Salud, Ciencia y Tecnología. 2024; 4:797
doi: 10.56294/saludcyt2024797

ORIGINAL

Hybrid Support Vector Machine for Predicting Accuracy of Conflict Flows in Software Defined Networks

Máquina de vectores de apoyo híbrida para predecir la precisión de los flujos de conflictos en redes definidas por software

B. Ananth[1]

[1]Deptartment of Computer Science and Engineering, Puducherry Technological University. Puducherry-605014, India.

Cite as: Ananth B. Hybrid Support Vector Machine for Predicting Accuracy of Conflict Flows in Software Defined Networks. Salud, Ciencia y Tecnología. 2024; 4:797. https://doi.org/10.56294/saludcyt2024797

Submitted: 06-11-2023 **Revised:** 12-01-2024 **Accepted:** 13-02-2024 **Published:** 14-02-2024

Editor: Dr. William Castillo-González

ABSTRACT

Software Defined Networking (SDN) is an infrastructure platform for delivering simplified and compliant services with flexible services. These are the means of centralized maintenance and adaptive functions. SDN is affected by various contention flows and causes network performance issues. In this case, we need to provide efficient solutions to handle conflicting flows with better priority and actions. In this paper, we propose a DeepQ Residue method for analyzing normal and conflicting flow scenarios in the load balancing phase. During simulation, an open SDN network is generated using TensorFlow. We use a Hybrid Support Vector machine with an improved decision tree method to predict accuracy and performance. In this case, we analyze threads from 1000 to 100 000 in increments of 10 000 threads in each iteration. Here, we train a deep belief network with a decision-free feature for environmental simulation. Based on the simulation results, the accuracy of our proposed method reaches 97 %, and we compare the results with the results of various existing methods. Our proposed algorithm provides a high-performance SDN application with different conflicting load-balanced flows.

Keywords: Software Defined Networks; Load Balancing; Conflict Flow; Accuracy; Prediction.

RESUMEN

Las redes definidas por software (SDN) son una plataforma de infraestructura para ofrecer servicios simplificados y compatibles con servicios flexibles. Se trata de medios de mantenimiento centralizado y funciones adaptativas. SDN se ve afectada por diversos flujos de contención y provoca problemas de rendimiento de la red. En este caso, necesitamos proporcionar soluciones eficientes para manejar los flujos conflictivos con mejor prioridad y acciones. En este artículo, proponemos un método DeepQ Residue para analizar escenarios de flujos normales y conflictivos en la fase de equilibrio de carga. Durante la simulación, se genera una red SDN abierta utilizando TensorFlow. Utilizamos una máquina de vectores de soporte híbrida con un método de árbol de decisión mejorado para predecir la precisión y el rendimiento. En este caso, analizamos hilos de 1000 a 100 000 en incrementos de 10 000 hilos en cada iteración. Aquí, entrenamos una red de creencia profunda con una característica sin decisión para la simulación del entorno. Basándonos en los resultados de la simulación, la precisión de nuestro método propuesto alcanza el 97 %, y comparamos los resultados con los de varios métodos existentes. Nuestro algoritmo propuesto proporciona una aplicación SDN de alto rendimiento con diferentes flujos conflictivos de carga equilibrada.

Palabras clave: Redes Definidas por Software; Equilibrio de Carga; Flujo en Conflicto; Precisión; Predicción.

INTRODUCTION

Software Defined Networks has data plane and control plane distributed networks. Each is choice producing system with data traffic and transfer flows. These network allows to the providers has flows rules such as routing,[1] qualify of services,[2] network traffic,[3] forwarding.[4] SDN has network architecture and split the network into layered services. In this case control and forwarding logic has various flows and components frameworks. Open flow is the major issue in SDN and need to maintain flow table for avoiding conflict and balance the loads based on the request. For example Amazon web services enables in build load balancer for monitoring the flow controls.[5]

While selecting SDN we need to consider controllers,[6] networking operating systems,[7] application programming interface[8] feature and flow table optimization.[9] The each action flows are recorded and controller will be activated based on open flow request. The open flow structure is shown in figure 1 and which has multiple connected components. Open flow is the controller which has link between flow table and priority index. Each flow table loads are configured by controller and each packet are processed by using store and forward feature.[10]

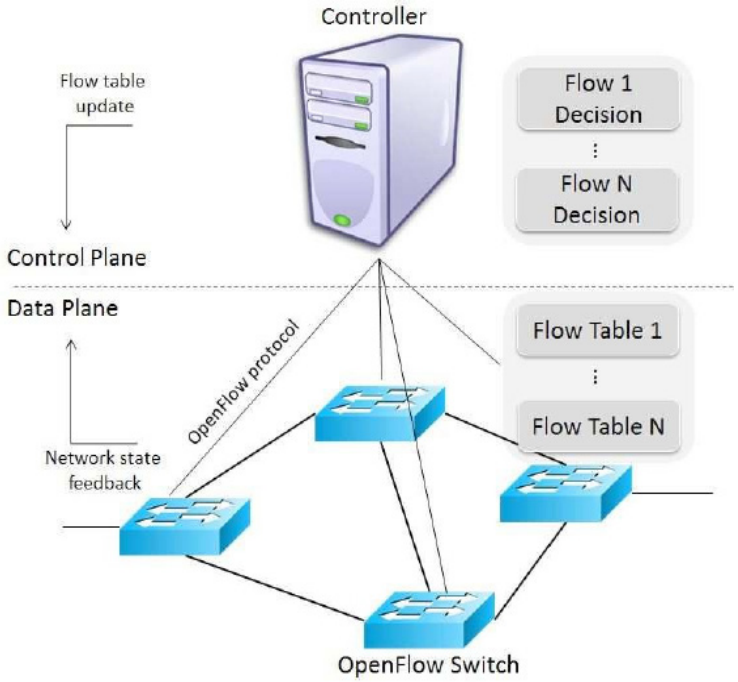

Figure 1. OpenFlow SDN operation flow and plane

Flow table has priority index,[11] matching the flows,[12] packet delivery,[13] counters[14] and conflict actions.[15] Cookies are enabler to select the incoming packet information, address the flow priority, matching the packet and conflict flow. In this case total response time, time out representations, idle request time, total turnaround time and waiting time are considered. The controller will adjust the filter and delete the unwanted/redundant flow data. This paper we propose hybrid support vector and enhanced decision tree approach for classification, accuracy and prediction index of conflict flows in SDN.[16]

Its conflict detection system is deployed by using deep learning features. The iteration based evaluation is done to assess the performance. In this paper Decision tree method is applied for generating deep belief network and support vector machine for analyzing performance. Detect and classify the conflict flows by measuring accuracy, precision, recall and execution time by varying flows at different intervals. This paper is organized as section 2 gives various related works, section 3 gives proposed system process, section 4 explain algorithms and SDN simulations and section 5 gives conclusion and future enhancement.

Related Work

SDN is a new technology which has high flexible factors. Application of SDNs is to manage balancing the loads, access controls, routing and performance. Manikandan et al, decoupling has logic of network control and network performance measurements. This case networking performance can be selected based routers, network flow features and infrastructure requirements. Wong et al, organizing the network based on network partitions and control function with fast forwarding features.[17]

Configuration control is another factor to set the efficiency of the network and fix the framework. Chiago et al, SDN has various advantages and one of the major requirements is centralized monitoring feature with efficient

https://doi.org/10.56294/saludcyt2024797

network capabilities. Based on various literatures control and data plane leads the hardware components and increase the choice the vendors. We test the simulations using commercial software with various networking devices such as routers, switches, gateways and storage servers.[18]

OpenFlow is the layered features which has centralized embedded features connected components functions. Control layer has networking functions with running environments. Sunaiman and Runge et al, proposed that infrastructure, control and application layers are major key players for evaluating the SDN. Application layer focused on business results, control layer has network services and protocols, infrastructure layer provide connected components with resources.[19]

Based on various related works table 1 shows that the algorithm and accuracy index of SDN dataset. Depending upon the data transmission we need an efficient approach to detect routing protocols, efficient transmission and gained network traffic.

Table 1. Various Literature result

Year	Dataset	Accuracy	Algorithm
2021	MNRE Set	87 %	CNN Classifier
2020	Information FlowSet	78 %	ReactJS
2018	CCN Set	79 %	Hypervisor
2015	MultiSet	82 %	RNN Flow
2014	ObjectSet	76 %	VM Optimizer
2010	DetectOFF	75 %	ModuleSPLIT

Source: IEEE CS,2022

From the various suggestion internet is major key player for affecting the network system with the aggregation of network virtualization and services.[20] We propose a hybrid approach to handle the network by means of multilingual dataset.

Proposed Method - Hybrid Support Vector Machine with Enhanced Decision Tree

Proposed method is implemented by using below figure 2. In this approach our system collects the conflict flows and normal flows by using open flow table results which is obtained from deep belief networks. In this method has three phases such as topological representations, creation of rules and open flow table for conflict flow.[21]

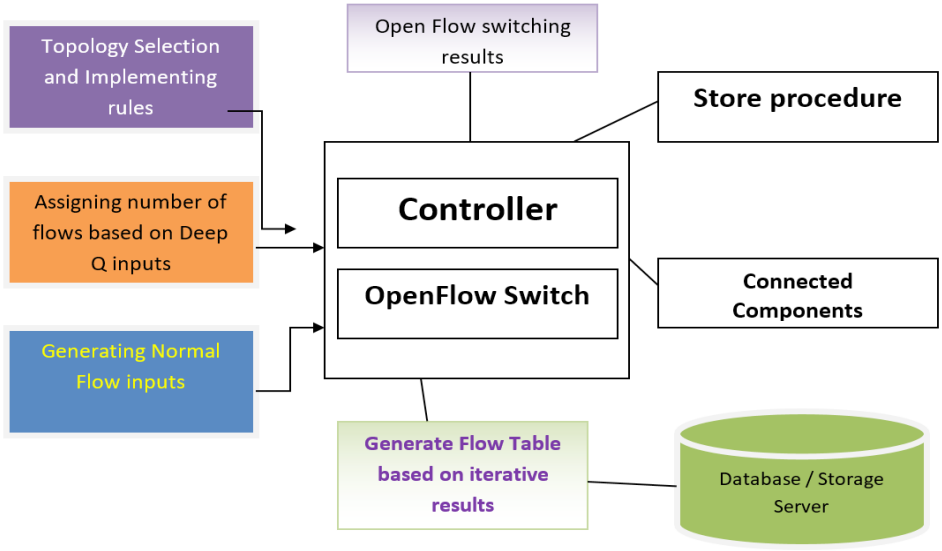

Figure 2. Proposed Generator model - Flow Table process

The proposed method has detection and classification phase. Detection phase is used to detect the flows such as normal and conflict flows. In this phase we used hybrid support vector machine is applied to controller plane to analyze the behaviours. In this case each feature is observed and stores the MAC, IP address and their

https://doi.org/10.56294/saludcyt2024797

actions. OpenFlow is used to forward the normal flows to open way and transmit the representation. Other scenario conflict phase representations are recorded. Each phase the classification is done for measuring the accuracy.[22]

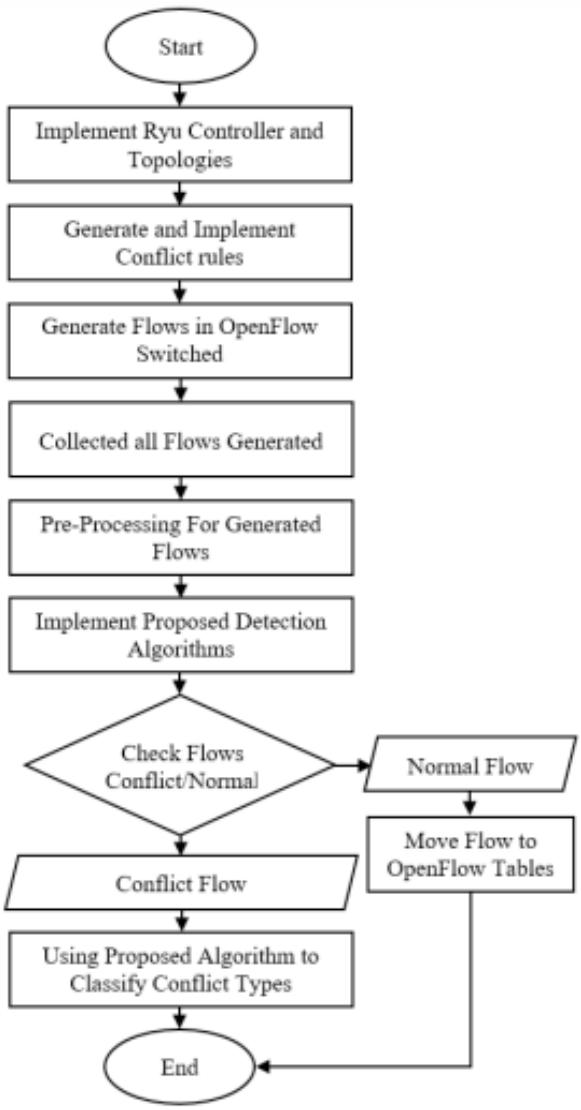

Figure 3. Flow chart for selection Support vector results using OpenFlow

Hybrid Support Vector Machine representations

Rule 1: Mac address is collected from each frame space (r) and corresponds to tuple representations (s,d) where they are source and destination address

Rule 2: The subset is observed by using 32 or 64 bit number which has IPv4 or v6 address with respect r

Rule 3: The segment phase is calculated from using s,d values and tuple representation results

Rule 4: Each tuple has source, destination, frame size, packet values, payload and denotation. It is represented as {s,d,f,p,x,y}

Function f : N -> number of flows rules

Which is obtained from each action field values and generated the decision tree as shown in figure 4

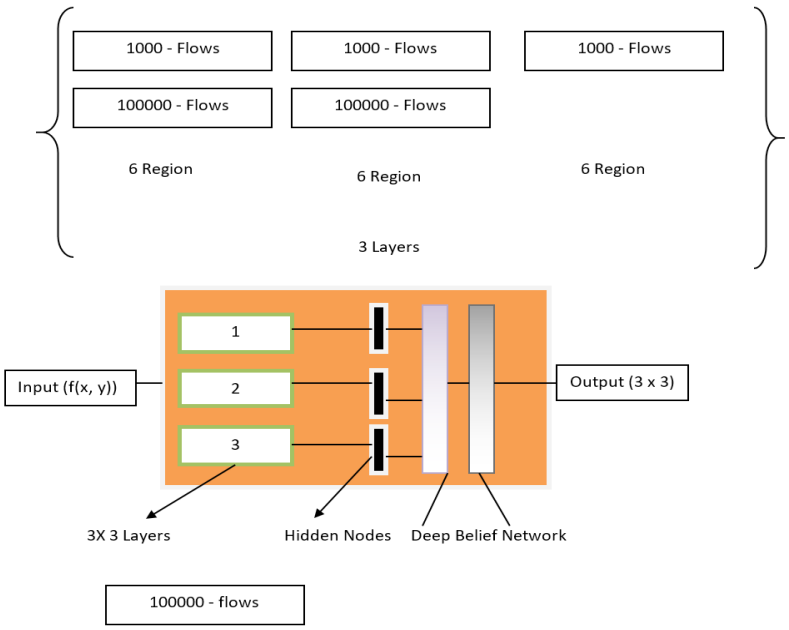

Figure 4. Deep Belief Network Generation by using Flow rule

Hence

R : f(N) based on ancient rule with setup transmitted and guided rules with respect to g,

Where as

If r = f(N) --- a and a = f(x,y) .... b then a,b belongs to region specific results

Below is the algorithm for generating SDN flow with respect to enhanced decision tree representations,

Algorithm 1: OpenFlow rules for generating decision tree of SDN using Conflict flow results

Input: Set of Flows L, R Host, Attack-D, P-Priority, T-Protocol, A-Action

Output: Conflict Rules and Deep Belief Networks

Stored - Procedure: topology(), Conflict_rule(), general_flow(), result()

1.topology (L,R,D,P,T,A)

2. L : (L X 10/1000) + D

3. D <- L/X + N

4. P<- X/2, Y/10, A(N)

5. For x is in N where (1, L+1)

6. Update the rules upto 6 = conflict rules

7. Conflict % == 10

8. if (P > P(n), T = T(n), A= A(n), Y, X. addr(IP). Addr & & Y.addr (IP)) then

9. return conflict_rule(x)

10. else if (P< P(n), T= T(n), A ≠A(n), X. addr(IP) && Y. addr(IP)) then

11. return topology(x,y)

12. else if (P < P(n), T = T(n), A ≠ A(n), X. addr (IP) || Y. addr(IP)) then

13. return general_flow(y)

14. else if (P < P(n), T=T(n), A ≠ A(n), X. addr(IP) ∩ Y. addr(IP)) then

15. return result(L,R,X,Y)

17. return conflict_rule()

18. else if (P < P(n), T=T(n), A = A(n), X. addr(IP) U Y. addr(IP)) then

19. return result(x,y)

20. else, (P= P(n), T = 0, A = A(n), X. addr(IP) ∩ Y. addr(IP) or N. addr(MAC) = R) then

21. return 1

22. Flow_Cookie = = 0

23. select L8

24. if buffer_Id = = 1000 then

25. classifier = conflict_flow(), cookie = topology(), buffer = 1, P= A(n)

26. else classifier =result(R,IP,MAC,N)

27. ip-address = packet.get()

28. src_ip = ip-address.src

29. dst_ip = ip-address.dst

https://doi.org/10.56294/saludcyt2024797

30. IP_Protocol = ip-address.protocol
31. Match = import.CSV || update.CSV

Algorithm of Enhanced Decision Tree procedure

Based on above representations support vector machine has selected the performance index of SDN. This algorithm detects the conflict flows based loads and summarized as

a. *Select and run algorithm values and their features*
b. *Check status of each flows*
c. *Apply open flow method to detect the normal and conflict*
d. *Classify the flows and generate deep belief network using below figure 4*

The phase 2 proposed based on classification index which is implemented by controller plane behaviours. Here priority factors are set as IP address, action and types. From the Figure 4 shows that detection model and classification model

Detection ← {Learning model, Training flow, Detection algorithm, Testing and Generate Flows}
Classification ← {Controller, Detection results, Decision Tree, Testing and Accuracy}

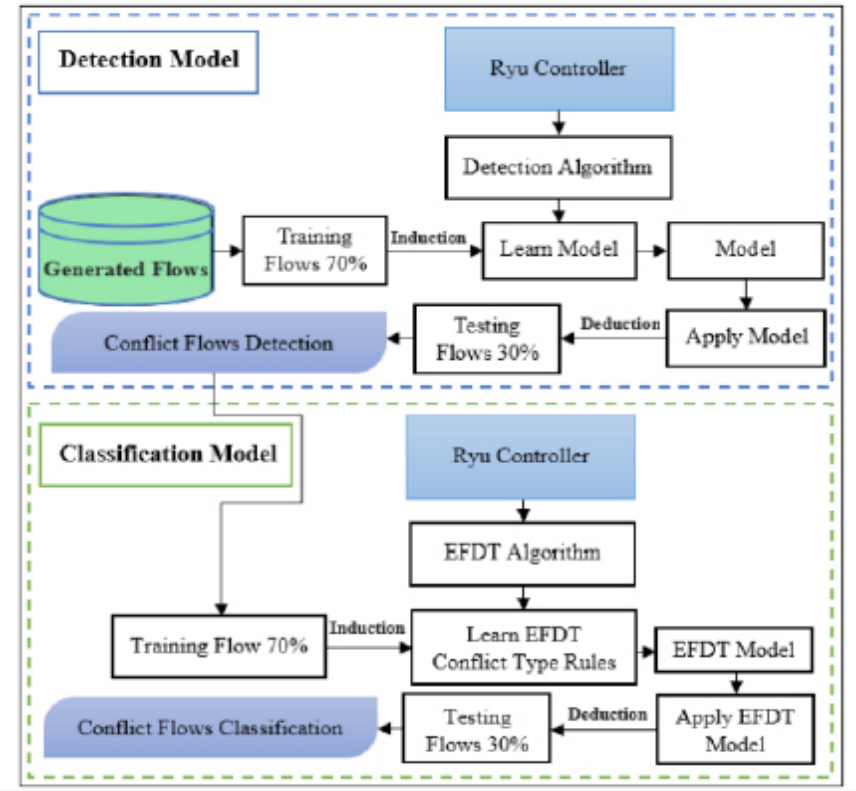

Figure 5. Detection and Classification Model Operations

The classification types has correlation index, generalization, shadowing techniques, overlapping factor and classification. Based on these categories we summarized as implement the decision tree classifier, apply detection flow rules, classify the priority and count the features based on iterations.

Experimental Setup and Simulations

Table 2. Specification of TensorFlow Simulators input

Hardware Specification	GPU Computing with 3.5GHz Intel i5 Processor, 1TB HDD and 8GB RAM
Operating System	Ubuntu above 15
OpenFlow	RYU switch OpenFlow Version
Software	Python Interpreter
Flows	1000 to 100000 with Iterations
Topology	Fat free topology

https://doi.org/10.56294/saludcyt2024797

Our research objective is to generate and collect the flows. Based on that we simulate the SDN and classify the flows as normal or conflict flows. In this case RYU classifier is proposed and TensorFlow is used for simulating the environment. Virtual box is enabled to activate the system and table 2 is listed for simulation specification of our network.

From the table 2 specification is applied for our simulation process. Fat free topology structure framed based on decision tree representation is shown in figure 5. In this case traffic generation is consisting of 1000 to 100000 flows with n number of iterations. Every server has multiple ports with respect to 8050, 9090, 7070 flow values. It is switching technique to hold the process and generate the accuracy index.

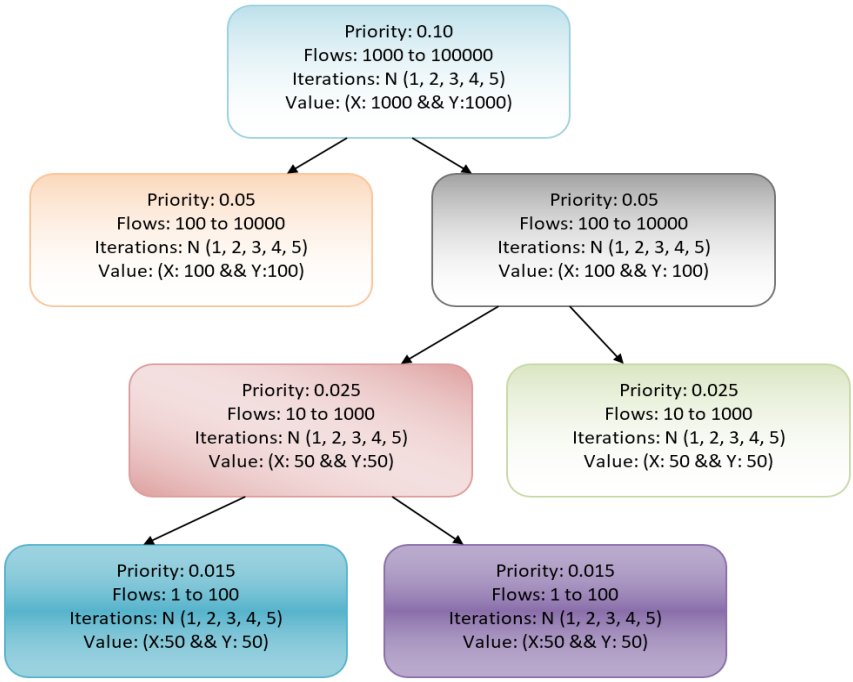

Figure 6. Decision Tree Generated based table 1 using TensorFlow

It is possible to detect the decision tree result using analytical representation using given dataset. In this case each reference parameters are analysed by using decision tree results. We test the results by using below step

a. Generate decision tree components in each control plane
b. Import all the generator rules and learning functions
c. prepare all the flows size value and train the dataset
d. Apply 70 % of generated flows and test the performance
e. Apply remaining 30 % to test dataset and prepare confusion matrix
f. Calculate the accuracy index for measuring performance

From the above procedure we calculated accuracy, precision, measure and recall factors. They are performance indicators for measuring the SDN performance by using conflict flows and load representations.

F(accuracy) = F(True_Positive) + F(True_Negative) / F(True_Positive + True_Negative + False_Positive + False_Negative)
F(Precision) = F(True_Positive) / F(True_Positive + False_Positive)
F(Measure) = 2 X (F(Accuracy) / F(Precision) / N
F(Recall) = F(True_Positive) / F(True_Positive + False_Negative)
Execution Time = Time (start) && Time (Finish)

From above representation we used TensorFlow simulator for testing the dataset. The below table is showing that the result of our proposed method.

From the above table 3 we test the dataset flow from 1 000 to 100 000 flow with the iteration of 1 000, 2 000, 5 000, 10 000 flows. We applied our proposed algorithm and achieved average accuracy of 97 %. Also we compare our proposed algorithm with various existing methods and iterations.

https://doi.org/10.56294/saludcyt2024797

Table 3. Accuracy result of our proposed method by using TensorFlow simulations

Iteration Flow	Dataset	Accuracy	Precision	Recall	Execution Time (ms)
1000	1000	96 %	15 %	88 %	1,12
	5000	97 %	17 %	86 %	1,23
	10000	95 %	18 %	86 %	1,54
	20000	96 %	17 %	84 %	2,03
	50000	96 %	15 %	85 %	2,25
	60000	97 %	16 %	86 %	2,87
	70000	96 %	17 %	87 %	3,45
	80000	97 %	17 %	86 %	4,05
	90000	96 %	16 %	87 %	5,34
	100000	98 %	15 %	85 %	6,23
2000	1000	97 %	16 %	86 %	1,27
	5000	96 %	17 %	87 %	1,56
	10000	97 %	17 %	86 %	2,23
	20000	96 %	16 %	87 %	2,56
	50000	98 %	15 %	85 %	3,25
	60000	97 %	16 %	86 %	4,23
	70000	96 %	17 %	87 %	5,23
	80000	97 %	17 %	86 %	5,45
	90000	96 %	16 %	87 %	6,02
	100000	98 %	15 %	85 %	7,02
5000	1000	95 %	18 %	86 %	1,46
	5000	96 %	17 %	84 %	2,34
	10000	96 %	15 %	85 %	2,56
	20000	97 %	16 %	86 %	3,02
	50000	96 %	17 %	87 %	3,59
	60000	95 %	18 %	86 %	4,25
	70000	96 %	17 %	84 %	5,34
	80000	97 %	17 %	86 %	6,27
	90000	96 %	16 %	87 %	7,45
	100000	98 %	15 %	85 %	8,28
10000	1000	96 %	17 %	87 %	2,24
	5000	97 %	17 %	86 %	3,45
	10000	96 %	16 %	87 %	4,45
	20000	98 %	15 %	85 %	5,34
	50000	97 %	17 %	86 %	6,46
	60000	96 %	16 %	87 %	7,27
	70000	97 %	17 %	86 %	8,45
	80000	97 %	17 %	86 %	9,05
	90000	96 %	16 %	87 %	10,34
	100000	98 %	15 %	85 %	11,23

From the above table 4 shows that comparison of proposed hybrid support vector machine with enhanced decision tree approach. We compared the accuracy factor with existing methods such as virtualized load balancer, CNN Classifier Sybase Pycode with various iterations and SDN Dataset. From that result our proposed system accuracy index is higher than other methods.

https://doi.org/10.56294/saludcyt2024797

Table 4. Comparison of Proposed method with existing methods

Iterations	Dataset	Virtualized Balancer	CNN Classifier	Sybase Pycode	Proposed Method
1000	1000	77 %	68 %	77 %	96 %
	50000	75 %	66 %	76 %	98 %
	100000	78 %	65 %	75 %	98 %
2000	1000	77 %	67 %	75 %	97 %
	50000	74 %	68 %	75 %	96 %
	100000	76 %	66 %	76 %	98 %
5000	1000	75 %	67 %	75 %	95 %
	50000	75 %	65 %	77 %	96 %
	100000	76 %	65 %	76 %	98 %
10000	1000	76 %	67 %	74 %	96 %
	50000	76 %	67 %	75 %	97 %
	100000	76 %	66 %	78 %	98 %

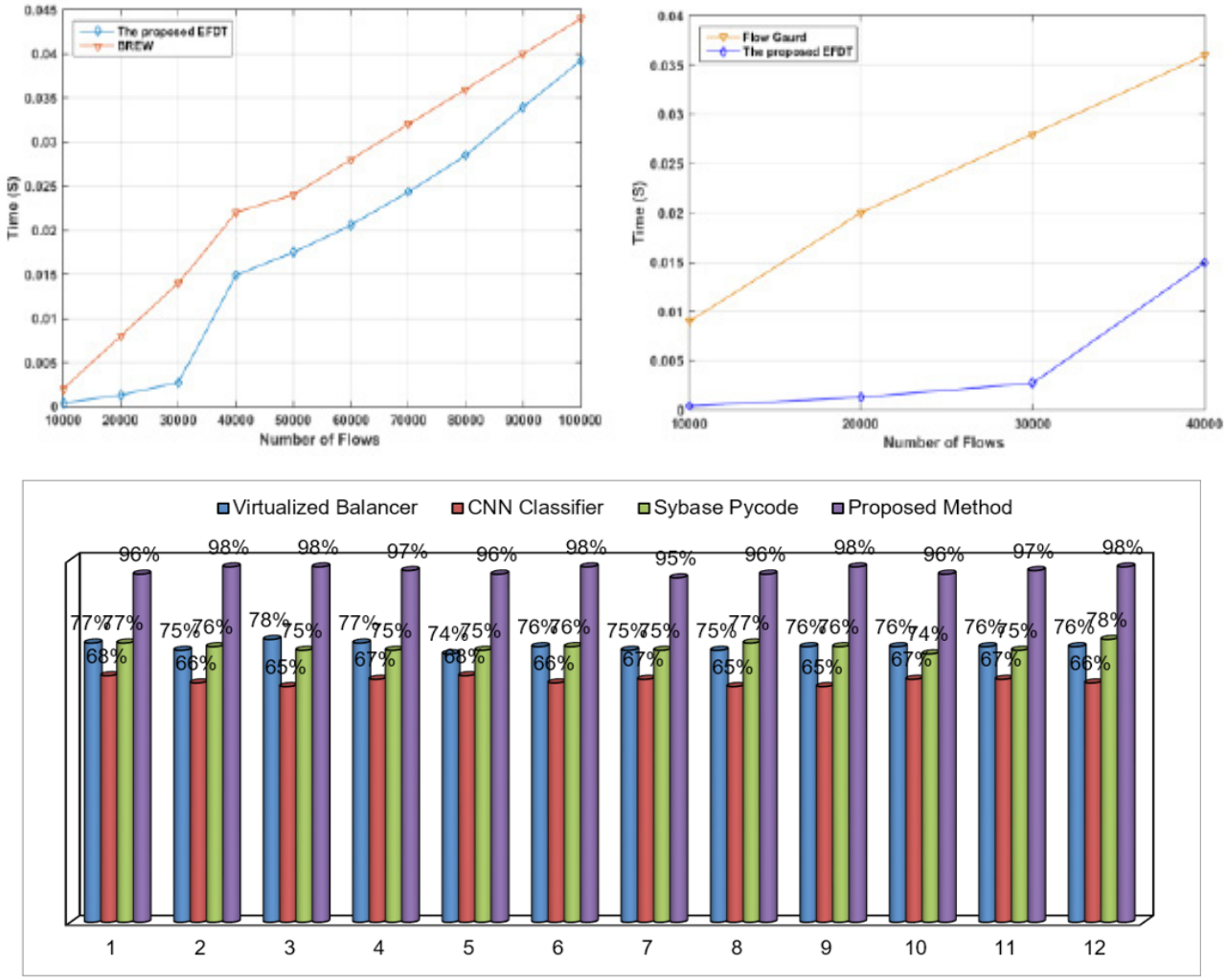

Figure 7. Comparison of proposed results with existing systems

CONCLUSION

This paper proposed Software Defined Network conflict flows accuracy prediction using deep learning. We used hybrid support vector machine with enhanced decision tree approach for calculating performance. In this case we tested 1 000 to 100 000 control flow dataset with multiple iterations. OpenFlow conflict rule are generated and deep belief network is obtained using TensorFlow. Fat free topology is prepared as decision tree results. From this we measured accuracy, precision, recall and execution time. We achieved 97 % accuracy

https://doi.org/10.56294/saludcyt2024797

result with various iterations flows and dataset. Also we compared results with existing method. In future we will apply this method to various real time smart applications.

REFERENCES

1. Manikandan. S, Dhanalakshmi. P, Priya. S, Mary Odilya Teena. A, "Intelligent and Deep Learning Collaborative method for E-Learning Educational Platform using TensorFlow", Turkish Journal of Computer and Mathematics Education, Vol.12 No.10 (2021), E-ISSN: 1309-4653, 2669-2676

2. M. A. A. Albadr, S. Tiun, M. Ayob, and F. T. AL-Dhief, ''Spoken language identification based on optimised genetic algorithm–extreme learning machine approach,'' Int. J. Speech Technol., vol. 22, no. 3, pp. 711–727, Sep. 2019.

3. S. Xu, X.-W. Wang, and M. Huang, ''Software-defined next-generation satellite networks: Architecture, challenges, and solutions,'' IEEE Access, vol. 6, pp. 4027–4041, 2018

4. A. A. Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, ''Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature,'' IEEE Access, vol. 6, pp. 14159–14178, 2018

5. B. A. A. Nunes, et al., "A survey of software-defined networking: Past, present, and future of programmable networks," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, 2014, doi: 10.1109/SURV.2014.012214.00180.

6. C. S. Khin, M. Z. Oo, and A. T. Kyaw, "Packet-in Messages Handling Scheme to Reduce Controller Bottlenecks in OpenFlow Networks," in 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), IEEE, 2020, pp. 502-505, doi: 10.1109/ECTICON49241.2020.9158127.

7. Hernández-Flórez N. Breaking stereotypes: "a philosophical reflection on women criminals from a gender perspective". AG Salud 2023;1:17-17.

8. Quiroz FJR, Oncoy AWE. Resiliencia y satisfacción con la vida en universitarios migrantes residentes en Lima. AG Salud 2023;1:09-09.

9. P. Monika, R. M. Negara, and D. D. Sanjoyo, "Performance analysis of software defined network using intent monitor and reroute method on ONOS controller," Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, pp. 2065-2073, 2020

10. H. T. Zaw and A. H. Maw, "Traffic management with elephant flow detection in software defined networks (SDN)," International Journal of Electrical & Computer Engineering (2088-8708), vol. 9, no. 4, p. 3203, 2019, doi: 10.11591/ijece.v9i4.pp3203-3211.

11. S. Manikandan and M. Chinnadurai, "Evaluation of Students' Performance in Educational Sciences and Prediction of Future Development using TensorFlow",International Journal of Engineering Education Vol. 36, No. 6, pp. 1783–1790, 2020, 0949-149X/91, TEMPUS Publications, Printed in Great Britain

12. Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in neural information processing systems. 2016

13. R. K. Arbettu, R. Khondoker, K. Bayarou, and F. Weber, "Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers," in 2016 17th International telecommunications network strategy and planning symposium (Networks), 2016: IEEE, pp. 37-44, doi: 10.1109/NETWKS.2016.7751150.

14. S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," in 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), IEEE, 2018, pp. 1-5, doi: 10.1109/ICCTAC.2018.8370397.

15. P. Raghav and A. Dua, "Enhancing flow security in ryu controller through set operations," in 2017 3rd IEEE International Conference on Computer and Communications (ICCC), IEEE, 2017, pp. 1265-1269, doi: 10.1109/

CompComm.2017.8322746.

16. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

17. Manikandan, S &Chinnadurai, M 2019, 'Intelligent and Deep Learning Approach OT Measure E-Learning Content in Online Distance Education', The Online Journal of Distance Education and e-Learning, vol.7, issue 3, July 2019, ISSN: 2147-6454.

18. M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A Novel SDN Intrusion Dataset," IEEE Access, vol. 8, pp. 165263-165284, 2020, doi: 10.1109/ACCESS.2020.3022633.

19. T. A. Assegie and P. S. Nair, "A review on software defined network security risks and challenges," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 17, no. 6, pp. 3168-3174, 2019, doi: 10.12928/telkomnika.v17i6.13119.

20. Caero L, Libertelli J. Relationship between Vigorexia, steroid use, and recreational bodybuilding practice and the effects of the closure of training centers due to the Covid-19 pandemic in young people in Argentina. AG Salud 2023;1:18-18.

21. Ogolodom MP, Ochong AD, Egop EB, Jeremiah CU, Madume AK, Nyenke CU, et al. Knowledge and perception of healthcare workers towards the adoption of artificial intelligence in healthcare service delivery in Nigeria. AG Salud 2023;1:16-16.

22. T. E. Ali, A. H. Morad, and M. A. Abdala, "Traffic management inside software-defined data centre networking," Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, pp. 2045-2054, 2020, doi: 10.11591/eei.v9i5.1928.

23. V. Danciu and C. N. Tran, "Side-Effects Causing Hidden Conflicts in Software-Defined Networks," SN Computer Science, vol. 1, no. 5, pp. 1-16, 2020, doi: 10.1007/s42979-020-00282-0.

24. S. Usman, I. Winarno, and A. Sudarsono, "Implementation of SDN-based IDS to protect Virtualization Server against HTTP DoS attacks," in 2020 International Electronics Symposium (IES), IEEE, pp. 195-198, 2020, doi: 10.1109/IES50839.2020.9231699.

FUNDING

The authors received no funding for the development of this research.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest

AUTHORSHIP CONTRIBUTIONS

Conceptualization: B. Ananth.
Data curation: B. Ananth.
Formal analysis: B. Ananth.
Research: B. Ananth.
Methodology: B. Ananth.
Supervision: B. Ananth.
Visualization: B. Ananth.
Writing - original draft: B. Ananth.
Writing - revision and editing: B. Ananth.