results indicated that the optimal false rate for the Bloom Filter approach was ranged between 0.04 to 0.06. Swami et al. [5] emulated the DDoS detection system using Entropy. The defined threshold during the simulation was one. Entropy could detect the attack efficiently on High-rate DDoS attacks. The authors concluded that the slow rate flooding mechanism was very harmful for the SDN environment, since entropy could not detect the attack that relatively behaved similarly to the legitimate network traffic. This problem is considered as the limitation of the statistics method. The other researcher used Machine Learning to overcome the limitation by giving the controller ability to learn and classify the packet's header. Sumadi et al. [6] proposed a Machine Learning approach by embedding classification modules on the SDN controller. The algorithms used during the experiment were SVM, MLP, GNB, KNN, RF, and Decision Tree. The authors also utilized a new scheme of DDoS datasets by extracting the information generated from the OFPMP_PORT_STATS request. The Linear and RBF kernels SVM algorithm showed the highest accuracy for predicting the DDoS packets pointed at 100%. Sangodoyin et al. [7] employed several algorithms (Quadratic Discriminant Analysis (QDA), GNB, KNN, and CART) to detect the HTTP, TCP, and UDP DDoS attacks which emulated using LOIC [9]. The results showed that CART algorithm could train the classification model and predict faster than the other algorithms. It also produced the highest accuracy at 98%. Alzahrani et al. [8] proposed Network Intrusion Detection System (NIDS) installed on SDN controller. The NIDS was developed using Machine Learning algorithms including the Decision Tree, Random Forest, and XGBoost for classifying NLS-KDD dataset. The modified XGBoost cloud predicts the DDoS with an accuracy at 95.95%.

The reactive-based solution implementation may burden the controller resources during the detection phase since the controller also executes the other management application, e.g., routing, proxy, and caching. One of the solutions for reducing the controller workload for detecting DDoS was implementing the stand-alone application that could monitor and classify the attack. The integration of SDN and Honeypot [10] could support the detection phase since the deployment of Honeypot sensors attracted the attack by opening several dominant IP protocol ports for acquiring or trapping the attack. Wang et al. [11] employed the hybrid architecture of SDN and Honeypot to detect the high-level of attack by migrating it into high-interaction Honeypot sensors. This mechanism provided high precision of data control on the network. The remaining papers [12]-[15] deployed the Honeypot on Software-Defined Internet of Things (SD-IoT) Network to identify DDoS and the other types of cyber-attack.

Based on the specified problems and previous works, all of the specified works did not implement Machine Learning as a basis of detection method on SD-Honeypot environment. No paper specifically established a flow modification message to block the attack as a mitigation scheme. Therefore, this paper was directed to create a stand-alone application in Modern Honeypot Network (MHN) Server for detecting DDoS using Machine Learning approaches including the SVM, MLP, GNB, KNN, and CART. The contributions are listed as follows:

- Integrating Honeypot on SDN architecture (SD-Honeypot Network)
- Creating detection modules based on Machine Learning algorithms
- The experiment dataset utilized the extracted data from Suricata sensor (ICMP flood)
- The mitigation process was conducted by installing a flow rule for blocking the attack using specific attributes on all of the available SDN switches

II. Material and Method

The experiment's topology illustrated in Fig. 1 was deployed in a real hardware scenario using several devices, namely the RYU [16] controller (C1), three Mikrotik [17] SDN switches (S1, S2, S3), and four Ubuntu hosts (H1, H2, H3, H4). In default, the communication between RYU controller and the available Mikrotik switches was regulated by the OpenFlow v1.1.0 [18] standard. The controller deployed a forwarding application to resolve network discovery between hosts using OFPT_PACKET_IN and OFPT_PACKET_OUT messages from the OpenFlow protocol. A simple switch program determined the resolution of the networking path. The attacker resided on H1 which sent ICMP flood for overwhelming H4 precisely transmitted at a range of 500 and 1000 packets per second (pps). The DDoS packet was constructed from a randomly generated IP and MAC source address. The attack was transmitted using TCPReplay [19] and crafted using Scapy [20]. Modern Honeypot Network was installed in H4 to attract cyber-attacks. Specifically, the deployed sensor on MHN server was Suricata [21].

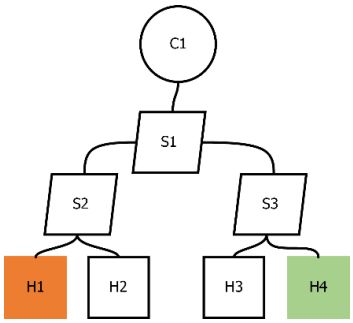

Fig. 1 Experiment's topology

The dataset used during the experiment was independent or self-made. The Machine Learning algorithms utilized this independent dataset to make classifiers for incoming packets at the MHN. The created dataset was a result of the MongoDB extraction process residing in the MHN. The packets number from normal flow was 30,000, while the DDoS was 60,000 packets. Then both of them were combined into a labeled dataset with the categories of "NORMAL" and "DDOS". Fig. 2, 3, and 4 describe each device's specific responsibility on topology in terms of the system workflow. The Mikrotik switches were used to filter the packet header's information based on the matching structure on available flow rules defined by the controller (Fig. 2). If match items could filter the incoming packets, the SDN switches performed the specified actions, e.g., forwarding, limiting, Quality of Service (QoS) operation, drop, or even crafting a reply packet.

If no flow could match the incoming packet, then the table miss event was generated. Subsequently, the SDN switches encapsulated the incoming packets into OFPT_PACKET_IN message then sent it to the controller for further processes, e.g., network discovery, link's detection, etc. The SDN switches also responded to the requestor command sent by the controller. Based on the research scenarios, the SDN switches installed in the specified flow originated from the OFPT_FLOW_MOD message.

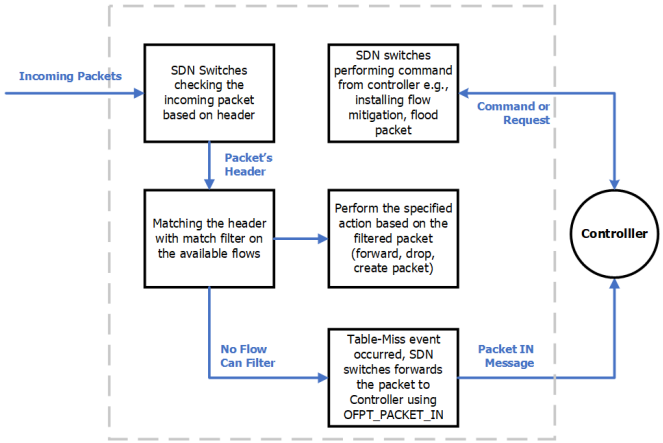

Fig. 2 SDN switches workflow

The controller performed the common assignment based on the installed application (Fig. 3). The experiment deployed only a simple switch application where the controller functioned as the main management node to resolve network discovery. Upon receiving a packet in message based on an unfiltered packet by SDN switches, the controller forwarded the encapsulated packet using OFPT_PACKET_OUT message. The controller should transmit the flow modification message when it is receiving the destined hosts to create a networking path between sender and receiver. In terms of the mitigation scheme for resolving the DDoS attack, the controller obtained the REST API command from the MHN server to forward the flow modification message for all available SDN switches to block the attack.

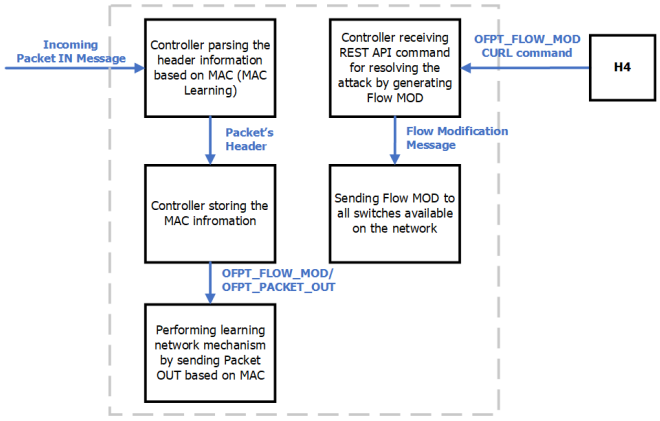

Fig. 3 Controller workflow

The entity that performed the classification process was MHN server (H4). The H4 was installed with MHN and supported the Suricata sensor. Unlike most previous works that utilized Machine Learning algorithms [6-8] to detect the DDoS reactively on the controller, the proposed scenarios deployed an open system separated from the controller for attracting the attack, namely the Honeypot. The attacker lured by opening all of the TCP/UDP ports for monitoring and detection. The integration of Honeypot on SDN architecture reduced the controller's load by allowing the MHN to create the detection module and send flow mitigation to block the attack.

Based on Fig. 4, The MHN server received the attack and stored the extracted packet's header information by Suricata sensor into MongoDB as a distributed database. The MHN server extracted the processed data by Suricata sensor every 30s duration. The attack was sent using six scenarios, where the distribution of data test at a range of 20%, 30%, and 40% transmitted at the rate of 500 and 1000 pps for every data test distribution. Upon receiving the attack, the MHN server application classified the data test using the generated model from the proposed algorithms (SVM, MLP, GNB, KNN, CART, and RF). When the application detected a DDoS packet, it performed the mitigation by selecting the Internet Protocol version 4 (IPv4) assigned protocol number. In case of the attacking scenarios, the detected protocol number was one as ICMP packets. The application employed REST API call by following the structure of the flow modification packet described in Table 1.

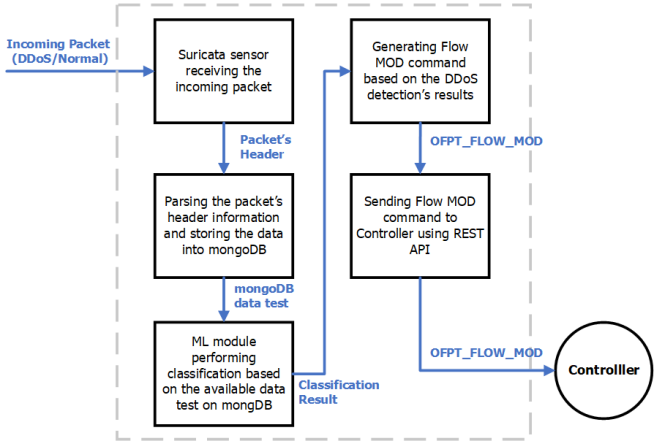

Fig. 4 MHN/Honeypot workflow

In order to evaluate the detection and mitigation modules, there were several variables extracted to measure the effectiveness: accuracy, precision, recall, and the duration for installing the mitigation flow from the first conducted attack. The last variable became one of the main contributions based on the evaluated results from the previous research.

TABLE I
FLOW MITIGATION STRUCTURE

Flow's Attribute	Value
Datapath	All Datapath Information (Switch's stats and Port stats Req)
Cookie	0
Table ID	0
Idle Timeout	10s
Priority	100
Flags	1
Matches	[IN_PORT, ETH_TYPE, IP_PROTO]
Actions	[ ] ~ Leave Blank for Blocking

III. Results and Discussion

The experiment was deployed in a real environment using three data distribution and two packet sending rates scenarios. The packet sending rates were constantly generated using the TCPReplay. The attacker performed an ICMP flooding attack that massively transmitted a randomly generated packet (IP and MAC source address) to imitate real DDoS attack activity. The attack affected the controller, and the associated hosts existed on the topology since the controller deployed a simple forwarding application and processed the dummy packets as unidentified packets. In response, the controller generated OFPT_PACKET_OUT message for resolving the end-to-end network learning. The results extracted during the implementation showed significant accuracy on Linear SVM compared to the other proposed classification methods depicted in Table 2.

TABLE II
CLASSIFICATION RESULTS (ACCURACY, PRECISION, RECALL)

Data Distribution		60:40:00	70:30:00	80:20:00
Packet Rate(pps)			**500**	
SVM	Accuracy %	93%	94%	94%
	Precision %	93%	94%	94%
	Recall %	93%	94%	94%
MLP	Accuracy %	67%	70%	70%
	Precision %	67%	70%	70%
	Recall %	67%	70%	70%
GNB	Accuracy %	70%	69%	70%
	Precision %	70%	69%	70%
	Recall %	70%	69%	70%
KNN	Accuracy %	69%	69%	70%
	Precision %	69%	69%	70%
	Recall %	69%	69%	70%
CART	Accuracy %	69%	69%	70%
	Precision %	69%	69%	70%
	Recall %	69%	69%	70%
Random Forest	Accuracy %	69%	69%	70%
	Precision %	69%	69%	70%
	Recall %	69%	69%	70%
Packet Rate(pps)			**1000**	
SVM	Accuracy %	93%	94%	94%
	Precision %	93%	94%	94%
	Recall %	93%	94%	94%
MLP	Accuracy %	69%	80%	72%
	Precision %	69%	80%	72%
	Recall %	69%	80%	72%
GNB	Accuracy %	70%	70%	70%
	Precision %	70%	70%	70%
	Recall %	70%	70%	70%
KNN	Accuracy %	69%	69%	70%
	Precision %	69%	69%	70%
	Recall %	69%	69%	70%
CART	Accuracy %	69%	69%	69%
	Precision %	69%	69%	69%
	Recall %	69%	69%	69%
Random Forest	Accuracy %	69%	69%	70%
	Precision %	69%	69%	70%
	Recall %	69%	69%	70%

The pointed accuracy, precision, and recall value are depicted at around 93-94 % for all data distribution scenarios. SVM could perform the classification efficiently even though the number of loss classification percentages displayed the second-highest value, more than 93% indicating that the MHN server was still overwhelmed by the incoming attack and processing the packet's class identification. In contrast, the less complex algorithm, namely CART (decision tree) and Random Forest only gained 69-70% accuracy.

TABLE III
CLASSIFICATION LOSS PERCENTAGES

Classifier Type	Data Distribution		
	60:40:00	70:30:00	80:20:00
Packet Rate (pps)		**500**	
SVM	95.00%	98.40%	98.00%
MLP	98.20%	97.90%	97.75%
GNB	33.82%	44.43%	46.40%
KNN	31.73%	34.85%	38.44%
CART	32.09%	33.26%	35.84%
Random Forest	33.72%	33.37%	35.22%
Packet Rate (pps)		**1000**	
SVM	94.00%	94.00%	96.25%
MLP	96.90%	97.60%	94.20%
GNB	38.00%	16.77%	8.80%
KNN	38.31%	42.59%	43.65%
CART	40.48%	40.23%	47.33%
Random Forest	35.35%	38.85%	46.99%

The classification results are directly related to the number of loss percentages during the identification process illustrated in Table 3. This variable existed because the MHN server was still overwhelmed by the DDoS attack and the classification process took more resources; therefore, the MHN server would drop incoming packets when there were data still in process. The algorithms that have the most classification loss are SVM and MLP.

TABLE IV
CPU USAGE DURING CLASSIFICATION PROCESS

Classifier Type	Data Distribution		
	60:40:00	70:30:00	80:20:00
Packet Rate (pps)		**500**	
SVM	0.17%	0.15%	0.19%
MLP	0.42%	0.26%	0.28%
GNB	0.02%	0.01%	0.02%
KNN	3.00%	2.01%	2.02%
CART	1.21%	1.08%	0.59%
Random Forest	1.34%	0.56%	0.50%
Packet Rate (pps)		**1000**	
SVM	0.19%	0.16%	0.13%
MLP	0.44%	0.31%	0.18%
GNB	0.02%	0.01%	0.01%
KNN	2.11%	2.00%	2.00%
CART	1.16%	1.17%	0.51%
Random Forest	1.17%	1.06%	1.07%

The classification loss percentage also affected the CPU utilization despite the average number of all algorithms not overcoming more than 3%. The highest CPU consumption

was deployed by KNN, where the number of loss classifications obtained was the second lowest. The CPU consumption also correlated with the complexity of the algorithms.

The value of accuracy, precision, and recall could not become the main point for determining the most efficient algorithm for resolving DDoS. The main notion of the system was directed to build an Intrusion Prevention System (IPS), which consisted of the identification and the mitigation module for blocking the incoming attack temporarily. The mitigation flow rule scheme consisted of several variables described in Table 1. The flow was intended to block the attack based on the physical port where the attack originated, the ethernet type for IPv4 environment, and IP protocol number for blocking a particular attack pattern (ICMP, TCP, or UDP flood). The defined action was left blank for defining the blocking approach. The flow rule was also composed of idle_timeout variable to remove the flow rule when no attacks were sent to the honeypot sensor (Suricata) within 10s duration.

The primary factor that can be used to define the most effective algorithm other than accuracy, precision, and recall, is the promptness for installing mitigation flow. The application embedded with the CART (decision tree) algorithm was the most responsive module to send the REST API command to deliver OFPT_FLOW_MOD was the application embedded with the CART (decision tree) algorithm. Even though CART only acquired 69-70% for accuracy, precision, and recall, it could perform the mitigation process immediately within 40ms on average compared to the SVM, which required 184,166.6ms to install the flow the accuracy pointed at 94%.

TABLE V
INSTALLATION DURATION OF MITIGATION FLOW IN SECOND (S)

Classifier Type	Data Distribution					
	60:40:00		70:30:00		80:20:00	
	Packet Rate (pps)					
	500	1000	500	1000	500	1000
SVM	126	112	137	140	305	285
MLP	272	190	256	285	239	178
GNB	2	2	2	2	2	2
KNN	1.001	0.837	0.92	1.081	1.138	1.324
CART	0.043	0.049	0.036	0.044	0.032	0.036
Random Forest	0.227	0.193	0.199	0.159	0.191	0.2

The consideration of practical deployment based on the mitigation flow installation also was correlated with the processing delay that occurred during the classification process. The resulting number indicated the time needed to mitigate after the first attack was identified. The most effective was the CART algorithm since it only required 40ms to detect and block the attack. In contrast, SVM and MLP, considered complex algorithms, needed more time, more than 3 minutes.

Compared to the other research [6]-[8], the prediction results did not obtain a better accuracy value. However, the previous works did not investigate the promptness of the mitigation scheme. Therefore, there was no conclusion from the previous works about which algorithm was the most effective method for mitigating the DDoS attack.

IV. CONCLUSION

Resolving DDoS attacks still became researchers' major concern where various approaches have been investigated and compared. The integration of Honeypot Sensor in SDN environment could provide a comprehensive system for developing DDoS-IPS. The results showed that SVM's most accurate algorithm to predict the DDoS attack. However, due to its complexity (O(n3)) [22] and processing delay, the capacity of the system embedded with the SVM model did not create an immediate response for blocking the attack. In contrast, the CART algorithm could defend the network promptly since the processing complexity was not too complicated (O(depth of the tree)). Therefore, in terms of all dependent variables (accuracy, precision, recall, and flow rule installation time), CART can be considered the most effective method for resolving DDoS in the SD-Honeypot environment. The authors will attempt to combine the statistical method for developing detection module in DDoS-IPS for further reference.

ACKNOWLEDGMENT

The authors are grateful to Informatics Laboratory at the University of Muhammadiyah Malang for supporting and providing hardware resources during the research implementation.

REFERENCES

[1] A. Praseed and P. S. Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 1, pp. 661-685, 2019.

[2] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey," *IEEE Access,* vol. 8, pp. 43920-43943, 2020.

[3] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine,* vol. 51, no. 7, pp. 36-43, 2013.

[4] V. Gupta, A. Kochar, S. Saharan, and R. Kulshrestha, "DNS Amplification Based DDoS Attacks in SDN Environment: Detection and Mitigation," in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, 2019, pp. 473-478.

[5] R. Swami, M. Dave, and V. Ranga, "Defending DDoS against Software Defined Networks using Entropy," in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 2019, pp. 1-5.

[6] F. D. S. Sumadi and C. S. K. Aditya, "Comparative Analysis of DDoS Detection Techniques Based on Machine Learning in OpenFlow Network," in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2020, pp. 152-157.

[7] A. O. Sangodoyin, M. O. Akinsolu, P. Pillai, and V. Grout, "Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning," *IEEE Access,* vol. 9, pp. 122495-122508, 2021.

[8] A. O. Alzahrani and M. J. F. Alenazi, "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks," *Future Internet,* vol. 13, no. 5, 2021.

[9] P. M. Ombase, N. P. Kulkarni, S. T. Bagade, and A. V. Mhaisgawali, "DoS attack mitigation using rule based and anomaly based techniques in software defined networking," in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, 2017, pp. 469-475.

[10] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, "A Comparative Analysis of Honeypots on Different Cloud Platforms," *Sensors,* vol. 21, no. 7, 2021.

[11] H. Wang and B. Wu, "SDN-based hybrid honeypot for attack capture," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 1602-1606.

[12] M. Du and K. Wang, "An SDN-Enabled Pseudo-Honeypot Strategy for Distributed Denial of Service Attacks in Industrial Internet of Things," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 1, pp. 648-657, 2020.

[13] H. Lin, "SDN-based In-network Honeypot: Preemptively Disrupt and Mislead Attacks in IoT Networks," *ArXiv,* vol. abs/1905.13254, 2019.

[14] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using MTD and SDN-based Honeypots to Defend DDoS Attacks in IoT," in *2019 Computing, Communications and IoT Applications (ComComAp)*, 2019, pp. 392-395.

[15] W. Tian, M. Du, X. Ji, G. Liu, Y. Dai, and Z. Han, "Honeypot Detection Strategy Against Advanced Persistent Threats in Industrial Internet of Things: A Prospect Theoretic Game," *IEEE Internet of Things Journal,* vol. 8, no. 24, pp. 17372-17381, 2021.

[16] 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)*, 2018, pp. 1-5..

[17] J. M. Ceron, C. Scholten, A. Pras, and J. Santanna, "MikroTik Devices Landscape, Realistic Honeypots, and Automated Attack Classification," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1-9.

[18] B. Heller et al., "*OpenFlow Switch Specification*", Version 1.3.0 (Wire Protocol 0x04)," pp. 1-105, 2012. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[19] Y. Li, R. Miao, M. Alizadeh, and M. Yu, "DETER: Deterministic TCP Replay for Performance Diagnosis," in *NSDI*, 2019.

[20] S. R. R, R. R, M. Moharir, and G. S, "SCAPY- A powerful interactive packet manipulation program," in *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, 2018, pp. 1-5.

[21] K. Nam and K. Kim, "A Study on SDN security enhancement using open source IDS/IPS Suricata," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1124-1126.

[22] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core Vector Machines: Fast SVM Training on Very Large Data Sets," *J. Mach. Learn. Res.,* vol. 6, pp. 363–392, 2005.