Regular Paper

Communications of the CCISA
Vol. 26 No. 4 Dec. 2020

一個有效的深度學習超參數選擇方法應用於入侵偵測系統

張維晏 [1,a*]、陳羿霖 [1,b]、陳煥 [1,c]、蔡崇煒 [2,d]
[1] 國立中興大學 資訊科學與工程學系、[2] 國立中山大學 資訊工程學系
[a] keoinn@gmail.com、[b] a0989735018@gmail.com、[c] huan@nchu.edu.tw
[d] cwtasi@mail.cse.nsysu.edu.tw

摘要

許多近期的研究利用機器學習或深度學習方法，作為入侵偵測系統的判別模組方法。部分研究主要注重於如何改良深度學習架構，來提升深度學習的準確率。其中一個影響深度學習的效能的原因是超參數的設定，包含了神經元的個數、層數或學習率等。然而目前超參數的設定，通常是依據個人的經驗或窮舉法方式來設定超參數的數值。在本論文中將差分進化演算法與深度學習結合，提出了一種可以自適應調整超參數的方法，並將其應用於入侵偵測系統中。本論文將所提出的方法與其他機器學習和深度學習進行比較。最終實驗結果說明，本論文所提的超啟發式演算法，在較複雜的入侵偵測問題上，可以有效的找到深度學習的最佳超參數，能有效提升深度學習模型對於入侵攻擊的檢測能力。

關鍵詞：入侵偵測系統、深度學習、超啟發式演算法、超參數

* 通訊作者 (Corresponding author.)

Regular Paper

Communications of the CCISA
Vol. 26 No. 4 Dec. 2020

An Effective Hyperparameter Selection for Deep Learning Algorithm in Intrusion Detection System

Wei-Yan Chang[1,a*], Yi-Lin Chen[1,b], Huang Chen[1,c], and Chun-Wei Tsai[2,d],
[1]Computer Science and Engineering, National Chung Hsing University, Taiwan, R.O.C.
[2]Computer Science and Engineering, National Sun Yat-sen University, Taiwan, R.O.C.
[a] keoinn@gmail.com, [b] a0989735018@gmail.com, [c] huan@nchu.edu.tw
[d] cwtasi@mail.cse.nsysu.edu.tw

Abstract

With the great success of artificial intelligence, machine learning and deep learning have been applied to intrusion detection as the core methods in recent years. The primary focus of many researches is on how to improve the performance by tuning the deep learning model and the associated hyperparameters. Hyperparameters are critical factors for deep learning-based method which includes how many layers should be used in the model; how many neurons are there for each layer; and how to set up the learning rate properly. However, most of studies set the hyperparameters depending on personal experience or using inefficient brute force exhaustive search. In this paper, we will present a self-adaptive method to decide the setting of hyperparameters that integrated deep learning and differential evolution. The proposed scheme is then be used for intrusion detection system. To evaluate the effectiveness of the proposed method, the performance of the scheme is compared to those of other machine learning and deep learning methods. Three performance metrics in terms of accuracy, recall and precision are used. Results show that the proposed algorithm can find the best hyperparameters setting for deep learning model than other methods compared in this paper.

Keywords: Intrusion detection system, deep learning, metaheuristic algorithm, hyperparameters.

壹、前言

物聯網 (internet of things, IoT) 應用隨著網路技術快速發展而崛起，並迅速融入到人們的日常生活中。因此許多研究與應用 [1] 嘗試將物聯網設備與技術引進至工廠之中，以減少人力資源以並加速生產，形成工業物聯網 (industrial internet of things, IIoT)。物聯網的因網路發展帶來的便利性，網路危害也相對接踵而至，例如：駭客利用網路釣魚將惡意軟體植入於電力公司網路，進一步透過遠端方式關閉電力控制系統造成 2015 年 12 月烏克蘭大停電 [2]。2020 年末 Juniper 網通設備業者揭露一個新型惡意程式[3]，專門鎖定基於 Linux Arm 或 MIPS 處理器的 IoT 裝置進行攻擊，遭植入惡意程式之設備將成為駭客組織的殭屍網路節點，進行加密貨幣挖礦或者是執行分散式阻斷服務攻擊 (distributed denial-of-service attack, DDoS)。上述的資安事件可以發現，目前物聯網的資訊安全成為了物聯網環境的主要挑戰[4]。為了抵禦網路中各式各樣的攻擊，許多研究都提出了許多有效的防禦機制或者系統例如：入侵偵測及防禦系統 (intrusion detection and prevention system, IDPS)，目前也因網路的發展使得網路攻擊方式逐漸複雜，傳統演算法已經無法精確地檢測出攻擊，需要有更高效且動態的檢測系統，才能應對網路攻擊方法變異快速之問題。因此近年來許多研究 [5–8] 都將深度學習應用於入侵偵測系統，以提供該系統檢測的精準度，其結果證明使用深度學習技術能夠有效偵測新型的網路攻擊方法。

上述研究結果說明深度學習方法應用於入侵偵測系統，可超越基於傳統演算法或規則開發之系統，然而實際建置入侵偵測系統時網路環境與訓練模型之資料集不盡相同，根據網路環境的改變，如何設定深度學習之超參數以最大化系統的檢測效能，是這些研究遭遇的共同問題。Yang 等人 [9] 使用粒子群最佳化演算法 (particle swarm optimization, PSO) 調整支持向量機 (support vector machine, SVM) 的超參數；Tuba 等人 [10] 採用蝙蝠演算法 (bat algorithm, BA) 調整 SVM。這兩篇文獻說明超啟發式演算法 (metaheuristic algorithm) 可以成功且有效地尋找機器學習適合的超參數組合。文獻 [11-12] 說明使用超啟發式演算法與神經網路結合，利用超啟發式演算法在有限時間內找出適合神經網路之超參數組合，可以節省因調整深度學習參數並重新訓練的計算成本，並將其計算資源用於最佳化模型之權重，以提升檢測模型的精確度。因此本研究採用差分進化演算法 (differential evolution, DE) [13] 調整深度類神經網路 (deep neural network, DNN) [14] 架構與長短期記憶神經網路 (long-short-term memory, LSTM) [15] 並應用於入侵偵測系統。

本論文章組織如下：第二章將介紹入侵偵測系統的背景以及類型，還有介紹近年的新型深度學習方法的架構及流程，第三章將介紹本篇論文的系統架構，包含使用到的超啟發式演算法以及深度學習如何將上述兩種方法結合成一個新型的演算法，並且將會描述整個方法的流程和檢測方式，第四章是實驗結果，在此章節會使用各種公開資料集進行效能測試，比較的演算法包含了本論文所提出的方法也會和近年的深度學習方法做比

較並分析結果，第五章將總結本篇論文所提出的方法的成效與優點以及提出未來可能發展的項目。

貳、文獻探討

目前市面上的入侵偵測系統大致上可以分成三種類型，分別為主機型入侵偵測系統 (host-based intrusion detection system, HIDS)、網路型入侵偵測系統 (network-based intrusion detection system, NIDS) 以及無線型入侵偵測系統 (wireless-based intrusion detection system, WIDS) [16]。主機型的入侵偵測系統通常建置於伺服器中，透過系統日誌進行掃描，該防禦方式屬於事後檢測方法，無法及時針對網路攻擊採取應對措施。網路型與無線型入侵偵測系統則是部屬於網路閘道並分析連線網路封包，針對發出可疑封包之來源端進行封鎖，以達到即時防禦或立即採取應對措施。因此本論文提出方法實現於網路型以及無線型入侵偵測系統，透過分析封包判斷連線方是否進行非正常操作，並及時阻擋以達到保護設備運作。

早期研究中 [17] 討論許多使用機器學習方法應用於入侵偵測系統，可以分成三種類型，單一分類器 (single classifiers)、集成分類器 (ensemble classifiers) 及混合分類器 (hybrid classifiers)。單一分類器即使用一種分類演算法解決分類問題，如：最近鄰居法 (*k*-nearest neighbor, *k*NN) [18]、決策樹 (decision tree) [19] 及支持向量機 (support vector machines, SVM) [20]。上述這些傳統方法進行簡單網路行為分類時擁有不錯的效果，隨著網路攻擊行為趨於複雜，僅靠單一分類器進行攻擊檢測已無法滿足及時防禦攻擊的可靠性，因此後續有許多研究結合兩種或兩種以上分類演算法提出混合分類器[21–23]，混合分類器建立演算法資訊交換機制，讓不同的單一演算法分類結果互相參考與比對，以提升分類器的分類準確度。另一些研究 [24–26] 指出混合分類器不斷地交換分類結果並進行參考，不考慮分類器的正確性，將會導致分類器準確率下降，因此提出集成分類器以解決上述問題。集成分類器集合多種單一分類器，以投票分式進行預測結果的判定，另一種則是從採取不同的資料集進行模型訓練，利用不同的模型之預測結果進行投票判定。

近年來深度學習技術發展運用已成熟，因此研究 [6] 使用深度神經網路建構入侵偵測系統並實現在軟體定義網路 (software-defined networking, SDN) 之中，該架構具有輸入層與輸出層和三個隱藏層，隱藏層中的神經元分別為 12 個、6 個與 3 個的小型神經網路用於解決分類問題，該研究也證明使用深度學習的神經網路構建入侵偵測系統相較於使用傳統演算法的入侵偵測系統有效。文獻 [5] 同樣使用類神經網路設計入侵偵測系統之分類器，並使用 NSL-KDD 資料集進行訓練與測試，該資料集是目前設計入侵偵測系統最常用來衡量方法之效能的資料集。研究 [7] 使用具有時序性的遞迴類神經網路應用在入侵偵測系統，並採用上述的資料及進行衡量，實驗結果證明使用遞迴神經網路

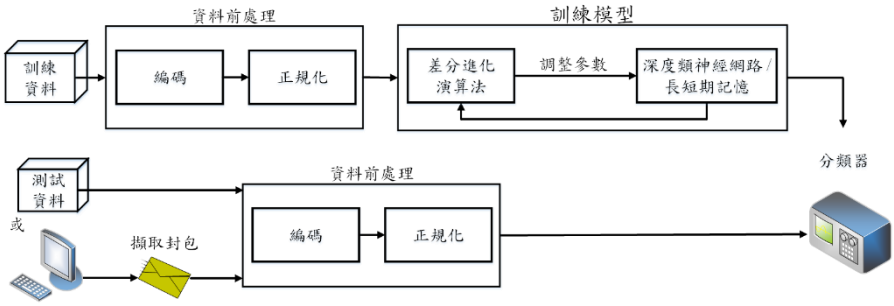
(recurrent neural networks, RNN) 的入侵偵測系統進行網路攻擊的二分類問題與多分類問題上擁有很高的準確率。文獻 [8] 使用長短期記憶模型設計入侵偵測系統之分類器，改善遞迴神經網路採用 S 型函數 (Sigmoid) 作為激勵函數導致訓練模型期間的梯度消失 (gradient vanishing problem) 及梯度爆炸 (gradient exploding problem) 之問題，並使用 10% 的 KDD CUP 99 資料集作為訓練與測試並對該系統進行效能評估，雖然該研究所提出的系統因長短記憶模型導致偵測誤報率上升，但上升幅度還在可接受範圍內，其結果擁有 96.9 % 的準確率及 98.8 % 召回率。

研究 [27] 中在設計入侵偵測系統採用自動編碼器 (autoencoder) 改良提出非對稱型堆疊式深度自動編碼器，使該編碼器讓分類模型具有無監督式學習及分層學習能力，使模型可以學習不同特徵之間的複雜關係。其結果在 NSL-KDD 資料集達到 85.42% 的準確率，KDD CUP 99 資料集可達到 97.85% 的準確率，證明使用自動編碼器技術能夠有效提升入侵偵測系統之分類準確率。因此研究提出新的深度學習模型稱為深度自動編碼器並應用於入侵偵測系統，該架構分成三個訓練步驟 1.) 預訓練階段：將每個神經網路之隱藏層使用自動編碼技術進行訓練以減少重建誤差 2.) 微調階段：進行預訓練之後將自動邊器的輸出作為輸入，使用訓練資料及標籤進行模型訓練 3.) 完全微調階段：使用監督式學習技術通過反向傳播方法對所有神經元層進行微調以提升模型準確度。其結果在 KDD CUP99 擁有 94.71% 之準確率。

從上述文獻回顧可以發現，目前入侵偵測系統的開發採用深度學習技術設計之系統相較於傳統演算法之系統，擁有較高的網路攻擊偵測準確率。然而深度學習中存在許多超參數 (hyperparameter) 使用試誤法或憑藉經驗設定參數，參數組合的優劣進一步影響深度學習開發分類模型之準確率，因此本研究重點為透過超啟發演算法尋找適合的深度學習參數組合，使入侵偵測系統之分類模型具有更精準的檢測能力。

參、系統架構與方法

本研究提出之入侵偵測系統架構圖，如圖一，主要分成兩大部分為資料前處理與訓



圖一：系統架構圖

練模型部分。資料前處理部分由編碼與正規化組成，將訓練資料或封包資訊編碼並正規

化成適合深度學習模型之輸入。訓練模型部分則採用深度神經網路及長短期記憶模型進行分類器的基礎，對於調整深度學習技術之參數採用差分進化演算法挑選適合的參數組合。

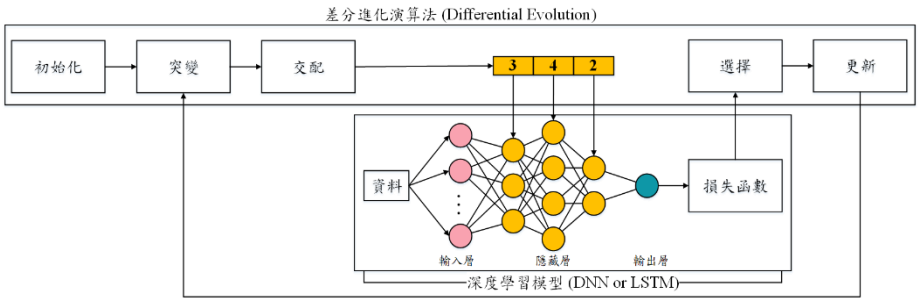
3.1 資料前處理

當輸入資料或訓練資料有問題，經過訓練或預測之結果擁有高的錯誤機率，因此將訓練資料輸入至訓練模型或是分類器前需要進行資料前處理。資料前處理主要分成三個部分：缺失值處理、資料編碼 (encoding) 及正規化 (normalization)。缺失值處理主要有兩種做法，其一為丟棄具有缺失值的資料，但深度學習模型訓練過程通常需要大量資料，若丟棄過多訓練資料時將導致模型準確率明顯下降。另一種則是補值，透過資料欄位之統計資訊進行缺失欄位之估計，因此本系統架構使用訓練資料欄位之平均值進行補值，以完成缺失值處理。一般來說訓練資料的欄位並非都是統一的資料型態組成，統一資料編碼方式採用 one-hot encoding 技術。舉例來說當欄位具有 A、B、C 與 D 四種組合，經由 one-hot encoding 編碼後產生 4 個欄位，每個欄位只有 0 或 1 兩種情況。經過此方式編碼轉換，擁有資料間的向量距離相等特性及擴增資料特徵之優點。最後為了避免部分特徵值過大或過小導致其他特徵之影響力下降，本論文採用 min-max normalization 將資料等比例縮放到 [0,1] 區間。

3.2 使用差分進化演算法調整深度學習之超參數

本論文採用差分進化演算法進行深度學習之超參數組合的搜尋。其模型訓練之流程圖如圖二所示。利用差分進化演算法調整深度學習神經元個數，主要步驟分成初始化、突變、交配、選擇、更新以及深度學習模型部分。以下將針對各步驟詳細說明。

初始化 (Initial)：差分進化演算法需要初始化個體，其過程需要設定每個維度的最大值與最小值，如公式 (1) ~ (3)。其中 V_{max} 及 V_{min} 分別為最大值與最小值的集合，j 為維度，x 為個體，而 $rand(0,1)$ 代表隨機亂數且該值介於區間 0 與 1 之間。經過初始



圖二：(Figure 2) 模型

化階段每個個體都會擁有不同的數值，這些數值皆會介於 V_{max} 與 V_{min} 間，這些個體

的組合稱之為一組解，如圖二。

$$V_{max} = \{v_{max}^1, v_{max}^2, \cdots, v_{max}^j\}, \quad (1)$$

$$V_{min} = \{v_{min}^1, v_{min}^2, \cdots, v_{min}^j\}, \quad (2)$$

$$x = rand(0,1) \times (V_{max} - V_{min}) + V_{min}. \quad (3)$$

- **突變 (Mutation)**：差分進化演算法透過突變策略達成個體差異，透過對解的每個個體進行修改，以期使群組更接近最佳解，如公式 (4)。其中 $M_{i,j}$ 表示經突變後的突變向量，t 代表當前迭代次數，$x_{r1,j}$、$x_{r2,j}$ 及 $x_{r3,j}$ 表示從個體中隨機選取的目標向量，F 則是突變步驟中的縮放因子。F 若設定過小將導致演算法收斂太過，使得解落入於區域最佳解；反之 F 設定過大則導致演算法無法收斂。

$$M_{i,j}(t+1) = x_{r1,j}(t) + F \times \Big(x_{r2,j}(t) - x_{r3,j}(t)\Big). \quad (4)$$

- **交配 (Crossover)**：交配過程主要是讓目標向量及突變向量進行資訊交換，計算方式如公式 (5)。$C_{i,j}$ 表示經過交配過程產生的試驗向量 (trial vector)，CR 為交配步驟之機率，r 則是產生介於區間 0 與 1 之間的隨機數。在此過程中先產生亂數 r，並假如 r 小於 CR，新子代選擇突變向量之值；反之則保留原始的個體值。解的每個個體皆會透過此步驟進行資訊交換，增加演算法的搜尋多樣性。

$$C_{i,j}(t+1) = \begin{cases} M_{i,j}(t+1) & \text{if } r \leq CR, \\ x_{i,j}(t) & \text{otherwise.} \end{cases} \quad (5)$$

- **選擇 (Selection)**：當解經由交配產生新的子代後進入選擇階段，本階段主要計算子代的適應值 (fitness value)，挑選出具有較佳適應值的解，如公式 (6) 所示。S_i表示經過選擇步驟後所保留下來的向量，$f(\ )$ 為衡量解之函數，該函數根據問題類型與定義有不同的計算方式。該過程採用貪婪策略，當交配之後的個體具有較好適應值，將會取代原始的個體；反之新的個體具有較差之適應值，則捨去該個體。

$$S_i(t+1) = \begin{cases} C_i(t+1) & \text{if } f\big(C_i(t+1)\big) \leq f\big(x_i(t)\big), \\ x_i(t) & \text{otherwise.} \end{cases} \quad (6)$$

- **更新 (Update)**：Lee 等人 [28] 提出多群體合作架構概念並實作於差分進化演算法上，在原始差分進化演算法的最後步驟加上更新機制，該機制會檢查當前最佳解是否已經重複 5 個迭代以上，若最佳解連續 5 個迭代不變則執行此機制。更新機制會從區域最佳解及當前最佳解中各別選擇一個維度並求其平均，將該值取代群體中隨機一個維度。當演算法執行過程落入區域最佳解時，可透過此機制跳脫區域最佳解。
- **深度學習模型**：本研究透過差分進化演算法尋找深度學習的參數組合，而一組解的適應值來自使用該組參數進行深度學習模型的訓練，訓練完成後會得到損失函數 (loss function) 值，該損失函數值會作為衡量解之適應值，再透過差分進化演算法尋找下一個迭代適合的深度學習參數組合。而入侵偵測系統的分類器屬於分類問題，因此適應值採用交叉熵 (cross-entropy) [29] 作為損失函數 $\mathcal{L}$，如公式 (7)。其中 N 為資料集數量，y_i 為第 i 筆資料實際類別，$\hat{y}_i$ 為模型預測機率。

$$\mathcal{L} = \frac{-1}{N} \times \sum_{i=1}^{N} y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i). \quad (7)$$

本研究使用差分進化演算法應用於深度學習完整演算法如演算法 1 所示，首先初始化需先設定最大值與最小值，本研究採用自動編碼器主要功能可以降低資料維度使演算法可以找出資料的特徵，如演算法 1 的第 2、3 行所示；而第 6 至 17 行為檢查突變過程或是交配過程產生的個體是否超越設定之範圍並將其修改；第 18 行為將演算法搜尋

演算法 1： 差分進化演算法應用於深度學習

Input : Population $\boldsymbol{n}$, Dimension $\boldsymbol{D}$, scaling factor $\boldsymbol{F}$, crossover rate $\boldsymbol{CR}$, maximum iteration $\boldsymbol{T}$
Output: global best solution

1: $t \leftarrow 1$
2: $V_{max} \leftarrow$ input size
3: $V_{min} \leftarrow$ category
4: $x_{i,j} \leftarrow$ Initial (V_{max}, V_{min})
5: **WHILE** $t \leq T$ **DO**
6: $M_{i,j} \leftarrow \text{Mutation}(x_{i,j})$
7: **IF** $M_{i,j} \leq v_{min}^{j}$ **THEN**
8: $M_{i,j} \leftarrow v_{min}^{j} + 1$
9: **ELSE IF** $v_{max}^{j} \leq M_{i,j}$ **THEN**
10: $M_{i,j} \leftarrow v_{max}^{j} - 1$
11: **END IF**
12: $C_{i,j} \leftarrow \text{Crossover}(M_{i,j}, x_{i,j})$
13: **IF** $C_{i,j} \leq v_{min}^{j}$ **THEN**
14: $C_{i,j} \leftarrow v_{min}^{j} + 1$
15: **ELSE IF** $v_{max}^{j} \leq C_{i,j}$ **THEN**
16: $C_{i,j} \leftarrow v_{max}^{j} - 1$
17: **END IF**
18: $f(x_i),\ f(C_i)$ ←Perform deep neural network or long-short-term memory
19: $S_i \leftarrow \text{Selection}(f(x_i),\ f(C_i))$
20: $x_{i,j} \leftarrow \text{Update}(S_i)$
20: $t \leftarrow t + 1$
22: **END WHILE**

的參數組合設定為深度學習的參數並使用訓練資料進行訓練，完成訓練取得損失函數值作為差分進化演算法之適應值。其中深度學習架構使用深度神經網路，可使用長短期記憶模型進行替換。第 19 行為選擇階段淘汰不良適應值的解；最後第 20 行為更新機制，當深度學習之參數組合 5 個迭代內未更動，強制使用更新機制修改解的一部份，使演算法擁有跳脫區域最佳解的能力。

肆、實驗結果與討論

本研究實驗環境為使用 Intel i9-7900x (3.3GHz, 13.75 cache, 10 cores) 處理器，搭載 16 G 記憶體及 GTX 2080Ti 顯示卡，作業系統為 Ubuntu 18.04 上進行開發與測試。本

研究實現機器學習之單純貝氏分類器 (naïve Bayes classifiers) [30]、隨機森林 (random forest, RF) [31]、支持向量機 (support vector machine, SVM) [20] 以及深度學習之深度類神經網路 (deep neural network, DNN) [14]、長短期記憶模型 (long-short-term memory, LSTM) [15]以上經典之方法進行比較。為了證明本研究提出之方法可以超越新型演算法，也實現深度自動編碼器 (deep auto-encoder) [32] 及非對稱型堆疊式自動編碼器 (stacked non-symmetric deep auto-encoders, SNDAE) [27] 進行比較。

上述機器學習方法之參數是根據文獻 [33] 的結果進行設定，隨機森林模型是由許多樹組合而成，隨機森林模型需要設定由多少樹組合其參數表示為 $n-estimators$ 並設定為 100；支持向量機需要給定懲罰係數 C ，該數值表示模型容忍誤差根據文獻設定為 1；深度學習方面需要設定隱藏層之神經元個數、訓練次數 (epoch) 及最佳化器 (optimizer)，神經元個數設定為 80；最佳化器採用 adam (adaptive moment estimation) 最佳化器；學習率 (learning rate) 設定為 0.5；批次大小 (batch size) 設定為 2048。經本研究測試長短記憶模型最佳化器選擇 RMSProp 擁有最佳的效能。而差分進化演算法有 4 個參數分別為個體數量 n、突變因子F、交配機率 CR 及迭代次數 t，個體數量設定為 20，突變因子表示演算法移動之步伐距離設定為 0.5；交配機率則為 0.3；迭代次數為 40 次。

4.1 資料集與衡量指標

本研究提出之入侵偵測系統使用 NSL-KDD [34] 資料集作為訓練及測試，其前身來自 KDD CUP 99 資料集。KDD CUP 99 存在許多重複或冗餘的資料紀錄，且訓練與測試的比數分配不均。NSL-KDD 資料集中有五個分類：正常行為及四種網路攻擊手段，分別為 1). 阻斷服務攻擊 (denial of service, DOS)：透過不停發送請求封包使目標設備的資源消耗殆盡，導致正常使用者無法存取或連線。2). 未授權遠端登入 (remote to login, R2L)：因存取遠端控制密碼安全性失能，導致攻擊者可以存取目標進行不當操作。3.) 取得未授權之最高管理權線 (user to root, U2R)：利用普通權限帳號繞過驗證或使用系統漏洞取得最高管理權限，進行不當行為。4.) 網路連接埠監視或掃描 (Probe)：透過程式對目標主機進行連接埠掃描以取得漏洞進行攻擊。另外 NSL-KDD 以事先將資料集切割分別為：訓練資料共 125,973 筆，測試資料 22,544 筆。

本研究採用分類模型中最常使用的混淆矩陣 (confusion matrix) 並計算指標。其中 True 和 False 表示資料真實分別為正類別或負類別；Positive 和 Negative 代表資料經模型預測後為正類別還是負類別；True Positive (TP) 表示資料實際上為正類別且被模型正確分類到正類別；True Negative (TN) 代表資料實際上為負類別且被模型正確分類到負類別；False Positive (FP) 為目標為負類別但被模型錯誤分類到正類別中；False Negative (FN) 表示目標為正類別但被模型錯誤分類到負類別中。根據混淆矩陣可以計算三種模型衡量指標分別是準確率 (accuracy, ACC)、召回率 (recall, DR) 及精密度

(precision, PR)，上述三種衡量指標定義如公式 (8)~(10)，準確率、召回率及機密度數值越高表示模型辨識能力越精準越好。

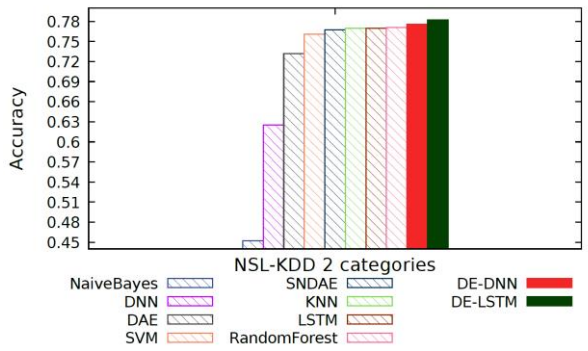
$$ACC = \frac{TP + TN}{TP + FP + TN + FN}, \tag{8}$$

$$DR = \frac{TP}{TP + FN}, \tag{9}$$

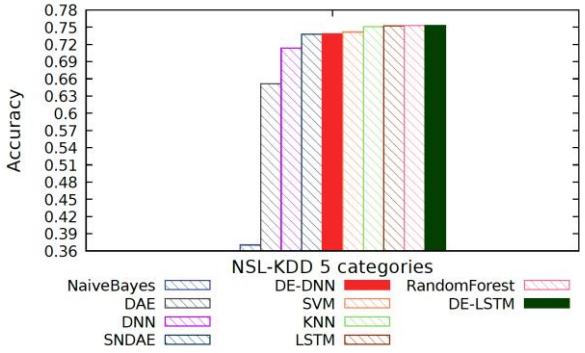
$$PR = \frac{TP}{TP + FP}. \tag{10}$$

4.2 實驗結果

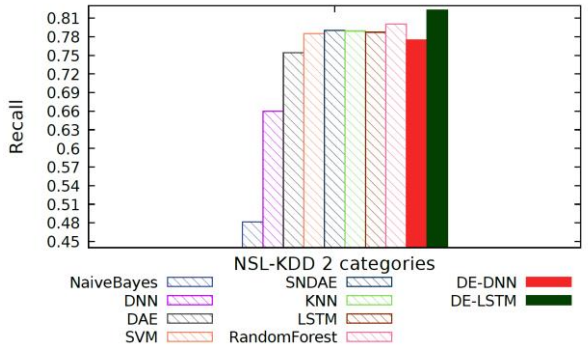
NSL-KDD 資料集擁有為二分類 (正常與非正常) 及五分類 (正常與四種網路攻擊手段) 兩種測試資料，其中有普通難度的測試資料集與困難測試資料集 (NSL-KDD Test-21)，所以實驗結果有四種測試資料。為了呈現所有演算法效能，實驗結果使用表格與長條圖表示，本研究提出方法以實心長條圖呈現，其他演算法為空心長條圖，如圖三及圖四。



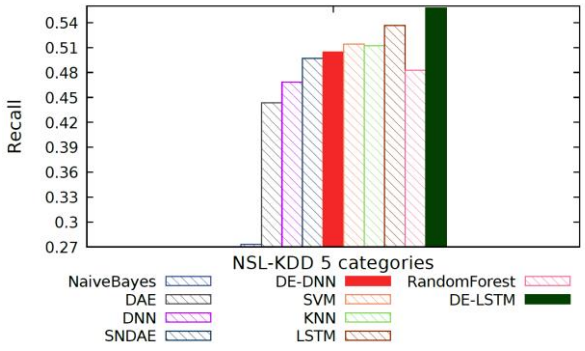
圖三：NSL-KDD 二分類-準確率



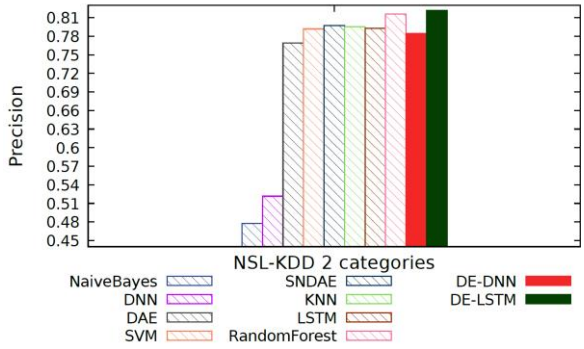
圖四：NSL-KDD 五分類-準確率



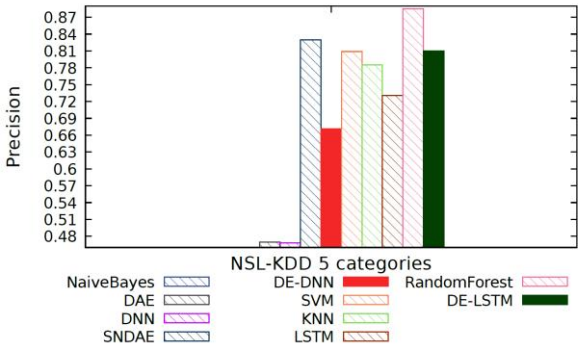
圖五：NSL-KDD 二分類-召回率



圖六：NSL-KDD 五分類-召回率



圖七：NSL-KDD 二分類-精確度



圖八：NSL-KDD 五分類-精確度

一般 NSL-KDD 測試資料集之準確率如圖三及圖四所示；一般 NSL-KDD 測試資料集之召回率如圖五及圖六所示;一般 NSL-KDD 測試資料集之精確度如圖七及圖八所示。本研究所提出方法在二分類問題準確率、召回率及精確度明顯超越其他演算法且搭配深度類神經網路或長短期記憶模型可以有效提升模型準確率，在深度類神經網路甚至可以提高約 15% 準確率。在五分類問題中本研究提出的方法與隨機森林演算法不分上下，但使用超啟發式演算法調整深度學習模型的超參數可以提升模型之 2.5% 準確率，如圖三及圖四所示。圖五及圖六呈現各演算法召回率，根據召回率公式分析，模型的 FN 值會影響該值的高低，而 FN 則代表是該模型的資料誤判比數，因此從召回率之長條圖觀察本研究提出演算法相較其他方法可以有效減少資料誤判情況發生。圖七與圖八則為各演算法的精確度之長條圖，根據精確度公式來看，本研究提出之方法模型的 FP 值數值稍微高，該值代表攻擊方法演算法未判斷出來，導致精確度相比其他演算法下略為遜色，但還是在可接受範圍內，詳細數據記載於表一。

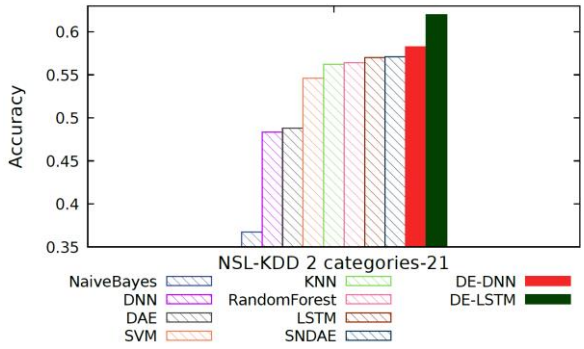
表一：NSL-KDD 實驗數據

演算法名稱	二分類			五分類		
	ACC	DR	PR	ACC	DR	PR
*k*NN	0.7696	0.7891	0.7955	0.7511	0.5120	0.7851
SVM	0.7607	0.7851	0.7918	0.7415	0.5142	0.8089
RF	0.7707	0.8002	0.8155	0.7528	0.4825	0.8850
naïve Bayes	0.4522	0.4812	0.4775	0.3703	0.2730	0.3427
DNN	0.6251	0.6596	0.5215	0.7135	0.4682	0.4681
LSTM	0.7698	0.7872	0.7926	0.7524	0.5365	0.7308
DAE	0.7316	0.7544	0.7688	0.6511	0.4437	0.4697
SNDAE	0.7673	0.7900	0.7973	0.7377	0.4968	0.8295
DE-DNN	**0.7757**	**0.7746**	**0.7843**	**0.7383**	**0.5046**	**0.6711**
DE-LSTM	**0.7822**	**0.8230**	**0.8219**	**0.7529**	**0.5575**	**0.8099**

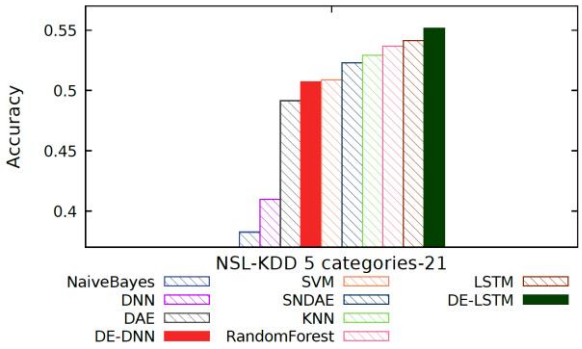
表二：NSL-KDD Test-21 實驗數據

演算法名稱	二分類			五分類		
	ACC	DR	PR	ACC	DR	PR
*k*NN	0.5621	0.6098	0.5653	0.5291	0.4376	0.6928
SVM	0.5459	0.6024	0.5609	0.5088	0.4371	0.7187
RF	0.5641	0.6851	0.6123	0.5367	0.4449	0.7098
naïve Bayes	0.3673	0.5794	0.5694	0.3825	0.3352	0.4583
DNN	0.4835	0.6938	0.6163	0.4097	0.3029	0.2245
LSTM	0.5700	0.6005	0.5597	0.5412	0.4591	0.6789
DAE	0.4878	0.5585	0.5355	0.4914	0.3571	0.3519
SNDAE	0.5711	0.6849	0.6130	0.4914	0.3571	0.3519
DE-DNN	**0.5824**	**0.6178**	**0.5701**	**0.5070**	**0.3928**	**0.4999**
DE-LSTM	**0.6199**	**0.7249**	**0.6340**	**0.5935**	**0.4780**	**0.6545**

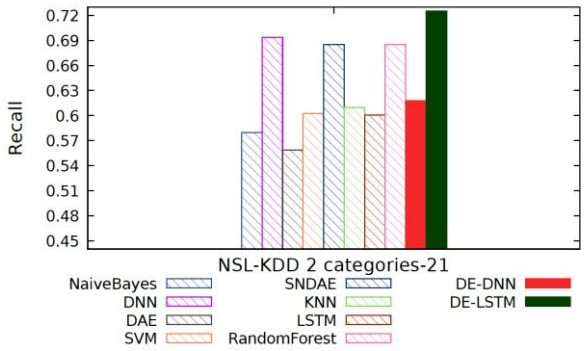
而圖九至圖十四則為 NSL-KDD 困難資料集之各演算法表現狀況，可以明顯發現使用超啟發式演算法動態調整深度學習的超參數可以有效提高二分類模型之分類準確率，大約提升 10% 的準確率如圖九。從二分類之召回率顯示本研究提出之方法大幅減少分類之誤判筆數，如圖十一中 DE-LSTM 之召回率約為 72.5%。從精確度之表現看出本論文所提出方法在五分類困難資料集表現較低，但準確率方面依然優於其他演算法，詳細實驗數據記錄於表二。從困難資料集之測試實驗來看，使用超啟發式演算法調整深度類神經網路在二分類問題上是有效提升模型準確率，將深度類神經網路替換成長短期記憶模型後，分類模型之準確率明顯上升。而五分類問題之表現更明顯說明長短期記憶模型準確率優於其他分類模型，因為網路封包資訊具有時序特性，使用具有處理時序性能力的長短期記憶模型相較於使用深度類神經網路作為基底時，更能在上述資料集獲得較優的成績。最後從實驗結果證明使用超啟發式演算動態尋找深度學習參數，能夠有效提升準確率，因此本研究提出方法可以更精準地檢測出網路攻擊行為或者更複雜的網路攻擊行為。



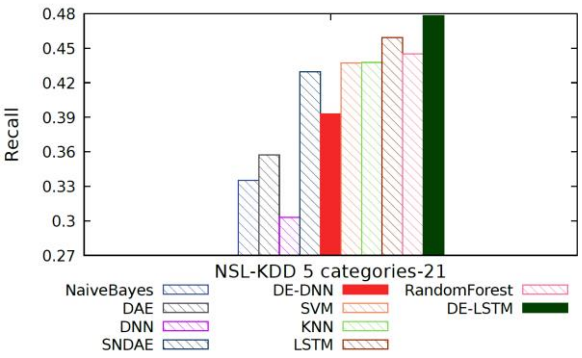
圖九：NSL-KDD-21 二分類-召回率



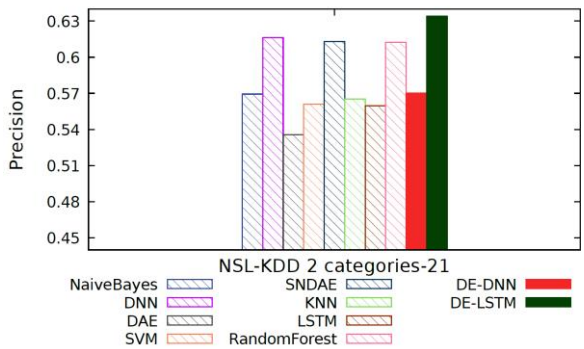
圖十：NSL-KDD-21 五分類-召回率



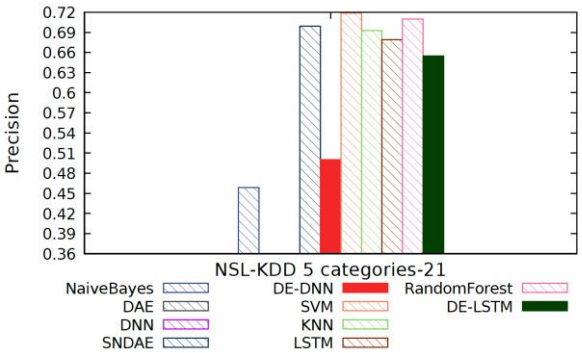
圖十一：NSL-KDD-21 二分類-精確度



圖十二：NSL-KDD-21 五分類-精確度



圖十三：NSL-KDD-21 二分類-精確度



圖十四：NSL-KDD-21 五分類-精確度

伍、結論

本論文提出了一個混合型演算法，其結合了差分進化演算法和深度學習模型，此演算法主要是利用超啟發式演算法的特性，在有效的時間內且有策略性的找尋近似於最佳的參數值，來提升深度學習模型的檢測能力。本論文使用了兩種來自不同環境的真實資料集作為評估本論文所提出的方法成果，並且也與其他著名的機器學習方法以及於 2018 年所發表的改良基於自動編碼器後的深度學習方法進行效能上的比較。在最終的實驗結

果證明了本篇論文所提出的方法在較為複雜的問題上比起其他演算法具有更出色的表現，且在簡單的問題中該演算法也是能夠維持一定的成果。在深度學習當中並非只有神經元個數可以調整，例如：學習率 (learning rate) 、批次大小 (batch size) 、隱藏層層數 (hidden layers) 甚至優化器 (optimizer) 的挑選，因深度學習參數多，因此未來大概會嘗試使用超啟發式演算法同時調整多個深度學習中的參數或者使用超啟發式演算法取代最佳化器的工作。

誌謝

本研究由科技部計畫 MOST-108-2221-E-110-076-MY3 及 MOST 109-2221-E-005-047 補助支持，特此誌謝。

參考文獻

[1] A. Gilchrist, Industry 4.0: The Industrial Internet of Things. New York: Springer, 2016.

[2] R. M. Lee, M. J. Assante and T. Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid: Defense Use Case., 2016, https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf (2021/01/14).

[3] A. Burt and T. Pott, "Gitpaste-12: A New Worming Botnet with Reverse Shell Capability Spreading via GitHub and Pastebin, 2020, https://blogs.juniper.net/en-us/threat-research/gitpaste-12 (2021/01/14).

[4] F. A. Alaba, M. Othman, I. A. T. Hashem and F. Alotaibi, "Internet of Things Security: A Survey," Journal of Network and Computer Applications, vol. 88, pp. 10-28, 2017.

[5] P. H. Duy and N. N. Diep, "Intrusion Detection Using Deep Neural Network," Southeast Asian Journal Sciences, vol. 5, no. 2, pp. 111-125, 2017.

[6] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi and M. Ghogho, "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking," in Proceedings of the International Conference on Wireless Networks and Mobile Communications, 2016, pp. 258-263.

[7] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," IEEE Access, vol. 5, pp. 21954-21961, 2017.

[8] J. Kim, J. Kim, H. L. T. Thu and H. Kim, "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection," in Proceedings of the International Conference on Platform Technology and Service, 2016, pp. 1-5.

[9] X. S. Yang, S. Deb and S. Fong, "Accelerated Particle Swarm Optimization and Support

Vector Machine for Business Optimization and Applications," in Proceedings of the International Conference on Networked Digital Technologies, 2011, pp. 53-66.

[10] E. Tuba, M. Tuba and D. Simian, "Adjusted Bat Algorithm for Tuning of Support Vector Machine Parameters," in Proceedings of the IEEE Congress on Evolutionary Computation, 2016, pp. 2225-2232.

[11] S. R. Young, D. C. Rose, T. P. Karnowski, S. H. Lim and R. M. Patton, "Optimizing Deep Learning Hyper-Parameters through An Evolutionary Algorithm," in Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, 2015, pp. 1-5.

[12] Z. Tian and S. Fong, "Survey of Meta-Heuristic Algorithms for Deep Learning Training," in Optimization Algorithms—Methods and Applications, IntechOpen, 2016, pp. 196-220.

[13] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.," Journal of Global Optimization, vol. 11, pp. 341-359, 1997.

[14] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning. United States: The MIT Press, 2016.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[16] H. J. Liao, C. H. R. Lin, Y. C. Lin and K. Y. Tung, "Intrusion Detection System: A Comprehensive Review," Journal Network Computer Applications, vol. 36, no. 1, pp. 16-24, 2013.

[17] C. F. Tsai, Y. F. Hsu, C. Y. Lin and W. Y. Lin, "Intrusion Detection by Machine Learning: A Review," Expert Systems with Applications, vol. 36, no. 10, pp. 11994-12000, 2009.

[18] J. M. Keller, M. R. Gray and J. A. Givens, "A Fuzzy K-nearest Neighbor Algorithm," IEEE Transactions on System, Man, Cybernetics, vol. 15, no. 4, pp. 580-585, 1985.

[19] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1, no. 1, pp. 81-106, 1986.

[20] C. Cortes and V. Vapnik, "Support Vector Networks," Machine Learning., vol. 20, pp. 273-297, 1995.

[21] T. Shon and J. Moon, "A Hybrid Machine Learning Approach to Network Anomaly Detection," Information Sciences, vol. 177, no. 18, pp. 3799-3821, 2007.

[22] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai and K. Dai, "An Efficient Intrusion Detection System based on Support Vector Machines and Gradually Feature Removal Method," Expert Systems with Applications, vol. 39, no. 1, pp. 424-430, 2012.

[23] S. M. Bridges and R. B. Vaughn, "Intrusion Detection via Fuzzy Data Mining," in

Proceedings of the Annual Canadian Information Technology Security Symposium, 2000, pp. 109-122.

[24] G. Giacinto, R. Perdisci, M. D. Rio and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers," Information Fusion, vol. 9, no. 1, pp. 69-82, 2008.

[25] D. Kang, D. Fuller and V. Honavar, "Learning classifiers for misuse and anomaly detection using a bag of system calls representation," in Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, 2005, pp. 118-125.

[26] S. Mukkamala, A. H. Sung and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," Journal Network and Computer Applications, vol. 28, no. 2, pp. 167-182, 2005.

[27] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[28] W. P. Lee and W. J. Chien, "A Novel Differential Evolution with Co-evolution Strategy," Journal of Computers, vol. 6, no. 3, pp. 594-602, 2011.

[29] Redstonewill, "Cross Entropy." 2020.

[30] D. Hand and K. Yu, "Idiot's Bayes: Not So Stupid after All?," International Statistical Review, vol. 69, no. 3, pp. 385-398, 2007.

[31] A. Liaw and M. Wiener, "Classification and Regression by Random Forest," R News, vol. 2, no. 3, pp. 18-22, 2002.

[32] F. Farahnakian and J. Heikkonen, "A Deep Auto-encoder based Approach for Intrusion Detection System," in Proceedings of the International Conference on Advanced Communication Technology, 2018, pp. 178-183.

[33] Z. H. Chen, Y. L. Chen, W. Y. Chang and C. W. Tsai, "A Hybrid Classification Algorithm for Intrusion Detection System," Communications of the CCISA, vol. 25, no. 1, pp. 14-27, 2019.

[34] University of New Brunswick, NSL-KDD dataset, https://www.unb.ca/cic/datasets/nsl.html (2021/01/18)