2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT) | 978-1-6654-9799-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/DS-RT55542.2022.9932068

Protocol Recognition in Virtual Avionics Network Based on Efficient and Lightweight Convolutional Neural Network

1st Mohamed Kerkech
Research & innovation
Capgemini Engineering
Blagnac, France
mohamed.kerkech@capgemini.com

2nd Van-Tuan Bui
Research & innovation
Capgemini Engineering
Blagnac, France
van-tuan.bui@capgemini.com

3rd Michel Africano
Research & innovation
Capgemini Engineering
Blagnac, France
michel.africano@capgemini.com

4th Lise Martin
Research & innovation
Capgemini Engineering
Blagnac, France
lise.martin@capgemini.com

5th Krishnan Srinivasarengan
Research & innovation
Capgemini Engineering
Blagnac, France
krishnan.srinivasarengan@capgemini.com

***Abstract*—Aerospace systems have complex internal interactions which allow a consistent behavior for the overall system. These are aided by communication protocols such as ARINC 629, AFDX, etc. These systems are too expensive and too critical to allow real experimentation, thus requiring extensive use of simulation. Thus an aircraft simulation test bench involves various simulation components with their own communication protocols, complicating its development process. One way to solve this issue consists of recognizing each communication protocol, decoding and encoding it in another protocol within a shared simulation environment. As part of a project to develop an interoperable simulator, we aim to build such a system that can recognize and decode avionics simulated communication protocols. In this work, we present AvioNet, a lightweight, computation-efficient neural network for virtual avionics network protocol recognition with accuracy and latency levels as required by aerospace systems. This method converts each packet into a common gray image, and then uses the depthwise separable convolution, pointwise group convolution and channel shuffle operations to automatically extract the appropriate spatial features. This reduces the computational complexity significantly while maintaining almost the same accuracy. This CNN-based classifier is verified on data that has non-avionic protocols mixed with avionic simulated protocols and is compared with the state-of-the-art methods. Experimental results show that the accuracy of the method exceeds 99.999% for avionics simulated dataset and outperforms other deep learning classifiers. Furthermore, the method provides low-latency guarantees that aerospace systems demand.**

***Index Terms*—interoperability, simulation, protocol recognition, avionics network, convolutional neural network, lightweight neural network**

ABBREVIATIONS

ARINC	Aeronautical Radio, Incorporated
AFDX	Avionics Full-Duplex Switched Ethernet
CNN	Convolutional Neural Network
IP	Internet Protocol
UDP	User Datagram Protocol
MAC	Media Access Control
LSTM	Long Short-Term Memory
VISTAS	Virtual Interoperable Simulation for Tests of Aircraft Systems
GPU	Graphics Processing Unit
PCAP	Packet Capture
MTU	Maximum Transmission Unit
FLOP	Floating-point Operation

I. INTRODUCTION

The complexity and criticality of engineering systems are increasing significantly. Due to the diversity of their components, their interactions and new requirements need to be considered, including performance, safety, etc. One of the major issues in systems engineering is the specification and validation of these complex systems as early as possible in the product development cycle. Thus, to reduce the associated costs, several manufacturers are increasingly using simulation tools, from the specification phase to validate concepts, models, laws, etc., before integrating them. All these simulators have common elements in the form of customer needs or via technical solutions. To develop a generic interoperable real-time simulator, that is, the ability to communicate seamlessly with different types of systems, the simulator should be equipped with a module that can recognize different protocols. With this context, we developed a protocol identification method for the case of protocols used in simulation of aerospace systems.

In recent years, with the new breakthroughs achieved by deep learning in image classification, natural language, object

recognition and so forth, deep learning methods are also gradually applied to the field of network traffic classification, more particularly to recognize application layer protocol of packets circulating on the network traffic. Their advantages are that it does not require feature engineering techniques and can automatically learn spatial features from raw data. Deep learning has gained prominence recently as an approach of representation learning for traffic classification as an end-to-end learning method. Some researchers have applied CNNs to automatically extract key features such as one-dimensional CNN (1D-CNN) and two-dimensional CNN (2D-CNN) to obtain better classification results than classic machine learning approaches.

Among the major constraints presented by aeronautical systems, are the needs for low latency ($= 10ms$) and high accuracy (an error rate of less than 10^{-5}, equivalent of 1 error out of 100,000 packets circulating on a network). That said, although recent work such as in [1] have shown that an accuracy exceeding 99% can be achieved in protocol recognition, this is not sufficient to meet industrial standards. In this paper, we propose a new neural network structure which can ensure higher classification accuracy while using fewer resources. In addition to the demonstration of accuracy, the latency which is a significant practical issue is also fully evaluated. The remainder of the paper is organized as follows: the next section describes related work. Section III provides necessary context for this work including a review of avionics simulated protocols and that of the approaches used to build efficient and lightweight CNN. Section IV describes the methodology that puts to use these new designs. Section V focuses on the experimental results and analysis. Section VI concludes the paper and proposes the future research direction.

II. Related Work

Traffic analysis through monitoring tools has been in use for a long time. For example, Wireshark is an open-source traffic analyzer that provides packet-level insight into the network. Packet-level network traffic allows one to develop add-ons that lead to various applications. The literature provides examples such as anomaly detection [2], network protocol recognition [1], intrusion detection [3], malware traffic classification [4], IoT security [5], etc. In the context of paper, we are interested in the works that enable the identification of protocols [1], which can be considered a form of network traffic classification. More precisely in network traffic that has different protocols with an Ethernet backend or particularly an AFDX backend in avionics network. Protocol identification methods vary based on their intended end use and the tools available.

Network traffic is divided into classifiable objects according to certain granularities as illustrated in Figure 1. The last object is byte, and multiple traffic bytes are combined to form a network packet. All packets that have the same 5-tuple (source IP, source port, destination IP, destination port, transport-level protocol) are defined as flow. A session includes both directions of flows where the source and destination IP/port are interchangeable [4].

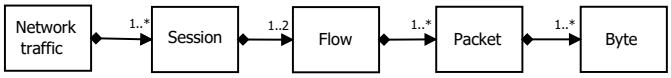


Fig. 1: Network traffic in UML (Unified Modeling Language)

Session and flow are used by most researchers as Flow-based approaches. They rely on statistical or time-series features. Statistical features are usually mean, variance, first, and third quartiles, flow size and maximum packet length. Time-series features can be flow duration, packet inter arrival time, etc. Much research work on flow features such as decision tree [6], support vector machine [7] and deep learning [1] [4] [8]. Flow-based methods do not investigate packet payload, so encrypted traffic classification is feasible. However, the method demands at least several milliseconds to collect packets to form a flow which is not practical in avionics network. Therefore, the basic object of our classification system is a packet. Packet-based approach takes every single packet as input and every byte in the packet as a feature. Deep learning is a promising method for this approach because it automatically learns spatial features from raw packets without feature engineering and expert experience.

In [9], the authors proposed a stacked auto encoder (SAE) to classify network protocol by using raw traffic data and achieved 99% on some protocols. In [4] a CNN is used to classify malicious traffic by taking traffic data as images. The final accuracy of the method is 99.41%. Data is divided into flows or sessions and only the first 784 (28x28) bytes of each flow or session are used. The method is protocol independent due to the use of the image classification approach and does not need to design features by hand for each protocol. The paper [8] proposed a CNN to first learn low-level spatial features of network traffic and then learns high-level temporal features using LSTMs. Experimental results showed that the average accuracy is 99.89%. Similarly, there are new deep neural networks, for example, 1D-CNN [10], PtrCNN [1], CNN and T-SNE [11], Convolutional Autoencoder [12], etc., whose overall accuracies are over 99%.

III. Context

A. Avionics Simulated Protocols

Aircraft systems have been using digital data buses for quite a while, starting with the various ARINC specifications in the 1970s (Mil-STD 1553, ARINC 429). To manage the increasing data load from newer aircrafts, several newer options including ARINC 629 were proposed. These, however, were complex and expensive. The latest iteration built ARINC 664 Part 7 (AFDX) based on Ethernet, a nearly ubiquitous technology, instead of building one from scratch. AFDX added some requirements on top of Ethernet to make it more robust and to ensure real-time operation that were critical to airplanes [13].

AFDX is patented and trademarked by international aircraft manufacturer Airbus. AFDX is based on commercial standards (IEEE802.3 Ethernet MAC addressing, IP, UDP) to provide deterministic timing and dual redundancy achieving the required

deterministic behavior for avionics applications [14]. The main differences between standard Ethernet and AFDX are shown in Table I.

TABLE I: Key Differences Between Ethernet and AFDX

Key Differences	Networks	
	Ethernet	***AFDX***
MAC Addresses	Source and Destination Addresses are unique, assigned by the IEEE	Source address contains a bit indicating network A or Network B. Destination address contains a Virtual Link Identifier (VLID,) it is not unique
IP Addresses	Source and Destination Addresses are unique, assigned by the IANA (Internet Assigned Numbers Authority)	IP source addresses use the private network and are unique within an AFDX network. IP Destination addresses can be the IP multicast
Switch	Send an incoming packet to one output port	Send an incoming packet to one or more output ports
	There may be multiple paths from source to destination	There is exactly one path from source to destination
	No dual redundancy	Dual redundancy
Timing	No timing defined by IEEE802.3	Periodical timing (every millisecond to every 128 milliseconds) based on kind of data
Protocols	TCP, UDP, DCCP, SCTP, etc	Primarily UDP

Physical test benches are an indispensable part of the aircraft development process, but they are complex platforms and very expensive. Virtual Testing is a promising solution to avoiding these difficulties. However, the aerospace industry brings specific challenges such as complex systems, hardware heterogeneity and multiple suppliers with different infrastructures. Therefore, an internationally recognized standard ED-247 (VISTAS) was created to provide a framework to support the interoperability of virtual and real avionics components. ED-247 defines rules to exchange data over virtualised aircraft components, in order to stay as close as possible to the exchanges in a real environment. Exchanging functional data can be Avionics data, AFDX messages, Ethernet frames, etc [15].

Another protocol which has the same functionalities as VISTAS is so-called CP-DSA (a communication protocol for distributed simulation applications). The official name of the protocol is hidden because of the proprietary issues. CP-DSA can be used to communicate between various simulation applications such as test benches, simulation models, etc. The data exchanged between these applications can be aircraft simulated data or any simulation environmental data. Both protocols VISTAS and CP-DSA exchange messages via the services of the UDP (Layer 4) with underlying IP (Layer 3). In our case, we focus on VISTAS and CP-DSA, which are normally used to simulate AFDX for realistic simulation of airplanes.

B. Convolutional Neural Networks

Neural networks are the elementary bricks of deep learning to model data with complex architectures combining different non-linear transformations. Theses neural networks are stacked together to form the deep neural networks. The most popular types of deep networks are CNNs. The CNN have enabled significant progress in image processing and analysis. They are based on convolution operators that improve a machine learning system by three important properties: sparse interactions, parameter sharing and equivariant representations. The early version of CNN, called LeNet by LeCun et al. [16] was a breakthrough in image pattern recognition to classify handwritten digits (MINIST datasets). The newer architectures from then on have typically had a standard structure: a stack of convolutional layers (optionally followed by normalization and max pooling), followed by one or more fully connected layers. However, these mainstream network structures tend to increase the depth (number of layers) and the width (number of neurons at each layer), which also makes the network more prone to overfitting and more parameters (increased use of computational resources).

A solution for this, as the paper [17] suggests, is to move from fully connected network to sparsely connected network architectures inside convolutional layers. This approach keeps the computational budget constant while increasing the depth and width of the network. The core concept of this sparsely connected architecture is the inception layer which is shown in Figure 2.

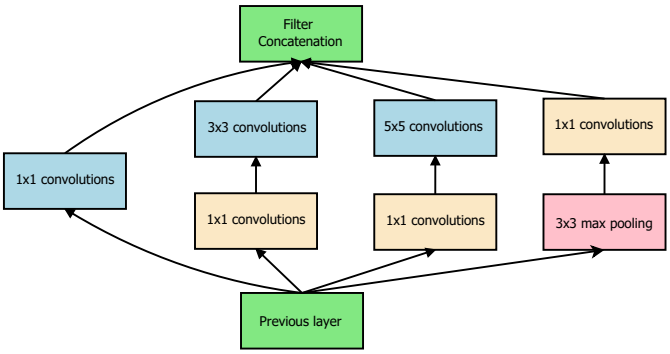


Fig. 2: Inception layer of GoogLeNet [17]

The inception layer uses multiple windows of convolutions to extract features from the input with different scales and then aggregated to enrich the information in each layer. Hence the next layer can abstract features from different scales simultaneously. The inception layer forms a concatenated layer using stacks of 1x1 convolutions, 1x1 followed by 3x3 convolutions, 1x1 followed by 5x5 convolutions and 3x3 max pooling layers followed by 1x1 convolutions. 1x1 convolutions used here have dual purpose: not only save computational power as dimension reduction but also make the network wider.

The hypothesis behind Inception layer is that cross-channel correlations (depth dimension) and spatial correlations (width and height dimensions) can be decoupled. An extreme version

of an Inception layer made a stronger hypothesis: these two correlations can be entirely decoupled. This extreme form is shown in Figure 3 which first uses a 1x1 convolution to map cross-channel correlations and then uses 3x3 convolutions to separately map the spatial correlations of every output channel.

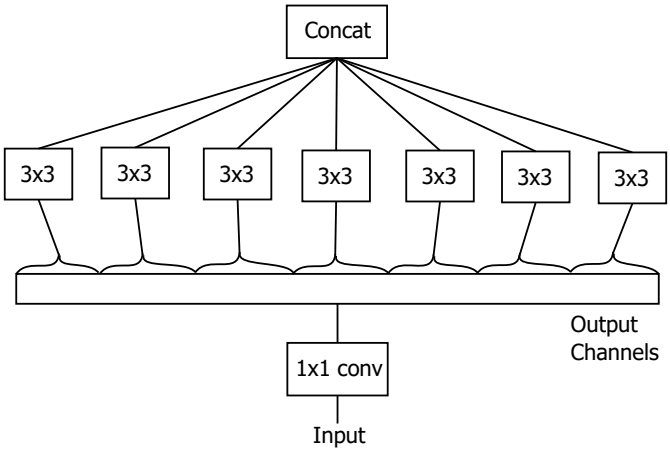
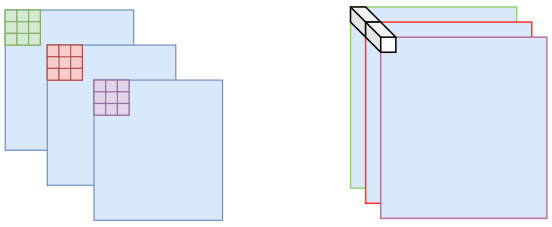


Fig. 3: An extreme version of Inception layer [18]

[18] remarked that this form is almost the same as the depthwise separable convolution, an operation that has been used as early as 2014 [19] to factorize traditional convolutions. Due to its effectiveness in computational cost with minimal loss in accuracy, depthwise separable convolution has been widely used in lighweight convolution networks [20] [21] [22].

Depthwise separable convolution replaces a standard convolution operation with a factorized version that splits convolution into two separate layers: depthwise convolutions in Figure 4(a) for filtering and pointwise convolutions in Figure 4(b) for combining. The depthwise convolution applies a single filter to each channel of an input. Pointwise convolution, a simple 1x1 convolution, combining the channels output by the depthwise convolutions in order to generate a new channel space. This factorization using $k \times k$ depthwise convolution has the effect of drastically reducing computation compared to traditional convolution by almost a factor of k^2 [20].



(a) Depthwise Convolution (b) Pointwise Convolution

Fig. 4: Depthwise separable convolution

However, pointwise convolutions also require considerable complexity. For example, in tiny networks, expensive pointwise convolutions lead to a limited number of channels to meet the complexity constraints, which often damages the accuracy of the model. In order to solve this problem, [22] proposed using a group convolution on a 1x1 convolution layer and a novel channel shuffle operation to reduce further the computational cost. The concept of group convolution, which was initially introduced in AlexNet [23] for accelerating the training process across two GPUs, has been widely applied in computational-efficient CNN architectures [22] [24] [25]. In theory, group convolution on g evenly distributed groups of input channels reduces the computational complexity by a factor of g. Channel shuffle operation as shown in Figure 5 used here to help information flow across group channels.

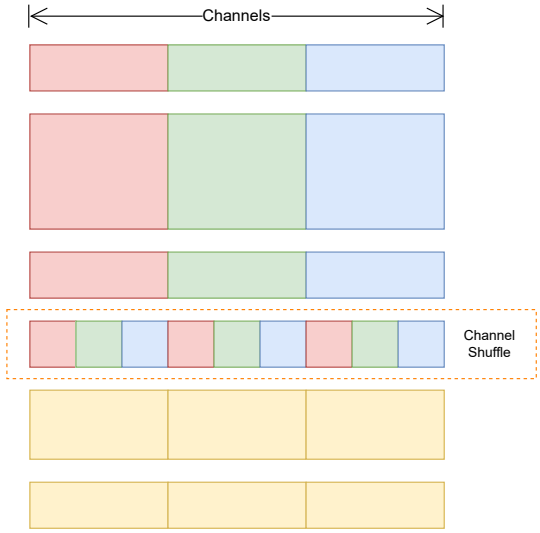


Fig. 5: Channel shuffle with group convolution [22]

Based on these observations on lower computational complexity of depthwise convolution and group convolution over the regular convolution while maintaining comparable accuracy, we will develop a convolutional neural network architecture based on these operations and we will next evaluate its performance on avionics simulated protocols.

IV. Methodology

A. Dataset

Avionics simulated data is acquired using the Wireshark software through our simulation platform. From acquisition, two "virtual" avionics protocols VISTAS and CP-DSA are obtained. To make a robust and reliable deep learning model, the avionics simulated data are mixed with other popular protocols which use UDP as transport layer. Two popular datasets USTC-TFC2016 [4] and CTU-13 [26] are chosen. In this paper, a total of 1.21 GB of raw traffic of five protocol types are selected as the experimental dataset, and they were saved in the PCAP files. Table II shows the details of our experimental dataset in this paper.

B. Data Preprocessing

In order to extract packets from the captured network traffic data and transform the packet into the standard input, the

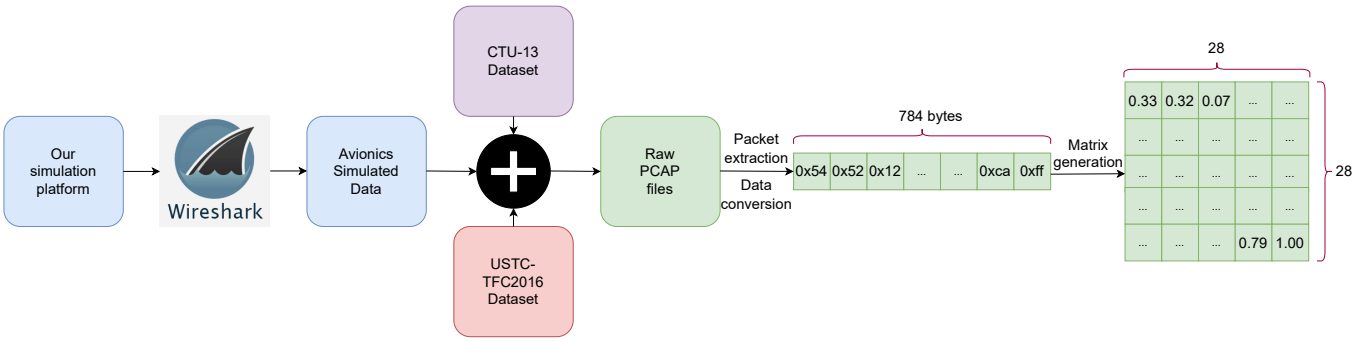


Fig. 6: Data acquisition and preprocessing procedures

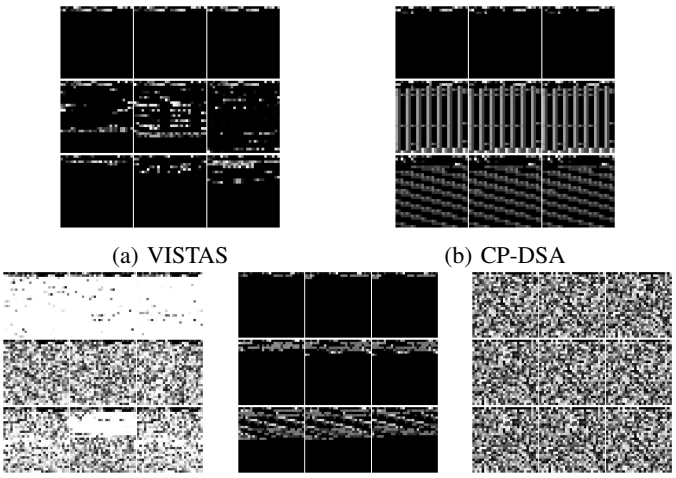
TABLE II: Dataset Description

Dataset	Protocol type	Number of packets
USTC-TFC2016	Facetime	6,000
CTU-13	DDOS	991,291
	DNS	30,016
Our simulation platform	CP-DSA	100,000
	VISTAS	1,213,064
Total		2,340,371

PCAP files need to be pre-processed through three steps: packet extraction, data conversion and matrix generation.

1) **Packet extraction:** select packets that use UDP as transport layer and may be the first fragment if there is IP fragmentation. IP fragmentation can happen when the packet size is bigger than the MTU, so it breaks the packet into smaller pieces (fragments).
2) **Data conversion:** We first fix MAC addresses in the data link layer and the IP addresses in the IP layer, which performs traffic anonymization. IPv4 addresses can be assigned 0.0.0.0 and MAC addresses are removed. Then, the deep neural networks require that the input data must be uniform, so only first 784 bytes of each packet are used. The packets with insufficient length will be padded with 0 at the end. This choice is motivated by three reasons. Firstly, the header information of protocol type is covered in the range. Secondly, the first data of UDP payload can reflect the characteristics of the application protocol. Thirdly, the LeNet model originally used grey image of size 28x28 as input to recognize hand-written digits. Moreover, experiments in some papers [1] [4] proved that 784 bytes are effective for CNN models.
3) **Matrix generation:** decimal value of each byte in a packet is normalized. Each value is divided by 255 so the range of values is [0,1]. Next, we convert 784 elements into a 28x28 two-dimensional matrix.

Figure 6 shows the whole process of data acquisition and preprocessing.



(a) VISTAS (b) CP-DSA

(c) Facetime (d) DNS (e) DDOS

Fig. 7: Data visualization

C. Visualization Analysis

Example images of packets of five protocol types are shown in Figure 7. The resulting images are generated by data preprocess procedure except for the normalization of every byte in a packet.

Nine selected images in every class are shown. The size of each grey image is 28x28. The values range from 0 to 255 (0x00 to 0xFF in hexadecimal) representing the density of black and white. Typically, 0 represents black and 255 represents white. From the visualized images, the texture of images in VISTAS, CP-DSA and DNS has different patterns and the patterns in the first rows of three protocols are quite similar. However, there is a great distinction between different other patterns of every class. Also, it is easily seen that the samples of the same class of Facetime and DDOS look similar.

D. AvioNet Architecture

The AvioNet structure exploits depthwise separable convolution as its building unit, which decomposes a standard convolution into a combination of a depthwise convolution and a pointwise convolution replaced by a group convolution and a channel shuffle operation to find a trade-off between

accuracy and latency. Figure 8 presents our block unit with depthwise convolution, 1x1 group convolution and channel shuffle operation followed by batch normalization [27] and ReLU layers.

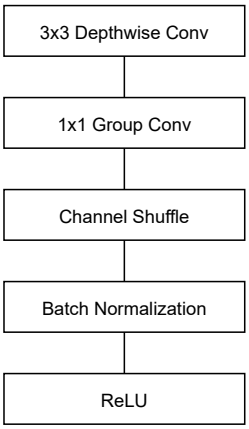


Fig. 8: AvioNet Block Unit

The channel shuffle operation used in AvioNet basic unit is exactly the same as that of ShuffleNet architecture [22]. A batch normalization layer is added before ReLU layer to accelerate training speed and improve the generalization capability of the network. Built on AvioNet units, we present the overall AvioNet architecture in Table III.

TABLE III: AvioNet Architecture

Type	Stride	Number of filters	Input size
3x3 Convolution	2	32	28 x 28 x 1
AvioNet Unit	1	32	13 x 13 x 32
AvioNet Unit	2	64	13 x 13 x 32
AvioNet Unit	1	64	6 x 6 x 64
Global Average Pooling			6 x 6 x 64
Fully Connected		5	64
Softmax			5

The first convolution layer is followed by a batch normalization and ReLU nonlinearity. Downsampling is handled with stride of 2 and valid convolution in the depthwise convolution within AvioNet unit as well as in the first layer. A final global average pooling reduces the spatial feature map to 1 before the fully connected layer. In total, the model includes one global average pooling layer and one fully connected layer, one convolution layer, and three AvioNet unit layers.

In AvioNet units, group number g controls the connection sparsity of pointwise convolutions. In this way, the trade-off between accuracy and latency becomes the key to our choice of group number g. To have a general and fair comparison, the complexity of a model is measured by the number of FLOPs, and the number of model parameters. Table IV shows the complexity of the AvioNet architecture with different group numbers g.

TABLE IV: Complexity of AvioNet

Model	Number of groups g	FLOPs	Parameters
AvioNet	1	1,080,350	9,701
AvioNet	2	686,110	6,117
AvioNet	4	488,990	4,325
AvioNet	8	390,430	3,429

V. Evaluation

A. Experiment Setup

In the experiment, 50% of the samples of five protocol types were selected for training, 10% for validation and the rest 40% were used for the testing. Table V describes the detail of our dataset after preprocessing the experimental dataset.

TABLE V: Evaluation Setup of Dataset

Protocol	Quantity		Train	Validation	Test
	Full	Selected			
Facetime	6,000	6,000	3,000	600	2,400
DDOS	991,291	200,000	100,000	20,000	80,000
DNS	30,016	29,998	14,999	2,999	12,000
CP-DSA	100,000	93,203	46,601	9,320	37,282
VISTAS	1,213,064	564,075	282,037	56,408	225,630
Total	2,340,371	893,276	446,637	89,327	357,312

There are 446,637 train packets, 89,327 validation packets and 357,312 test packets selected in the dataset. All AvioNet models were implemented using the TensorFlow framework [28] and trained on the Windows 10 64-bit OS. We use the Adam optimizer with a base learning rate of $1e-4$ and drop the learning rate to $1e-5$ after 20 epochs. We set batch size to be 300 and train for up to 100 epochs with early stopping if the validation loss has not improved in the last 10 epochs.

Finally, we convert Tensorflow trained models to Tensorflow Lite models [29] to evaluate the latency on the Raspberry Pi 4 with a 32-bit Buster operating system for protocol recognition in real-time which will be integrated into the simulation platform we are developing.

B. Classification Performance

The results on the test sets of our study are shown in Table VI. Four deep learning AvioNet architectures are compared to understand the behaviors of the group number g, and to find the most reliable architecture and score for our application.

From the results, we see that the model with group convolution ($g = 2$) performs better than the counterpart without pointwise group convolutions ($g = 1$). When group numbers become relatively large ($g = 8$), the error score increases, because this reduces the number of input channels for each convolutional filter in each group, which in turn diminishes the representation capability. With benefits of low computational

TABLE VI: Classification Performance of AvioNet Models

Model	Misclassified packets	Latency (ms)
AvioNet $g = 1$	2	7
AvioNet $g = 2$	**1**	**7**
AvioNet $g = 4$	2	7
AvioNet $g = 8$	3	7

cost and optimized Tensorflow Lite models of our architecture, they take only $7ms$ during inference.

C. Performance Comparison

To better analyze the performance of AvioNet on protocol recognition, we compare our method with LeNet, 2D-CNN, 1D-CNN and PtrCNN. These models have been evaluated in several papers [30], [4], [10], [1] and showed excellent performance for traffic classification. All four methods are based on CNN and use uniform size of 784 bytes as input. In order to better quantitatively analyze the recognition performance of avionics simulated protocols of the algorithm proposed in this paper, we select four metrics to evaluate the performances on the test set: Global Accuracy, Precision, Recall and F1-score. Global Accuracy is used to evaluate the overall performance of the classifier. Precision and recall are used to evaluate the performance of prediction of each protocol type. F1-score is used to balances both the concerns of precision and recall in one number.

TABLE VII: Performance Comparison of Different Methods

Method	FLOPs	Parameters	Misclassified packets	Latency (ms)
LeNet	845,042	61,281	33	8
2D-CNN	27,833,886	3,269,509	7	22
1D-CNN	39,712,734	5,825,413	76	27
PtrCNN	10,602,686	2,674,661	15	16
AvioNet $g = 2$	**686,110**	**6,117**	**1**	**7**

Table VII, VIII show the performance comparison and four metrics on the test dataset between our proposed method and the state-of-the-art methods. The global accuracy of AvioNet reaches 99.999%, that means only one error out of 357,312 packets in the test dataset while reducing the number of network parameters and the computational complexity compared to other deep neural networks. Overall, the AvioNet model gains the highest performance when it comes to the accuracy and latency, which makes more suitable to be deployed in a real-time protocol recognition application in the simulation of aircraft avionics systems.

VI. Conclusion And Future Work

Based on the critical constraints of accuracy and latency of aeronautical systems, a new efficient and lightweight convolutional neural network combining group convolution and depthwise separable convolution was proposed. Experimental evaluation shows that our method outperforms other deep neural networks in the domain of traffic classification both in terms of accuracy and model complexity. More specifically, AvioNet obtains a very good result of 99.999% accuracy in the classification of protocols which use UDP as transport layer and the inference time is around $7ms$ for each classification which has very promising feasibility for practical use in a simulated avionics network environment. In future work, we plan to study simulation interoperability in other domains such as automotive industry, railway industry, etc. Moreover, our method does not verify on encrypted protocols. To further improve system performance and robustness, encrypted traffic classification is worth exploring.

TABLE VIII: Performance Metrics of Different Methods

Method	Protocol type	Metrics (%)			
		Precision	Recall	F1-Score	Global Accuracy
LeNet	Facetime	100	100	100	99.990
	DDOS	100	99.995	99.997	
	DNS	99.841	100	99.920	
	CP-DSA	99.989	99.959	99.974	
	VISTAS	99.995	99.993	99.994	
2D-CNN	Facetime	100	100	100	99.998
	DDOS	100	100	100	
	DNS	100	100	100	
	CP-DSA	100	99.981	99.990	
	VISTAS	99.996	100	99.998	
1D-CNN	Facetime	100	100	100	99.978
	DDOS	100	99.998	99.999	
	DNS	100	100	100	
	CP-DSA	100	99.799	99.899	
	VISTAS	99.966	100	99.983	
PtrCNN	Facetime	99.958	100	99.979	99.995
	DDOS	100	99.996	99.998	
	DNS	100	100	100	
	CP-DSA	100	99.967	99.983	
	VISTAS	99.993	100	99.996	
AvioNet $g = 2$	Facetime	100	100	100	**99.999**
	DDOS	100	100	100	
	DNS	100	100	100	
	CP-DSA	**100**	**99.997**	**99.998**	
	VISTAS	**99.999**	**100**	**99.999**	

References

[1] W. Feng, Z. Hong, L. Wu, M. Fu, Y. Li, and P. Lin, "Network protocol recognition based on convolutional neural network," *China Communications*, vol. 17, no. 4, pp. 125–139, 2020.

[2] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30 387–30 399, 2020.

[3] K.-M. Yu, M.-F. Wu, and W.-T. Wong, "Protocol-based classification for intrusion detection," vol. 3, 03 2008.

[4] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717.

[5] B. Issac and N. Israr, *Case Studies in Intelligent Computing: Achievements and Trends*, 08 2014.

[6] W. Li and A. Moore, "A machine learning approach for efficient traffic classification," 11 2007, pp. 310 – 317.

[7] A. Santos Da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in sdn," in *2015 IEEE 14th International Symposium on Network Computing and Applications*, 2015, pp. 134–141.

[8] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[9] Z. Wang, "The applications of deep learning," 2015.

[10] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 43–48.

[11] J. Xue, Y. Chen, O. Li, and F. Li, "Classification and identification of unknown network protocols based on cnn and t-sne," *Journal of Physics: Conference Series*, vol. 1617, p. 012071, 08 2020.

[12] K.-C. Chiu, C.-C. Liu, and L.-D. Chou, "Capc: Packet-based network service classifier with convolutional autoencoder," *IEEE Access*, vol. 8, pp. 218 081–218 094, 2020.

[13] D. Koppel. (2017) Why are there so many avionics communications specifications? [Online]. Available: https://www.mil-1553.com/why-many-avionics-communications-specifications

[14] Afdx/arinc664p7 tutorial. [Online]. Available: https://www.aim-online.com/products-overview/tutorials/afdx-arinc664p7-tutorial/

[15] EUROCAE, "Technical standard for virtual interoperable simulation for tests of aircraft systems in virtual or hybrid bench," Tech. Rep., March 2020.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[18] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[19] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," *arXiv preprint arXiv:1403.1687*, 2014.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[22] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[23] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[24] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[25] K. Sun, M. Li, D. Liu, and J. Wang, "Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks," *arXiv preprint arXiv:1806.00178*, 2018.

[26] S. Garca, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 09 2014.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[29] Deploy machine learning models on mobile and edge devices. [Online]. Available: https://www.tensorflow.org/lite

[30] W. Jia, Y. Liu, Y. Liu, and J. Wang, "Detection mechanism against ddos attacks based on convolutional neural network in sinet," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, 2020, pp. 1144–1148.