domains [10].

With regards to the proposed HMM framework, the TL approach with parameter exchange is deployed to leverage the source's HMM model, λ_S, as a staring point for calculating the target's, λ_T. The authors in [15] emphasise that the main research problem in TL is in determining whether a given source domain is suitable in transferring knowledge and determining the amount of knowledge to be transferred.

One intuitive approach would be to transfer more knowledge when the source is closely related to the target, and transfer less knowledge otherwise. To this end, the authors in [16] propose using a rate, $\zeta = 1/t$, inversely to the time instance, t, where t corresponds to a parameter learning iteration. Thus, a modified λ_{new} can be obtained using:

$$\lambda_{new} = \lambda_{new} \times (1 - \zeta) + \lambda_{odd} \times \zeta \tag{18}$$

V. EXPERIMENTAL SETUP

A. OVERVIEW OF EVALUATED DATASETS

Two public datasets that were used in this paper are the DARPA 2000 [11], developed by the Defense Advanced Research Projects Agency (DARPA), and the CSE-CIC-IDS2018 [12], developed by the Communications Security Establishment (CSE) in collaboration with the Canadian Institute for Cybersecurity (CSE-CIC-IDS2018). The DARPA 2000 dataset is a popular MSA comprising of two Distributed Denial of Service (DDoS) scenarios with packets collected from two nodes, namely, the inside and the demilitarised zone (DMZ). In [34], the five-phase MSA is summarised as Internet Protocol (IP) sweeping, Sadmind probing, Sadmind exploit, DDoS software installation and launching.

On the other hand, the CSE-CIC-IDS2018 dataset was formulated to address the over-reliance sub-optimal datasets, for evaluation of intrusion detection and intrusion prevention approaches, which are unreliable and outdated [35]. Unlike the DARPA 2000 dataset which presented an MSA scenario, the CSE-CIC-IDS2018 dataset comprises of seven sequential attack scenarios: Brute-force, Heartbleed, Botnet, Denial of Service (DoS), DDoS, Web attacks, and infiltration of the network from inside [12]. This dataset was formulated to address the lack of reliable, public and recent datasets, which has resulted in anomaly-based IDS approaches suffering from accurate deployment, analysis and evaluation [35].

B. SNORT CONFIGURATION

The acquisition of observations and states for representation required first extracting the essential features from the dataset. In this paper, Snort IDS Version 2.9.11.1 [13] has been deployed in Ubuntu 18.04.1 operating system to generate alerts. MATLAB R2020a programming language was used to extract features from the alerts, remove duplicates and assign observation and state symbols.

1) DARPA 2000

The Snort IDS default configuration, with rules aggregated by PulledPork 0.7.4 [36] has been used to generate alerts. This Snort IDS configuration was unable to detect the first four stages of the DDoS attack, hence, customised rules as described in [34] were introduced to improve the detection accuracy. The rules were extracted from [37] and slightly modified whenever alerts were not generated and this resulted in 2306 alerts for Inside dataset and 2949 alerts for DMZ dataset as shown in Table 2.

TABLE 2. Alerts generated from Snort IDS using both default and customised configuration for DARPA 2000 dataset and only default configuration for CSE-CIC-IDS2018.

Phase	Number of alerts			
	DARPA 2000		CSE-CIC-IDS2018	
	Inside	DMZ	CSE-CIC1	CSE-CIC2
1	40	785	8139	204
2	227	263	101	329
3	74	173	335	295
4	15	12	411	761
5	18	48	1240	79
Total	**2306**	**2949**	**10226**	**1668**

2) CSE-CIC-IDS2018

Unlike in the DARPA 2000, where both the default and customised rules were used, in the CSE-CIC-IDS2018 dataset, the default configuration was used. This is because the default configuration was able to detect the attacks. Furthermore, the dataset contains multiple attacks and it's precise customisation would be beyond the scope of this work. Nevertheless, the obtained alerts are sufficient for HMM construction, and conceptually, in this paper, the individual attacks are

aggregated sequentially as proposed in [38]. The two separate datasets (CSE-CIC1 and CSE-CIC2) are obtained by creating similar attack scenarios. As observed in Table 2, there are five phases which represent DoS (phase 1), DDoS (phase 2), Brute Force-Web, Brute Force-XSS and SQL Injections (phase 3), infiltration-Dropbox download (phase 4) and infiltration-nmap and portscan (phase 5). In addition to having different sizes, the two datasets are carefully selected in order to have different distributions, as an example, the DoS attack in CSE-CIC1 and CSE-CIC2 uses SlowHTTPTest and Hulk tools, respectively.

VI. METHODOLOGY

All the experiments were conducted using MATLAB 2018b software and scripts were written to extract the fields from Snort IDS alerts, pre-process, train and evaluate the CML and TL HMM. Figure 2 illustrates the methodology adopted where it is shown that there are two datasets that were categorised into the source and target domain i.e the Inside and CSE-CIC1 for source domain and the DMZ and CSE-CIC2 for the target domain.

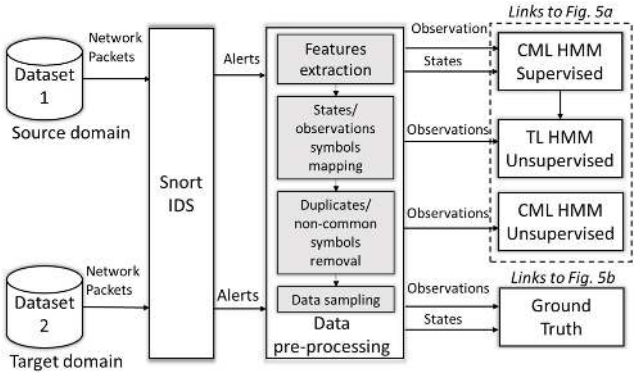

FIGURE 2. Experimental methodology illustrating the deployed transfer learning approach for hybrid supervised and unsupervised hidden Markov model.

A. PRE-PROCESSING

During pre-processing, the alerts are mapped to observation symbols based on the alert description. By following a careful Snort rule design, alert descriptions may have unique Snort IDs, and each ID can simply be mapped to an observation symbol. For DARPA 2000 and CSE-CIC-IDS2018 dataset, the states are assigned as indicated in Section V-A and V-B2, respectively. Thereafter, duplicate alerts based on timestamp, alert category, alert description, IP address, and port number, were removed. Furthermore, in order for the source and target domains to have similar features, non-common observation symbols were removed.

B. OBSERVATION SAMPLING

Data sampling entails splitting each of the phases of a dataset into training and evaluation parts. This was an important task during pre-processing, as it was necessary to ensure similar observation features between training and evaluation datasets. The source domain dataset was used in it's entirety, with assumption of full knowledge at the source domain. The target domain was split in training and evaluation datasets in two different ratios: 30% training and 70% evaluation and inversely.

In this paper, four data sampling techniques are proposed, i) sequential, ii) uniform, iii) sequential with sample injection, and iv) uniform with sample injection. In sequential sampling, consecutive samples from each phase were taken depending on the split ratio (i.e. a phase was not scanned throughout). In uniform sampling, the dataset was uniformly scanned to sample observations throughout its whole duration, i.e. from the beginning to the end. In sequential sampling with injection, the training and evaluation datasets are compared and missing observations, in any dataset, are replaced with observations from the full dataset. The same approach of injecting observations is performed for uniform sampling with injection.

C. POST-PROCESSING

During post-processing, a CML-based HMM of the source domain is constructed utilising full knowledge of the domain, i.e. both the observation and state sequences, with an SL approach (see Figure 3a). The SL CML approach ensures two points. Firstly, the full knowledge in the source domain was used, including states. Secondly, all examined TL methods had a common starting model, since a UL CML technique would not converge to the same λ model. For the target domain, the UL approaches for both CML and TL are implemented using only the observation sequences and a percentage of the target dataset. In TL, the parameter transfer approach is adopted, therefore, the resulting UL CML model is used as a starting point before fine-tuning the target model, as illustrated in Figure 3a.

This introduces a bias in favour of TL, since the number of UL CML iterations are already embedded in the transferred model. To address this, fixed-point iteration (FPI) is deployed, whereby the UL CML approach uses the same number of iterations to that of TL. For each iteration the trained parameters from the previous iteration are recursively used to initialise the current training parameters. The same training data is used at each iteration. The formula for the maximum number of iterations for UL CML, $Maxiter_{CML}$, is shown in (19) where $Size_{CML}$ and $Size_{TL}$ are the lengths of the training data points in UL CML and TL, respectively.

$$Maxiter_{CML} = \frac{Size_{TL} \times Maxiter_{TL}}{Size_{CML}} \tag{19}$$

In total, 16 different combinations of experiments are possible and are presented in Table 3. An analysis in the results section will highlight the best experimental setting among the evaluated parameters: use of FPI, data sampling technique, training/testing ratio and HMM learning algorithm.

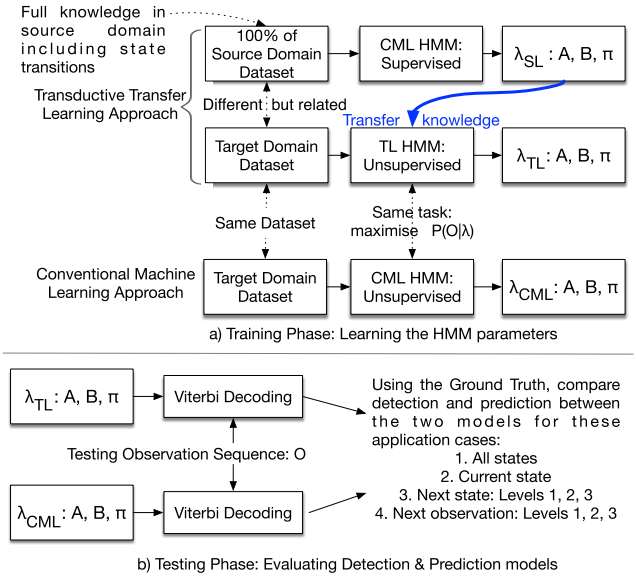

FIGURE 3. a) Training phase: Transfer learning (TL) scheme when compared with conventional machine learning (CML) approach. Note that there is full knowledge in the source domain, hence SL CML is used. The derived model is transferred to the TL framework and then adapted to the Target domain using UL approach. Solid lines represent the diagram's flow, whereas dotted lines represent notional explanations. b) Testing phase: The performance evaluation for both CML and TL was based on the same testing observation sequence.

TABLE 3. Illustration of 16 different experiments performed using TL and CML with and without fixed point iteration (FPI), and using two different ratios for training and evaluation datasets.

No.	Experiment description			
	Data sampling	Training (%)	Evaluation (%)	Iteration
1	Sequential	30	70	FPI
2	Uniform	30	70	FPI
3	Sequential-Injection	30	70	FPI
4	Uniform-Injection	30	70	FPI
5	Sequential	70	30	FPI
6	Uniform	70	30	FPI
7	Sequential-Injection	70	30	FPI
8	Uniform -Injection	70	30	FPI
9	Sequential	30	70	No FPI
10	Uniform	30	70	No FPI
11	Sequential-Injection	30	70	No FPI
12	Uniform -Injection	30	70	No FPI
13	Sequential	70	30	No FPI
14	Uniform	70	30	No FPI
15	Sequential-Injection	70	30	No FPI
16	Uniform-Injection	70	30	No FPI

D. HMM EVALUATION METRICS

For evaluation purposes, four performance metrics have been used and these are accuracy (for prediction detection), Bayesian inference criterion (BIC), mean square error (MSE) and adjusted random index (ARI) (BIC, MSE, ARI for model parameter evaluation). With regards to accuracy, both the detection and prediction estimates are determined based on an estimated window sample size of 150, as was experimentally established in [1]. The window size is shifted by one sample each time until all samples are processed and the results are averaged together. Furthermore, a two-fold cross-validation approach is performed by swapping the training and evaluation datasets and averaging the results. The Viterbi decoding is used for AS detection with the last state used for CS detection. On the other hand, the forward algorithm has been used in conjunction with A and B transition matrices for predicting NS and NO, respectively. The evaluation procedure is illustrated in Figure 3b.

Specifically, as explained in detail in [1], the probability that the NS is j at $t+1$, $P(q_{t+1} = s_j)$, is computed by multiplying the current forward variable, $\alpha_t(i)$, by the corresponding state transition, a_{ij}, as shown in (20). On the other hand, the probability that the NO is v_k at $t+1$, $P(o_{t+1} = v_k)$, is computed by multiplying the previous forward variable, $\alpha_t(i)$ with the respective observation probability, $b_j(v_k)$, as indicated in (21).

$$P(s_{t+1} = s_j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} \tag{20}$$

$$P(o_{t+1} = v_k) = \sum_{j=1}^{N} \alpha_t(j) b_j(v_k) \tag{21}$$

With respect to the MSE, the complete target domain dataset is first trained using SL and the obtained results for the B parameter is compared between the λ_{ideal} and λ_{TL}. Note that only the B parameter has been considered in this paper, as it includes information from both the observations and states.

In addition to MSE, the BIC value has been used instead of the log-likelihood in order to include the penalisation of over-parameterised models as explained in [39]. The BIC is computed as in (22), where $numPar = N^2 + 2N - 1$ and the best HMM is the one with the lowest BIC value,

$$BIC = -2 \times ln(P(O|\lambda)) + numPar \times ln(T) \tag{22}$$

Lastly, the adjusted random index (ARI) score, proposed in [40], addresses the challenge of comparing two different partitions of a finite set of clusters. With respect to HMM, the ARI is a measure of agreement between the estimated sequence of states and the ground truth [41]. The ARI is computed as shown in (23), where Ag represents the number of state agreements, EI is the expected index for adjustment and Ta stands for the total pairs of entities.

$$ARI = \begin{cases} 0, & Ta = EI \\ \dfrac{Ag - EI}{Ta - EI}, & otherwise \end{cases} \tag{23}$$

VII. RESULTS AND DISCUSSION

The following section extensively analyses the results on the DARPA 2000 dataset through 16 experimental settings, as described in Table 3. The best performance among these settings is identified in Sections VII-A1 and VII-A2. A full analysis of the identified optimum experimental settings is presented in Section VII-A3. These settings are also used, in Section VII-B, for analysis of the CSE-CIC-IDS2018 dataset.

A. RESULTS FOR DARPA 2000 DATASET

1) ACCURACY RESULTS FOR ALL APPLICATIONS

The detection of all the states (AS), current state (CS) and the prediction of the next state (NS) and next observation (NO) have been considered. For NS and NO, a maximum of three probable levels have been used, where level 1, level 2 and level 3 consider only the most probable symbol, first two probable symbols, and the three probable symbols, respectively. Results for all applications, where an application refers to AS, CS, next state level 1 (NS1), next state level 2 (NS2), next state level 3 (NS3), next observation level 1(NO1), next observation level 2 (NO2) and next observation level 3 (NO3), are presented in Tables 4 and 5.

Table 4 shows the results for the maximum percentage gain of TL over CML in each application across all the evaluated training techniques. In other words, this indicates which is the highest performance improvement that can be achieved in each application, regardless of the technique (DE, GD, etc). As can be observed, Experiments 6 and 14 have better performance for AS, CS, NS1 and NS2. Experiment 9 had the highest accuracy score for NS3 prediction, whereas Experiments 5 and 13 performed better for all NO predictions. Despite the considerable performance of mentioned experiments, Experiment 16 had the best overall performance. It should be highlighted that the accuracy gain being considered is with respect to TL improvement over CML.

TABLE 4. Experimental results for maximum percentage improvement using TL over CML based on each application: detection of all states (AS) and current state (CS), and prediction of next state (NS) and next observation (NO). Bold values indicate the highest gain for each application.

No.	Maximum gain for application (%)								Overall
	AS	CS	NS1	NS2	NS3	NO1	NO2	NO3	gain (%)
1	55.61	55.54	57.85	54.71	49.37	38.60	43.15	48.64	50.43
2	54.55	68.47	67.89	57.42	49.26	29.09	51.12	63.84	55.21
3	63.91	71.24	65.97	57.53	50.70	38.20	42.94	48.41	54.86
4	54.55	68.47	67.89	55.17	49.26	29.09	51.25	64.74	55.05
5	42.04	33.38	34.60	46.32	41.42	**40.46**	**61.85**	**67.17**	45.91
6	**65.01**	**79.55**	**78.44**	**60.43**	54.05	29.86	51.32	60.93	59.95
7	52.53	49.26	47.11	46.43	41.59	39.97	60.83	66.35	50.51
8	64.94	79.47	78.36	60.36	53.99	29.83	51.26	59.86	59.76
9	55.57	55.65	57.85	51.88	**54.71**	38.60	43.15	48.64	50.75
10	54.55	68.47	67.89	55.17	49.26	29.09	51.12	64.35	54.99
11	54.55	68.47	67.89	55.17	49.26	29.09	51.12	64.35	54.99
12	54.55	68.47	67.89	55.17	49.26	29.09	51.12	63.84	54.92
13	48.09	40.19	38.69	46.32	41.42	**40.46**	**61.85**	**67.17**	48.02
14	**65.01**	**79.55**	**78.44**	**60.43**	54.05	29.86	51.32	60.93	59.95
15	55.60	69.72	61.10	55.72	41.59	39.97	60.83	66.35	56.36
16	64.94	79.47	78.36	60.36	53.99	29.83	51.26	63.80	**60.25**

Table 5 shows the average percentage gain of TL over CML in each application. In other words, these are the averaged results across all techniques for each application. It is observed that Experiment 14 performs better than the rest for AS, CS, and all NS levels. Nevertheless, it should be noted that Experiment 7 performed better for NO1, whereas Experiment 13 performed better for NO2, and NO3 prediction (slightly higher than that of Experiment 14). The overall gain of 34.53% is observed for Experiment 14. Experiment 6 had the second best performance for AS, CS, and all NS levels. Experiments 6 and 14 both used uniform sampling with 70% training for the target domain. The only difference between the two experiments is that Experiment 6 used FPI, whereas Experiment 14 did not. This shows that the introduction of FPI slightly degrades the performance, however, FPI translates to a fair performance comparison between CML and TL, as explained in Section VI-C. Thus, the settings of Experiment 6 would be opted as the most suitable for analysing the application performance.

TABLE 5. Experimental results for average percentage improvement using TL over CML based on application: detection of all states (AS) and current state (CS), and prediction of next state (NS) and next observation (NO). Bold values indicate highest gain for each application.

No.	Average gain for application (%)								Overall
	AS	CS	NS1	NS2	NS3	NO1	NO2	NO3	gain (%)
1	24.03	22.47	27.28	20.86	16.90	13.61	14.54	12.70	19.05
2	36.06	44.15	42.85	37.14	25.42	10.61	15.21	24.11	29.44
3	30.36	29.25	30.71	27.07	18.87	12.26	14.27	14.05	22.11
4	35.75	43.70	41.78	32.79	20.81	11.39	16.98	24.85	28.51
5	20.22	20.00	20.32	20.45	18.56	24.64	27.93	24.19	22.04
6	41.59	50.72	49.44	43.95	32.90	10.82	17.59	23.95	33.87
7	21.79	21.64	20.39	14.43	12.18	**26.13**	27.59	21.90	20.76
8	41.20	49.77	48.00	39.88	32.07	10.73	16.95	24.62	32.90
9	25.21	24.27	25.25	21.84	18.26	13.29	14.29	12.76	19.40
10	34.31	42.62	40.90	33.12	22.17	11.21	17.99	26.21	28.56
11	34.31	42.62	40.90	33.12	22.17	11.21	17.99	26.21	28.56
12	33.43	41.13	39.43	31.51	21.91	11.68	17.35	25.23	27.71
13	20.77	22.09	22.31	22.19	18.82	23.72	**28.26**	**26.56**	23.09
14	**41.85**	**51.15**	**49.99**	**44.04**	**35.26**	11.50	17.16	25.28	**34.53**
15	18.57	23.19	21.64	18.66	16.18	24.47	27.72	23.54	21.75
16	40.94	50.09	47.09	40.05	32.55	11.34	17.63	26.42	33.26

2) ACCURACY RESULTS FOR EACH TECHNIQUE

In addition to analysing the performance of each application, an analysis is made based on individual training techniques (BW, GD, etc). The maximum percentage improvement of TL over CML in each technique across all applications is shown in Table 6. In other words, these are the highest TL performance gains achieved for every technique, regardless of the application scenario.

As observed in Table 6, five techniques, namely uniform BW, uniform DE, uniform GD, count-based DE and count-based SA, performed better with the settings of Experiment 6. Similar performance is observed for Experiment 14 expect for count-based SA. Uniform and count-based VT perform better using the setting of Experiments 8 and 16. Furthermore, using the setting Experiment 4, the random DE and count-based GD performed better.

Overall, Experiment 14 has the best gain for all individual techniques and this is followed by Experiment 6. This matches the suggestion made in the previous subsection where Experiment 6 is considered as a candidate experiment worthy of exploring.

Table 7 shows the results for the average percentage improvement of TL over CML based on each technique, across all applications. In other words, the results of each technique for all applications are averaged. A total of six techniques performed better with the settings from Experiments 6 and 14, whereas two techniques, random BW and random VT performed better using settings of

TABLE 6. Experimental results for maximum percentage improvement using TL over CML based on each technique: Baum Welch (BW), Viterbi training (VT), different evolution DE), gradient descent (GD) and simulated annealing (SA). Bold values indicate the highest gain for each technique.

No.	Maximum gain for individual technique (%)															Overall gain (%)
	Uniform					Random					Count					
	BW	VT	DE	GD	SA	BW	VT	DE	GD	SA	BW	VT	DE	GD	SA	
1	31.49	28.24	31.54	11.19	44.77	57.85	48.16	51.88	0.32	43.57	5.91	22.65	31.54	1.78	34.62	29.7
2	39.43	66.92	68.47	23.06	60.89	39.43	53.50	63.84	48.49	53.44	**27.10**	52.47	67.89	26.33	57.42	49.91
3	31.27	29.68	37.26	18.86	**71.24**	56.71	49.73	57.53	6.15	40.65	5.99	22.51	37.26	3.13	50.65	34.57
4	39.31	66.54	68.47	43.22	56.33	39.05	53.76	**67.89**	28.39	58.96	20.04	53.82	29.48	**64.74**	28.39	47.89
5	40.46	35.83	36.24	15.67	60.35	29.16	**54.22**	67.17	60.76	63.08	11.44	34.74	36.24	20.84	58.86	41.67
6	**42.41**	69.43	**79.55**	**46.86**	44.43	53.14	53.14	66.3	47.27	55.57	17.31	64.57	**78.44**	57.79	**66.09**	56.15
7	39.97	35.4	44.82	13.46	65.68	30.01	50.74	66.35	60.03	41.99	11.44	34.19	42.13	26.65	44.82	40.51
8	41.46	**69.57**	79.47	44.59	47.83	52.68	53.39	66.33	46.92	62.29	19.72	**67.44**	78.36	46.51	57.74	55.62
9	31.49	28.24	31.54	11.19	0.50	57.85	48.16	51.88	0.50	7.58	5.96	22.65	31.54	11.19	44.07	25.62
10	39.43	66.92	68.47	23.06	64.03	38.6	53.5	63.84	**64.35**	54.34	26.98	52.47	67.89	26.33	47.53	50.52
11	31.27	29.68	37.26	11.00	51.69	**58.16**	49.73	57.53	0.52	54.06	6.04	22.51	37.26	7.61	50.91	33.68
12	39.31	66.54	68.47	29.48	57.16	37.12	53.76	63.84	59.15	53.44	19.91	53.82	67.89	26.91	55.49	50.15
13	40.46	35.83	36.24	15.67	65.53	29.29	**54.22**	67.17	36.24	20.84	11.58	34.74	36.24	15.67	56.27	37.07
14	**42.41**	69.43	**79.55**	**46.86**	53.14	53.24	53.14	66.3	55.57	68.42	20.04	64.57	**78.44**	57.79	55.57	**57.63**
15	39.97	35.4	44.82	13.46	57.87	30.15	50.74	66.35	63.93	**69.72**	11.57	34.19	42.13	26.65	60.83	43.19
16	41.46	**69.57**	79.47	44.59	60.26	52.98	53.39	66.33	55.21	63.80	20.73	**67.44**	78.36	46.51	58.75	57.26

TABLE 7. Experimental results for average percentage improvement using TL over CML based on individual techniques: Baum Welch (BW), Viterbi training (VT), different evolution DE), gradient descent (GD) and simulated annealing (SA). Bold values indicate the highest gain for each technique.

No.	Average gain for individual technique (%)															Overall gain (%)
	Uniform					Random					Count					
	BW	VT	DE	GD	SA	BW	VT	DE	GD	SA	BW	VT	DE	GD	SA	
1	26.72	20.73	19.51	0.68	37.77	35.89	38.64	42.97	-4.80	28.54	2.99	13.77	8.38	-2.90	16.86	19.05
2	26.87	36.60	40.83	7.61	41.52	26.70	30.20	50.10	17.61	37.28	**11.37**	25.06	33.67	10.18	46.07	29.44
3	26.63	22.55	23.51	3.17	43.94	36.06	**39.01**	46.87	-1.47	28.75	3.18	14.20	12.32	-6.01	38.91	22.11
4	26.74	36.61	40.83	8.08	33.48	26.62	30.72	50.10	17.37	40.67	8.03	25.46	33.66	14.00	35.22	28.51
5	17.99	16.84	21.68	9.35	28.44	17.99	29.49	45.55	31.66	32.30	2.10	14.79	22.34	5.16	34.91	22.04
6	**28.11**	**39.26**	**45.88**	**22.12**	31.50	28.97	29.77	51.91	33.18	40.57	7.55	35.06	**39.83**	24.42	**49.91**	33.87
7	18.31	12.87	26.68	0.76	36.99	18.36	25.10	50.14	22.82	24.54	2.17	9.87	26.96	4.45	31.31	20.76
8	25.67	37.64	45.83	19.50	30.95	28.83	30.29	**51.92**	30.60	46.30	8.37	**35.52**	39.77	23.18	39.17	32.90
9	26.72	20.73	19.51	0.68	23.50	35.91	38.64	42.97	-0.04	25.48	2.93	13.77	8.38	0.88	30.92	19.40
10	26.87	36.60	40.83	7.61	**46.43**	25.78	30.20	50.10	23.74	34.83	11.36	25.06	33.67	10.65	24.74	28.56
11	26.63	22.55	23.51	0.60	36.24	**36.22**	**39.01**	46.87	-0.06	34.99	3.06	14.20	12.32	-3.58	28.96	21.44
12	26.74	36.61	40.83	8.08	40.41	25.37	30.72	50.10	24.20	28.28	8.24	25.46	33.66	11.14	25.83	27.71
13	17.99	16.84	21.68	9.35	35.15	18.16	29.49	45.55	34.41	39.90	1.89	14.79	22.34	8.41	30.39	23.09
14	**28.11**	**39.26**	**45.88**	**22.12**	42.02	28.77	29.77	51.91	**34.51**	**51.82**	8.42	35.06	**39.83**	**26.28**	34.15	**34.53**
15	18.31	12.87	26.68	0.76	32.51	18.56	25.10	50.14	25.51	46.52	2.07	9.87	26.96	4.70	25.61	21.75
16	25.67	37.64	45.83	19.50	41.16	28.64	30.29	**51.92**	31.92	39.60	9.17	**35.52**	39.77	23.63	38.69	33.26

Experiment 11. As has already been alluded to in the performance of Tables 4, 5 and 6, Experiment 6 is the ideal experimental setting and this corresponds to the use of FPI, uniform sampling and 70% of training data. Therefore these settings have been been adopted for producing the next results.

3) PERFORMANCE COMPARISON FOR OPTIMUM EXPERIMENTAL SETTINGS

The results in this subsection are produced using FPI and uniform sampling for both TL and CML. A percentage of observations were extracted from each phase for training with the remaining portion reserved for evaluation. Uniform sampling entailed two experiments with different percentages for evaluation and training. In the first experiment, training and evaluation were allocated 30% and 70%, respectively. In the second experiment, training and evaluation were allocated 70% and 30%, respectively. In other words, the presented results are an aggregate of Experiments 6 and 2 (see Table 3), such that the entire dataset is used for training and evaluation in a two fold cross-validation.

The overall results for AS detection are shown in Figure 4 which depicts a consistent improvement of TL over CML. From the results, the highest improvement in TL is achieved by count-based SA (SA-Count), which has a gain of about 53.7%. This does not signify that SA-Count is the best overall technique as it corresponds to a detection accuracy of about 69.7%, whereas the DE based technique reaches up to 76.3% accuracy. Thus, all DE based techniques are recommended for AS detection followed by VT, which has a detection accuracy of 72.5% when using TL.

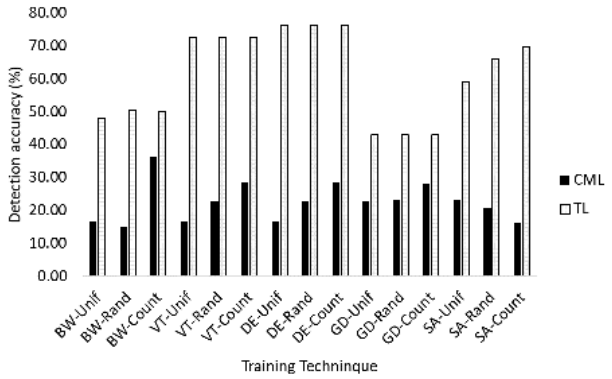

FIGURE 4. Comparison of all states detection accuracy for HMM training techniques when applying TL and CML.

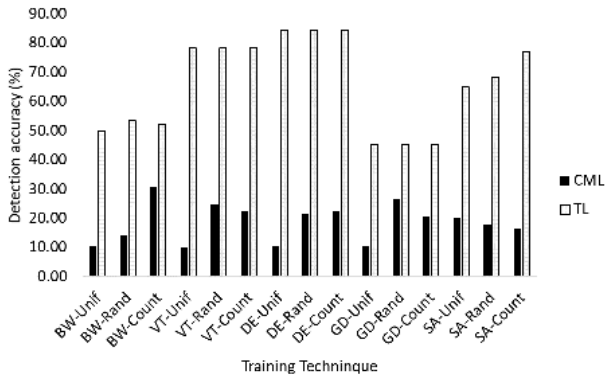

FIGURE 5. Comparison of current state detection accuracy for HMM training techniques when applying TL and CML.

In Figure 5, the results for CS detection are similar to the performance of AS, where the TL outperforms CML for all techniques. Specifically, the highest gain is demonstrated by uniform DE, which has an accuracy improvement of 74.0% followed by VT-uniform at 68.2%. The least gain is observed in count-based BW with an accuracy of 21.5%. When compared to uniform BW and random BW, it was anticipated that the count-based technique would perform better, since it uses part of the observation information for initialisation. This is justified in CML, where count-based BW outperforms its counterparts, but not as clearly manifested in TL.

The NS prediction results are shown in Figures 6, 7, and 8 for NS1, NS2, and NS3, respectively. TL performed better than CML in all the three levels with the uniform and count-based DE being the best overall techniques, with prediction accuracy of about 96.3% for NS3. As the number of levels increases, the accuracy gain drops from 73.2% (in NS1) to 52.7% (in NS3). The random SA is the second best technique with an accuracy of about 96% for NS3 followed by count-based SA at 94.11%. An interesting observation is noted in DE, VT and SA performance, where the TL converges to the same point regardless of the initialisation technique.

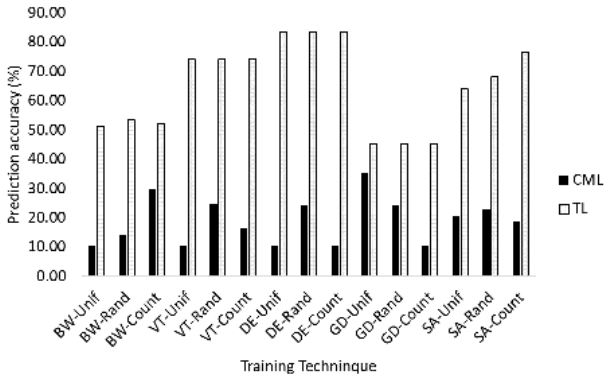

FIGURE 6. Comparison of next state level 1 prediction accuracy for HMM training techniques when applying TL and CML.

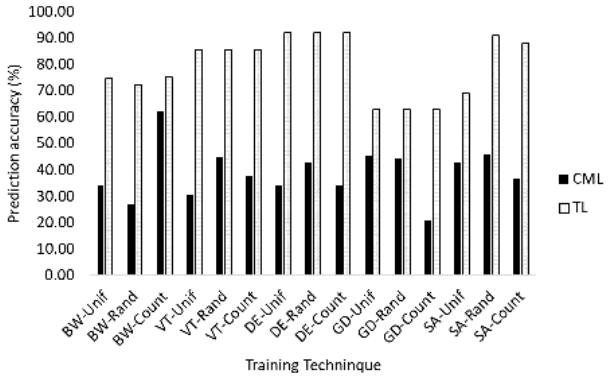

FIGURE 7. Comparison of next state level 2 prediction accuracy for HMM training techniques when applying TL and CML.

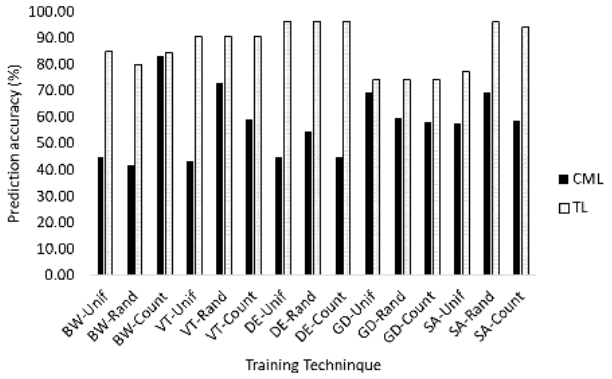

FIGURE 8. Comparison of next state level 3 prediction accuracy for HMM training techniques when applying TL and CML.

Lastly, the results for NO prediction are shown in Figures 9, 10, 11 for NO1, NO2, and NO3, respectively. To begin with, for NO1, it is observed that the highest gain of TL over CML is achieved by random based DE at about 29.5%. This is followed by random GD at 27.5% and then the SA based techniques at 26.4%, 25.4% and 19.7% for uniform, count-based and random initialisation, respectively. The rest

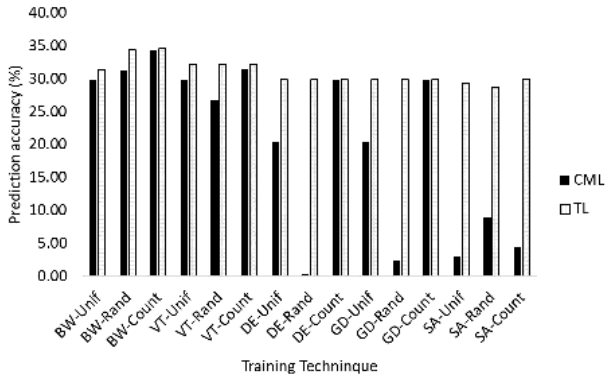

FIGURE 9. Comparison of next observation level 1 prediction accuracy for hidden Markov model training techniques when applying transfer learning (TL) and conventional machine learning (CML).

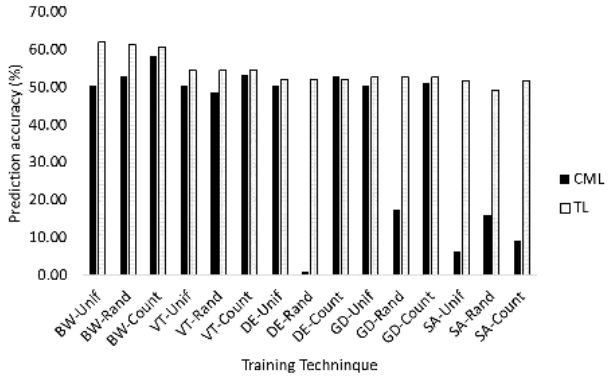

FIGURE 10. Comparison of next observation level 2 prediction accuracy for hidden Markov model training techniques when applying transfer learning (TL) and conventional machine learning (CML).

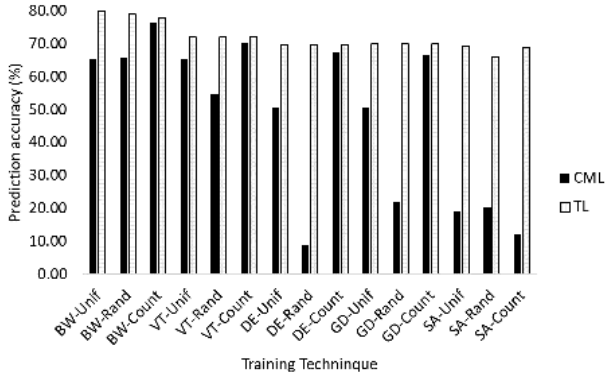

FIGURE 11. Comparison of next observation level 3 prediction accuracy for hidden Markov model training techniques when applying transfer learning (TL) and conventional machine learning (CML).

of the techniques range from 9.4% to 0.4%. The exception is count-based DE and GD where there is insignificant gain in TL over CML.

It should be highlighted that the actual maximum accuracy value for NO1 in TL is peaks at about 34.6% for count-based BW followed by random BW at 34.3% and then count-based VT at 32.2%. It can be deduced that a technique manifesting a high maximum improvement of TL over CML, does not necessarily imply that it is the best overall technique. It should also be noted that in contrast to NS prediction, which has 5 unique states, the NO prediction is lower due to the increased search space of 25 unique observations.

Concerning NO2 prediction in Figure 10, random DE achieved a maximum improvement of about 51.2% for TL over CML. Uniform and count-based SA followed at 45.5% and 42.4% respectively. Then, random GD and random SA achieved an accuracy gain of 35.1% and 33.2%, respectively. There was a negative transfer of −0.9% for count-based DE, which reiterates the need for careful consideration when applying TL. Nevertheless, this is the only instance with a negative TL and −0.9% is insignificant, hence, overall, TL is still of great benefit over CML. For the actual performance, all BW based techniques performed the best with 61.2%, 61.2% and 60.65% for uniform, random and count-based initialisation, respectively. BW has demonstrated better performance with TL when integrated with HMM unlike in the AS and CS detection and NS prediction.

For NO3 prediction, random DE maintains its consistently good performance with a gain of about 60.7% for TL over CML. Also, SA based techniques are the next best performing techniques with about 56.6%, 50% and 45.7% for count-based, uniform and random initialisation, respectively. The uniform GD achieved a gain of about 19.5%, whereas the uniform DE obtained a gain of about 19.3%. Considering the actual accuracy values, though the percentage improvement of TL over CML for BW and VT based techniques restricted up to 20%, the uniform, random and count-based BW techniques performed the best with n overall of 79.9%, 78.9% and 77.9%, respectively. This is followed by all the VT techniques which all produced a prediction accuracy of about 72.%.

It is interesting to note that unlike in CML where the actual performance of individual techniques varies due to different initialisations, in TL, the performance of VT, DE and GD is consistent for all applications. This can be attributed to the fact that the models converge to the same parameters. On the contrary, the BW and SA techniques achieve different results as they have separate convergence points for different initialisation techniques.

Overall, for TL, the DE-based techniques perform better than the rest with an average gain of 73% for all applications irrespective of the initialisation method. The achieved TL gain over CML, is about 51% and 43.4% for the random and uniform initialisation, respectively.

4) HMM PARAMETER EVALUATION

In addition to the detection and prediction accuracy, BIC, MSE and ARI have been used to measure HMM parameter alignment of TL and CML to the ideal parameters generated when training the whole target dataset with SL CML. As seen in the BIC results (Figure 12), the increase performance of TL over CML is observed mainly in SA based techniques.

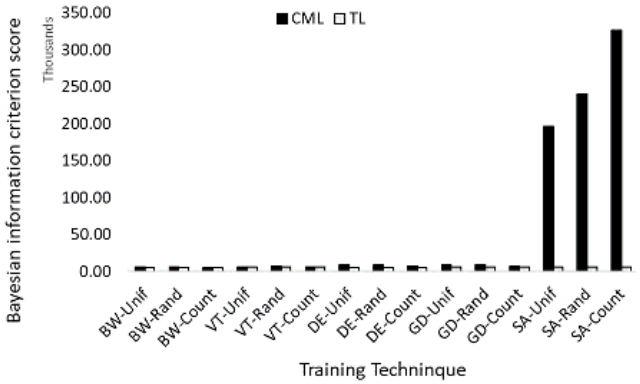

FIGURE 12. Bayesian information criterion for HMMs trained using TL and CML. Lower value indicates better technique.

Because a lower BIC value indicates a better technique, TL outperforms CML for all SA initialisation techniques. An in-depth exploration of the BIC indicates that values are less for TL than CML for all training techniques.

Though the performance of TL is better based on BIC, there is no clear correlation when compared with the accuracy values for all applications. For instance, as it was highlighted previously, the DE performs better in most of the applications, but this is not clear when considering BIC values.

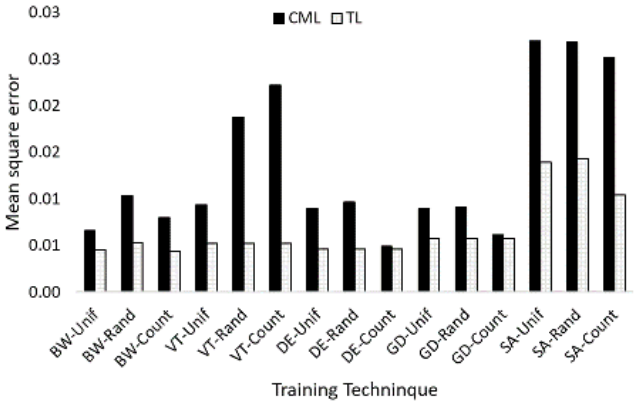

FIGURE 13. Mean square error for emission probabilities of HMMs trained using TL and CML. Lower value indicates a better technique.

When using the MSE metric (Figure 13), the comparison is based only on emission probabilities, B, rather than the whole model λ. In terms of MSE, TL performs better than CML for all training and initialisation techniques. This concurs with both accuracy and BIC performance. There is an evident increase in performance between TL and CML across all techniques, not as clearly manifested when looking at the BIC values. Nevertheless, there is still no association of accuracy with the MSE metric. For instance, the increased accuracy performance of TL over CML using DE (as asserted in Section VII-A3) is not indicated in Figure 13. The most plausible reason is that MSE metric only considers the training data and not the evaluation data which is used for accuracy determination.

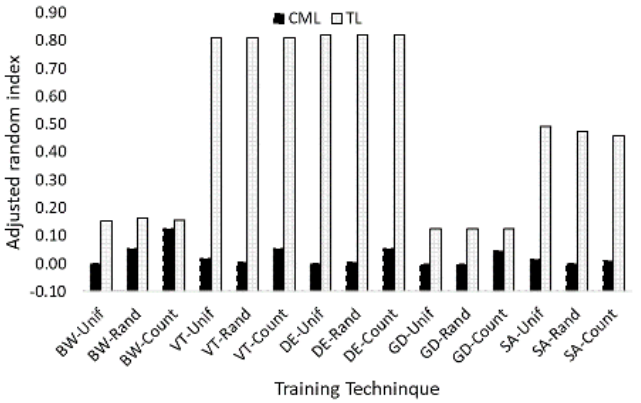

FIGURE 14. Adjusted random index for HMMs trained using TL and CML. Higher values indicate a better technique.

Lastly, the ARI performance is illustrated in Figure 14, where a comparative performance is observed similar to that of AS and CS detection in Figures 4 and 5, respectively. The VT and DE techniques are shown to be the best performing techniques just like in AS and CS detection. SA techniques come third in performance gains.

It can be deduced that the ARI is a better performance metric than the BIC and MSE. It would, therefore, be suggested for analysis of HMMs. A simple algorithm can thus be deduced such that when the ARI for TL performs less than that for CML, then CML would be deployed, thereby avoiding negative transfer. After model parameters for TL, λ_{TL} and CML, λ_{CML} are learned, the ARI for TL, ARI_{TL} and CML, ARI_{CML} are compared as in Algorithm 4.

Algorithm 4 Selection of Best Model Based on ARI

1: Learn parameters λ_{CML}, λ_{TL}
2: Compute ARI_{CML}, ARI_{TL}
3: **if** $ARI_{TL} > ARI_{CML}$ **then**
4: $\quad \lambda = \lambda_{TL}$
5: **else**
6: $\quad \lambda = \lambda_{CML}$

5) MULITPLE RUNS FOR RANDOM BASED TECHNIQUES

In order to statistically analyse random based techniques and the stochastic nature of SA, multiple runs of experiments were proposed for random based BW, VT, DE, GD and all SA techniques. To this end, a total of 38 runs were made using these stochastic algorithms for all applications.

Figure 15 presents boxplot results focusing on AS detection for all techniques. It is observed that TL outperforms the CML in all the training techniques. The expectation was that there would be stochastic results for all techniques. Instead, random VT, DE and GD have deterministic response when using TL. There is an insignificant interquartile range (IQR) of 46.7% - 47.8% for random VT when performing TL. All SA based techniques had larger IQR, up to 11.4% using TL on uniform SA. A plausible reason for the significant

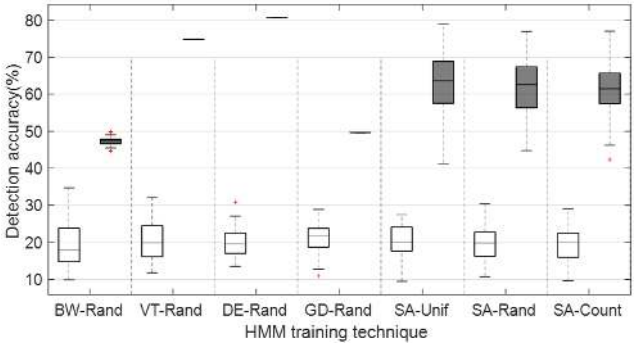

FIGURE 15. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in detecting all states of an observations sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

IQRs in the case of SA is that it incorporates at least two random processes, since it selects arbitrary elements and also adds random values to λ. The random process implies the possibility of a wide range of solutions. Unlike TL, where most techniques have constant values, in CML all techniques have varying IQR at 9.1%, 8.4%, 5.5%, 5%, 6.5%, 6.2% and 6.6% for random BW, random VT, random DE, random GD, uniform SA, random SA and count-based SA, respectively. Also, it is noted that for all HMM training techniques, there is no overlap between TL and CML values.

Regarding the multiple runs for CS detection, shown in Figure 16, TL outperforms CML for all the training techniques. There is no variance seen in TL using random VT, DE and GD. For random BW, a small IQR range is observed, as was the case in AS detection. There are outliers in random VT, DE and GD for CML, and also in random BW and GD for TL. There is no overlap between the IQRs of TL and CML. It should also be highlighted that all the CML techniques have a significant IQR within the same ranges of AS detection.

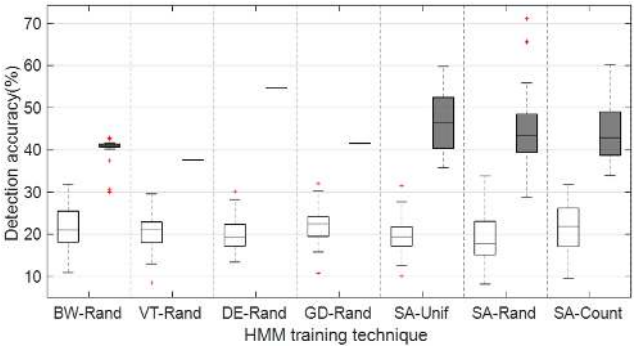

FIGURE 16. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in detecting the current state of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

Considering the performance for NS1, NS2 and NS3 prediction as shown in Figures 17, 18, and 19, respectively, a similar pattern is observed for the multiple runs of these applications. For all the applications, there is no overlap of

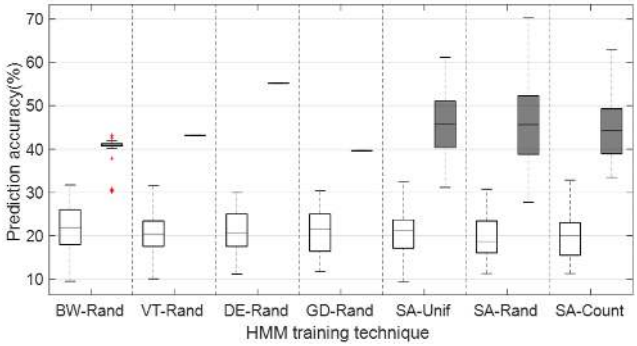

FIGURE 17. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next states level 1 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

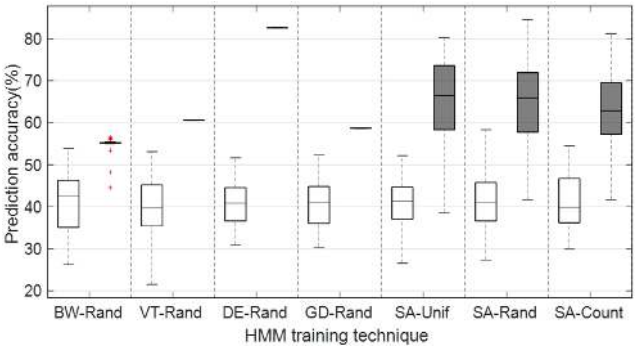

FIGURE 18. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next states level 2 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

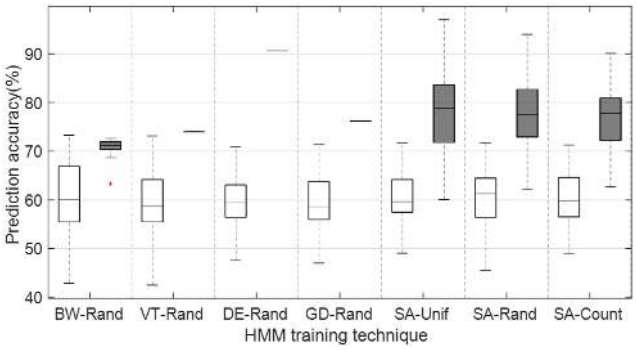

FIGURE 19. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next states level 3 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

IQRs between TL and CML. Furthermore, TL performance surpasses that for CML in all training techniques. Similar to AS and CS detection, the random VT, DE and GD for TL yield deterministic solutions. Also, random BW using TL has a narrow IQR and it is the only technique that has outliers. There is a corresponding increase in performance as the number of levels increases for all techniques. This is anticipated as the increase in the number of levels corresponds to inclusion of many possible NSs, which increases the prediction probability.

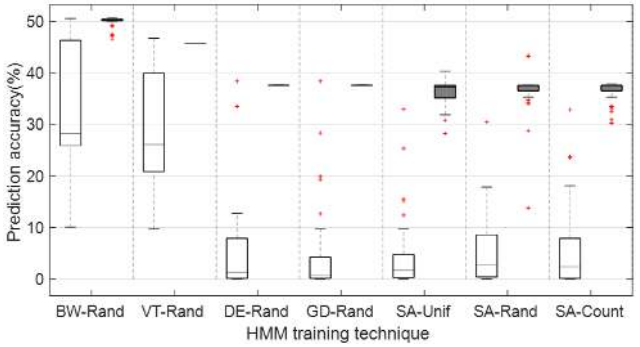

FIGURE 20. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next observation level 1 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

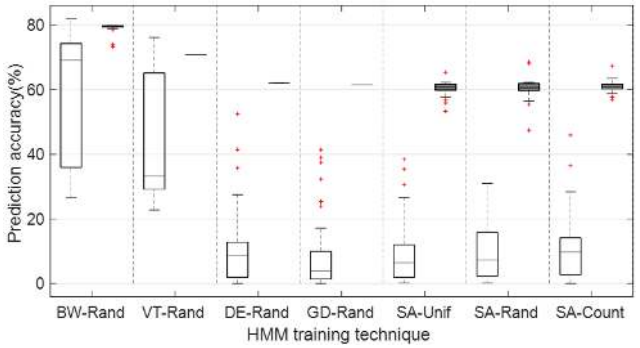

FIGURE 21. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next observation level 2 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

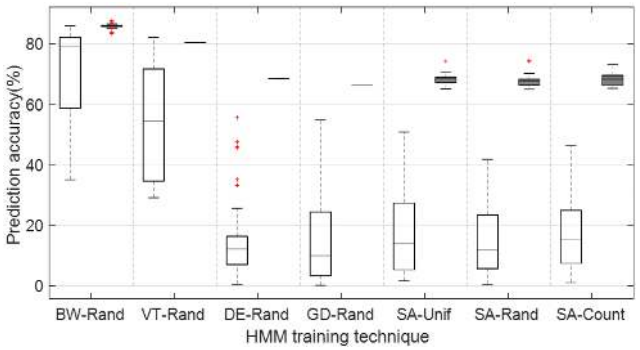

FIGURE 22. Boxplot for 38 runs of random BW, VT, DE and GD, and all SA based techniques in predicting next observation level 3 of an observation sequence. White-filled boxplots indicate CML whereas black-filled boxplots represent TL.

Regarding NO performance, for each level, as shown in Figures 20, 21, and 22, it can be observed that there are more outliers when compared to previous results. Nevertheless there is no overlap between IQRs of CML and TL. The constant performance for random VT, DE and GD in TL is still maintained for all applications just like in AS, CS and NS results. It is observed that for all TL techniques, the performance is either constant or within a small IQR, which can be interpreted as the convergence benefit of TL over CML. In NO prediction, it is challenging to get better performance than NS prediction, since the number of observations are much more than those for states. Similar to NS prediction, when the number of levels increases, there is a corresponding increase in performance.

Overall, although there are stochastic solutions for randomly initialised BW, VT, DE, GD and all SA based techniques, TL still overshadows the CML based techniques in all applications. There is no overlap of IQRs for CML and TL, hence the improvement in the performance of TL over CML as presented in Section VII-A3 and VII-A4 still upholds. The multiple runs do not present a noticeable benefit for CML, hence TL usage performs better than CML in stochastic techniques as well.

B. RESULTS FOR CSE-CIC-IDS DATASET

After the extensive analysis of TL performance over CML, it was considered to identify an appropriate technique that can be replicated onto the CSE-CIC-IDS2018 dataset. The uniform DE has consistent performance in all evaluated scenarios with DARPA 2000 and, thus, was considered to analyse the CSE dataset as well. The results (Figure 23) reinforce the previous observation of significant performance improvement of TL over CML with the highest gain shown for NO3 at 60.9% (i.e. an increase from 20.6% for CML to 81.5% for TL). Even though CML barely yields any results for the applications of CS, NS1 and NO1, TL is able to detect and predict both observations and states with considerably higher accuracy values.

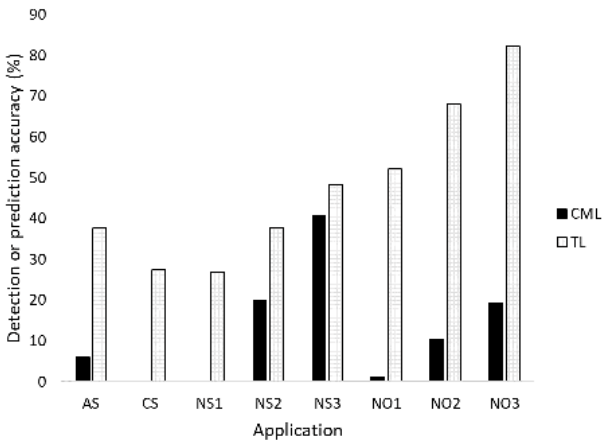

FIGURE 23. Comparison of all states (AS) and current state (CS) detection accuracy and next state (NS) and next observation (NO) prediction accuracy between TL and CML using the uniformly initialised differential evolution technique.

VIII. CONCLUSION

This paper addresses the challenge of detecting and forecasting complex, cyber-attacks that are manifested in a sequential order, or phases. Each phase may appear benign or malicious

when analysed independently, however, when phases are analysed in sequence, the real motives are revealed. There are several challenges that make this task even more challenging. In particular, it is very difficult to access labelled cyber-attack datasets. In addition, previously generated machine learning models become obsolete very quickly, due to the rapidly changing attacking techniques and attitudes, as well as, the dynamic nature of the underlying benign traffic.

To address these challenges, this paper examines the effectiveness of transfer learning techniques on sequential network attacks. The primary driver of this research work is to leverage prior modelling capability and knowledge derived from known, labelled datasets and use it to efficiently develop models for new, yet unlabelled, datasets in an unsupervised fashion. Ultimately, new models may be trained (learned) by learning from prior knowledge. This approach opens a new paradigm in adapting derived models to new trends in sequential attacks and different underlying network dynamics.

An extensive experimental analysis, examining five parameter learning algorithms, three parameter initialisation techniques, four data sampling techniques and two pairs of training to evaluation ratios has been conducted. The results show that TL outperforms CML for all evaluated algorithms, i.e. BW, VT, DE, GD and SA using uniform, random, and count-based initialisation techniques. Specifically, the DE algorithm, a globally optimum solution, regardless of any initialisation technique, gives the highest gains in TL in comparison to the rest of the deployed techniques. In addition, DE-based TL also shows the best actual performance overall for the applications of AS, CS and NS. However, for NO prediction, the BW-based TL technique gives the highest prediction values overall.

Besides demonstrating the significantly improved performance in detection and forecasting using TL, this paper has utilised fixed point iteration for a fair comparison towards CML. Notably, all three metrics used for HMM parameter evaluation, BIC, MSE, and ARI, attest to the increased benefit of using TL over CML. The ARI metric in particular showcases a behaviour that aligns with and justifies very well the detection results in AS and CS. This observation suggests that a simple algorithm may be deployed that intelligently decides which specific technique (say, BW or DE) or approach (TL or CML) would be best to deploy, based on preliminary results derived from the ARI metric.

Finally, future work will include the application of TL approaches in Deep Neural Networks for SNA analysis and further analysis on alternative transfer learning approaches.

Note: The MATLAB source code for this work is available in Code Ocean (https://codeocean.com/capsule/3814125) and in Loughborough University's repository (https://doi.org/10.17028/rd.lboro.12666950).

REFERENCES

[1] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, "Analysis of hidden Markov model learning algorithms for the detection and prediction of multi-stage network attacks," *Future Gener. Comput. Syst.*, vol. 108, pp. 636–649, Jul. 2020.

[2] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, and J. Disso, "Cyber-attack modeling analysis techniques: An overview," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud Workshops (FiCloudW)*, Aug. 2016, pp. 69–76.

[3] E. Hutchins, M. Cloppert, and R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues Inf. Warfare Secur. Res.*, vol. 1, no. 1, p. 80, 2011.

[4] I. Ghafir, K. G. Kyriakopoulos, S. Lambotharan, F. J. Aparicio-Navarro, B. Assadhan, H. Binsalleeh, and D. M. Diab, "Hidden Markov models and alert correlations for the prediction of advanced persistent threats," *IEEE Access*, vol. 7, pp. 99508–99520, 2019.

[5] C. Ventures, "2019 official annual cybercrime report," Herjavec Group, Toronto, ON, Canada, Tech. Rep., 2019. [Online]. Available: https://www.herjavecgroup.com/wp-content/uploads/2018/12/CV-HG-2019-Official-Annual-Cybercrime-Report.pdf

[6] T. Shawly, A. Elghariani, J. Kobes, and A. Ghafoor, "Architectures for detecting interleaved multi-stage network attacks using hidden Markov models," *IEEE Trans. Dependable Secure Comput.*, early access, Oct. 23, 2019.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[8] A. Ng, "Nuts and bolts of building AI applications using deep learning," in *Proc. NIPS Keynote Talk*, 2016, pp. 1–5.

[9] S. Sun, H. Liu, J. Meng, C. L. P. Chen, and Y. Yang, "Substructural regularization with data-sensitive granularity for sequence transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2545–2557, Jun. 2018.

[10] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[11] Lincoln Laboratory. *2000 DARPA Intrusion Detection Scenario Specific Datasets*. Accessed: Jun. 1, 2020. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scen%ario-specific-datasets

[12] Canadian Institute of Cybersecurity. *CSE-CIC-IDS2018 on AWS*. Accessed: May 18, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[13] CISCO & Affiliates. *Snort—Network Intrusion Detection & Prevention System*. Accessed: Jun. 6, 2020. [Online]. Available: https://www.snort.org

[14] J. Zhao, S. Shetty, J. W. Pan, C. Kamhoua, and K. Kwiat, "Transfer learning for detecting unknown network attacks," *EURASIP J. Inf. Secur.*, vol. 2019, no. 1, pp. 1–13, Dec. 2019.

[15] L. Yang, L. Jing, J. Yu, and M. K. Ng, "Learning transferred weights from co-occurrence data for heterogeneous transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2187–2200, Nov. 2016.

[16] W. Wang, Q. Chen, X. He, and L. Tang, "Cooperative anomaly detection with transfer learning-based hidden Markov model in virtualized network slicing," *IEEE Commun. Lett.*, vol. 23, no. 9, pp. 1534–1537, Sep. 2019.

[17] P. Holgado, V. A. Villagra, and L. Vazquez, "Real-time multistep attack prediction based on hidden Markov models," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 134–147, Jan. 2020.

[18] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[19] H. Tan and S. Ma, "Learning partially observable Markov decision model with EM algorithm," in *Proc. 7th Int. Conf. Appl. Inf. Commun. Technol.*, Oct. 2013, pp. 1–4.

[20] B. Benyacoub, S. E. Bernoussi, and A. Zoglat, "Credit scoring model based on baum-welch method," in *Proc. ACS Int. Conf. Comput. Syst. Appl. (AICCSA)*, May 2013, pp. 1–5.

[21] A. A. R. Sá, A. O. Andrade, A. B. Soares, and S. J. Nasuto, "Estimation of hidden Markov models parameters using differential evolution," in *AISB Convention Communication, Interaction and Social Intelligence*. Beijing, China: AISB, 2008.

[22] K. Ghanem, A. Draa, E. Vyumvuhore, and A. Simbabawe, "Differential evolution to optimize hidden Markov models training: Application to facial expression recognition," *J. Comput. Inf. Technol.*, vol. 23, no. 2, pp. 157–170, 2015.
[23] J. Salazar, M. Robinson, and M. R. Azimi-Sadjadi, "A hybrid HMM-neural network with gradient descent parameter training," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2003, pp. 1086–1091.
[24] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035–1074, Apr. 1983.
[25] J.-S. Lee and C. Hoon Park, "Hybrid simulated annealing and its application to optimization of hidden Markov models for visual speech recognition," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 40, no. 4, pp. 1188–1196, Aug. 2010.
[26] M. Lalaoui, A. El Afia, and R. Chiheb, "Hidden Markov model for a self-learning of simulated annealing cooling law," in *Proc. 5th Int. Conf. Multimedia Comput. Syst. (ICMCS)*, Sep. 2016, pp. 558–563.
[27] A. E. Afia, M. Lalaoui, and R. Chiheb, "A self controlled simulated annealing algorithm using hidden Markov model state classification," in *Procedia Computer Science*, vol. 148. Amsterdam, The Netherlands: Elsevier, 2019, pp. 512–521.
[28] D. Paul, "Training of HMM recognizers by simulated annealing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. ICASSP*, Apr. 1985, pp. 13–16.
[29] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2015.
[30] R. Ribani and M. Marengoni, "A survey of transfer learning for convolutional neural networks," in *Proc. 32nd SIBGRAPI Conf. Graph., Patterns Images Tuts. (SIBGRAPI-T)*, Oct. 2019, pp. 47–57.
[31] J. Brownlee. (2017). *A Gentle Introduction to Transfer Learning for Deep Learning*. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/
[32] W. Fan, I. Davidson, B. Zadrozny, and P. S. Yu, "An improved categorization of Classifier's sensitivity on sample selection bias," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2005, pp. 605–608.
[33] A. Argyriou, M. Pontil, Y. Ying, and C. A. Micchelli, "A spectral regularization framework for multi-task structure learning," in *Proc. Adv. Neural Inf. Process. Syst.*, New York, NY, USA: Curran Associates, 2008, pp. 25–32.
[34] T. A. Shawly, A. A. Elghariani, J. Kobes, and A. Ghafoor, "Architectures for detecting real-time multiple multi-stage network attacks using hidden Markov model," 2018, *arXiv:1807.09764*. [Online]. Available: https://arxiv.org/abs/1807.09764
[35] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[36] J. Cummings, M. Shirk, and P. Team. *PulledPork*. Accessed: May 28, 2020. [Online]. Available: https://github.com/shirkdog/pulledpork
[37] E. Eldondev. (2011). *Snort Rules*. Accessed: May 28, 2020. [Online]. Available: https://github.com/eldondev/Snort
[38] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, "Contemporary sequential network attacks prediction using hidden Markov model," in *Proc. 17th Int. Conf. Privacy, Secur. Trust (PST)*, Aug. 2019, pp. 1–3.
[39] B. Roblès, M. Avila, F. Duculty, P. Vrignat, S. Begot, and F. Kratz, "Methods to choose the best Hidden Markov Model topology for improving maintenance policy," in *Proc. 9th Int. Conf. Modeling, Optim. Simulation MOSIM*, 2012, p. 1.
[40] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.
[41] E. Ramasso, "Inference and learning in evidential discrete latent Markov models," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 5, pp. 1102–1114, Oct. 2017.

TIMOTHY CHADZA (Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Malawi (UNIMA)-Polytechnic, in 2005, the M.Tech. degree in advanced IT–networking and telecommunications from the International Institute of Information Technology, India, in 2011, and the M.Sc. degree in mobile communication from Loughborough University, in 2016, where he is currently pursuing the Ph.D. degree with the Wolfson School of Mechanical, Electrical, and Manufacturing Engineering. He has undergone training in wireless networking at the International Centre of Theoretical Physics, Italy, and National Information Technology, Poland. He is currently a Lecturer in telecommunications with the Department of Electrical Engineering, UNIMA-Polytechnic. He is interested in wireless networking security applied in wireless networks and mobile communication.

KONSTANTINOS G. KYRIAKOPOULOS (Member, IEEE) received the B.Sc. degree in electrical engineering from the Technological Education Institute, Larisa, Greece, in 2003, and the M.Sc. degree in digital communication systems and the Ph.D. degree in computer networks from Loughborough University, Loughborough, U.K., in 2004 and 2008, respectively. From 2008 to 2015, he was a Research Associate with the School of Electronics, Electrical, and Systems Engineering, Loughborough University, involved mainly in the EPSRC projects and successfully licensed research output from his work. Since 2016, he has been an Academic Member of the Wolfson School of Mechanical, Electronics, and Manufacturing Engineering, Loughborough University. He has extensive experience in computer network security, anomaly detection, contextual awareness, and performance measurements in emerging network paradigms and their applications. His research interests are in the areas of intelligent decision-making using machine learning techniques, evidence theory, and soft computing techniques.

SANGARAPILLAI LAMBOTHARAN (Senior Member, IEEE) received the Ph.D. degree in signal processing from Imperial College London, London, U.K., in 1997. He was a Visiting Scientist with the Engineering and Theory Center, Cornell University, Ithaca, NY, USA, in 1996. He was with Imperial College London, until 1999, where he was a Postdoctoral Research Associate. From 1999 to 2002, he was with the Motorola Applied Research Group, U.K., as a Research Engineer, working on various projects, including the physical-link layer modeling and performance characterization of the GPRS, EGPRS, and UTRAN. From 2002 to 2007, he was a Lecturer with King's College London, London, and a Senior Lecturer with Cardiff University, Cardiff, U.K. He is currently a Professor in digital communications and the Head of the Signal Processing and Networks Research Group, Loughborough University, Loughborough, U.K. He has authored more than 250 conference papers and journal articles in his research areas. His research interests include wireless communications, convex optimizations, game theory, blockchain, and machine learning. He is currently an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.