Demonstration of P4 Neural Network Switch

Francesco Paolucci[1], Lorenzo De Marinis[2], Piero Castoldi[2], Filippo Cugini[1]

1: CNIT, Pisa, Italy 2: Scuola Superiore Sant'Anna, Pisa, Italy
francesco.paolucci@cnit.it

Abstract: A programmable P4 node performing neural network acceleration is demonstrated. The node implements both traffic feature extraction and neural networking computations. Cyber security use case targeting DDoS attack detection is deployed.

1. Introduction

Bringing in-network Artificial Intelligence (AI) is hot topic in the development of cloud and edge nodes capable of performing high-speed forwarding. The idea of offloading specific functions to intelligent Software Defined Networking (SDN) devices is nowadays reaching a consensus, especially targeting packet-optical edge and 5G scenarios. In these cases, a number of functions related to traffic tagging, encryption, tunnelling, monitoring, and security can significantly benefit from programmable resources of a network element (switch pipeline, network interface card – NIC), thus reducing dedicated CPU resources as well as network latency/jitter due to virtualization and CPU/storage resource contention.

However, so far, AI acceleration has been delegated to specific hardware platforms, programmed independently with AI-specific languages and libraries (e.g., PyTorch and TensorFlow).

The P4 language, the de-facto SDN data plane programmability language, besides edge network telemetry [1], provides stateful processing [2] and computation capabilities. Such potentials can be exploited to define and implement a neural network inside a programmable SDN switch, regardless of the hardware platform used to deploy the switch. This means that the P4 SDN framework can be used as a unified programming language for both forwarding and AI elaborations. Moreover, the P4 capabilities to create and update selected features used for the neural network are numerous. For example, all the packet protocol layer header fields, but also specific metadata referred to traffic statistics or event occurrences may be utilized by the P4 switch as an input to a neural network running inside the same switch. Finally, leveraging on wire-speed programmable P4 ASICs has the potential to guarantee unprecedented performance to AI acceleration. So far, a pioneering preliminary proposal to apply P4 for AI has been considered in the Network Interface Cards (NIC) using simplified binary neural networks [3].

In this demo, we show an implementation of a neural network based on int8 elements running inside a P4 switch suitable for edge networking. The considered use case is cyber security, where the switch is able to tag malicious traffic referred to a number of different attack types, including distributed denial of service (DDoS) attacks.

2. Demo overview

Fig. 1 depicts the functional scheme of a P4 switch implementing a neural network. Besides the implementation of the required Layer 1-7 network headers (e.g., Ethernet, IP, TCP), the ingress switch pipeline has been selected as the candidate section where to implement the neural network. When the packet enters the switch, it is first parsed to extract all the required protocol header values. Then, all the packet features required by the neural network, enclosed within a P4 packet metadata structure, called *nn_meta*, are updated for the considered packet. The features may be stateless (e.g., packet header field) or stateful (e.g., cumulated statistics referred to a set of traffic packets previously processed). In the case of stateful features, P4 enables the utilization of registers and counters that may be allocated and updated upon packet processing. The feature updates are triggered by matching the desired traffic flows within a number *N* of dedicated flow tables. At the end of update, the *nn_meta* instance encloses all the packet features required by the neural network and the packet is subject to the neural network. More precisely, the *nn_meta* set is submitted to a macro action that implements the neural network.

The neural network is implemented as 3 layers of neurons. Each neuron is defined in the *nn_meta* as an int8 element. At the first layer, each related neuron output is computed as the weighted sum of all the inputs followed by an activation function, the rectified linear unit (ReLU) which simply sets all negative values to zero. At the other layers, each neuron output is subject to the same operations using the output values of the last layer, using different weights. Each neural network weight, used to determine whether an input stimulus is excitatory or inhibitory, is a parameter of the macro action and is submitted as control plane flow entry. The last layer produces the outputs of the neural network, that will be written again in the *nn_meta* instance. In the considered implementation use case, the output is boolean and identifies benign and attack packets. This way, a last flow table called *enforcer* matches the neural network output

value imposing, e.g., different forwarding to the packets, thus realizing online segregation between benign traffic and attack traffic, which will be diverted to a specific out-of-band output interface for further traffic analysis and for dropping operation.

The demo will show the detailed implementation of the node, including the key P4 code sections, along with a live lab session. The live session will include a traffic source generating a portion of the traffic traces obtained by the UNSW-NB15 Network Intrusion Detection data set, the data set utilized to perform the training of the P4 switch [4], profiling a number of intrusion attacks, including DDoS. The considered P4 software switch is the Behavioral Model version 2 software switch programmed via P4 version 16. The switch will be configured as a typical edge gateway element with four optical ethernet interfaces. With reference to Fig. 1, interface 1 is the uplink towards the xHaul network receiving packets to be processed, interface 2 is connected to the local edge node, interface 3 is connected to an isolated network, interface 4 is connected to the SDN control plane. The switch will perform traffic segregation for traffic received at interface 1, allowing online traffic inference and redirecting the malicious traffic to interface 3, while allowing forwarding for benign traffic towards the edge using interface 2. The switch will be equipped with internal metadata extraction features and postcard-based telemetry, capable to retrieve online monitoring of selected features, processing time, switch latency. Moreover, a dedicated PyTorch visual application will show the network's learning phase as a function of training iteration.

Fig. 2 shows the detailed structure of the *meta_nn* metadata structure. The considered features for the cyber security use case are selected from the full set of features of the UNSW-NB15 data set. In particular, stateless features, easily extracted by the P4 parsers, are the layer 4 source and destination ports (*src_port*, *dst_port*), the IP protocol (*IP_proto*), the packet length (*s_bytes*), the IP time-to-live (*s_TTL*), the TCP advertised window (*swin*). Moreover, two stateful features are considered, built resorting to internal P4 counters and registers: the number of packets having the same IP source (*ct_src_ltm*) or destination (*ct_dst_ltm*) address of the submitted packet within the last processed 100 packets. The *meta_nn* structure includes also the int8 neurons. Neurons are computed inside the *compute_nn* action resorting to the P4 arithmetic. The action takes the neural network weights as flow entry parameters. Each neuron is computed following the neural network structure, using the weighted summation of the former layer, then quantized using a non-linear threshold-based operator. More precisely, the neuron output y is computed as $y = \sigma(I_1W_{11} + \ldots + I_8W_{18} + \theta_1)$, where $\sigma(x)$ is the ReLU non-linear function, I_x are the neurons output of the previous layer (or the features values in the case of the first layer), W_{1x} are the weights and θ_1 is a bias parameter. Both weights and bias are constant values learned by the neural network during the training phase.

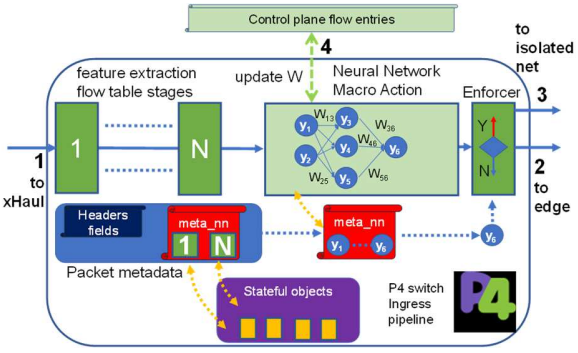

Fig. 1: Internal pipelines of the proposed P4 NN switch.

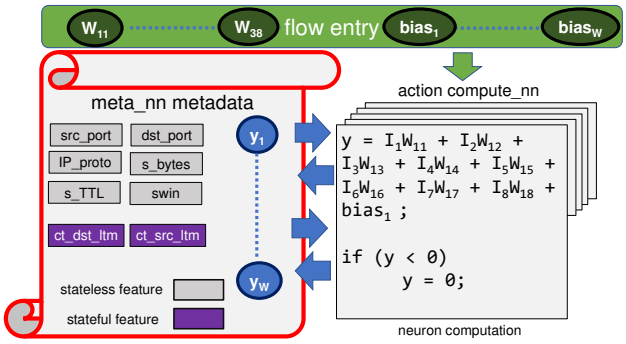

Fig.2. P4 metadata and neuron computation.

The neural network (NN) has been previously designed, trained and evaluated using the PyTorch library. Fig. 3 depicts a schematic of the developed PyTorch NN model, where different colors in the graph lines represent different weights. The model has 8 input neurons, 2 hidden layers with 8 neurons each and an output layer with 2 neurons. As for the nonlinearity, the ReLU function has been selected because of its versatility and ease of implementation [5][6]. The model has been trained using regular floats with an Adam optimizer over a cross entropy loss function. Then, a post-training dynamic quantization [7] is applied to convert its parameters to int8 type, i.e. converting weights, bias terms and the ReLU activations.

The choice of using int8 elements for quantization is the result of a tradeoff between ease of computation, memory consumption and information loss. Moreover, considering the UNSW-NB15 dataset use case, the use of int8 allows

the features from the dataset to be only lightly adapted for the neural computations, since they are already represented as integers. In our implementation, only the IP protocol feature has been really modified in order to be represented as an integer and not as a string, thus mapping *udp* as 0, *tcp* as 127 and all other as 255.

The main instance of the model has been trained with all the 8 aforementioned features selected from the UNSW-NB15 Network Intrusion Dataset. To further investigate the impact of such features, two additional instances of the model were trained with stateless only or stateful only features, reshaping the input layer with 6 and 2 neurons respectively. The accuracy results of all the trained models, and their quantized counterparts, are reported in Tab. 1.

The results highlight a minimum loss in accuracy on the test set in all cases (up to a few percent) when applying the dynamic quantization with int8, outperforming the 85% accuracy achieved by binary NN in [3]. Moreover, the network exhibits noticeable learning and generalization capabilities even when trained with stateless or stateful only features, with an accuracy drop < 3%. Such results are significant for the design policies related to P4 programmability over different software/hardware platforms. For example, results suggest that possible P4 switch implementations may be employed using a small set of stateful features, in all cases where there is the need to implement a reduced NN (e.g., in software switches). Alternatively, when stateful processing is limited due to memory constraints (e.g., using FPGA), it is possible to select a wider set of stateless features only, achieving a 96% accuracy with a very limited performance degradation.

3. Innovation

This demo will show the first AI-driven switch operated through SDN/P4 coordinated control and encompassing effective in-network AI-analysis. The goal is to show the capabilities to host dedicated neural networking resources processing network traffic directly at the networking devices, without involving selected GPU/FPGA resources inside the edge/cloud node.

4. Relevance

This demo is specifically designed for the OFC audience, mainly telco and cloud/edge operators and vendors, interested in the potential innovation capabilities driven by a novel extended SDN data plane programmability including coordination and direct enforcing of AI-based analysis.

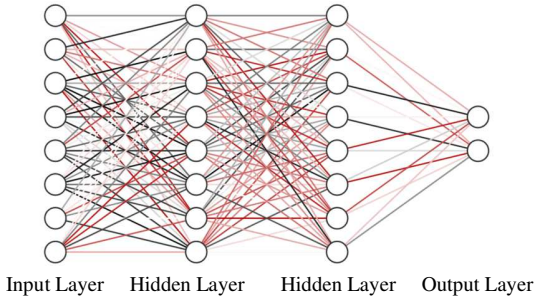

Fig. 3: Neural Network, line colors represent weights.

Used Features	Trained NN Accuracy	Quantized NN Accuracy
Stateless + Stateful (8)	99.4 %	96.9%
Stateless only (6)	98.8%	96.6%
Stateful only (2)	96.8%	96.8%

Tab.1: Accuracy on test set for the as trained and quantized NN.

Acknowledgment. This work has been supported by the BRAINE Project, funded by ECSEL Joint Undertaking under grant agreement No. 876967.

References

[1] I. Pelle et al. "Telemetry-Driven Optical 5G Serverless Architecture for Latency-Sensitive Edge Computing." OFC 2020.
[2] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security", Journal of Optical Communications and Networking (JOCN), 11(1), A84-A95, 2019.
[3] G. Siracusano, S. Galea, D. Sanvito, M. Malekzadeh, H. Haddadi, G. Antichi, R. Bifulco, "Running Neural Networks on the NIC". arXiv preprint arXiv:2009.02353. Sept. 2020.
[4] N. Moustafa, J. Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." in 2015 Military Communications and Information Systems conference (MilCIS), pp. 1-6. IEEE, 2015.
[5] A.A.M. Al-Saffar, H. Tao and M.A. Talab "Review of deep convolution neural network in image classification", in IEEE Proc. ICRAMET, pp. 26-31, Oct 2017.
[6] I. Hubara et al., "Quantized neural networks: Training neural networks with low precision weights and activations." Journal of Machine Learning Research, vol 18 n.1, pp. 6869-6898, 2017.
[7] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius. "Integer quantization for deep learning inference: Principles and empirical evaluation", arXiv preprint arXiv:2004.09602. 2020.