Kölner Beiträge zur technischen Informatik
Cologne Contributions to Computer Engineering
Band 1/2024

Ergebnisse der Workshops 2023 und 2024 des Forschungsschwerpunkts Vernetzte intelligente Infrastrukturen und mobile Systeme (VIMS)

Technology
Arts Sciences
TH Köln

Impressum:

Forschungsschwerpunkt Vernetzte intelligente Infrastrukturen und mobile Systeme
Technische Hochschule Köln
Fakultät für Informations-, Medien- und Elektrotechnik
Betzdorfer Str. 2
D-50679 Köln
tobias.krawutschke@th-koeln.de
Stand: April 2024
ISSN: 2193-570X

Inhaltsverzeichnis

Editorial

Mit diesem Band im Jahr 2024 wird die Veröffentlichung in der Reihe *Kölner Beiträge zur technischen Informatik* nach dem Band 2/2022 fortgesetzt. Die Fakultät für Informations-, Medien- und Elektrotechnik am Institute of Computer and Communication Technology (ICCT) ermöglicht Master Studierenden nicht nur aus dem Bereich der technischen Informatik eine Möglichkeit Ihre Forschung zu veröffentlichen, die im Rahmen von Forschungs- und Entwicklungsprojekten an der TH Köln und/oder bei Projektpartnern entstand.

Ziel ist es, die Ergebnisse laufender Arbeiten aus den Forschungs- und Entwicklungsaktivitäten des Forschungsschwerpunkts nach außen zu kommunizieren und Informatiker:innen und Informationstechniker:innen außerhalb des Forschungsschwerpunkts z.B. aus dem Kölner Raum einzuladen, neue Ergebnisse aus Wissenschaft und technischer Anwendung im Rahmen von *Cologne Open Science* zu publizieren.

In zwei Workshops wurden die hier veröffentlichten Themen diskutiert. Am 16.6.2023 wurden *A novel approach for the optimal scheduling of a rail network (Tom Wenk, Markus Siebertz, Julian Kartte, Chunrong Yuan)* und *Comparison of Missing Data Imputation Techniques using Autoencoders (Rubaiya Kabir Pranti, Gernot Heisenberg, Sven Woehrle)* vorgestellt. Am 15.3.2024 wurden *FPGA-based acceleration of neural net simulation (Felix Sebastian Nitz)*, *Improving Traffic Flow Analysis: A Novel Approach to Sound-Based Traffic Density Classification (Leon Schex, Luca Schex)*, und *Open RAN Threat Analysis of the O-RAN SC Reference Implementation (Henrik Wittemeier, Arn Jonas Dieterich, Andreas Grebe, Thomas Karl)* vorgestellt.

Der Herausgeberkreis freut sich, diesen neuen Band der Reihe der Fachöffentlichkeit zur kritischen Prüfung und zur möglichen Mitwirkung vorlegen zu können.

Rainer Bartz
Andreas Behrend
Andreas Grebe
Tobias Krawutschke
Hans W. Nissen
Beate Rhein
René Wörzberger
Chunrong Yuan

A novel approach for the optimal scheduling of a rail network

Tom Wenk, Markus Siebertz, Julian Kartte
TH Köln - University of Applied Sciences
Cologne, Germany
{tom_frederik.wenk, markus.siebertz, julian_alexander.kartte}@smail.th-koeln.de

Chunrong Yuan
Autonomous Systems Lab
TH Köln, Cologne, Germany
chunrong.yuan@th-koeln.de

***Abstract*—Creating optimal timetables for passenger transport in a network of railway systems is a well-known but not fully solved problem. In this work, we propose an approach capable of reducing passenger delays and thus ensuring customer satisfaction for networks of arbitrary configurations with different parameters regarding passengers, trains, lines and stations. The approach has been compared to three other ones and evaluated based on the criteria of robustness and quality. The proposed scheduling algorithm performs best on both metrics. It has been shown through experimental study that for a wide variety of different scenarios, our approach can provide high quality timetables with minimal delays.**

***Index Terms*—railroad network, timetable optimization, scheduling strategy, InformatiCup, round-based model**

I. Introduction

In 2021, Deutsche Bahn introduced the Deutschlandtakt, a concept toward coordinated scheduling for passenger transport in Germany. It aims at making local public transport in cities and rural areas more attractive. The main idea is to interlock different schedules and make smart and coordinated planning of different travel requests so as to reduce the travel time and minimize the overall delay as well as waiting times in a network system.

In 2022, the German Society for Computer Science organized a coding competition called InformatiCup that tackled this problem of creating optimal timetables for a rail network [1]. The goal of the contest was to minimize the overall delay of all passengers and thus improve customer satisfaction with rail transportation. This so simply formulated task contains a lot of challenges: thousands of passengers want to be transported from start to destination every day, as quickly as possible. However, only a limited number of trains are available to transport these passengers. A huge rail network connects all cities with each other. Smooth operation of all rail vehicles must be ensured to make optimum use of resources.

Clearly, this kind of logistic optimization problem is faced not only by railroad companies, but also by many other logistics companies and organizations involving transportations of people or goods. The algorithmic core of this InformatiCup challenge is thus relevant in an economic context on a daily basis. And an optimal solution to it can be applied to many areas in our modern society, where all resources should be used efficiently and carefully.

According to the requirement of InformatiCup, the railroad network is represented as a round-based model. The four elements of it are station, line, train and passenger. A round is a fictive time unit and stands for the shortest interval for action scheduling within the network. Each of the four elements has certain features, as will be detailed in the following paragraphs.

At a **station**, passengers can get on and off a train as well as stay for any amount of time. A station can only hold a maximum number of trains simultaneously and hence has a train capacity. However, a station does not need to have a passenger capacity, as train stations in Germany are generally not overcrowded.

Every **line** connects bidirectionally two stations and it is guaranteed that all stations are directly or transitively connected by lines. Each line can accommodate a certain number of trains. The length of the lines influences the travel time of trains. The exact position of the trains on the track is irrelevant for the underlying task.

Every **train** can transport up to a maximum number of passengers with a certain velocity on lines between stations. Passengers cannot board (or leave) a train in the round of its departure (or arrival). If a train starts on a line, it cannot execute another action, until it reaches the next station. By multiplying the passed number of rounds on the current line with the speed of the train, one gets the (fictive) distance it has travelled. The train arrives at the next station in the round in which the distance travelled on the current line is greater than or equal to the length of the line. The train will then inevitably stop at this station and passengers get the possibility to board or leave the train. It can happen that a train arrives at the next station in the same round in which it has departed from the previous station. Each train is allocated to a given start station. In case no specific starting station is yet assigned to a train, which will be referred to as having an arbitrary starting station in the following, a suitable station will be determined for it.

Passengers are not modelled individually. They are grouped together and each group represents all the members travelling together from a start station to a destination. The size of a group should be equal or greater than one. Additionally, for every passenger group, a desired arrival time is defined in the input. This is later used to calculate the delay and therefore the quality of the algorithm. If passengers arrive at their destinations later than the desired arrival time, they are delayed. The goal is to generate a timetable leading to no or minimal passenger delay. For each passenger group, the delay p_{delay} is calculated as the difference between the actual and desired arrival time at destination. The overall delay d_{total} is the sum of the delays for all passengers, which is calculated by Eqn. 1 and 2.

$$d_{\text{total}} = \sum_{p \in P} p_{\text{delay}} \tag{1}$$

$$p_{\text{delay}} = \begin{cases} p_{\text{at}} - p_{\text{dt}}, & \text{if } p_{\text{at}} > p_{\text{dt}} \\ 0, & \text{else} \end{cases} \tag{2}$$

If passengers arrive earlier than desired, it is regarded to have no influence on their satisfaction, meaning the delay does not get lowered.

Every train and every group of passengers can execute one action per round. Passengers can either board or leave a train or they can wait at a station or travel on a train. Trains can either wait at a station, wait on a line or execute one of the following moves to drive along the railroad network:

- Leave a station and start on a line
- Leave a station, start on a line, and arrive at the next station
- Continue driving on a line
- Continue driving on a line and arrive at the next station.

Trains are only allowed to leave their current station if they have no passengers that want to board or leave the train.

II. Related Work

The train scheduling problem is widely considered as being NP-hard, in some specific problem formulations NP-complete, and very complex to solve. Although the problem was considerably discussed and different methods have been developed for tasks with specific settings [2] [3] [4] [5] [6] [7], none of the available solutions is directly transferable to the problem defined in Section I.

Among the participants of InformatiCup 2022, several teams have made their submissions publicly available. This makes it possible to compare different approaches. In the following, the methods developed by three other teams will be summarized.

A. Team ZZE (*Zügig zum Erfolg*)

The scheduling problem is formulated by ZZE as a variation of the Knapsack problem [8], with the size of passenger groups defined as weight and their desired arrival time as value. While the original problem is intended to maximize a value, ZZE tries to minimize the arrival time of passengers [9].

First, passengers are further grouped into blocks such that each block contains passengers with the same starting station and destination. Each block is then sorted by the desired arrival time in an ascending order. Trains are then assigned to passengers based on their positions and distances to the locations of passengers. Trains that already have a start station are prioritized for a calculation. If there is no train in a station with waiting passengers, a train must be called there to pick up the passengers.

Another criterion for train assignment is to ensure that at least one train whose passenger capacity is equal or greater than the maximal block size is available at a station on the route network. If this is not the case, a train with a sufficient capacity has to be placed at the location of the passenger as soon as possible.

For each passenger and each suitable train, the Dijkstra algorithm [10] is used to calculate the route from the starting station to the destination. Additionally, if a train needs to be called, the time at which that train could start at the starting station of the passenger is determined. Both measures are then used to determine which train would arrive at the destination at the earliest time. Once determined, that train is assigned to the selected group.

If a route or a station is occupied for a certain time, the delay is determined. If a station is occupied for an indefinite period of time, the station is cleared by driving the train out. If a route of a journey can be travelled without problems, this journey is saved permanently and can no longer be changed afterwards. All subsequent assignments must be based on this fixed assignment.

B. Team OBO (*Off By One*)

The team OBO built their solution under the premise that the train scheduling problem is NP-Hard and that there does not exist an optimal solution for the problem [11]. For the purpose of finding an approximation algorithm that can provide a sufficiently good solution, three methods have been taken into consideration:

- Genetic search [12], for reproducing and improving the ”fittest” solutions until an optimal solution is found
- Simulated annealing [12], by making random changes to an existing solution and then evaluating its impact based on the quality of the solution
- Tabu-enhanced genetic search [13], by using a tabu-list during search

After evaluation, the Tabu-enhanced genetic search has been adopted by the team OBO, where movements that lead to bad results are stored in a tabu list and omitted from the following iterations. If such movements are found at an early stage, it will help the system arrive faster at a good solution. It can also prevent cyclical behaviour so that the system will not get stuck in suboptimal regions.

C. Team HS (*HeiSpeed*)

Similar to [14], the HS solution [15] is based on the use of an algorithm ensemble, which consists of three algorithms: the Divide-and-Conquer algorithm [16], the Basic Greedy algorithm [10] [12] and the Branch-and-Bound algorithm [17]. The results of these methods are compared and the best solution is chosen among them.

The Divide-and-Conquer method will break down a larger problem into smaller subproblems which can then be solved independently. The advantage of this approach is that it limits

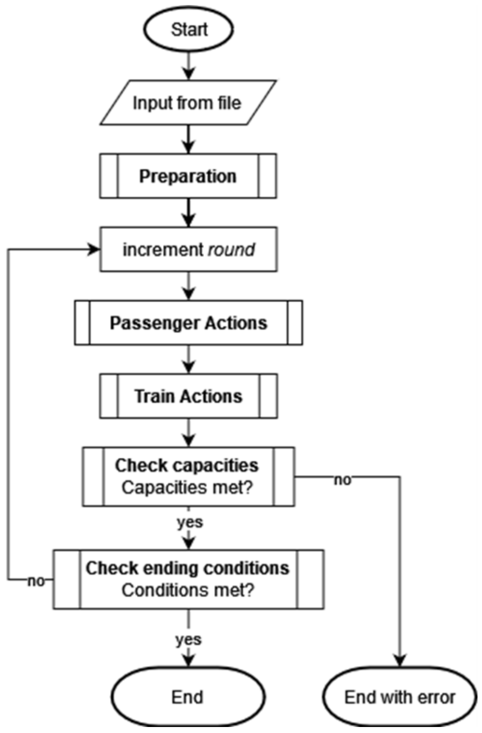


Fig. 1. Overview of the scheduling process.

TABLE I
ENTITIES AND THEIR ATTRIBUTES

Variable	Description
Train $t \in T$	Train entity
t_{cap}	Capacity of a train regarding passengers
t_{pas}	All passenger groups on a train
t_s	Speed of train
t_l	Location of train (line or station id)
t_d	Destination of train
t_{route}	Route of train
t_{blocked}	Trains performing certain actions
Line $l \in L$	Line entity
l_{cap}	Line capacity regarding trains
l_{trains}	All trains on a line
l_l	Length of line
$l(s_0, s_1)$	Line that connects stations s_0 and s_1
Station $s \in S$	Station entity
s_{cap}	Station capacity regarding trains
s_{trains}	All trains in a station
s_{pas}	All passengers at a station
Group of passengers $p \in P$	Passenger group entity
p_l	Location of passenger group
p_d	Destination of passenger group
p_s	Size of passenger group
p_{at}	Actual arrival time at destination
p_{dt}	Desired arrival time at destination
p_{delay}	Delay of passenger group

the risk of overflowing the routes and deadlocks. Computationally intensive operations such as searching for the shortest path within independent blocks become faster due to a reduced problem size. In order to use the Divide-and-Conquer algorithm, shortest paths between stations have to be precalculated so that a balanced partition of the railway network can be achieved. Trains are then assigned on the basis of balanced capacities among stations and routes.

The Greedy approach uses the Dijkstra algorithm to find the shortest path from start to destination for a single group of passengers. This means that it follows a simple resource reservation strategy and cannot reliably prevent deadlocks, as it uses a global iterative approach.

In the case of Branch and Bound, a brute force search is not applied to all possible solutions. By using an upper bound to determine if a solution is promising enough, and a lower bound that approximates the cost, this leads to the reduction of computational costs at an early stage.

III. SYSTEM OVERVIEW

The approach proposed in this work aims at the construction of optimal timetables for arbitrarily configured networks. It consists mainly of processing rules and scheduling strategies designed for a preparation phase and an iterative round-based phase, as is shown in Fig. 1.

Two kinds of tasks are involved in the preparation phase. One is the construction of a connected bidirectional graph for the representation of a rail network having an arbitrary configuration. The other contains the initializations of the four elements of a rail network together with their attributes using a standardized format, where a number of entities relating to trains, passengers, lines and stations are set up properly. All the entities and attributes are listed in Table 1.

Once the initializations are performed, a structural model is built which is capable of converting task scenarios with arbitrarily configured rail networks into a proper representation. The model consists of a compact data structure with features related to the underlying scheduling tasks and a graph representation of the network. With the features of these entities initialized properly, they can then interact with each other in the iterative scheduling phase: trains can pick up passengers and will then iteratively transport the passenger with the highest priority to its destination.

In order to achieve an optimal solution for a task scenario, it is important to set passenger priority and to assign trains to a properly selected start station. Both are determined during the initialization process as well. Details on the preparation and initializations regarding these two aspects will be discussed in Section IV. In Section V, strategies dealing with the iterative scheduling process will be presented.

IV. GRAPH CONSTRUCTION AND INITIALIZATIONS

Before the round-based part of the algorithms begins, a connected bidirectional graph G is constructed that represents the relations between stations and lines. The graph is later used to find the shortest path between two stations. It consists of a set of nodes representing the stations and a set of edges representing the lines. The cost of each edge is equivalent to its length.

Passenger groups are then allocated to their individual starting stations in the graph. Based on the desired arrival time, passengers are sorted in an ascending order, leading to an ordered set of passengers shown in Eqn. 3:

$$O_{\text{pas}} = \{p_0, p_1, \dots, p_n \mid p_{i_{dt}} \leq p_{(i+1)_{dt}},\ 0 \leq i \leq n-1\} \quad (3)$$

This process also functions as a prioritization of passengers. The earlier the desired arrival time of a passenger group, the higher its priority and the sooner it will be handled in the round-based phase.

Trains are allocated to their individual starting stations in the graph as well. An ordered set of trains is constructed, by sorting the speed of each train in a descending order.

$$O_{\text{trains}} = \{t_0, t_1, \dots, t_n | t_{i_s} \geq t_{i+1_s},\ 0 \leq i \leq n-1\} \quad (4)$$

For trains with an arbitrary starting station, they have to be assigned to a station $s^\alpha \in S$ that fulfills these conditions:

$$|s^\alpha|_{\text{pas}} > 0 \quad (5)$$

$$(s^\alpha)_{\text{cap}} - |s^\alpha|_{\text{trains}} > 0 \quad (6)$$

$$\nexists s \in S - s^\alpha : s_{rp} \leq (s^\alpha)_{rp} \quad (7)$$

$$s_{rp} = |s|_{\text{pas}} - \sum_{t \in s_{\text{trains}}} t_{\text{cap}} \quad (8)$$

As determined by the conditions shown in Eqn. 5 to 8, such a train will get allocated to a station s^α that:

- has passengers, as determined by Eqn. 5;
- can accommodate the additional train (i.e., station capacity is bigger than the number of trains at station s^α), which is decided by Eqn. 6;
- has the highest need for additional trains, i.e., has the biggest number of passengers who are waiting to be transported, which is determined by Eqn. 7 and 8.

The the number of waiting passengers at a station s is estimated as the difference between the number of passengers at station s and the number of passengers that could be held by existing trains at s.

If no station exists that meets all these conditions, the train can be allocated to a station s with no passenger. But still s must satisfy the conditions specified in Eqn. 6 to 8. Fig. 2 shows the above initializations performed in the preparation phase. The step of assigning a proper starting station for a train is shown on the bottom right part of Fig. 2.

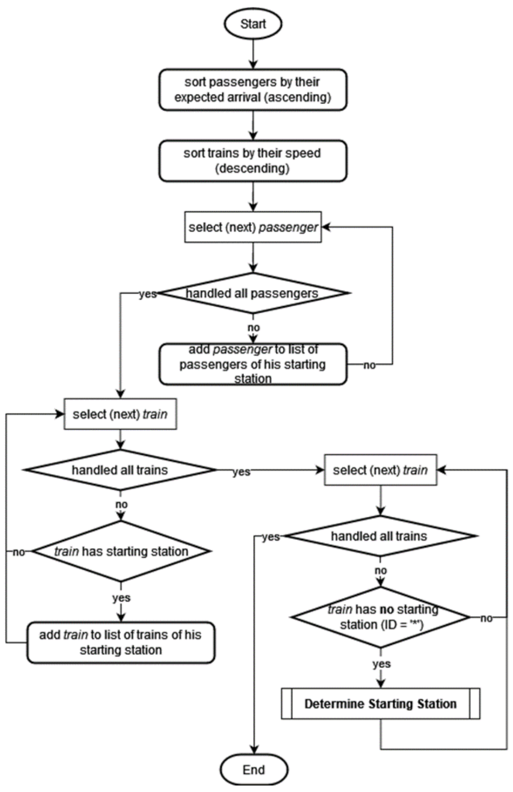

Start
sort passengers by their expected arrival (ascending)
sort trains by their speed (descending)
select (next) passenger
handled all passengers
yes
no
add passenger to list of passengers of his starting station
no
select (next) train
handled all trains
yes
no
train has starting station
no
yes
add train to list of trains of his starting station
select (next) train
handled all trains
yes
no
train has no starting station (ID = "*")
no
yes
Determine Starting Station
End

Fig. 2. Initializations during the preparation phase.

V. Iterative round-based scheduling

Our round-based scheduling approach deals with the determination of actions and routes for passengers based on the graph model and entities initialized in Section IV. In each round, passenger and train actions have to be decided properly. These actions have to be executed in a sequence such that in the end, all passengers will arrive at their desired destinations and the delay over all passengers is minimized.

After each round, it is checked whether all passengers have arrived at their destination station, determined by Eqn. 9:

$$\forall p \in P : p_l = p_d \quad (9)$$

Once this condition is met, the algorithm will end successfully. If this is not the case, the following conditions shown in Eqn. 10 to 12 will be checked:

$$\forall s \in S : s_{\text{cap}} \geq |s_{\text{trains}}| \quad (10)$$

$$\forall t \in T : t_{\text{cap}} \geq |t_{\text{passengers}}| \quad (11)$$

$$\forall l \in L : l_{\text{cap}} \geq |l_{\text{trains}}| \quad (12)$$

A. Passenger actions

In every round, the algorithm iterates through all passengers. Each passenger $p \in P$ executes one action depending on its current location p_l. Passengers on trains and passengers waiting at stations will perform different actions (see Fig. 3).

For a passenger p on a train t, possible actions are simply staying on it or getting off if the current location of the train is the same as the desired destination, i.e. $t_l = p_d$.

If a passenger p with $p_l \neq p_d$ is waiting at a station s, every train in station s will be checked to determine whether it satisfies the following condition:

$$p_s \leq t_{\text{cap}} - \sum_{p_k \in t_{\text{pas}}} p_k \quad (13)$$

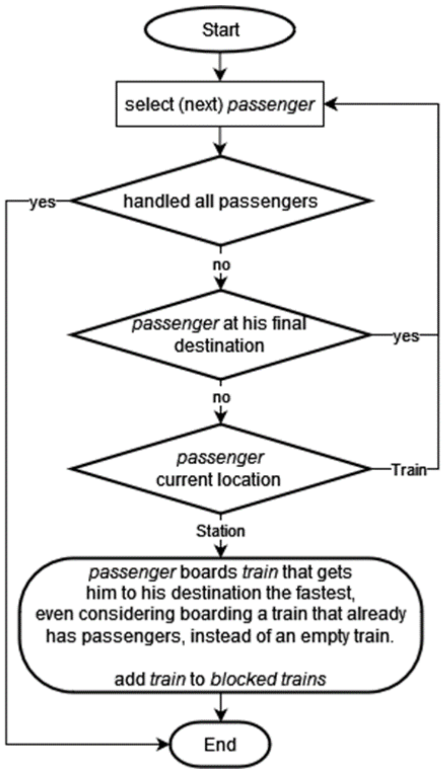

Start
select (next) passenger
handled all passengers
yes
no
passenger at his final destination
yes
no
passenger current location
Train
Station
passenger boards train that gets him to his destination the fastest, even considering boarding a train that already has passengers, instead of an empty train.
add train to blocked trains
End

Fig. 3. Passenger actions.

The goal is to find candidate trains that are located at station s and whose remaining capacity is not smaller than the group size of p so that it has enough capacity left to accommodate this passenger group. If no train meets these conditions, then p must wait. Otherwise, the task becomes the selection of a train from all those candidate trains so that passenger p can get on it.

For each train among these candidates, we estimate a possible arrival time p_{est} for passenger p. The train with the least delay for all onboard passenger groups will be chosen. In case the candidate train t has yet no passenger onboard, then p would be the only passenger which also has the highest priority on this train. The estimated arrival time p_{est} can be calculated as the travel time from the current station to the destination of p on the shortest route. And the delay of selecting this route is $p_{\text{est}} - p_{\text{dt}}$, as the decision has no influence on other passengers and p_{est} is the earliest possible time it would take train t to drive from p_l to p_d.

If one or more passengers are already on the train t, and t is currently scheduling along a route t_{route} toward the destination of an onboard passenger group with the highest priority, then there are two possibilities. If the current route t_{route} would later pass p_d, then p_{est} can be determined as the number of rounds it takes the train t to drive from p_l to p_d along t_{route}. If p_d is not a station on the current route of the train t, a balance must be struck between conflicting consequences. If we let passenger p board the train t and satisfy its need first so that p could travel to the desired destination p_d as quickly as possible. However, all other passengers who are already on the train will be delayed by the boarding of p, since the train must stop at an extra station p_d, leading to an increased delay for all passengers on the train.

In order to be able to weigh up among different options, the delay for each passenger p_i on train t is estimated and all delays are summed up to d_t, as is shown in Eqn. 14 and 15.

$$d_t = \sum_{i=0}^{n} (p_{i_{\text{est}}} - p_{i_{\text{dt}}}) \tag{14}$$

$$p_{i_{\text{est}}} = \begin{cases} \tau_t(t_l, p_{i_{\text{d}}}), & \text{if } i = 0 \\ p_{(i-1)_{\text{est}}} + \tau_t(p_{(k-1)_{\text{d}}}, p_{k_{\text{d}}}), & \text{else} \end{cases} \tag{15}$$

In Eqn. 15, $\tau_t(s_1, s_2)$ represents the time (i.e., number of rounds) it takes t to drive from s_1 to s_2. And the estimated arriving time for passenger p corresponds to the case $i = 0$. All passengers already onboard correspond to cases with $i > 0$.

Based on the calculated total delay d_t for each candidate train, an optimal solution can be found among all of them by selecting the train with the minimal total delay for all involved passengers.

Boarding a train will prevent that train from performing further actions in the current round, i.e., it belongs now to a set of blocked trains, the t_{blocked}. If the passenger group is not able to find a suitable train in the current round, it must wait for the next round. Since passengers are sorted by their desired arrival time, the iterative process ensures that passengers with higher priority will board the most suitable trains first.

Like boarding, leaving a train prevents that train from performing further actions in the current round.

B. Train actions

Once all passengers have executed an action, actions for individual trains will be determined based on its current location. The overall process of train activities is illustrated in Fig. 4. Since trains located at stations should perform different actions as trains that are driving on lines, details regarding the determination of train activities in both cases are further illustrated in Fig. 5-6 and Fig. 7, respectively.

For each train located at a station, it is necessary to determine its destination t_d together with a route t_{route}. This process depends on whether the train has passengers on board or not.

In case a train t at a station has passengers, i.e., $|t_{\text{pas}}| \neq \emptyset$, the onboard passengers are sorted by their desired arrival time, leading to a sorted set t_{pas}, as is shown in Eqn. 16.

$$t_{\text{pas}} = \{p_0, p_1, \ldots, p_n \,|\, p_{i_{\text{dt}}} \leq p_{(i+1)_{\text{dt}}}, 0 \leq i \leq n-1\} \tag{16}$$

The destination of train t is set to the destination of the passenger with the lowest desired arrival time, resulting in

$$t_d = (p_0)_{\text{d}} \tag{17}$$

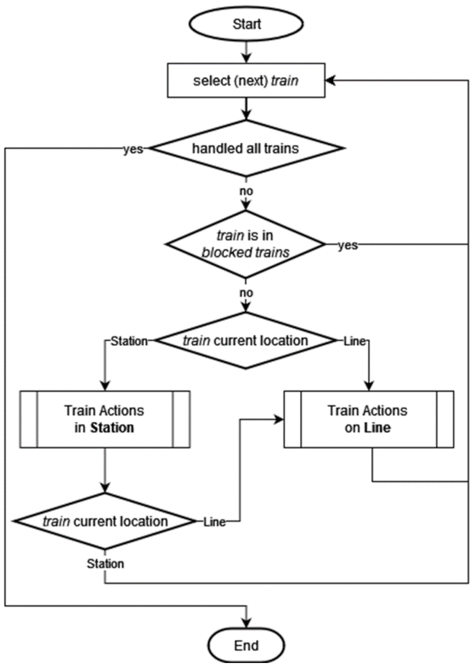


Fig. 4. Train actions.

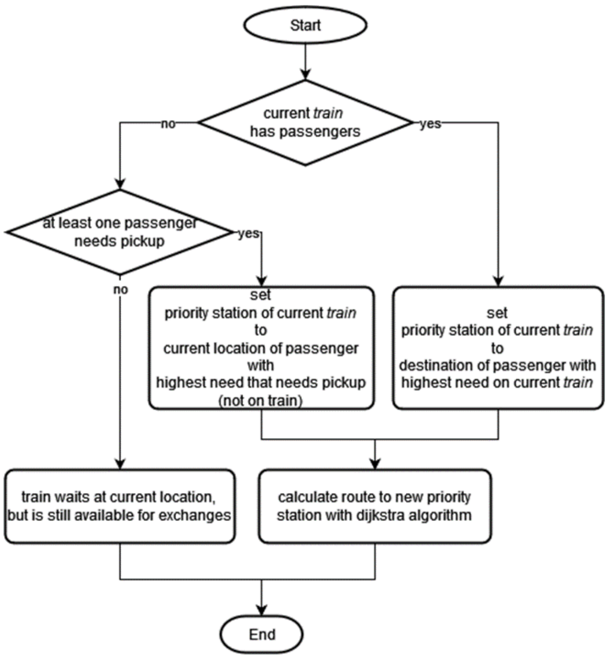

Start
current train has passengers
no
yes
at least one passenger needs pickup
yes
no
set priority station of current train to current location of passenger with highest need that needs pickup (not on train)
set priority station of current train to destination of passenger with highest need on current train
train waits at current location, but is still available for exchanges
calculate route to new priority station with dijkstra algorithm
End

Fig. 6. Determining routes for trains located at stations.

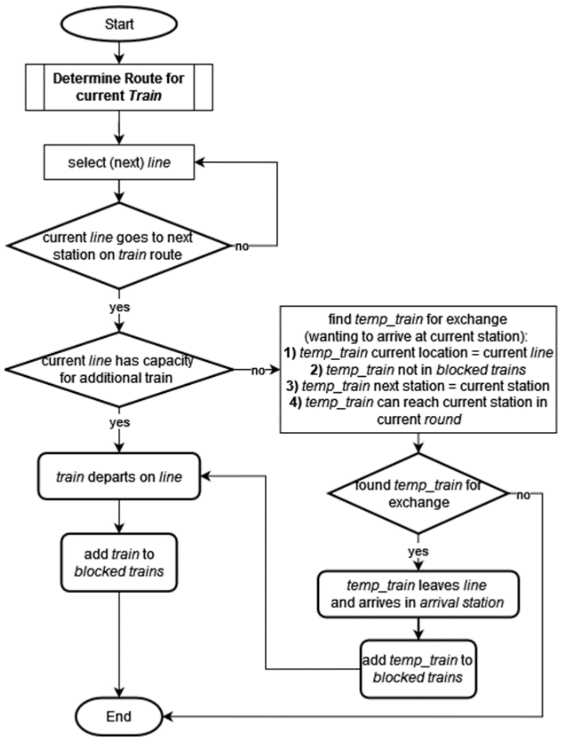

Start
Determine Route for current Train
select (next) line
current line goes to next station on train route
no
yes
current line has capacity for additional train
no
yes
find temp_train for exchange (wanting to arrive at current station):
1) temp_train current location = current line
2) temp_train not in blocked trains
3) temp_train next station = current station
4) temp_train can reach current station in current round
found temp_train for exchange
no
yes
train departs on line
add train to blocked trains
temp_train leaves line and arrives in arrival station
add temp_train to blocked trains
End

Fig. 5. Determine actions for trains at stations.

This means, the lower the arrival time, the higher the priority, and the sooner the passenger gets handled and arrives at the destination. Once the destination is set, the Dijkstra algorithm is used to find the shortest path from t_l to t_d in the constructed graph G.

For each train at a station, an optimal route t_{route} is set to the one that takes train t the smallest number of rounds to get from t_l to $(p_0)_{\text{d}}$, as is formulated as Eqn. 18:

$$t_{\text{route}} = \{s_0, s_1, \ldots, s_n \mid s_n = (p_0)_{\text{d}}\} \tag{18}$$

Once the route is set for the train t at a station t_l with $t_l \in S$, it checks the lines that are adjacent to t_l and chooses the line $l(t_l, s_0)$ that would bring it to the next station on t_{route}. Then train t will start on l, where the condition regarding the remaining capacity of the line is met: $l_{\text{remcap}} = l_{\text{cap}} - l_{\text{trains}} \geq 1$. Once the train t has executed this action, it is moved to t_{blocked}, meaning a new action for it will not be allowed until the next round.

If the line has no more capacity to hold train t, this means the network has a local bottleneck, which has to be dealt with properly.

As is illustrated on the right part of Fig. 5, a possible solution to such kind of bottleneck will be the search for another train t_{temp} which is currently driving on the line and could make room for train t so that t could start on the line and t_{temp} would stop temporarily at station t_l. Suppose there exists a train t_{temp} which is heading toward the station t_l. If the train t_{temp} has not yet executed an action in the current round, it will arrive at station t_l in this round and then be able to swap places with train t. After that, both trains will be moved to t_{blocked}.

If there does not exist a train t_{temp} that meets the above condition, then train t must wait at station t_l. In the next round, it has to search again for the possibility of making a swap operation.

In the simple case that train t at station t_l has no onboard passenger, i.e., $|t_{\text{pas}}| = \emptyset$, it will check whether there are passengers who need to be picked up at other stations, as is shown in Fig. 6. A sorted set of waiting passengers at other stations whose boarding on train t would not violate its capacity can be obtained based on Eqn. 19:

$$P_w = \{p_0, p_1, \ldots, p_n \,|\, p_{i_l} \neq t_l, p_{i_s} \leq t_{\text{cap}}, p_{i_{\text{dt}}} \leq p_{(i+1)_{\text{dt}}}\} \tag{19}$$

If $P_w = \emptyset$, the train just waits at its current station. If such a set of waiting passengers P_w does exist, the destination of train t will be set to the location of passenger $p_0 \in P_w$ with the lowest desired arrival time so that $t_d = p_{0l}$. A route for train t is then determined by the Dijkstra algorithm, leading to $t_{\text{route}} = \{s_0, s_1, \ldots, p_{0l}\}$.

The above process is repeated in each round so that t_{route} can be adjusted dynamically based on the current situations, making it possible to deal with changes or disturbances occurred in the network.

Until now we have put forward several strategies toward iterative action generations for trains that are located at stations. Of course, trains can also be already driving on lines. Fig. 7 shows how actions can be generated for these running trains.

For each train t driving on a line, we have $t_l \in L$. In each round, it is necessary to check whether the train should keep driving on its current line or stop. Suppose C_t^l is the number of rounds the train has already spent on the line, then the stop condition is: $C_t^l \cdot t_s \geq l_{t_l}$, where t_s is the speed of the train t and l_{t_l} is the length of the line.

If the stop condition is not met, the train simply continues driving on the line, C_t^l is incremented by 1, and t is added to t_{blocked}.

If a train reaches its end station s_e on the line, the capacity of s_e has to be checked. If $s_{e_{\text{cap}}} - |s_{e_{\text{trains}}}| \geq 1$, the train can enter the station, and it is added to t_{blocked}. Otherwise, there is a bottleneck situation at station s_e. This has to be treated again with a swap operation. Here there are two possibilities: One way is to find a train which is about to leave station s_e, the other is to find a train in station s_e that has no passengers onboard.

For the former case, a train t_{temp} that fulfills the following conditions will be looked for:

$$t_{\text{temp}_l} = s_e \tag{20}$$

$$t_{\text{temp}_{route}} = \{s_e, s_1, \ldots, s_n\} \neq \emptyset \tag{21}$$

$$t_{\text{temp}} \notin t_{\text{blocked}} \tag{22}$$

If such a train cannot be found, another search will be carried out, by trying to find a train t_{temp}, where the following conditions shown in Eqn. 23-25 are met:

$$t_{\text{temp}_l} = s_e \tag{23}$$

$$|t_{\text{temp}_{\text{pas}}}| = 0 \tag{24}$$

$$t_{\text{temp}} \notin t_{\text{blocked}} \tag{25}$$

If one of the above searches is successful, a swap action will be executed. Then t and t_{temp} will be added to t_{blocked}. Otherwise, train t has to wait until the next round.

VI. Experimental evaluation

The proposed approach has been experimentally evaluated using test data containing various network configurations. In order to test the proposed approach under more difficult conditions than those provided by the InformatiCup, we have realized an automatic data generator. Based on this extended data set, a comparative study has been carried out.

A. Test data and data generator

Each piece of test data consists of a network configuration with an arbitrary set of trains, stations, lines, and passengers, formatted according to the specification of InformatiCup and stored in a file. An example is shown in Fig. 8. It can be observed that for a train whose starting station is not yet set, there is an asterisk (*) shown directly beside the train, as is the case of train T2.

Using our algorithm explained in Section IV, a graph with nodes and edges can be constructed. For a test scenario with five stations, the constructed graph is visualized in Fig. 9, where the number of passengers waiting at each station is shown in color red. Shown in color green are the maximal numbers of trains that can be held at individual stations.

According to our algorithm designed for the preparation phase, a train with an arbitrary starting station will be assigned to station s_1. It has the highest need for additional trains, with twenty groups of waiting passengers, as is shown in Fig. 9.

The initial test data are provided by the organizing society and a few attending teams of the InformatiCup. The original data set contains only a small number of test cases, with few of the test scenarios having more than a hundred elements.

For the automatic generation of test data with larger numbers of elements and more complex configurations, we have built a data generator, where a Python script has been written so that additional test scenarios can be created on the fly. In an interactive process, all the necessary parameters that a test scenario must contain, as is regulated in the specification of the InformatiCup, can be instantiated through user input.

Using the data generator, those attributes t_{cap}, s_{cap}, l_{cap}, p_s, t_s, l_l can be initialized randomly. The number of trains with an arbitrary starting station can be set randomly as well. Random values can also be generated for setting the starting stations for both passengers and trains.

In order to ensure that the generated network was connected, the data generator will first create all possible lines among the stations. Then it will repeat a process, where in each iteration, a line is randomly removed while keeping the graph connected.

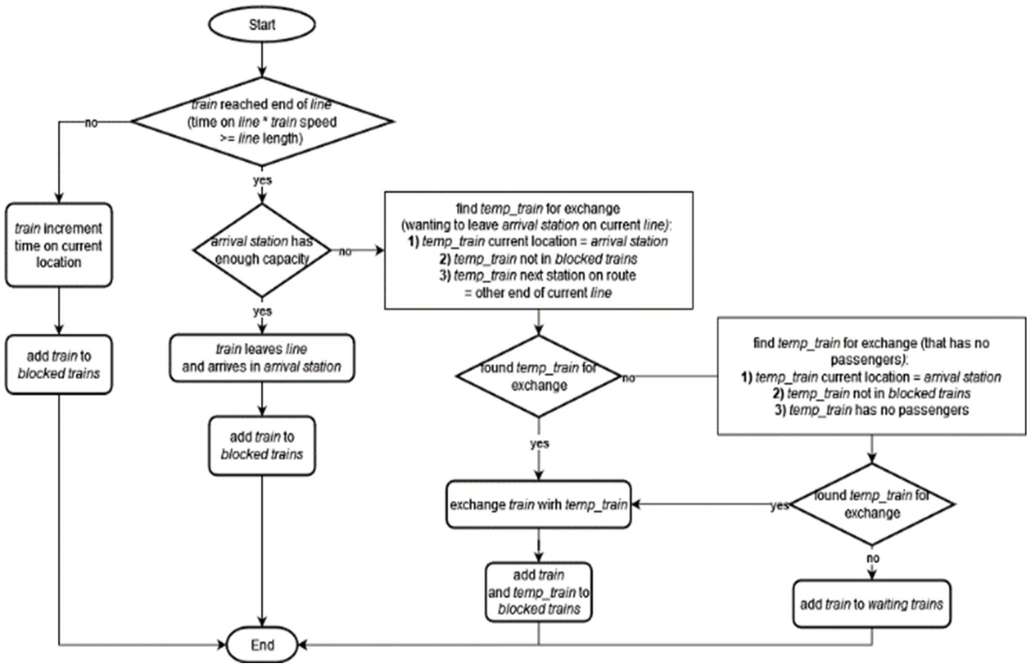
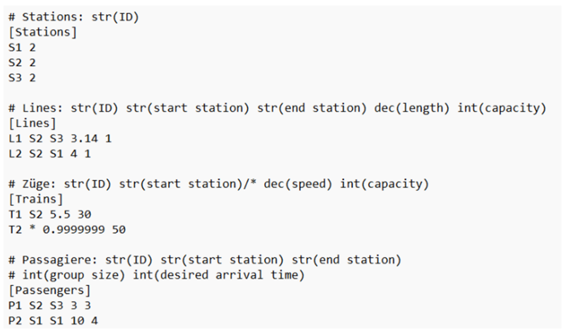

Start
train reached end of line (time on line * train speed >= line length)
no
yes
train increment time on current location
add train to blocked trains
arrival station has enough capacity
no
yes
train leaves line and arrives in arrival station
add train to blocked trains
find temp_train for exchange (wanting to leave arrival station on current line): 1) temp_train current location = arrival station 2) temp_train not in blocked trains 3) temp_train next station on route = other end of current line
found temp_train for exchange
no
yes
find temp_train for exchange (that has no passengers): 1) temp_train current location = arrival station 2) temp_train not in blocked trains 3) temp_train has no passengers
exchange train with temp_train
yes
found temp_train for exchange
no
add train and temp_train to blocked trains
add train to waiting trains
End

Fig. 7. Determine actions for trains driving on lines.



```
# Stations: str(ID)
[Stations]
S1 2
S2 2
S3 2

# Lines: str(ID) str(start station) str(end station) dec(length) int(capacity)
[Lines]
L1 S2 S3 3.14 1
L2 S2 S1 4 1

# Züge: str(ID) str(start station)/* dec(speed) int(capacity)
[Trains]
T1 S2 5.5 30
T2 * 0.9999999 50

# Passagiere: str(ID) str(start station) str(end station)
# int(group size) int(desired arrival time)
[Passengers]
P1 S2 S3 3 3
P2 S1 S1 10 4
```

Fig. 8. Example of a network configuration.

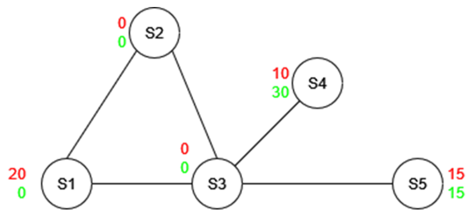


Fig. 9. Visualization of the constructed graph.

The iterative process stops when the desired number of lines was reached.

In order to ensure that a test scenario will definitely result in delay, a few test scenarios have been constructed in such a way that it will enforce a delay. This was achieved by setting the desired arrival time for each passenger to a value only achievable by the fastest train available in the network. This will definitely lead to delay for some passengers who have to travel with a slower train. In the following, these so generated test data will be referred to as delay-enforcing ones.

TABLE II
NETWORK CONFIGURATIONS OF EIGHT SELF-GENERATED TEST CASES

Stations	Passengers	Trains	Lines	Total
6	7	5	10	28
11	15	8	31	65
19	32	12	97	160
33	68	18	301	420
57	143	27	934	1161
97	301	41	2896	3335
165	633	62	8978	9838
281	1330	93	27832	29536

A total of 67 test cases have been used for the purpose of comparative study, where our approach has been compared with the three algorithms introduced in Section II. Within the total test data, eight cases have been constructed with the technique of delay enforcing. The relevant parameters involved in these eight cases are shown in Table II, where each row stands for one test case. From top to bottom, one can observe an exponentially enlargement of network complexity, which is due to the rapid increase in the number of total elements within the network.

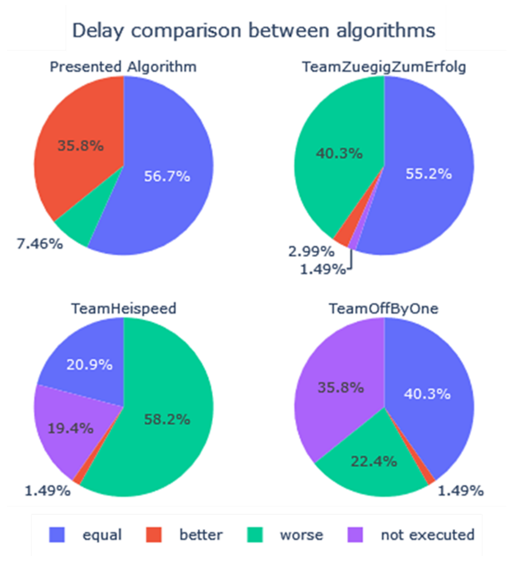


Fig. 10. Robustness and quality comparison.

B. Comparative study

A comparative study has been carried out, involving the performance evaluation of four approaches using two evaluation criteria: one is robustness, and the other is quality. Here, robustness describes the proportion with which an algorithm returns a valid passenger timetable and train timetable. The timetables are defined as valid if they conform to the formalized format described in [1]. The quality of an algorithm depends on the delay over all passengers, i.e., d_{total}.

As demonstrated in Fig. 10, the presented algorithm constructs valid timetables for all the test cases, reaching a robustness of 100%. Team ZZE achieved a robustness of 98.5%, only failing to return valid timetables for one test case. The algorithm from team OBO did not return timetables for 24 cases, resulting in a robustness of 64.2%. Team HS achieved a robustness of 80.6%.

For each algorithm and each input test case, the delay over all passengers has been calculated from the corresponding passenger timetable. The quality was then labelled as better, equal, worse, or not executed, based on the criteria described in Table III.

The distributions of the quality labels are displayed for each algorithm in Fig. 10 as well. The proposed algorithm achieved the highest quality on 35.8% of the test cases and a quality equal to the highest quality of the other three algorithms in 56.7% instances. This means that the presented algorithm was not among the best algorithms in only 7.46% of the time. Team ZZE performed best on two cases and was among the best in 55.2% of all test scenarios. The algorithm performed worse on 40.3% of the inputs. Team HS and team OBO both achieved the highest quality on one test case and were among the best algorithms in 20.9% and 40.3% instances, respectively. Team HS performed worse on 58.2% of the inputs, team OBO on 22.4% of the inputs.

TABLE III
QUALITY LABELS BASED ON PERFORMANCE COMPARISON

Quality labels	Performance compared among the four algorithms
Better	The algorithm achieved the highest quality on the input
Equal	No other algorithm achieved a higher quality on the input, but at least one other algorithm achieved the same quality
Worse	At least one other algorithm achieved a higher quality on the input
Not executed	The algorithm was not able to return valid timetables for the input

TABLE IV
TOTAL DELAY RENDERED BY THE FOUR SCHEDULING ALGORITHMS

	Presented Algorithm	Team HS	Team ZZE	Team OBO
Mean	13.275	1610.5	20.55	23.95
Median	0	3	0	0
Min	0	0	0	0
Max	400	60319	653	715
Sum	531	64420	822	958

It should be noted that a lot of the test cases were constructed in a way such that an overall delay of zero, i.e., $d_{\text{total}} = \sum_{p \in P} p_{\text{delay}} = 0$, could be achieved easily, which explains the high occurrences of the quality label "Equal".

Excluding inputs for which at least one of the algorithms failed to create a valid timetable, one obtains the values of total delays achieved by the four algorithms, as is displayed in Table IV. Among these inputs, the presented approach has outperformed the other three, achieving the shortest delay with the lowest values regarding all the measures including mean, median, minimum, and maximum and sum. Team ZZE achieved the second lowest values, followed by team OBO. Compared to these three teams, the team HS achieved a substantially longer delay with particularly larger values regarding the mean, sum, and maximum measures.

The results achieved by three of the four algorithms on the delay-enforcing data are illustrated in Fig. 11. Here team OBO

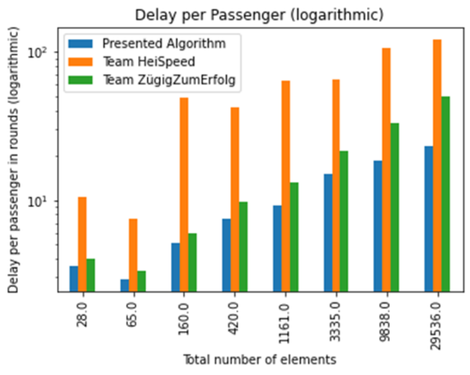


Fig. 11. Delay per passenger on delay-enforcing data.

was excluded in the graphic, as it could not provide valid timetables in 7 out of 8 cases. The proposed approach delivers the timetables with the highest quality for each test instance, followed by team ZZE. Team HS achieved a considerably longer delay on all eight cases.

VII. Conclusion

We have developed an approach for train scheduling with the goal of solving the 2022 InformatiCup challenge. The approach has been validated not only with test scenarios defined by the original InformatiCup but also with more complex scenarios having much more complicated network configurations and conditions. The proposed approach has achieved the highest robustness, and for most test scenarios, the highest quality among all the compared algorithms. Through experimental study, it has been demonstrated that the proposed scheduling strategies designed for an arbitrarily set of passenger requests in different configurations of network resources (i.e., a network with trains and lines) can achieve a comparatively low delay for all passengers.

Using the proposed approach, it is possible to deliver satisfactory timetables for a network of rail transportation systems. Another algorithmic advantage lies in its flexibility and adaptability. If the network configuration has been changed, it is possible to generate an updated timetable with minimal delay while accommodating dynamic variations for trains and passengers within the network. This is due to the fact that during the iterative scheduling phase, passenger and train actions are determined dynamically, by applying the same principle for the minimization of the overall delay.

It should be noted that in the current design of round-based iterations, the assignment of round has an abstract nature, since the line length, the train speed, and the time are given in fictive units. However, the use of rounds can be applied in the real world as well, where metric units (e.g., kilometres per hour) are used instead. While one round could be equivalent to any amount of time, its value should be chosen properly, as a balance must be struck: on the one hand, choosing a shorter time interval for one round would lead to a scheduling result with higher quality; on the other hand, it would require that scheduling decisions have to be made more frequently, which would lead to a higher consumption of computation time and resources.

References

[1] "InformatiCup 2021 - Abfahrt!," 2021. [Online]. Available: https://github.com/informatiCup/informatiCup2022.

[2] K. Ghoseiri, F. Szidarovszky and M. J. Asgharpour, "A multi-objective train scheduling model and solution," Transportation, 2004, pp. 927–952.

[3] X. Cai and C.J. Goh, "A fast heuristic for the train scheduling problem," Computers & Operations Research, Vol. 21, 1994, pp 499-510.

[4] X. Li, D. Wang, K. Li, Z. Gao, "Green train scheduling model and fuzzy multi-objective optimization algorithm," Applied Mathematical Modelling, vol. 37, no. 4, 2013, pp. 2063–2073.

[5] Y. Wang, "Passenger-demands-oriented train scheduling for an urban rail transit network," Transportation Research Part C: Emerging Technologies, vol. 60, 2015, pp. 1-23.

[6] L. Yang, J. Qi, S. Li and Y. Gao, "Collaborative optimization for train scheduling and train stop planning on high-speed railways," Omega, vol. 64, 2016, pp. 57–76.

[7] X. Xu, K. Li, and X. Li, "A multi-objective subway timetable optimization approach with minimum passenger time and energy consumption," Journal of Advanced Transportation, vol. 50, no. 1, 2016, pp. 69–95.

[8] S. Martello, "Knapsack problems: algorithms and computer implementations," New York: J. Wiley & Sons, 1990.

[9] B. Ünal, H. Yang, L.H. Hsu and L. Dreyer, "Team Zügig zum Erfolg - Project Report," 2022. [Online]. Available: https://drive.google.com/file/d/1MlNx56C3BgLioU3_gALE6AEKQRJX3vcb/view?usp=share_link.

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math. 1, 1959, pp. 269–271.

[11] L. Carstens-Behrens and O. Herrmann, "An Application Of Tabu-Enhanced Genetic Search To A Railway Optimization Problem," 2022. [Online]. Available: https://drive.google.com/drive/folders/1oYtIKKOKqyTLJL1nZQeGhl97w6LFdPCq?usp=share_link.

[12] S.J. Russell and P. Norvig, "Artificial intelligence: a modern approach," Printince hall, 2010.

[13] F. Glover and E. Taillard, "A user's guide to tabu search," Annals of Operations Research, vol. 41, 1993, pp 1–28.

[14] X. Zhou and M. Zhong, "Bicriteria train scheduling for high-speed passenger railroad planning applications," European Journal of Operational Research, vol. 167, no. 3, 2005, pp. 752–771.

[15] J. Freyberg, M. Walther, J. Wildberger and H. Wünsche, "Team Heispeed - Project Report," 2022. [Online]. Available: https://drive.google.com/file/d/1KwNQX27NTHtL-JjdSbGBNHNb9Ana0EeU/view?usp=share_link.

[16] D. Hou and W. Zhang, "Multi-Warehouse Location of Logistics Based on Dijkstra and Divide-and-Conquer Algorithm," 2017 10th International Symposium on Computational Intelligence and Design (ISCID), 2017, pp. 442-447.

[17] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," The Computer Journal, Vol. 8, 1965, pp. 250–255.

Comparison of Missing Data Imputation Techniques using Autoencoders

1st Rubaiya Kabir Pranti
Communication Systems and Networks
Cologne University of Applied Sciences
Cologne, Germany
rubaiya_kabir.pranti@smail.th-koeln.de

2nd Gernot Heisenberg
Information and Communication Sciences
Cologne University of Applied Sciences
Cologne, Germany
gernot.heisenberg@th-koeln.de

3rd Sven Woehrle
Information and Communication Sciences
Cologne University of Applied Sciences
Cologne, Germany
sven.woehrle@th-koeln.de

***Abstract*—Missing data in large-scale surveys reduces reliability in data analysis. Demographic and Health Surveys (DHS) are significant surveys including household information from which users can take valuable data insights for social activities and business purposes. DHS contain numerous missing values for various reasons. In this study, we aim to show a methodical comparison of autoencoder-based (AE) three techniques (simple autoencoder (AE), denoising autoencoder (DAE), and variational autoencoder (VAE)) which are advanced neural networks to impute missing values effectively with remarkable imputation results. For this purpose, we downloaded the raw datasets initially from the DHS'S website, from which we consider only household water supply information for our study. Before training the AE methods, we perform data pre-processing (including feature engineering), and exploratory data analysis to identify important features and then we transform the data to impute missing values precisely. We make the training dataset with complete data and generate artificial missingness in the test data and later we measure performance metrics from these variables. For evaluation purposes, we apply the k-fold cross-validation technique for five different missing ratios together with integrated hyperparameter regularization for improved imputation performance. Finally, the missing data are imputed on unseen test data after the final training of the autoencoders (AEs). In this regard, we analyze model performances using performance metrics such as root mean square error (RMSE) and visual inspection. Finally, we compare the performances of three AE-based imputation methods that provide expected imputation outputs. Especially, our study reveals that DAE provides better results compared to other two techniques (AE, VAE) in terms of RMSE, but VAE is more successful at determining all the underlying patterns and relationships of data features though RMSE values are higher than DAE's. Hence, AE falls behind DAE and VAE due to its simple architecture to capture patterns.**

***Index Terms*—missing data imputation, autoencoders, deep learning, cross-validation, AE, DAE, VAE, DHS**

I. Introduction

Using traditional missing data imputation methods such as mean and method imputation methods for handling missing data alone are insufficient techniques to deal with this issue. By building sophisticated neural network architectures, trained under carefully designed loss functions, state-of-the-art models obtain impressive performance [1]. Therefore, in this study, we execute artificial neural networks (ANN) named as autoencoders including AE, DAE, and VAE for the effective imputation of missing data to address the problem.

Firstly, simple/regular AE, this deep neural network has latent space in between the input data and output data. AE is trained to minimize the reconstruction loss between the input and output data. For our missing data imputation tasks, AE primarily learns the internal correlations from the complete input training dataset. After learning from the training process, missing values are imputed on unseen test dataset with the help of learned representation during the training phase.

DAE, on the other hand, is a more effective approach than traditional AE (baseline) where noise is incorporated into input data. Then reconstruction of the input data is possible from the latent representation. In our research, DAE is trained on complete corrupted data where gaussian noise is added to the input data. Therefore, DAE learns to denoise or reconstruct the original data from the noisy input. DAE can considerably impute missing values more than simple AE.

Furthermore, VAE is a generative autoencoder that learns to reconstruct the input data and generate new data from the learned distribution. By following this, VAE fills in the missing values because of its homogeneity where we can sample from the latent space much smoother and it can generate a more meaningful output from every sample data location. Variational autoencoder produces more diverse and realistic imputations than both DAE and AE.

The evaluation of the AE, DAE, and VAE methods primarily depends on the root mean square error (RMSE) metric. This metric is preferred due to its ability to provide an acceptable performance measure.

Based on the results of tests, the DAE has the best performance with mean RMSE, followed by VAE, where the simple autoencoder has higher mean RMSE values.

II. Data Retrieval Process

DHS surveys gather information from eligible respondents by using various types of questionnaires. For our study, we focus on only household questionnaires (from sub-saharan africa region people) and concentrate on analyzing water supply-related features (https://dhsprogram.com/methodology/Survey-Types/DHS-Questionnaires.cfm). The household questionnaire related raw datasets are retrieved from the DHS program website [2]. Through proper pre-processing of data, a total of 970637 rows of values contain seventeen columns including three vital features named 'source of drinking water (simplified)', 'time to get to a water source (minutes)', and 'location of source for water' related to the water supply. After that, by executing the data pre-processing step on the final dataset, all records with missing values in the original survey dataset are deleted. By employing the feature engineering process, we only keep these mentioned three features. In this paper, we mention these features as 'Source', 'Time', and 'Location' in short. These three columns have a correlation with each other. For instance, if the 'Time' column has a value of 0 minutes, the 'Location' column must have location features such as 'in own dwelling' or 'in own yard/plot'. But, if the 'Time' column contains more than 0 minutes, the 'Location' column has to provide 'elsewhere' category as mandatory relationships. In this way, the resulting dataset with fully known observations contains 679187 rows of data which are classified into fifteen different categories in the 'Source' column, three different categories in the 'Location' column, and minutes ranging from 0 to 1000 minutes in the 'Time' column.

III. Methodology

A. Data Pre-processing

We assess data pre-processing to identify crucial features and transform the data to enhance the effectiveness of the AE-based methods. For the next step, we use the cleaned dataset as input for the autoencoder-based imputation methods (AEs).

B. Feature Engineering

We conduct feature engineering to reduce the dimensionality of the dataset. In our case, the relevant features are named in short as Source, Time, and Location features.

C. Splitting Dataset

The dataset is divided into training and testing sets using a random split of 80% for training and 20% for testing. We use the training dataset to train the AEs, the validation dataset with missing values for validation purposes and the testing set with missing values to evaluate the performance.

D. Training Methods

We train three autoencoder-based methods during the training procedure using TensorFlow and Keras by customizing the architecture, loss function, and optimizer and by adding regularizer and initializer to specific layers which differ for these three methods.

E. Evaluation through Cross-Validation

The evaluation criteria for imputing missing values take root mean square error (RMSE) as a measure [3]. For better evaluation, we perform cross-validation to get an estimation of the performance of our deep learning methods through RMSE (reconstruction loss). Here, we apply k-fold cross-validation on the train folds, and later we obtain imputed data from test folds.

F. Hyperparameter Optimization

We execute hyperparameter regularizations for all three AEs (simple AE, DAE, VAE). Though, it is not recommended to use more layers and nodes to avoid complexity and increased time costs, still, for analysis purposes, different combinations of complex architecture with different bottleneck nodes are investigated with fixed steps which allows a precise standard of symmetry between the encoder and decoder, which is beneficial for the learning process [4]. To prevent overfitting in the AE and DAE, we integrate L2 regularizer. As an optimizer, we use the Adam optimizer. For activation function, ReLU (Rectified Linear Unit) was proven to be the most effective one. Another hyperparameter used in our AE methods is the He uniform initializer for initializing weights in the neural network layers. Additionally, for every autoencoder method, we chose an epoch size of 50 for time and resource constraints.

IV. Missing Value Imputation

A. Experimental Setup

To evaluate our methods, we clean (without any missing values) the dataset as the first step with some additional cleaning of unexpected rows. We also carry out label encoding on the cleaned and complete dataset. Then, we split the dataset into training, test, and validation set. Later, we introduce artificial missingness to the test set. Then, the mode imputation method (most frequently observed values in the dataset) fills the missing values in the test dataset temporarily before applying autoencoders as AEs should not be trained with NaN values. Therefore, we introduce the AE and its variants' networks and compile these with some configurations. These above-mentioned steps are repeated again within the k-fold cross-validation process. The RMSE is calculated between the actual (ground truth) values and imputed values. We estimate average RMSE by employing 5-fold cross-validation for each missing ratio 10%, 20%, 30%, 40%, and 50% by averaging the results between the 5 folds [5]. After getting the results for five different missing ratios, the unbiased examination is completed through a cross-validation technique to see how the AEs will perform on unseen test data. Then we train the AEs again as the final assessment step. Next, the AE techniques predict and impute missing values on the unobserved test dataset to assure AEs' real-world performance [6].

B. Evaluation Metrics

For our research project, we have mainly selected the Root Mean Squared Error (RMSE) metric and scatter plots to evaluate and compare the performance of different AEs in handling missing values. Here, RMSE can give a precise understanding of the deviation between ground truth values and imputed values. To assess the variability of the average RMSE values, the standard error of the mean RMSE and the confidence interval of the RMSE are calculated using the standard error. In this case, a 95% confidence interval is computed, which is based on a z-score of 1.96.

V. Overall Comparison

In this section, we compare the overall results for the DHS dataset regarding missing data imputation with the help of a 5-fold cross-validation process separately for every AEs such as regular AE, DAE, and VAE (Table 1.). And then we also present inconsistent results and performance (Table 2.).

A. Discussion on the Tables

From Table 1., for missing ratios (from 10% to 50%), we can observe that Denoising Autoencoder (DAE) is superior in most cases across various architectures upon understanding the underlying patterns and relationships well, exhibiting lower RMSE values than the other two AEs. Besides, we find that VAE and DAE outperform AE steadily for all architectures. And in certain cases, VAE even surpasses DAE and AE.

From Table 2., we see that there have been some inconsistencies and unrecognized patterns in the output of AE, DAE, and VAE methods. We acknowledge that for some rows, AEs have imputed missing values incorrectly as the location with 'elsewhere' label having time equals 0 minutes which does not match with the original training data. In this way, some rows are not acceptable though the RMSE values are lower. Regarding variational autoencoder (VAE), it outperforms other AE and DAE for some architectures except for (128,10) architecture. VAE has been always uniform with stable results without any contradictory output as earlier mentioned. But for VAE, RMSE values are higher than DAE in the majority of cases. For DAE, there are also evident inconsistent rows found, but the RMSE values are comparatively lower than the other two AE methods which makes it the optimal method for our dataset [7].

Architecture	M. Ratio	Comparison (RMSE Result)
(64,5)	10% - 50%	VAE > DAE > AE
(64,10)	10% - 50%	DAE > VAE > AE
(128,5)	10% - 50%	DAE > VAE > AE
(128,10)	10% - 50%	DAE > AE > VAE
(256,5)	10% - 50%	VAE > DAE > AE
(256,10)	10% - 50%	DAE > VAE > AE

Table 1. Performance Comparison for different architectures

Detailed Individual Autoencoder Results

The following tables are regarding Table 3. to Table 5. which actually demonstrate the performance of the AEs in terms of missing data imputation considering 'Time' feature only. We compute missing data imputation tasks using three different Autoencoder approaches (AEs), the tables depict the average RMSE values through 5-fold cross-validation (for missing ratios ranging from 10% to 50%). It should be noted that a lower RMSE value represents better performance. We also observe that all methods perform worse as the missing rate increases, which is to be expected because a dataset with a higher missingness proportion is also harder to impute [6]. Due to time constraints, we focused on evaluating the bottleneck layer with 5 and 10 nodes only for each AE method, considering only three features.

Architecture	AE	DAE	VAE
(64,5)	11786 rows	279 rows	21 rows
(64,10)	585 rows	319 rows	0 rows
(128,5)	17 rows	504 rows	0 rows
(128,10)	11486 rows**(UI.)**	11090 rows**(UI.)**	12176 rows**(UI.)**
(256,5)	10898 rows	0 rows**(UI.)**	4 rows
(256,10)	0 rows**(UI.)**	902 rows	0 rows

Table 2. Inconsistent and Unrecognized Results Comparison for Different Architectures for Missing Ratio=30%, where UI=Unidentified Patterns

B. Imputation Results of Regular AE

Table 3. represents the RMSE results with uncertainty measures of a 5-fold cross-validation process for regular AE for the time column. The architecture combination (64, 10) performs acceptably. It can be concluded that when the bottleneck layer has fewer nodes, AE struggles to capture all key patterns in the input data.

C. Imputation Results of DAE

Table 4. displays the RMSE results with uncertainty measures for DAE using a 5-fold cross-validation process for the time column. Among the architecture combinations, (64,5) shows poor performance, characterized by higher RMSE values and inconsistent rows. Specifically, they only capture the Tubewell or borehole label in the source column and incorrectly identify the location as elsewhere with 0 minutes. These outputs fail to capture many significant patterns and result in higher RMSE values. The most meaningful architectures are again (64,10) like AE.

D. Imputation Results of VAE

For VAE, Table 5. shows RMSE results with uncertainty measures using a 5-fold cross-validation process for the time column. VAE is good at supplying more consistent outputs than AE and DAE. But RMSE values can be seen higher in most cases. Almost all architectures are acceptable for the VAE method for its consistency, except the (128,10) combination which deals with many unidentified patterns (only captured source= Tubewell or borehole and location= elsewhere with 0 minutes) by leaving many significant patterns (Table 2) in outputs with higher RMSE values.

Conclusion

In this study, we worked on missing data imputation tasks using AE, DAE, and VAE methods in the DHS dataset. In

Architecture	Bottleneck	10%	20%	30%	40%	50%
64	5	9.737±0.720	12.994±0.677	16.302±0.784	18.468±0.706	20.256±0.343
64	10	9.715±0.729	12.934±0.669	16.199±0.788	18.330±0.704	20.076±0.333
128	5	9.752±0.712	13.008±0.671	16.289±0.784	18.421±0.703	20.189±0.333
128	10	9.711±0.728	12.936±0.672	16.248±0.793	18.380±0.708	20.126±0.336
256	5	9.743±0.722	12.996±0.666	16.278±0.782	18.415±0.700	20.167±0.333
256	10	9.707±0.725	12.922±0.667	16.186±0.789	18.317±0.706	20.066±0.334

Table 3. Average RMSE Results for Autoencoder (Time)

Architecture	Bottleneck	10%	20%	30%	40%	50%
64	5	9.343±0.455	12.439±0.817	15.439±0.451	18.160±0.427	20.191±0.713
64	10	9.104±0.775	12.917±0.634	15.966±0.438	17.870±0.208	19.707±0.900
128	5	9.104±0.775	12.920±0.632	15.964±0.437	17.874±0.209	19.707±0.899
128	10	9.104±0.776	12.916±0.631	15.964±0.436	17.872±0.211	19.710±0.897
256	5	9.099±0.775	12.919±0.632	15.967±0.435	17.877±0.210	19.712±0.898
256	10	9.103±0.773	12.923±0.628	15.970±0.425	17.877±0.211	12.923±0.628

Table 4. Average RMSE Results for Denoising Autoencoder (Time)

Architecture	Bottleneck	10%	20%	30%	40%	50%
64	5	9.721±0.724	12.938±0.669	16.212±0.788	18.353±0.701	20.129±0.354
64	10	9.725±0.726	12.941±0.668	16.213±0.784	53.712±0.726	20.104±0.333
128	5	9.722±0.726	12.953±0.656	16.210±0.790	18.349±0.700	20.099±0.334
128	10	18.284±15.370	12.943±0.6677	16.216±0.786	18.354±0.703	20.109±0.346
256	5	9.728±0.734	12.942±0.672	16.261±0.785	18.348±0.704	20.093±0.332
256	10	9.722±0.725	12.941±0.668	16.215±0.791	18.348±0.706	20.099±0.339

Table 5. Average RMSE Results for Variational Autoencoder (Time)

this regard, we extended the existing deep learning architectures by introducing specific hyperparameter optimizations that are well-fit for particular scenarios. We investigated various missing rates during the cross-validation process. Our experiments consistently showed that DAE outperformed the other AE methods in all aspects. Therefore, DAE surpassed other methods due to the little effect of the noise. In contrast, VAE had higher RMSE values than DAE because it was strongly affected by hyperparameters. However, when it comes to generating consistent (valid) imputations, the VAE exceeds both the DAE and AE architectures.

References

[1] Śmieja, Marek, Maciej Kolomycki, Łukasz Struski, Mateusz Juda, and Mário Figueiredo. ”Iterative Imputation of Missing Data Using Auto-Encoder Dynamics.” In Iterative Imputation of Missing Data Using Auto-Encoder Dynamics, 258-269. 2020. doi:10.1007/978-3-030-63836-8_22.

[2] Abubakar, Ismaila Rimi. 2017. ”Access to Sanitation Facilities among Nigerian Households: Determinants and Sustainability Implications” Sustainability 9, no. 4: 547. https://doi.org/10.3390/su9040547.

[3] Gaudette, L., Japkowicz, N. ”Evaluation Methods for Ordinal Classification.” In Advances in Artificial Intelligence. Canadian AI 2009, edited by Y. Gao and N. Japkowicz, Lecture Notes in Computer Science(), vol. 5549. Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-01818-3_25.

[4] Pereira, R.C., Santos, M., Rodrigues, P., Henriques Abreu, P. (2020). ”Reviewing Autoencoders for Missing Data Imputation: Technical Trends, Applications and Outcomes.” Journal of Artificial Intelligence Research 69 (2020): 1255-1285. doi:10.1613/jair.1.12312.

[5] Psychogyios, K., Ilias, L., and Askounis, D. ”Comparison of Missing Data Imputation Methods using the Framingham Heart study dataset.” In 2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Ioannina, Greece, 2022, pp. 1-5. doi:10.1109/BHI56158.2022.9926882.10.1109/BHI56158.2022.9926882.

[6] Psychogyios, K., Ilias, L., Ntanos, C., and Askounis, D. ”Missing Value Imputation Methods for Electronic Health Records.” IEEE Access 11 (2023): 21562-21574. doi:10.1109/ACCESS.2023.3251919.

[7] Ryu, S., Kim, M., Kim, H. ”Denoising AutoencoderBased Missing Value Imputation for Smart Meters.” IEEE Access 8 (2020): 40656-40666. doi: 10.1109/ACCESS.2020.2976500.

FPGA-based acceleration of neural net simulation

Felix Sebastian Nitz
TH-Koeln, University of Applied Science
Cologne, Germany
felix.nitz@smail.th-koeln.de

***Abstract*—The widespread integration of `Artificial Intelligence` (AI) in both Industry 4.0 and personal life necessitates an objective examination of the factors driving its adoption. This study explores modern hardware designed to accelerate AI applications, which plays a crucial role in optimizing AI performance. The article categorizes AI accelerators into two types: centralized and edge accelerators. Centralized accelerators, such as `Graphics Processing Units` (GPUs) and `Tensor Processing Units` (TPUs), focus on large-scale data centers and are utilized for cloud-based AI tasks. Conversely, edge accelerators, exemplified by `Multiprocessor Systems-on-Chips` (MPSoCs) with a `Field Programmable Gate Arrays` (FPGAs)-part, are intended for decentralized applications that necessitate prompt outcomes similar to those observed on smartphones or robots. The paper presents an in-depth examination of the techniques and characteristics utilized in both areas, with a focus on the enhancement of AI in the edge domain through FPGA technology. A closer look at the FPGA-based edge reveals that Intel is currently the only vendor using dedicated hardware for AI acceleration. Other manufacturers are focusing on reconfiguring existing hardware by enhancing the interface between a `Central Processing Unit` (CPU) and an FPGA in one device.**

***Index Terms*—Machine Learning, `Deep Learning`, ASIC, FPGA, MPSoC, Decentralization, AI acceleration, AI on edge, Centralized AI, Energy Efficiency**

I. Introduction

Since `Artificial Intelligence` (AI) is increasingly important in both the Industry 4.0 movement and our personal lives, it is intriguing to examine the reasons behind its profitability. Public acceptance and interest in AI-based products, ranging from autonomous driving to rice cookers with fuzzy control, are a contributing factor [34]. An AI-based system can significantly improve a product's efficiency, safety, and usability [1] [2]. To achieve optimal performance, modern hardware is crafted specifically for accelerating AI-driven applications [35]. As stated by Synopsis, an AI accelerator is a high-performance parallel computing machine designed to efficiently process AI workloads, including DL [3]. There are two main spaces for accelerating AI-based operations: Centralized and edge computations. The first approach involves using large, centralized data centers to either train neural networks or deploy massive AI-based services. Examples of this include voice assistants in smart homes and the advancements in cloud based training of AI-based tasks like autonomous driving [4]. The edge represents the second main space. Its application requires an almost immediate outcome, such as for interactive programs used on smartphones or robots. Due to this decentralized function, energy efficiency and compact size are key factors.

This paper evaluates potential hardware-based methods for accelerating AI-based tasks. It begins by outlining the main characteristics of the two main spaces and describing the methods used to accelerate AI processes at the hardware level. The focus then shifts to an edge space example, featuring a practical experiment on `Field Programmable Gate Array` (FPGA)-based AI acceleration. Finally, this study compares the two main spaces, with a particular emphasis on FPGA technology as a representative of edge computing. The research aims to determine the relative capabilities of each space.

II. AI accelerator main spaces

As previously stated, AI accelerators can be divided into two main areas: The centralized and the edge accelerators. To enhance comprehension of each space, a representative will be used to explain them individually. Considering the latest industry advancements in centralized accelerators, apart from supercomputers, this paper will concentrate on `Graphics Processing Units` (GPUs) and `Tensor Processing Units` (TPUs) as representatives of this category. In the edge computing realm, FPGAs serve as proxies due to their multiplexing capabilities, compact form factor, and wide availability, making them powerful computing tools. [36].

The centralized space of AI acceleration focuses on the clustering of processor units, most commonly of the graphics or tensor type [35] [37]. These units are frequently utilized for cloud-based applications or in data centers. Initially, Graphics Processing Units were used due to their availability and computing power. Subsequently, companies such as Nvidia and AMD began developing new types of GPUs that could accelerate AI-related operations at the hardware level. This was achieved through the optimization of communication between CPUs and GPUs, creating a specialized computing path for AI-related tasks. The introduction of CUDA and the resulting improvements in parallel computing tasks within a single GPU have accelerated AI-based operations [39]. Through further developments of CUDA, an enhanced acceleration could be attained through cross-talk between multiple GPUs and by intelligently assigning tasks to each unit. This configuration enables an increase in both compute and memory bandwidth, resulting in a significant acceleration of AI-specific tasks [35]. Additionally, implementing the IEEE 754 standard for `Floating Point` (FP) arithmetic in GPUs has further enhanced precision and speed [5] [6] [7]. By increasing the number of `Arithmetic Logic Units` (ALUs), the GPU can execute multiple multiplications and additions simultaneously. In addition, `Floating-Point Units` (FPUs) were introduced as FP or `single precision` units to the GPUs. An FPU is a specialized logic unit that directly performs arithmetic operations with Floating-Point values [40]. FPUs have been present in CPUs since the 1990s [8]. However, because of increased power and space requirements, they were only implemented in GPUs through the adoption of IEEE 754 in 2008 [9]. This, in turn, has accelerated AI tasks utilizing GPUs.

Since May 2016, Google introduced an alternative hardware solution for speeding up AI processes [38]. The `Tensor Processing Unit` (TPU), a custom-designed `Aplication-Specific Integrated Circuit` (ASIC), is specifically created to optimize and accelerate the execution of machine learning algorithms that involve matrix operations. Such operations are frequently employed in neural networks. TPUs are constructed for efficient handling of the intensive computational demands of `Deep Learning` (DL), a subgroup of AI tasks, as compared to traditional CPUs or GPUs. TPUs are parallel processors that specialize in matrix computations and are optimized for high-speed, low-precision operations, making them particularly well-suited for training and running large-scale Deep Learning tasks. Google developed TPUs initially for internal use in their data center to power various Google services such as Google Search, Google Translate, and Google Photos. However, since February 2018, Google has made TPUs available to external developers through the Google Cloud Platform. This move has enabled others to utilize the power of TPUs for their own machine learning workloads. TPUs are frequently utilized in various fields, including natural language processing, computer vision, speech recognition, and recommendation systems, among others, that require large-scale DL models. They have substantial speed and efficiency benefits over CPUs and GPUs, resulting in quicker training and inference times, which can be critical in scenarios that necessitate real-time or near-real-time processing. To explain the advantages that TPUs have over GPUs and CPUs would necessitate a separate paper. In summary, TPUs [41]:

1) Are optimized for matrix processing (multiplication and other tensor operations in parallel) due to a high number of processing units.
2) Have a high memory bandwidth, allowing for fast data access and movement. Which is crucial for deep learning workloads, as neural networks often involve processing large amounts of data. The high memory bandwidth enables TPUs to rapidly access data, reducing bottlenecks and accelerating computation.
3) Use a reduced-precision arithmetic of 16-bit or even 8-bit, in contrast to GPUs and CPUs. This reduces memory usage and the size of FPU processing elements while increasing processing speed. Deep Learning models can tolerate this precision reduction without significant accuracy loss.
4) Are designed to work in clusters, enabling easy scaling for larger and more demanding machine learning tasks. Multiple TPUs can be connected together to form a TPU pod, which allows for distributed training of large-scale models. This scalability is particularly beneficial for training complex models and accelerating research and development in the field of machine learning.

By combining these features, TPUs offer significant performance improvements over CPUs and GPUs when it comes to Deep Learning tasks [41]. They provide faster training and inference times, increased throughput, and improved energy efficiency. Therefore being the TPUs are the preferred choice for large-scale machine learning workloads.

A. *AI acceleration on the edge*

AI acceleration on the edge refers to the process of running AI algorithms and models directly on edge devices –such as smartphones, industrial machinery, automobiles, `Internet of Things` (IoT) devices, and embedded systems–rather than relying on cloud-based servers for AI processing [10] [11] [12]. These edge devices are equipped with dedicated hardware, such as GPUs or specialized AI chips like ASICs or FPGAs. Such Hardware components are designed to facilitate AI computations and handle intricate workloads effectively. An inherent limitation of edge AI is its reduced computational capacity compared to centralized acceleration. Hence, efficient data management and preprocessing are crucial in developing an AI model. The issue lies in the fact that edge AI systems frequently handle significant amounts of data, and transmitting all the unprocessed data to the cloud for processing might not be feasible due to concerns over privacy or limitations in bandwidth. A solution to this predicament is to perform data preprocessing and filtering directly at the edge. This entails the processing of raw data, the extraction of relevant features, and the transmission of only essential information to the cloud or the ability to make local predictions directly on the device. This method reduces data transmission and conserves network bandwidth and thereby enhancing responsiveness for real-time on-edge computations.

While it is possible to train an AI model directly on the edge device, it is more common to train the model on a high-performance computer or in the cloud before deploying it to the edge device. Popular frameworks like PyTorch, TensorFlow, or Caffe may be used during the training phase. The model is optimized to efficiently operate on edge devices' hardware, taking into account variables including power consumption, memory constraints, and computational capabilities. Employing various optimization techniques can enable this objective. For example, implementing model compression techniques, such as removing redundant parameters and applying knowledge distillation, can yield smaller, more efficient models. `Post-Training Quantization` (PTQ) reduces the precision of weights and activations in a model, lowering memory and computing demands, resulting in more resource-efficient use without sacrificing accuracy [13]. Pruning reduces computational complexity by removing unimportant connections or filters from the model (refer to figure

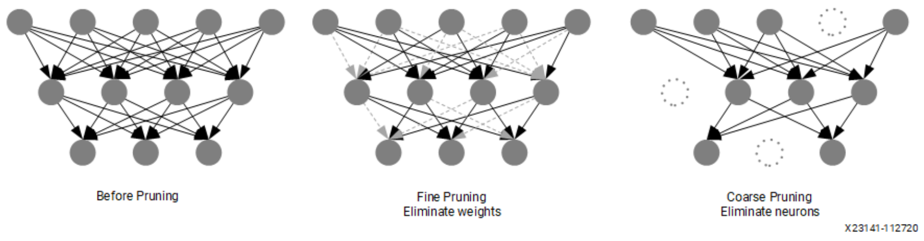


Figure 1 Example of two pruning methods [31]

1) [14] [31]. Optimization methods help attain a more equitable trade-off between model size, computational requirements, and precision.

After the AI model is deployed on the edge device, it has the capability to apply real-world data for predictions or decision-making. The inference process frequently happens directly on the edge device without relying on a cloud connection within the context of AI on the edge. This method removes the need for data transmission to remote servers and reduces latency, thus allowing for real-time decision-making.

Specialized hardware optimized for matrix calculations and parallel processes is employed to accelerate edge-based AI, similar to centralized acceleration. Offloading computations to hardware accelerators on edge devices improves inference task performance compared to relying solely on the device's general-purpose CPU [15].

III. Hands-on FPGA-based AI acceleration

As mentioned previously, the domain of AI acceleration on the edge includes a wide variety of devices, ranging from entire smartphones to single `Integrated Circuits` (ICs), such as FPGA. FPGAs present several advantages for executing AI tasks on the edge, which involves deploying AI algorithms on devices with embedded or limited resources. Some advantages of utilizing FPGAs include [42] [43] [44] [45]:

1) Their ability to provide low inference latency makes them ideal for real-time AI applications, like image and speech recognition, where low latency is crucial.
2) Their high energy efficiency, as they can be tailored to execute particular AI operations without expending energy on superfluous computations.
3) The ability to integrate them into compact form factors, making them adaptable for edge devices with confined space.
4) Their highly parallel nature, allowing them to handle multiple AI workloads concurrently, which is advantageous in edge applications that require multitasking.
5) Their versatility and adaptability for a wide range of edge applications, as they can be reconfigured to accommodate changing AI requirements or support different AI models.
6) They can be customized to accelerate specific AI workloads, such as `Convolutional Neural Networks` (CNNs) or `Recurrent Neural Networks` (RNNs), resulting in significant performance improvements.

The ensuing section provides a detailed analysis of the hardware acceleration component of FPGA-driven AI acceleration from the explanation of several hardware acceleration methods employed by vendors. This is followed by a particular instance demonstrating the flow of activities from model training to execution.

A. *AI acceleration on FPGAs*

As is the case with GPUs, FPGAs have not been specifically designed for AI-based tasks, but were initially used because of their aforementioned advantages. It is only since the middle of 2020 that vendors such as AMD, Microchip, and Intel have developed special `Multiprocessor Systems-on-Chips` (MPSoCs) with a FPGAs-part that accelerate AI-based tasks at the hardware level [16] [17] [18].

AMD, operating under the name Xilinx, and Microchip use their own invented `intellectual property` (IP) cores to accelerate AI-based tasks at the hardware level. These IP cores

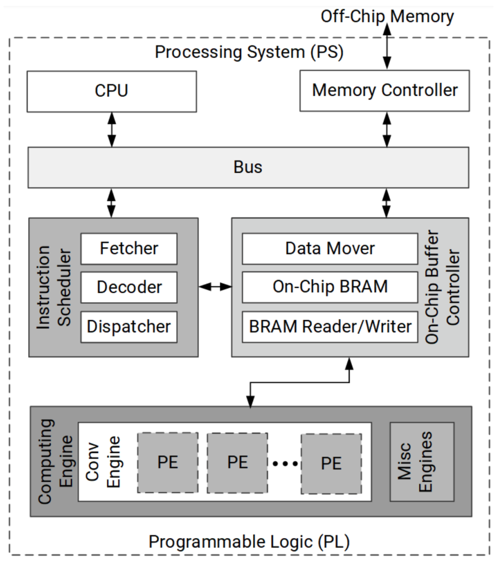


Figure 2 AI DPU High-Level Diagram [20]

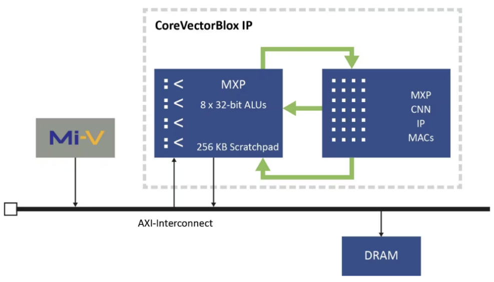


Figure 3 AI CoreVectorBlox High-Level Diagram [21]

are dedicating specific tasks of the inference process within the MPSoC to either the `Processing System` (PS) (CPU-part) or the `Programmable Logic` (PL) (FPGA-part). This means both vendors achieve acceleration by a special configuration of existing hardware. A high-level overview of Xilinx's `Deep learning Processor/Processing Unit` (DPU) and Microchip's CoreVectorBlox IP cores is shown in figures 2 and 3, respectively. Obviously, both vendors utilize dissimilar names for their developed IP core; however, upon closer examination of their documentation, it becomes evident that they essentially offer identical products. IP core-based acceleration utilizes particular sections of the PL in the FPGA-part to accelerate AI tasks. This is accomplished by delegating specific tasks to dedicated hardware resources within the FPGA-part. The hardware consists of ALUs that perform element-wise tensor operations, including addition, subtraction, exclusive OR, shifting, multiplication, dot products, and more [18] [19] [21]. To implement the layers of the CNN, a two-dimensional array of `Multiply Accumulate` (MAC) functions is employed via math blocks (refer to Figure 3). At Xilinx, the `Processing Engines/Elements` (PEs) are responsible for the operations and the implementation of the CNN (refer to Figure 2) [20]. At Microchip, the `Matrix Processor` (MXP) handles element-wise tensor operations, while the MXP CNN IP is utilized to implement the CNN layers [21]. To facilitate data and instruction exchange, IP core-based acceleration incorporates FPGA-external memory, including `Dynamic Random-Access Memory` (DRAM). This external memory functions as a repository of data accessible and manipulable by the acceleration cores. Furthermore, the interaction between the Programmable Logic and PS is enhanced by bus systems such as the `Advanced eXtensible Interface` (AXI) (refer to Figure 3 and 2). The AXI interface facilitates efficient transfer of data and control signals between PL and PS, enabling them to collaborate and accelerate AI-based operations. In both instances, with CoreVectorBlox IP and DPU, it is feasible to overlay multiple networks simultaneously and switch between them dynamically at runtime [20] [21].

Intel also utilizes its own IP core, the `Intel FPGA AI Suite IP`, which operates similarly to Xilinx's and Microchip's [24]. Additionally, Intel employs its optimized tensor arithmetic block, the `AI Tensor Block`, to accelerate AI-based tasks on their MPSoCs. This improved `Digital Signal Processor` (DSP) block architecture is designed to handle standard matrix-matrix or vector-matrix multiplications, and it is equipped for INT32 and `single-precision Floating Point` (FP32) cascade and accumulation capabilities through a two-column tensor structure. It also facilitates INT8 and INT4 data computations, and provides shared exponent support for FP16 and FP12 block Floating Point representations. Intel claims that the AI Tensor Block has the ability to substitute two DSP blocks while executing complex-number multiplication [25]. Additionally, multiple AI Tensor Blocks may be linked to accommodate larger matrices. Figure 4 presents a High-Level overview of the architecture.

It is apparent that acceleration of AI on MPSoCs is primarily achieved by utilizing specialized IP cores to configure the FPGA-part for AI-based tasks. Intel appears to be the only vendor utilizing a hardware accelerator resembling a TPU for MPSoCs, instead of just reconfiguring existing hardware.

Recent studies seek to improve current methods of acceleration. A compelling concept is to replace a fraction of the PL with `Tensor Slices`. These slices function similarly to DSP slices in DSP-operations, following Intel's concept of AI Tensor Blocks [46]. These slices possess a systolic array of Processing Engine/Element that facilitate various tensor operations, multiple dynamically selectable accuracies, and can dynamically break down into individual multipliers and MACs. Additionally, the inputs of the slices feature a local crossbar that reduces the routing congestion resulting from large FPGA blocks. The integration of coarse-grained hard blocks dedicated to DL on FPGAs increases their computational density and enhances their efficiency as hardware accelerators for DL applications. This is accomplished while maintaining the capability to configure the majority of the FPGA at a fine granularity. Compared to Intel's AI Tensor Blocks, Tensor Slices provide full support for INT8, INT16, FP16, and BF16 precision, while having a larger block area but requiring fewer routing wires.

B. *Kria KV260 example*

During the research phase of this study, a detailed analysis of FPGA-based AI acceleration was conducted, with a special focus on the Xilinx environment. The main objective was to evaluate potential hardware-based methods for accelerating AI with FPGAs. This analysis was conducted using the `Kria KV260 Vision AI Starter Kit` [23]. The board is advertised as a MPSoC evaluation kit that is easy to use and comes with an incorporated hardware accelerator (refer to figure 5). Users can train and operate AI models directly on the Kria K26 `System On Modul` (SOM), which is built around the Zynq UltraScale+ MPSoC. After following the guide on getting started [23], the focus shifted to various techniques for running and accelerating models on the Kria KV260.

Several methods exist for obtaining a functional model of the Kria KV260, each tailored to specific utilization scenarios and needs.

One option is to access pre-optimized models from a model repository called `Model Zoo`. These models are usually pre-trained and fine-tuned for specific tasks [28]. By directly loading them onto the Kria KV260, it is possible to rapidly deploy and operate in various Artificial Intelligence and machine learning contexts. This approach is practical and requires minimal model development.

It is also possible to use the Xilinx Appstore by containerization via Docker [27]. Certain Xilinx MPSoCs, such as the Zynq Ultrascale+, possess adequate resources to operate a simplified version of Ubuntu or PetaLinux. This enables Docker to run directly on the MPSoCs Processing System without any hindrances. This method facilitates the deployment of pre-packaged solutions for various tasks on the Kria KV260, allowing for facile switching between them and diminishing the requirement for extensive custom development.

For greater customization and improved control of model design, the Vitis `High-Level Synthesis` (HLS) tool can be utilized [29]. This method involves implementing models at the C/C++ level, which will be synthesized by the tool to functioning `Register Transfer Level` (RTL) code for the MPSoC. It allows for optimization of the models to particular hardware capabilities and requirements, empowering the enhancement in resource utilization and performance.

Another option is to use the tools provided by the Vitis AI development kit, which are designed for Xilinx's IP core-based acceleration (refer to Figure 6) [18]. These three tools form a coherent pipeline, taking a FP pre-trained

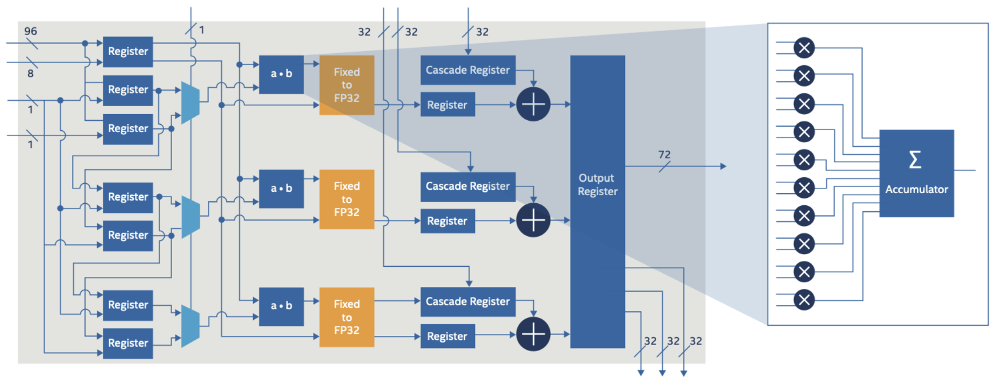


Figure 4 AI Tensor Block High-Level Diagram [22]

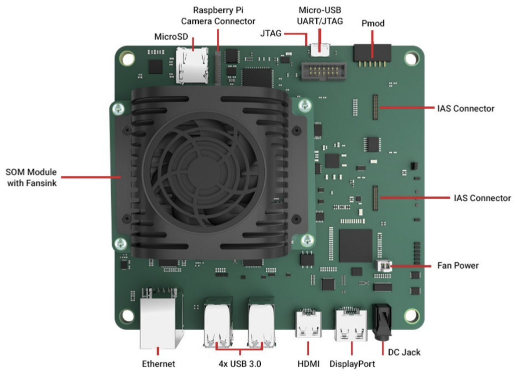


Figure 5 Kria KV260 Vision AI Starter Kit [23]

model as input and producing instructions for a DPU in the output. This approach requires the optimization of a pre-existing neural network, followed by its quantization, and ultimately the establishment of dedicated hardware resources in the PL to expedite the processing of AI and machine learning workloads.

In summary, the Kria KV260 provides a variety of options for deploying AI and machine learning models, ranging from basic model loading from a Model Zoo to more intricate methods such as `Very High Speed Integrated Circuit Hardware Description Language` (VHDL)-level implementation, Docker-based applications deployment, and development kit acceleration through the Vitis AI workflow.

After thorough analysis of the model deployment process using either of the aforementioned methods, it is evident that hardware acceleration in these methods is reliant on the utilization of DPUs [30] [26] [18]. To effectively operate any AI model on a Xilinx MPSoC, the recommended approach is to comply with the Vitis AI development kit tool pipeline, depicted in figure 6. This process entails various crucial steps to guarantee peak performance on the FPGA hardware. First, training an AI model on a standard PC using popular frameworks such as TensorFlow or PyTorch is a standard practice. This enables the development, fine-tuning, and validation of models without the limitations of FPGA hardware, facilitating experimentation and iteration. After completing model training, the subsequent step is to stabilize its state through freezing. This process involves converting the model into a static format that is suitable for deployment on a MPSoC. Deploying on a MPSoC requires models with fixed architectures. After freezing the model, it is crucial to optimize the network by leveraging Xilinx's AI optimizer to prune it. As previously mentioned, pruning entails removing unnecessary connections, layers, or weights from the model. This optimization step streamlines the model for deployment on the MPSoC by lessening computational complexity and speeding up inference [31].

Following the pruning process, the next step is PTQ, which requires transforming the pruned model into a representative format using `8-bit integers (INT8)` [31] [47]. The AI Quantizer, seen in the bottom-right of Figure 6, accepts a FP model and conducts pre-processing (such as folding batch norms and removing nodes that are not required for inference) before quantizing the weights/biases and activations to the specified bit width. Before quantization, the model is inspected by an inspector step. The output of the inspector displays partition information, identifying which operators can execute on either the DPU or CPU device. Typically, the DPU in the Programmable Logic provides faster processing than the CPU in the Processing System. Therefore, it is advisable to execute as many operations as possible on DPUs. Upon completion of partitioning, the resulting partition will contain messages identifying which operators cannot be run on DPUs. To accomplish the task at hand, a calibration image dataset is required as input for the PTQ process, which involves conducting multiple rounds of inference to calibrate activations, enhance quantized model accuracy, and collect activation statistics. The Quantizer is typically effective when given 100-1000 calibration images. An unlabeled data set is sufficient because backpropagation [32] is unnecessary. Backpropagation is unnecessary because the compression of the model is achieved without any alterations to the initial training methodology, model structures, and parameters. After calibration, the quantized model is converted into a format that can be deployed on a DPU, which is compatible with the input data format of the AI Compiler.

The last and vital stage requires feeding the quantized model into the AI compiler, which is tailored to a specific version of the Xilinx DPU [31]. The AI Compiler serves as a collective interface for a range of compilers aimed at boosting neural network computations throughout different DPUs. Each compiler optimizes a network model into a specialized DPU instruction series. Figure 6 on the lower left portion presents a simplified outline of the AI Compiler framework. The framework analyzes the topology of the quantized and optimized input model and produces an internal computation graph as an `intermediate representation` (IR) that matches the representations of control flow and data flow. Then, the framework executes a variety of optimizations, including combining computation nodes, as in the case of combining batch norm with a presiding convolution, and efficient instruction scheduling using inherent parallelism or data reuse. Finally, the `Xilinx Intermediate Representation` (XIR)-based compiler processes the quantized model as input. The input models are initially converted to the XIR format, serving as the basis for subsequent processes. This phase involves eliminating most framework variations and transferring to a unified representation in XIR. Following this stage, the graph undergoes various optimizations and is divided into multiple subgraphs based on whether the operation can be performed on the DPU. Architecture-aware optimizations are applied to each subgraph as necessary. The compiler generates the instruction stream and attaches it to the DPU subgraph. Then, the optimized graph, including essential information and `Vitis AI Runtime` (VART) instructions, is serialized into a compiled xmodel-file, which can be copied to the target MPSoC board via the Linux scp command. After installing the VART environment on the target board, the model can be run by simply calling it like a shell script [33]. By following this comprehensive procedure, one can transition effortlessly from a trained AI model to a streamlined deployment on a Xilinx MPSoC outfitted with a DPU.

In addition to utilizing a vendor's acceleration method, it is possible to use additional VHDL modules to further accelerate the AI-based task. Figure 7 depicts an image pre-processor module that optimizes incoming image data prior to the inference process.

IV. Conclusion

In summary, this paper presents a comprehensive overview of AI acceleration, covering centralized and edge spaces, with a particular emphasis on FPGA technology. The study of hardware acceleration methods, based on hands-on experimentation and comparison of various acceleration strategies, indicates that FPGAs are capable of effectively accelerating AI applications in hardware, yet they are not without limitations. As demonstrated by this study, Intel

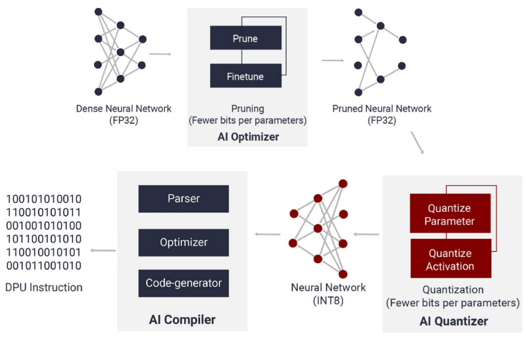


Figure 6 AI Example workflow of IP core-based acceleration [18]

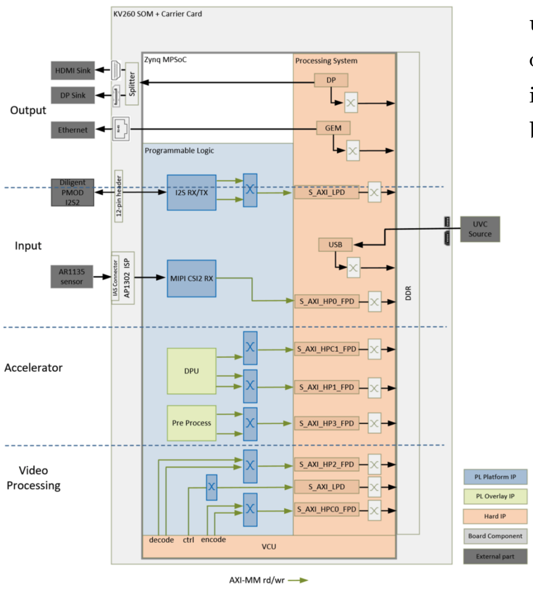


Figure 7 High-level diagram of the vision AI hardware architecture [26]

is currently the sole vendor implementing recently developed, specialized hardware to boost AI tasks, while other vendors opt to reconfigure pre-existing hardware in a more favorable manner. Since FPGA-based MPSoCs have only been specialized for AI acceleration tasks since mid-2020, it remains to be seen whether other vendors invent similar, or even improved, versions of Intel's AI Tensor Blocks. Regardless, both methods - confugurational acceleration and specialized hardware-based acceleration - are certainly a step in the right direction. Although it is possible to run a trained model on a PC using a GPU as an accelerator without any additional steps beyond the training process, creating an executable model on a MPSoC requires additional steps, such as utilizing the previously mentioned Vitis AI pipeline. Nevertheless, deploying this approach ensures a high-performing model on the edge, with no significant decrease in speed or accuracy when compared to executing the same model on a GPU.

The investigation of hardware acceleration techniques, hands-on experiments, and comparisons of various hardware acceleration methods contribute to a comprehensive understanding of the present AI hardware acceleration landscape. As the industry evolves, it is essential to evaluate acceleration technologies' capabilities and drawbacks to fuel innovation and satisfy the increasing AI application requirements in the business and personal arenas.

References

[1] Alexy Thomas, "How AI and automation make data centers greener and more sustainable" https://www.ey.com/en_in/technology/how-ai-and-automation-make-data-centers-greener-and-more-sustainable#:~:text=Traditional%20data%20centers%20monitor%20and,centers%20and%20thus%20minimize%20downtimes. accessed on 22.02.2023 at 10:18 o'clock

[2] Wang, Ting and Xia, Yu and Muppala, Jogesh and Hamdi, Mounir, "Achieving Energy Efficiency in Data Centers Using an Artificial Intelligence Abstraction Model", doi=10.1109/TCC.2015.2511720, https://ieeexplore.ieee.org/abstract/document/7364218, accessed on 22.02.2023 at 10:30 o'clock.

[3] Synopsys, "What is an AI Accelerator?", https://www.synopsys.com/ai/what-is-an-ai-accelerator.html, accessed on 30.01.2023 at 16:04 o'clock.

[4] electrek, "Tesla unveils new Dojo supercomputer so powerful it tripped the power grid", https://electrek.co/2022/10/01/tesla-dojo-supercomputer-tripped-power-grid/, accessed on 22.02.2023 at 9:58 o'clock.

[5] Nvidia, "Floating Point and IEEE 754 Compliance for NVIDIA GPUs", https://docs.nvidia.com/cuda/floating-point/index.html#references___1, accessed on 22.02.2023 at 10:30 o'clock.

[6] AMD, "FLOATING-POINT ARITHMETIC IN AMD PROCESSORS", https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiE_t_Q9_n-AhXSgf0HHav9BGQQFnoECAgQAQ&url=https%3A%2F%2Fcommunity.amd.com%2Fsdtpp67534%2Fattachments%2Fsdtpp67534%2Fopencl-discussions%2F6370%2F1%2FFLOATING-POINT%2520ARITHMETIC%

2520IN%2520AMD%2520PROCESSORS.pdf&usg=AOvVaw2fXgb4IrkptHsS1dy0vmGN, accessed on 22.02.2023 at 10:53 o'clock.

[7] Title: IEEE Standard for Floating-Point Arithmetic, Author: IEEE Computer Society, Year: 2019.

[8] ANY DIFFERENCE BETWEEN, "Difference Between ALU and FPU", https://anydifferencebetween.com/difference-between-alu-and-fpu/, accessed on 23.02.2023 at 10:53 o'clock.

[9] Erick Roch Moraguez, "What is the FPU (floating point unit): how does it work and what is it for?", https://lovtechnology.com/en/que-es-la-fpu-unidad-de-coma-flotante-como-funciona-y-para-que-sirve/, accessed on 23.02.2023 at 12:01 o'clock.

[10] J. Deng et al., "5G and AI Integrated High Performance Mobile SoC Process-Design Co-Development and Production with 7nm EUV FinFET Technology", 2020, IEEE Symposium on VLSI Technology, Honolulu, HI, USA, 2020, pp. 1-2, doi: 10.1109/VLSITechnology18217.2020.9265074.

[11] W. Kim and I. Jung, "Smart Parking Lot Based on Edge Cluster Computing for Full Self-Driving Vehicles", in IEEE Access, vol. 10, pp. 115271-115281, 2022, doi: 10.1109/ACCESS.2022.3208356.

[12] C. Chansri, J. Srinonchat, E. G. Lim and K. L. Man, "Low Cost Hand Gesture Control in Complex Environment Using Raspberry Pi", 2019 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2019, pp. 186-187, doi: 10.1109/ISOCC47750.2019.9027669.

[13] M. -H. Horng and T. -W. Jiang, "The artificial bee colony algorithm for vector quantization in image compression", 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, Shenzhen, China, 2011, pp. 319-323, doi: 10.1109/ICBNMT.2011.6155949.

[14] K. Kollek, M. A. Aguilar, M. Braun and A. Kummert, "Improving Neural Network Architecture Compression by Multi-Grain Pruning", 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 2021, pp. 6-13, doi: 10.1109/PIC53636.2021.9687071.

[15] Apple Inc, "Deploying Transformers on the Apple Neural Engine", Speech and Natural Language Processing Article, June 2022, https://machinelearning.apple.com/research/neural-engine-transformers accessed on 29.10.2023 at 17:24 o'clock.

[16] Microchip, "Microchip Reveals Software Development Kit and Neural Network IP for Easily Creating Low-Power FPGA Smart Embedded Vision Solutions", https://www.microchip.com/en-us/about/news-releases/products/mchp-reveals-vectorbloxsdk-and-ip-for-fpga-smart-embedded-vision, accessed on 29.10.2023 at 17:58 o'clock.

[17] Intel, "Intel has just announced its first AI-optimized FPGA – the Intel® Stratix® 10 NX FPGA – to address the rapid increase in AI model complexity", https://community.intel.com/t5/Blogs/Products-and-Solutions/FPGA/Intel-has-just-announced-its-first-AI-optimized-FPGA-the-Intel/post/1332696, accessed on 29.10.2023 at 18:08 o'clock.

[18] AMD, "Vitis AI User Guide (UG1414)", https://docs.xilinx.com/r/1.2-English/ug1414-vitis-ai/Revision-History, accessed on 29.10.2023 at 18:09 o'clock.

[19] AMD, "DPU for Convolutional Neural Network", https://www.xilinx.com/products/intellectual-property/dpu.html#overview. accessed on 30.10.2023 at 10:25 o'clock.

[20] AMD, "DPUCZDX8G for Zynq UltraScale+ MPSoCs Product Guide (PG338)", https://docs.xilinx.com/r/en-US/pg338-dpu/Introduction?tocId=3xsG16y_QFTWvAJKHbisEw, accessed on 30.10.2023 at 10:30 o'clock.

[21] Microchip, "VectorBlox™ Accelerator Software Development Kit for PolarFire® FPGAs", https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/vectorblox, accessed on 30.10.2023 at 14:03 o'clock.

[22] Intel, "Intel Stratix 10 NX FPGA", https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/a1137206-stratix-10-nx-product-brief.pdf accessed on 30.10.2023 at 14:34 o'clock.

[23] AMD, "Getting Started with Kria KV260 Vision AI Starter Kit", https://www.xilinx.com/products/som/kria/kv260-vision-starter-kit/kv260-getting-started/getting-started.html, accessed on 20.10.2023 at 12:35 o'clock.

[24] Intel, "Intel® FPGA AI Suite: IP Reference Manual", https://www.intel.com/content/www/us/en/docs/programmable/772497/2023-2-1/version-2023-2-release-notes.html, accessed on 06.11.2023 at 9:30 o'clock.

[25] Intel, "Intel® FPGAs and SoCs with Intel® FPGA AI Suite and OpenVINO Toolkit Drive Embedded/Edge AI/Machine Learning Applications", https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2022-12/ai-fpga-whitepaper.pdf, accessed on 06.11.2023 at 9:53 o'clock.

[26] AMD, "Hardware Architecture of the Platform", https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/docs/hw_arch_platform.html, accessed on 06.11.2023 at 12:33 o'clock.

[27] AMD, "Accelerated Applications for Adaptive Platforms", https://www.xilinx.com/products/app-store/kria.html#kria-apps, accessed on 06.11.2023 at 14:22 o'clock.

[28] Xilinx, "Vitis AI Model Zoo", https://github.com/Xilinx/Vitis-AI/tree/master/model_zoo, accessed on 06.11.2023 at 14:24 o'clock.

[29] AMD, "Vitis High-Level Synthesis User Guide (UG1399)", https://docs.xilinx.com/r/en-US/ug1399-vitis-hls, accessed on 06.11.2023 at 14:26 o'clock.

[30] Xilinx, "OFA depthwise resnet50", https://github.com/Xilinx/Vitis-AI/blob/master/model_zoo/model-list/pt_OFA-depthwise-

resnet50_3.5/model_info.md, accessed on 06.11.2023 at 15:26 o'clock.

[31] AMD, "Vitis AI User Guide (UG1414)", https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview, accessed on 08.11.2023 at 12:02 o'clock.

[32] Neural Networks and Deep learning, Michael Nielsen, "How the backpropagation algorithm works", http://neuralnetworksanddeeplearning.com/chap2.html, accessed on 08.11.2023 at 14:04 o'clock.

[33] Xilinx, "Vitis AI v2.0", https://github.com/Xilinx/Vitis-AI/blob/2.0/setup/mpsoc/VART/README.md#step3-run-the-vitis-ai-examples, accessed on 08.11.2023 at 14:45 o'clock.

[34] Andy Morris, "6 Trends Driving the AI Everywhere Boom", https://community.intel.com/t5/Blogs/Tech-Innovation/Artificial-Intelligence-AI/6-Trends-Driving-the-AI-Everywhere-Boom/post/1459069#:~:text=AI%20is%20being%20used%20in,lives%20easier%20and%20more%20convenient., accessed on 08.11.2023 15:30 o'clock.

[35] Jensen Huang, "Accelerating AI with GPUs: A New Computing Model", https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/, accessed on 08.11.2023 at 15:25 o'clock.

[36] RD. Pai, "Developing FPGA based edge-AI Solutions", https://www.linkedin.com/pulse/developing-fpga-based-edge-ai-solutions-rd-pai, accessed on 08.11.2023 at 15:40 o'clock.

[37] Faizan Fahim, "CPU vs GPU vs TPU: Understanding the Difference Between Them", https://serverguy.com/comparison/cpu-vs-gpu-vs-tpu/, accessed on 08.11.2023 at 15:50 o'clock.

[38] Google, "Introduction to Cloud TPU", https://cloud.google.com/tpu/docs/intro-to-tpu, accessed on 08.11.2023 at 15:55 o'clock.

[39] Nvida, "CUDA C++ Programming Guide", https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, accessed on 08.11.2023 at 16:01 o'clock.

[40] Sergio Marchese, "AI Chips Must Get The Floating-Point Math Right", https://semiengineering.com/artificial-intelligence-chips-must-get-the-floating-point-math-right/, accessed on 08.11.2023 at 16:08 o'clock.

[41] Deepgram, "Tensor Processing Unit (TPU)", https://deepgram.com/ai-glossary/tensor-processing-unit-tpu, accessed on 08.11.2023 at 16:25 o'clock.

[42] DigiKey, "Use the FPGA Fast Path to Building High-Performance Power-Efficient Edge AI Applications", https://www.digikey.de/en/articles/use-the-fpga-fast-path-to-building-edge-ai-applications, accessed on 08.11.2023 at 16:58 o'clock.

[43] run.ai, "FPGA for Deep Learning", https://www.run.ai/guides/gpu-deep-learning/fpga-for-deep-learning, accessed on 08.11.2023 at 7:02 o'clock.

[44] Intel, "FPGA vs. GPU for Deep Learning", https://www.intel.com/content/www/us/en/artificial-intelligence/programmable/fpga-gpu.html, accessed on 08.11.2023 at 17:10 o'clock.

[45] TS2, Marcin Frackiewicz, "AI FPGA: The Future of Machine Learning and Deep Learning Acceleration", https://ts2.space/en/ai-fpga-the-future-of-machine-learning-and-deep-learning-acceleration/#, accessed on 08.11.2023 at 19:20 o'clock.

[46] Association for Computing Machinery, Aman Arora, Moinak Ghosh, Samidh Mehta, Vaughn Betz, and Lizy K. John, "Tensor Slices: FPGA Building Blocks For The Deep Learning Era", https://dl.acm.org/doi/10.1145/3529650, accessed on 12.11.2023 at 11:54 o'clock.

[47] Zhihang Yuan at al, "Benchmarking the Reliability of Post-training Quantization: a Particular Focus on Worst-case Performance", https://arxiv.org/pdf/2303.13003.pdf, accessed on 12.11.2023 at 13:21 o'clock.

Improving Traffic Flow Analysis: A Novel Approach to Sound-Based Traffic Density Classification

Leon Schex
TH-Köln
Köln, Deutschland
leon_nicolas.schex@smail.th-koeln.de

Luca Schex
TH-Köln
Köln, Deutschland
luca_jannis.schex@smail.th-koeln.de

***Abstract*— This work presents an innovative approach to classify traffic density on highways using audio data. In view of the increasing challenges in urban traffic systems, the work aims to classify the states "free", "full" and "congestion". The methodology includes the recording of traffic noise with several microphones with different directional characteristics. By applying several data pre-processing steps, such as the use of Mel Frequency Cepstral Coefficients (MFCC) for feature extraction, a robust data foundation is created. The work focuses on dimension reduction using Principal Component Analysis (PCA), which serves as an essential step in optimizing the Random Forest classifier. PCA helps to increase the efficiency and accuracy of the model by transforming correlated variables into a set of linearly uncorrelated variables through dimension reduction. Additionally, hyperparameter tuning is conducted to determine the optimal parameters for the classification model. The evaluation of the model shows a high precision and robustness with an accuracy of 98.34 % which offers potential for future development of advanced traffic monitoring systems.**

Keywords*— Traffic Density Classification, Machine Learning, Mel Frequency Cepstral Coefficients (MFCC), Principal Component Analysis (PCA), Acoustic Traffic Analysis, Traffic Monitoring Systems, Signal-to-Noise Improvement, Sound Pattern Recognition, Audio-Based Monitoring Technologies.

I. Introduction.

The increasing complexity and density of traffic in urban and highway systems represents one of the most challenging issues for modern traffic management strategies. A key component in overcoming this challenge is the classification of traffic density, which forms the basis for adaptive control systems and efficient traffic flow analysis. Against this background, the aim of this research project is to explore the technical possibilities of using microphones instead of traditional camera monitoring systems for traffic density classification. The focus of this work is the development of a model that is trained to classify basic traffic density states – free, full and congestion – from a data set created specifically for this purpose. The "free" state means that there are vehicles on the road, but the traffic flow is not affected in any way. The "full" state means that traffic is flowing, but the traffic is close to the critical density. The "congestion" state means that the traffic flow is severely impaired. The vehicles have to stop temporarily on the road and are moving very slowly. (See Fig. 1) The classification of the recorded audio samples was based on visual observation of the lanes during the recording period.

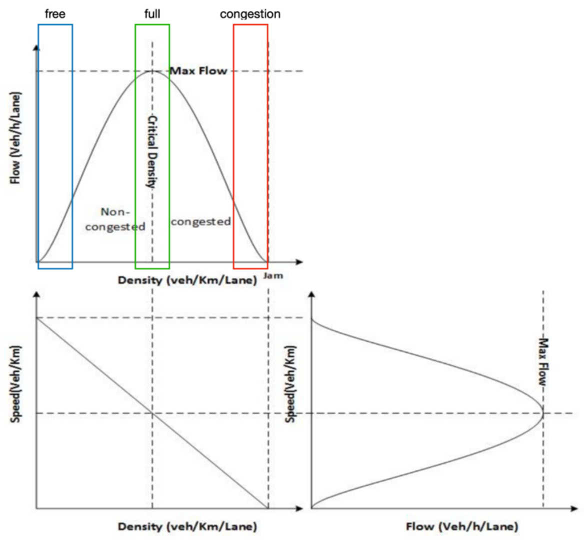


Fig. 1. Fundamental diagram of traffic flow [1]

Signal processing techniques and machine learning are used to extract patterns and information from the acoustic signals. The motivation for this approach is based on several advantages that microphones offer in this context.

On the one hand, the use of audio surveillance addresses important privacy concerns that are becoming increasingly important in public perception and legal regulation. By focusing on acoustic rather than visual data, the privacy of road users is better protected as no identifiable images are captured. Secondly, microphones are simpler to install than cameras as they can be easily positioned close to the ground at the edge of the road. Cameras require a clear line of sight, which can often only be achieved by positioning them on bridges or masts. This can reduce installation costs and make microphones an economically attractive alternative to extensive camera surveillance systems. Microphones also require less maintenance in terms of sensitivity to dust, as their functionality remains largely unaffected, unlike cameras whose vision can be impaired by dust particles on the lens. The independence from visual conditions also enables continuous and reliable data acquisition.

The methodological approach of this work includes the recording and pre-processing of audio data. With the help of several microphones with different directional characteristics, traffic sounds are recorded to capture a comprehensive spectrum of traffic sounds. This audio data is then segmented, labeled, and prepared for analysis. The Python programming language and various machine learning libraries and frameworks are used for this purpose. A special focus lies on the use of Mel Frequency Cepstral Coefficients (MFCC) and Mel spectra and other spectral techniques to analyze and extract the characteristic patterns of traffic noise.

After processing the data, feature selection and dimension reduction follows using Principal Component Analysis (PCA) to optimize efficiency and model performance. A Random Forest classifier is used for classification. In addition, fine-tuning of the hyperparameters of the model helps to increase the accuracy and generalization capability of the classifier. The evaluation of the model shows promising results based on the analysis of the confusion matrix and various validation techniques.

By using audio data, techniques of data processing and machine learning, this work contributes to the improvement of traffic management systems and at the same time opens new avenues for future research in this field.

II. Data acquisition and creation of a data set

Four different microphones were used to create the audio data. Two microphones from Sennheiser were used: the ME 64 with a cardioid polar characteristic and the ME 67, which has a directional characteristic. Both were operated with Sennheiser K6 power adapters and aimed directly at the highway, as shown in Fig. 2. In addition, two ZOOM XYH-6 unidirectional microphones were used, which also have a cardioid characteristic. These are positioned offset by 90° in the horizontal axis and aligned at an angle of 45° to the highway. In the vertical axis, they were pointed directly at the highway. A ZOOM H6 was used as the recorder. The recording was made at a sampling rate of 44.2 kHz. The quantization was set to 24 bits.

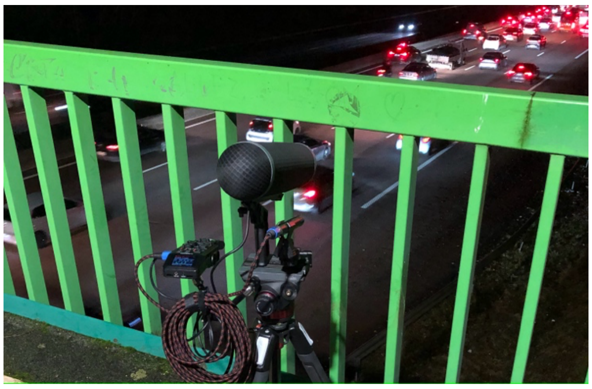


Fig. 2. Microphone set-up facing the A4, 93160 Noisy-le-Grand, France

The use of different microphone types and their specific positioning made it possible to capture a wide range of sound patterns. This helps to increase the diversity of the data set and the number of data points. In addition, the performance of the different microphones can later be compared by analyzing the misclassified samples. In this way, a test environment that is as realistic as possible can be ensured to achieve a model that is robust and reliable. Nevertheless, all microphones are positioned on the side of the highway that should be recorded. We avoided using omnidirectional microphones, as the primary aim was to record the traffic density on the specific side of the road and minimize background noise.

During data collection, environmental factors remained constant. The recording times for each traffic category were specifically selected to ensure that the traffic flows on the highway corresponded to the defined classes. In addition, a constant traffic density was observed during the audio recordings to ensure consistent data quality.

As a preprocessing step for further analysis and modeling of the audio data, the audio files, which are available in WAV audio format, are segmented into 4-second samples. The unprocessed data set therefore contains 5568 data points. In addition, the segments, which were created from the audio recordings of the respective microphones, have a 1-second offset to each other. This means that the segments of the respective microphones differ not only in terms of the different sound patterns, but also in terms of the temporal information.

The decision to choose a sample length of 4-seconds is based on tests of different sample lengths applied to a Random Forest classifier. The exact data preprocessing steps, as well as the selection of the algorithm, are explained in more detail in sections 4 and 5. Table 1 shows the test results over a 5-fold cross validation as Accuracy – i.e. the percentage of correct predictions in relation to the total number of predictions.

TABLE I. Accuracy for different segment legth[a]

Segment Length	2 sec.	3 sec.	4 sec.	8 sec.	12 sec.	16 sec.
Accuracy in %	96.89	96.74	96.96	96.57	96.92	97.10

[a.] default Random Forest, MFCCs as input features.

Table 1. shows that a sample length of 4-seconds is a good choice for a good performance of the algorithm. While the information relevant for the classification in the recorded traffic noise is quite short and repetitive, a reduction of the sample length leads to decreasing results. Similarly, increasing the sample length does not necessarily lead to a significant improvement in the results, as shown in Table 1. Therefore, the segmentation of the audio files into 4-second samples offers a good balance between the accuracy of the classification and the ability to react to changing situations.

III. Discover and visualize the data to gain insights

To analyze the data, histograms are created from all averaged data points per class. These show the frequency components calculated by the Short-time Fourier Transformation (STFT) in relation to time with the loudness level in dB (Fig.3.a) and the Mel spectrum in relation to time with the loudness level in dB (Fig.3.b).

In comparison to the STFT spectrum, the Mel spectrum provides a representation format for audio signals based on human auditory perception. Since human hearing does not respond linearly to frequencies, some areas of the frequency spectrum are more sensitive than others. The Mel spectrum provides a better representation of the acoustic properties that are mainly relevant for speech recognition. In the Mel spectrum, the frequency axis is scaled logarithmically to better represent the differences in the perception of low and high frequencies. Low frequencies are displayed with a higher resolution than high frequencies (Fig. 4). This different frequency resolution is also visible when comparing Fig. 3.a with Fig. 3.b. The Mel spectrum also appears to be useful in application areas outside of speech recognition.

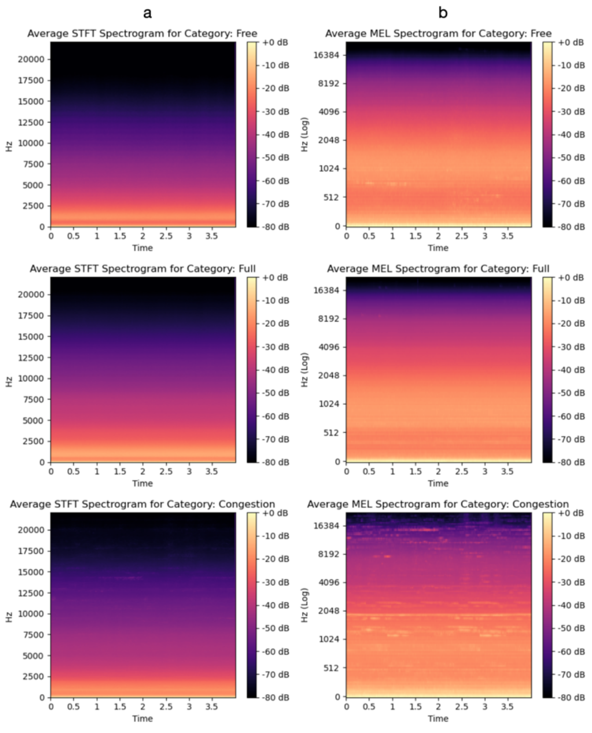


Fig. 3. Average STFT- and Mel-Spectrograms of each class

When looking at the individual histograms in Fig. 3, the spectrum is significantly more unstable in the congested state compared to the free or full state. It is also noticeable that the higher frequency ranges, especially from 14 kHz, are slightly more dominant in the congested state. This characteristic could indicate a more intense presence of specific acoustic elements in this scenario, for example the more frequent occurrence of horn tones or the sound of the brakes grinding. In addition, Fig. 3.b shows a slight increase in the amplitude values at a frequency of around 750 Hz in free traffic.

Overall, the three classes show a clear similarity, which was to be expected given the averaging applied over all data points of each class.

IV. Prepare the Data for Machine Learning Algorithms

A. *Balance the data*

Despite an overall balanced class distribution, the number of data points per class is equalized by reducing the data points of the overrepresented classes. This ensures that each class is equally represented in the data set. This can avoid distortions in model performance, as the model may tend to learn the more frequent class better and neglect the rare classes [2].

B. *Factorize the labels*

Since most machine learning algorithms prefer numbers [2], the different classes, i.e. their labels, which are previously stored as strings, are factorized to numerical values from 0-2.

C. *MFCC - Mel Frequency Cepstral Coefficients*

For correct classification of audio samples, it is important that the statistical data show a small variation within the respective class and differ clearly from the statistical data of other classes. In general, spectral information is extracted from the audio signal to use this data as features.

This is done using a Short-Time Fourier Transform (STFT). It is defined as follows:

$$X(\tau,\omega) = \int_{-\infty}^{\infty} x(t) \cdot \omega(t-\tau) \cdot e^{-j\omega t}\, dt \qquad (1)$$

Where $x(t)$ is the input signal in the time domain, with t as the time variable. $\omega(t-\tau)$ is the Hanning window applied to the signal to smooth the ends of the considered signal section to minimize artificial discontinuities [3]. τ is the shift of the window over the signal. [4]

The Mel spectrum is commonly used in speech recognition, music processing and other audio signal processing applications and can be used as input for various machine learning techniques to identify patterns in the audio data. It is generated by logarithmizing the intensity of the frequencies and normalizing them to a range of (normalized_mel $\in$ [-1,1]). This process involves the application of a non-linear Mel filter bank to generate the Mel spectrum from the linear spectrum. As shown in Fig. 4, the Mel filter bank consists of a series of triangular bandpass filters that overlap and are logarithmically distributed across the frequency band. Followed by a conversion of the intensities into Mel decibels (mel_db). [5] [6]

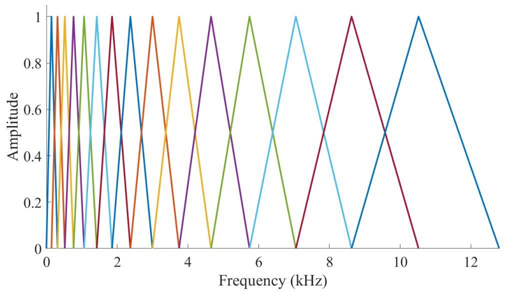


Fig. 4. Mel filter bank [7]

Another common example as input for various machine learning methods are the Mel Frequency Cepstral Coefficients (MFCC). A cepstrum is the inverse Fourier transform of a logarithmized spectrum. The calculation of the MFCC involves the use of the "Mel spectrum". After the Mel spectrum has been generated, the Discrete Cosine Transform (DCT) is applied to it. This transformation is used to reduce the correlation between the Mel spectral bands and compress the most important information into a smaller number of coefficients. Several MFCCs together form an MFC (MEL-Frequency Cepstrum). [5] [6]

The sequential results of this process are visualized in Figure 5 using a single input sample.

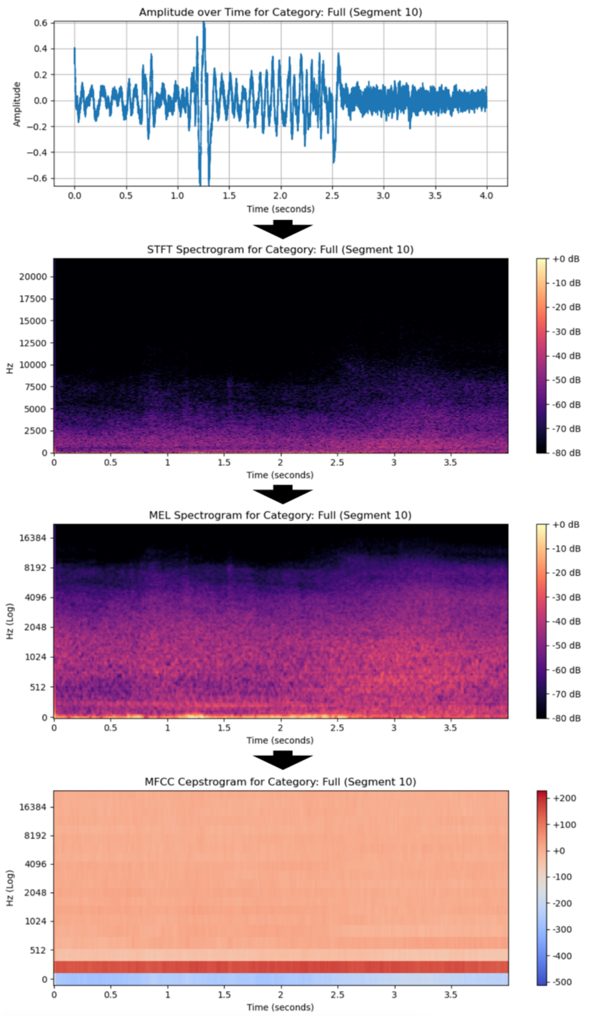


Fig. 5. Visual transformation process based on a single segment

The Python library librosa is used to extract spectral information from audio signals to use as features for audio classification. Librosa offers functions for processing and analyzing audio signals and for calculating features such as MFCC, MEL spectrograms and linear spectrograms. The MFCCs, the MEL spectrum and the linear spectrum are calculated for each audio sample in the data set. When comparing the shape of these features per segment, it becomes noticeable that the MFC has only 20 coefficients per 345 time ranges, while the MEL spectrum has 128 coefficients per 345 time ranges and the linear spectrum has 1025 coefficients per 345 time ranges.

The compact representation of features as MFCCs brings various advantages for audio classification. It captures important information about the spectral shape of audio signals. However, it concentrates only on the most important frequency components and thus removes redundancy or less relevant information. In addition to reducing the amount of data, this can improve the performance and accuracy of classification algorithms. This is also reflected in the test results in Table 2:

TABLE II. ACCURACY DIFFERENTS MFCC, MEL, STFT[b]

	MFCC	MEL	STFT
Accuracy in %	96.96	97.51	95.28

[b] default Random Forest.

The accuracy over a 5-fold cross validation on the training data set when using the entire frequency spectrum of the STFT as input features is about 1.7 % worse compared to the use of MFCCs. Overall, the best results were achieved when using the Mel spectrum. However, the MFCCs are used as features in this use case because they performed best after applying the PCA.

For completeness, the 5-fold cross validation was also performed on a training data set exclusively with data from a single microphone (ME 67). The accuracy is 96.24 % with MFCCs as input features.

D. *Train-Test-Split*

The data is split into training and test sets using the "`train_test_split`" function from Scikit-Learn. As the data set is relatively small, the split is performed with a test size of 20 % [2]. The parameter "`random_state`" ensures the reproducibility of the split [2], and "`stratify`" ensures that the distribution of classes in the test and training set is proportional to that in the entire data set. The test set is then used for validation. The training set is split again by the cross-validation used to test and optimize the model.

E. *Feature Scaling*

As all features are available as MFCCs and therefore already use the same scaling, additional feature scaling is not required for the accuracy of the model. This makes the use of a standard scaler obsolete, as no significant improvement in model accuracy is to be expected.

F. *Reshape MFCCs from 2D- to 1D-Array*

To utilize the scikit-learn's Random Forest algorithm, the original 3D data structure (consisting of segments of 2D MFCC arrays) needs to be transformed into a 2D format. This is achieved by reshaping the 2D MFCCs of each segment into 1D arrays. The outcome is a 2D array where each segment is represented by a 1D array, containing all MFCCs in a flat, one-dimensional structure, making it compatible with the Random Forest algorithm.

With a reshape of the MFCCs in the row direction, an accuracy of 96.75 % is achieved with a 5-fold cross validation on the training data set. When reshaping the MFCCs in the column direction, an accuracy of 96.96 % is achieved. Based on this marginally higher result, we decided to continue with the column direction as the preferred data structure.

G. *Dimensional reduction, principal component analysis (PCA)*

Principal component analysis is a statistical method for reducing the dimensionality of data. It transforms the data into a new space where the axes (principal components) are orthogonal to each other and represent the variance of the data in descending order. The principal components are determined by calculating the eigenvectors and eigenvalues of the covariance matrix of the data. By projecting the original data onto these new axes, selecting the principal components with the largest variance, the dimensionality of the data set is reduced without losing significant information. [8]

In general, PCA is used to speed up the training time by reducing the input features without having to accept a significant degradation in prediction quality. In certain cases, however, PCA can also help to remove redundancy and reduce noise by transforming correlated variables into a set of linearly uncorrelated variables through dimension reduction. This transformation can result in improved prediction quality. [2]

The PCA class from Scikit-Learn is used to implement the dimension reduction. The visualization of the cumulative explained variance as a function of the number of principal components in Fig. 6 allows a visual evaluation of the data reduction efficiency. The cumulative explained variance represents the percentage of the total variance of the original data that is represented by a given number of principal components; a higher value indicates a better representation of the data [2]. Fig. 6 shows a turning point in the curve, from which the increase in explained variance due to additional dimensions is only marginal. This shows that a reduction to 28 dimensions preserves the majority of the data variance.

A comparison of the model accuracy using a 5-fold cross validation applied to the training dataset shows promising results: While the model without PCA achieves an accuracy of 96.96 %, this increases to 98.20 % when PCA is applied. This improvement in accuracy, although marginal, is remarkable given the fact that the number of features is reduced from 6900 to only 28.

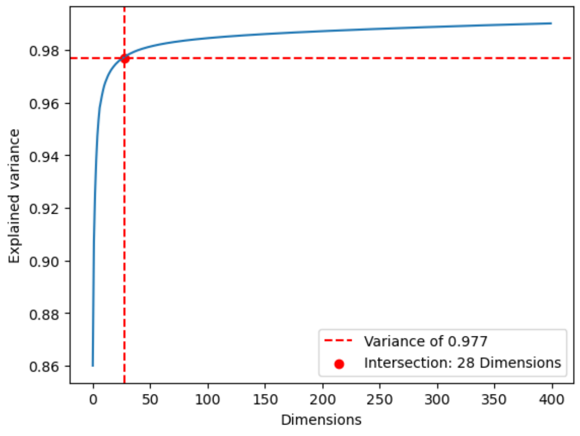


Fig. 6. Explained variance as a function of the number of dimensions

As a result, applying PCA to the MFCCs reduces the data size by around 99.6 % of the original size and increases the accuracy of the classifier by around 1.2 %. This is due to the removal of redundancies and noise reduction.

V. Select and Train a Model

A. *Random Forest*

When selecting the appropriate model for traffic sound classification, various machine learning approaches are under consideration, including neural networks. Although neural networks (NN) are known for their ability to recognize and learn complex patterns, a conscious decision was made not to use this approach. The main reason for this is the good performance of the Random Forest classifier, which already demonstrates an impressive accuracy of 98.46 %. As well as it can efficiently handle the existing data size and structure without the need for the deep architecture and computationally intensive training requirements of an NN. [2]

The Random Forest classifier from Scikit-Learn is used for classification. The Random Forest algorithm is a suitable method for classifying sounds due to its ability to recognize complex and non-linear relationships. It is based on an ensemble technique where multiple decision trees are combined to achieve a robust classification. In addition, it has a built-in feature evaluation function to identify relevant information and patterns in the audio data. [2]

Due to the relatively small size of the training dataset with around 4344 data points, the Random Forest classifier proves to be advantageous as it is less sensitive to overfitting. If the data set is small, there is a risk that a model with high complexity, such as a single decision tree, will overfit the data and generalize poorly to new data. By using a random selection of features and training examples for each decision tree in the ensemble, the correlation between the trees is reduced. As a result, the classifier learns different aspects of the data and can react more robustly to new data, even with a small data set. [2]

B. *Hyperparameter Tuning*

To improve the performance and generalization capability of the model, the hyperparameters of the model are optimized. For this purpose, the GridSearchCV function of Scikit-Learn is used, which systematically tests different combinations of hyperparameters of a model to find the best configuration. Different combinations of the hyperparameters `n_estimators` and `max_depth` of the Random Forest classifier were tested. The respective accuracy of the model was evaluated using 5-fold cross validation on the training data and displayed in a diagram (Fig. 7).

The hyperparameter `n_estimators` specify how many decision trees are to be used in the ensemble. A higher number of estimators generally improves the performance of the model but increases the computational complexity of the model. The hyperparameter `max_depth` specifies the maximum depth of a decision tree and controls the complexity of the model. A deeper tree can potentially capture more information and learn more complex patterns. However, if the tree depth is too deep, there is also a risk of overfitting.

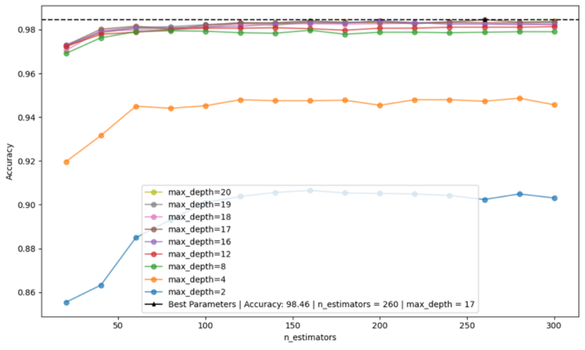


Fig. 7. Accuracy for different max_depth and n_estimators

Fig. 7 shows that only marginal differences in accuracy can be determined above a tree depth of 8. In addition, above a tree depth of 16, no significant change in model accuracy appears to occur. This could be since the maximum depth of a balanced tree for 5430 data points is about 12, based on the formula:

$$log_2(i) = d \qquad (2)$$

Where i is the number of data points and d is the maximum depth of a balanced tree.

Regarding the number of trees in the ensemble, there are only marginal effects on the accuracy above a number of 40 trees. This indicates that the model is relatively stable against changes of this hyperparameter. For use in a resource-efficient environment, a combination of `n_estimators` = 40 and `max_depth` = 8 would be well suited, as an accuracy of around 97.50 % is already achieved here, which represents an efficient balance between performance and computational effort.

However, the optimal results are achieved with a tree depth of `max_depth` = 17 and `n_estimators` = 260 trees in the ensemble, with an obtained accuracy of 98.46 %. Furthermore, Fig. 7 shows that the accuracy decreases slightly with a number of trees above 260 and does not improve further, which could be an indication that the model starts to show a slight overfitting at these values.

C. *Model validation*

To improve the evaluation of the model quality, further parameters are considered, which were calculated by a 5-fold cross validation on the training data set.

A high recall value (the proportion in percent of correctly positive instances) is associated with a lower precision value (how many of the instances predicted as positive are actually positive) and vice versa. This means that if the recall is increased, a reduction in precision must be accepted. However, since it is equally important for the model whether it has a high percentage of true positives and a low percentage of false positives, a high F1 score is more important. The F1 score is the harmonic mean of recall and precision and, with 98.46 %, is already sufficiently high to have a model that can make both accurate and comprehensive predictions.

To evaluate the model more detailed, a confusion matrix is created (Fig. 8). A confusion matrix is a tabular representation that shows the number of correct and incorrect predictions of a classification model for different classes [2].

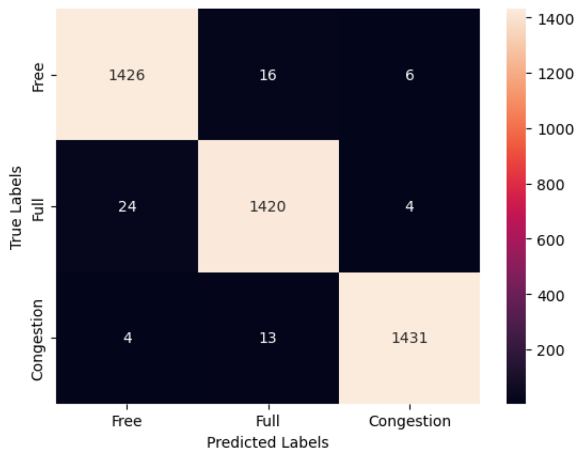


Fig. 8. Confusion matrix

Fig. 8 shows that the model most frequently confuses the state “full” with “free” during prediction and vice versa. To better identify the problem, a script was written that can output the incorrectly predicted segments and save them as a WAV. By listening to the incorrectly predicted segments, it becomes clear that it is also a challenge for a human to classify the audio tracks correctly, as they are difficult to distinguish due to their similarity, respectively the variance is noticeably lower.

Furthermore, in 4 cases the class "congestion" is classified as "free" and in 6 cases the other way around. When listening to these segments, it becomes obvious that this is due to incorrect sample quality, as the audio tracks contain noises. This means that incorrect information, such as sirens from emergency vehicles, loud squealing brake noises due to poor brakes despite free traffic, short gusts of wind, is contained in these segments. The segments that were incorrectly classified as "full" instead of "congestion" and vice versa have the same problem. These problems could be prevented by selecting the most representative class from a set of, for example, 4 consecutive segments to be classified, if no such short interval length of 4-seconds is required for the classification. Secondly, only one microphone used (ME 67) was sufficiently protected from the effects of wind. The analysis of the misclassified segments confirms the assumption that a microphone with a strong directional characteristic, which is also protected from wind, should be used in practice in order to achieve a higher prediction quality.

If we compare the accuracy of 98.46 % from the cross validation on the training data with the prediction accuracy of 98.34 % on the test data, it demonstrates that the model does not overfit and shows a robust performance that generalizes well to new data. The same result can also be seen from the F1 score, which is 98.46 % on the training dataset with cross-validation and 98.35 % validated on the test data.

VI. Conclusion

This work provides a systematic approach for modern traffic management systems on how microphones can be used to record acoustic input signals to monitor and control traffic flow.

By recording the traffic noise using various microphones and several processing methods, including MFCC transformation, a solid database was created. By using PCA, the amount of data can be effectively reduced by 99.60 % of the original size, which in combination with a Random Forest classifier results in an impressive increase in classification accuracy to 98.34 % validated on the test dataset. The work also demonstrates that the use of PCA is not only suitable for data reduction, it can also improve the model accuracy and generalization capability of the system by removing redundancies and noise.

Varying weather conditions and their potential impact on the acoustic data were not considered at this stage and represent an area for future investigation. Furthermore, the excellent performance achieved by the model leaves room for extending the classification with additional categories to allow a finer classification of traffic conditions.

In conclusion, this work shows how the combination of audio data analysis and machine learning can be used to develop a robust and precise model that can classify the fundamental states of traffic flow – free, full and congestion – only using audio signals.

References

[1] A. Fares and W. Gomaa, "Freeway Ramp-Metering Control based on Reinforcement Learning," in Proc. 11th IEEE Int. Conf. on Control & Automation (ICCA), Taichung, Taiwan, June 18-20, 2014. [Online]. Available: https://www.researchgate.net/publication/264660603_Freeway_Ramp-Metering_Control_based_on_Reinforcement_Learning. Accessed: December 25, 2023.

[2] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems," 2nd ed., Sebastopol, CA, USA: O'Reilly, 2021.

[3] SciPy Community, "scipy.signal.windows.hann," in SciPy v1.12.0 Reference Guide. 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.windows.hann.html. Accessed: January 22, 2024.

[4] J. B. Allen, "Short term spectral analysis, synthesis, and modification by discrete Fourier transform," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-25, no. 3, pp. 235-238, June 1977.

[5] T. Rohdenburg, "Klassifikation von Audio-Signalen, Diplomarbeit," Bremen, Deutschland, May 27, 2003.

[6] H. Veisi and H. Sameti, "Speech enhancement using hidden Markov models in Mel-frequency domain," Speech Communication, vol. 55, no. 2, pp. 205-220, 2013. ISSN 0167-6393. [Online]. Available: https://doi.org/10.1016/j.specom.2012.08.005. Accessed: January 14, 2024.

[7] Aalto University. (n.d.). Cepstrum and MFCC. [Online]. Available: https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC. Accessed: December 20, 2023.

[8] I. T. Jolliffe und J. Cadima, "Principal component analysis: a review and recent developments," in Phil. Trans. R. Soc. A, vol. 374, April 13, 2016.

Open RAN Threat Analysis of the O-RAN SC Reference Implementation

Henrik Wittemeier
Institute of Computer and
Communication Technology (ICCT)
Technische Hochschule Köln
Cologne, Germany
henrik.wittemeier@th-koeln.de

Arn Jonas Dieterich
Institute of Computer and
Communication Technology (ICCT)
Technische Hochschule Köln
Cologne, Germany
arn_jonas.dieterich@th-koeln.de

Prof. Dr. Andreas Grebe
Institute of Computer and
Communication Technology (ICCT)
Technische Hochschule Köln
Cologne, Germany
andreas.grebe@th-koeln.de

Thomas Karl
PROCYDE GmbH
Breitscheidt, Germany
thomas.karl@procyde.com

***Abstract*— The diversifications of 5G and upcoming 6G networks led to the development of open architectures and interfaces for the Radio Access Networks (RAN), named Open RAN. The introduction of new radio network control functions and open interfaces results in an extended threat landscape. The O-RAN Software Community (SC) is creating a reference implementation for proof of concept and sample functionality. This paper summarizes threats and vulnerabilities of the O-RAN SC reference implementation, and the underlying virtualized infrastructure setup and software development process.**

***Keywords*— Open RAN Vulnerabilities, O-RAN SC Reference Implementation G-Release, Kubernetes**

I. Introduction

Mobile communication technologies are part of critical infrastructures. The availability, integrity and confidentiality of 5G and upcoming 6G cellular networks are crucial. With growing demand for mobile telecommunications services and national security concerns about mobile network technology, the idea of a Radio Access Network (RAN) supporting 3GPP specifications for 5G and 6G networks, split into several functional entities with open interfaces, has been derived [1,2].
To address these upcoming challenges the Open RAN project was launched. The O-RAN Alliance [2] is a community of mobile operators, manufacturers, vendors, and scientific institutions that are working on open specifications and standards for radio access networks for 5G and 6G mobile networks. Part of the O-RAN Alliance is the O-RAN SC (Software Community) that implements the specification into a reference implementation. This reference implementation is being developed as a proof of concept of the specifications and for use in industrial deployments [3].
In comparison to the former 3GPP standards for cellular networks, the Open RAN design aims at dividing the RAN into smaller components with open specified interfaces, which can be developed by different vendors. Network operators can choose the best performing components to work in their networks.
The introduction of a new architecture with additional interfaces, the release of open-source implementations and use of virtualization introduces additional security threats and potential vulnerabilities. More interfaces and more participants cause a greater attack surface. As the Open RAN is in an early state it is necessary to eliminate possible threats, since later changes need a higher effort to be integrated into a downwards compatible system.
To guarantee a safe deployment of a mobile network in the future the development principles should be based on the “security-by-design” approach.
The research project 5G-FORAN [4] evaluates the identification and handling of security incidents in an Open RAN reference implementation, developed by the O-RAN Software Community (SC). The O-RAN Alliance specifications and related work is evaluated to identify potential vulnerabilities in G-Release reference implementation. In this paper we focus on the O-RAN SC Reference Implementation G-Release, which has been published in December 2022.

II. Architecture

The work of the O-RAN Alliance comprises several key areas, one of which is the creation and release of specifications for an Open RAN architecture and the release of a reference implementation.
The O-RAN architecture given in Fig. 1, has been created by O-RAN Alliance Work Group 1 (WG1). Components that were part of this research are described in the following chapter [5].

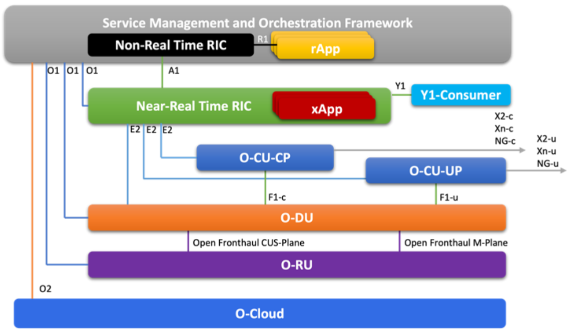


Figure 1: O-RAN Alliance Open RAN Architecture

The O-RAN specifications split the RAN into parts with similar time requirements. Functions with a control loop cycle higher than one second are handled by the Non-Real Time RIC (RAN Intelligent Controller), while more time critical functions are handled by the Near-Real Time RIC and real time functions are steered by the O-DU (O-RAN Distributed Unit). The interconnection between those components is handled via open interfaces.

The Non-RT RIC gives policy based guidance to the Near-RT RIC through the A1 interface. The exchange of policies is facilitated through an HTTP API.

The E2 interface interconnects the Near-RT RIC with the O-DU, the O-CU (O-RAN Central Unit) and the O-eNB (O-RAN eNodeB). It is used to control cellular RANs by transmitting policies and guidelines. The policies apply to a single User Equipment (UE), a group of UEs, radio cells or an area of cells. Communication on the E2 Interface is performed with the E2 Application Protocol encapsulated in SCTP.

To determine the impact of policy changes a control loop is implemented and feedback data is provided through the O1 interface to the Non RT-RIC. The protocols defined for O1 interface are HTTP and NETCONF over SSH.

The Non-Real Time RIC gains its intelligence by adding rApps to the RIC. An example use case of rApps is the management of cell shutdowns for energy savings during periods of low network utilization. The Near-Real Time RIC gets intelligence through xApps. xApps could manage cell handover, quality-of-service management or performance improvements.

III. Known and proSpected O-RAN vulnerabilities

In several research studies threats of the O-RAN specifications were analyzed. These results indicate potential threats in the reference implementation of the O-RAN Software Community. The following section summarizes some of the major results.

A. *Architectural Vulnerabilities*

The Federal Office for Information Security (BSI), Germany, published an extensive evaluation of the O-RAN architecture in 2022 [6]. It provides a meticulous survey of vulnerabilities in the O-RAN specifications. Vulnerabilities are evaluated for the protection goals confidentiality, integrity, accountability, availability, and privacy. The results are compared to the 3GPP specifications to show improvements or adverse effects on cybersecurity in the mobile networks.

General recommendations include that security protocols should have as few vulnerabilities as possible. To ensure integrity and privacy of all data, it is necessary that data is protected during transport as well as at rest, so that attackers or insiders with filesystem access are limited in violating confidentiality. In regard to the interfaces of the Open RAN infrastructure it is requested to implement a role-based access control model. Since Open RAN has many powerful interfaces like the O1 interface which has connections to many components, all components should only have the minimum privileges required to function. This decreases and limits the capabilities of malicious components.

With the multi-vendor approach of Open RAN it is necessary to isolate each component as much as possible to protect other components from being compromised. The implementation of a Zero Trust Architecture is recommended as no entity in the RAN should be trusted. This also applies to manufacturers, system integrators, and insiders with privileged access, such as a RAN or cloud operator, who are the main threat actors considered in this paper.

While many types of attacks can be prevented, DoS attacks cannot be entirely prevented. It is important that DoS attacks have limited impact on the performance of the O-RAN control system, to ensure its availability.

The A1 interface is only optionally secured by a TLS session. Since the A1 interface mostly transmits policies and RAN steering data, the protection goals confidentiality and integrity are not violated, but the network performance for users could be decreased significantly.

Unencrypted transmission on the E2 interface leads to a high risk for all stakeholders, however the O-RAN specification states encryption is only an optional requirement that could be achieved by tunneling the connection through IPSec. The E2 interface has a relevant role in Near-RT RIC communication. Therefore, a DoS attack on the E2 interface could have a high influence on the performance of the whole system.

The O1 interface uses the NETCONF protocol over SSH. SSH is a very powerful protocol that could give system wide access in the case of misconfiguration. In addition, the specification requires a backwards compatibility to prior vulnerable versions of SSH [7].

With an attack on the O2 interface it may be possible to control the entire RAN. Weaknesses in the specification cannot be addressed because they are not finalized at this point.

B. *App-related Vulnerabilities*

To improve the expandability of the RICs rApps and xApps were added to the RAN. rApps control the Non-Real Time RIC via the R1 interface. The BSI research team addressed the weak separation between different rApps as the specifications do not require physical separation. Since there are no requirements regarding programming languages, unsafe languages such as C or C++ could be used, which pose a higher risk of attacks through buffer overflows.

Due to the similar integration of rApps and xApps in the RIC the same points of concern apply to xApps. These use the A1 and E2 interfaces and are therefore closer to the 3GPP interfaces, which could mean a higher attack surface that can only be estimated as the security principles for the applications are not yet finalized.

The introduction of xApps aims to control and optimize radio components from the Near-RT RIC using the Radio Resource Management (RRM) functionality exposed by the gNB. However, without coordination, different xApp developers pose a risk of introducing xApps that conflict other xApp's actions and degrade the overall function or performance. Furthermore, malicious xApps can cause a DoS by signaling conflicting actions on purpose.

A role concept that would implement the principle of least privileges is suggested but not specified until now [8].

C. *Infrastructure-related Vulnerabilities*

The widespread use of virtualization in mobile networks has reached the RAN. Virtualization of the RAN poses new risks due to the mere logical isolation of components. The host system has access to all data in the RAM and hard drives and is therefore able to access the data of the workloads. Therefore, all layers from hardware, to firmware, to the hypervisor and the container orchestrator must be trusted to operate a trusted containerized application [9]

The reference implementation of the O-RAN SC is based on containerization, with Kubernetes as container orchestrator tool. There are various threats that containerized environments are exposed to, therefore these must be taken into consideration when deploying such critical software.

First and foremost, common misconfigurations, whether malicious or unintentional can help an attacker to perform reconnaissance and gain insights into the infrastructure. However, in the worst case, misconfigurations can lead to the compromise of the entire cluster. To avoid such scenarios, best practices should be implemented, such as following the guidelines of the Cybersecurity and Infrastructure Security Agency (CISA) [9] to harden deployments.

Containerization is often mistaken for complete isolation from the host. However, there are numerous ways to circumvent this isolation. Therefore, the use of sandboxing such as with “Kata” [11] and “gVisor” [12] are recommended. Furthermore, Linux Security Modules (LSMs) help to limit the application’s capabilities, file system access, network access and the available system calls. Without sandboxing and LSMs in place, isolation of Kubernetes pods might not suffice to protect the host system [13].

The most dangerous means of running a pod entails the use of the privileged flag in container environments. This grants the container all capabilities, with access to all the host’s devices and networks. This flag limits the isolation from the host to a minimum, making container escapes more feasible. Further flags concern the direct access to the host file system (hostPath), the system process space (hostPid) and the use of excessive capabilities such as the ability to use raw sockets for packet crafting through the CAP_NET_RAW capability. There are various other dangerous flags and capabilities that should only be granted according to the least-privilege model. The possibility of a container escape is not only rooted in misconfigurations. Numerous highly scored CVEs have been discovered over the years, such as the impactful RunC vulnerability CVE-2019-5736. Furthermore, a series of container escape vulnerabilities have been published in Feb. 2024 in the “Leaky Vessel” series published by Snyk Limited [13,14,15].

D. Internal network attacks

The Kubernetes infrastructure depends on essential protocols and network services such as ARP and DNS. The Address Resolution Protocol (ARP) is a layer two protocol that facilitates the mapping of IP addresses to MAC addresses. |In an ARP spoofing attack, an adversary can bind his MAC address to a legitimate IP, thereby diverting traffic to the malicious MAC address. This allows the attacker to intercept, modify, or even discard the traffic that was intercepted. In the context of Kubernetes, it allows for diverting traffic from one pod to another, for all pods on the same node.

The Domain Name System (DNS) protocol is used extensively within Kubernetes for inter-pod communication and service discovery. This allows services to be advertised using a permanent DNS-name instead of configuring IP-addresses. Since ARP spoofing is possible, this allows an attacker to divert traffic from the Kubernetes hosted DNS-service and modify responses. This opens up several possibilities, such as DNS spoofing or a MiTM attack.

E. O-RAN reference implementation development

The work of the O-RAN Alliance comprises several key areas, two of the most important focal points are the creation and release of specifications for an Open RAN architecture and the release of a reference implementation. Working Group 11 (WG11) is dedicated to addressing all security relevant aspects of the Open RAN ecosystem. WG11 has created a comprehensive threat model that outlines the threat surface and threat actors posing a risk to Open RAN operations. The following section discusses threat categories that apply to the Open RAN Alliance SC, with a focus on development principles regarding the global inter-organizational development effort [16].

1) Application Lifecycle Management

The category Application Lifecycle Management includes threats related to the use of outdated or unverified software as well as incorrect removal of containers, applications, and data. The open-source nature of O-RAN components and interface specifications fosters collaboration amongst experts and aids in the identification and resolution of security flaws. However, slow rollout times and release cycles could risk lengthy exposure to new vulnerabilities. Once a vulnerability is disclosed, it is publicly available for anyone to investigate using vulnerability databases such as the National Vulnerability Database (NVD). If such public flaws in software implementations are not addressed in a timely manner, threat actors may utilize their thorough understanding of the identified vulnerabilities and white-box view of the system to their advantage, to develop exploits and compromise security of Open RAN operations [17].Therefore, the influence of Open Source Software (OSS) widens the attack surface, and must be taken into account when evaluating the security posture of the RAN. Meticulous vulnerability management and supply chain risk analysis are required to securely integrate and operate OSS in practice [9].

2) Unsecured Credentials and Keys

The exposure of unsecured credentials, labeled T-GEN-5 by WG11 [16] is a general threat, specifically applicable to open-source development. Adversaries may search applications, containers, virtual machines, and data stores for further credentials. Hard-coded certificates, private SSH keys, plaintext passwords, API-tokens and other unsecured credentials are a quick win for adversaries to elevate privileges and access otherwise unauthorized systems.

Global cross-organizational coding efforts may lead to more room for errors. It only takes one person with a certain level of access to make a mistake or leak credentials online for an adversary to quickly gain a foothold. Hence, the use of hardcoded and unsecured credentials in an open-source project comes with a high level of risk and can quickly lead to adverse effects.

Mobile Network Operators (MNO) using parts of this code as the basis for their RAN infrastructure, introduce direct access for threat agents. Furthermore, some credentials enable access to the code repository itself. This provides the adversary with the possibility to introduce hidden backdoors into the repository.

IV. THREAT ANALYSIS OF O-RAN ECO SYSTEM

This chapter analyzes the applicability and feasibility of the previously introduced vulnerabilities in the context of the O-RAN SC Reference Implementation, G-Release.

A. O-RAN Component: A1 Policy Management Service

With access to the A1 interface, it is possible to read and write policies to the database hosted in the A1 Policy Management Service in the Near-RT RIC. Tests yield that transmitted policies are directly stored in the database. As description fields in policies are not limited in their size it is possible to utilize the whole RAM capacity by writing policies with sizes of multiple GBs. When the RAM is fully consumed the policy database is dropped resulting in a denial of service due to the inaccessibility of policies for xApps. Furthermore, as the database grows and consumes more memory, other components are negatively impacted due to less available memory, leading to performance drops.

B. Interfaces

Central to the ecosystem of the O-RAN Architecture are the specified interfaces. These interfaces connect the components and enable the operation of the RAN.

1) E2 Interface

The specification suggests protecting the E2 traffic through an IPSec tunnel. In the O-RAN reference implementation the protocol stack used by the E2 interface does not use TLS, nor is it encrypted by an IPsec tunnel. Thus, the E2 interface is vulnerable to confidentiality and integrity violations because data is not protected at all.

2) A1 Interface

The A1-interface is specified as HTTP RESTful API with endpoints on the Non-RT RIC and Near-RT RIC. The interface does not use TLS so a simple API call can be done to read a policy from the A1 interface. The A1 interface on the Near-RT RIC must be accessible from the Non-RT RIC and vice versa to facilitate communication. This means that an attacker with access to the network between the RICs could execute these attacks. Attackers that have this level of access are at least cloud providers. However, this attack could also be a part of another attack, which already gained access to the networks of the datacenter or machine.

C. xApps / rApps

xApps and rApps play a significant role in the Open RAN deployment. These apps lay the foundation for the RICs functionality and intelligence. A typical RAN deployment includes multiple xApps and rApps, where each app implements a specific functionality offered to the RAN for control or optimization as well as security purposes. This subset of functions only requires a limited set of privileges. In the O-RAN SC reference implementation xApps and rApps are not restricted in their access to the corresponding interfaces. Therefore, misbehaving apps can influence control functions that are not part of their intended feature set.

O-RAN Component	**Vulnerability**	**Source**
A1 Policy Management Service	Pod resource theft	5G-FORAN
E2 Interface	Unencrypted E2AP	O-RAN WG11
A1 Interface	Unencrypted HTTP	O-RAN WG11
A1 Interface	Unauthenticated Access to A1 interfaces	O-RAN WG11
xApps, rApps	No Principle of least privilege	O-RAN WG11

Table 1 Summary of Vulnerabilities found in O-RAN SC Reference Implementation (G-Release)

D. Infrastructure

The following section discusses vulnerabilities relating to the infrastructure. The analysis covers vulnerabilities regarding computer networks and the container orchestrator Kubernetes.

1) ARP spoofing

ARP spoofing is an attack that involves sending malicious ARP messages on the local network with the goal of poisoning the ARP cache. This allows the adversary to intercept and manipulate traffic, by assuming the role of the default gateway for all devices on the local network.
If the malicious host drops the traffic it would lead to a DoS, because all packets in the network are discarded. If the malicious host routes the packets to their original destination the change would not be observed by other components, but the malicious host is able to sniff that traffic and modify it if not encrypted. In combination with hardcoded credentials and certificates even encrypted traffic could be sniffed and manipulated.
In summary, the protection goals availability, integrity, and confidentiality would be violated. By sending the MAC address of our own component to all ARP requests from specific hosts, we can redirect traffic to our component. Thereby, the network abstraction of the supplied Kubernetes installation does not prevent ARP spoofing and an attack is successful.

2) Kubernetes

The O-RAN SC's implementation of Open RAN is based on container orchestration using Kubernetes. The maintainers of Kubernetes employ fast release cycles of four months, with a short support period of only 14 months. This can lead to mobile network operators utilizing out-of-date container orchestration software, that will not receive patches even if a vulnerability comes to light [18].
The O-RAN SC G-Release includes installation scripts, with Kubernetes version v1.16 for the Near-RT RIC. This Kubernetes version is end-of-live since August 2020, while the G-Release was published in December 2022.
The following table depicts infrastructure related vulnerabilities, mostly affecting the vulnerable Kubernetes version operated in the Near-RT RIC's G-Release implementation. Each vulnerability is identified with a

Common Vulnerability and Exposure (CVE) ID and only high-scoring vulnerabilities are included. The Common Vulnerability Scoring System (CVSS) framework classifies scores above 7.0 as high and scores above 9.0 as critical [19].

Kubernetes Component	Vulnerability	CVE	Source
Kubelet and kube-proxy (K8s)	Exposure of node-local services to adjacent hosts	2020-8558	NVD
Container volumes (K8s)	File access outside container volume	2021-25741	NVD
Kubernetes API server (K8s)	Malicious Custom Resource Definitions enable privilege escalation	2022-3294	NVD
Kubernetes API server (K8s)	Missing input validation leading to DoS	2019-11253	NVD
Internal network (K8s)	ARP spoofing	1999-0667	NVD

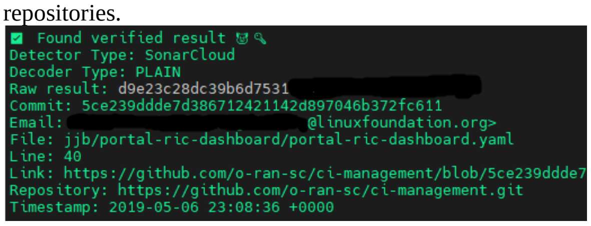
Table 2: List of infrastructure related vulnerabilities affecting O-RAN SC used software [20][21]

E. Development Principles

The development of an O-RAN reference implementation by the O-RAN SC is a complex endeavor. The involvement of numerous institutions, standardization bodies and corporations, each with disparate coding conventions and guidelines raises the intricacies involved when securing the software, infrastructure, and development environment. Furthermore, the use of OSS makes for a complex software supply chain, due to the various dependencies involved. The result is a large, ramified network with multiple potential sources of vulnerabilities.

The Cloud Security Alliance (CSA) published a report in 2022 regarding the “Top Threats to Cloud Computing”. The number one security issue identified is inadequate identity, credentials, access, and key management. Furthermore, the number one issue comprises the use and failure to decommission privileged accounts [22].

The O-RAN SC is affected by this security issue as well. The use of hardcoded certificates and credentials is common, and industry best practices are not applied. Tools such as “truffelhog” [23] allow scanning for unsecured credentials in GitHub repositories and organizations. For example, the following figure depicts a functional and verified API-token for the SonarCloud [24] with some degree of access to one GitHub repository of the O-RAN Software *Community* (SC) repositories.



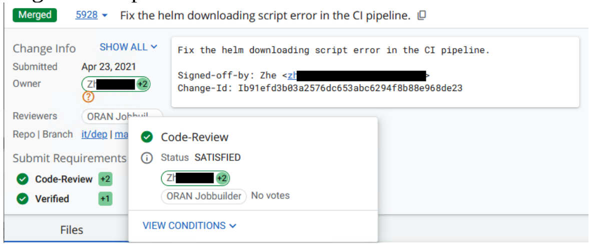
```
✅ Found verified result 🐷🔑
Detector Type: SonarCloud
Decoder Type: PLAIN
Raw result: d9e23c28dc39b6d7531[redacted]
Commit: 5ce239ddde7d386712421142d897046b372fc611
Email: [redacted]@linuxfoundation.org>
File: jjb/portal-ric-dashboard/portal-ric-dashboard.yaml
Line: 40
Link: https://github.com/o-ran-sc/ci-management/blob/5ce239ddde7
Repository: https://github.com/o-ran-sc/ci-management.git
Timestamp: 2019-05-06 23:08:36 +0000
```

Figure 2: Hard-coded SonarCloud API-Token in in one of the O-RAN the Alliance SC repositories

Furthermore, the ONAP part of the O-RAN SC implementation uses hardcoded credentials extensively, a list of over 50 different hardcoded credentials is published in their Wiki [25]. As the past has shown that hardcoded credentials can remain in production code, they should already be avoided at in the development stage.

Moreover, ONAP uses hardcoded certificates in its services. If certificates remain in production environments private data could be decrypted and changed.

A big benefit of OSS is the large community that works on software projects and reviews code to ensure functionality and security. To ensure that reviews cannot be skipped, repositories should have implemented rules sets that require reviews before code can be pushed to the main branch. In the repositories of the SC these rules are weakly implemented. Figure 3 illustrates that a developer can merge a pull request without reviews from other developers. This is a threat as a single developer could introduce malicious code.



Merged 5928 Fix the helm downloading script error in the CI pipeline.

Change Info SHOW ALL

Submitted Apr 23, 2021

Owner Zh[redacted] +2

Reviewers ORAN Jobbuil

Repo | Branch it/dep | ma

Submit Requirements

✓ Code-Review +2

✓ Verified +1

```
Fix the helm downloading script error in the CI pipeline.

Signed-off-by: Zhe <zh[redacted]>
Change-Id: Ib91efd3b03a2576dc653abc6294f8b88e968de23
```

✓ Code-Review

Status SATISFIED

Zh[redacted] +2

ORAN Jobbuilder No votes

VIEW CONDITIONS

Files

Figure 3: Unreviewed merge to main branch

In Table 3 threats are summarized that come from weak infrastructure or development principles. If possible, a Common Weakness Enumeration (CWE) ID is matched. As the name suggests, common vulnerabilities are listed in the CWE database [26].

O-RAN Component	Vulnerability	CWE	Source
Near-RT RIC	Outdated Kubernetes Version	1104	O-RAN WG11
ci-management repository	Unsecured Credentials	522	5G-FORAN
Non-RT RIC repository	Hardcoded Credentials and Certificates	798	5G-FORAN
Non-RT RIC repository	Insufficient merge restrictions	-	5G-FORAN

Table 3 Summary of Vulnerabilities related to O-RAN SC Reference Implementation (G-Release) Infrastructure and Development Practices

V. Further Research and Mitigations

This research has its focus on interfaces, infrastructure and development principles, therefore further research projects should take the O-RAN components more into account. The research took place using a passive deployment of an Open RAN, hence the impact of attacks could only be estimated. A deployment with UEs and live data will allow evaluations with more precise results.

The vulnerabilities and weaknesses identified in this research let us infer that the O-RAN SC does not develop their RAN implementation (G-Release) with a Zero Trust Architecture (ZTA) in mind. The O-RAN SC prioritized the implementation of functionality, with security being a lesser concern. ZTA features the isolation and protection of functions and components using micro-perimeters. Furthermore, authentication and authorization are strictly enforced for all entities, whether internal or external, human, or digital. No implicit trust is assigned to any entity. Moreover, data is protected in all states, thereby confidentiality and integrity for data in-transit, at-rest and in-use is enforced. Lastly, extensive logging, monitoring, and alerting are required for the purposes of troubleshooting and Digital Forensics and Incident Response (DFIR) [27].

As mobile networks are a part of the critical infrastructure of a country a zero-trust approach would be appropriate. This includes that traffic on all interfaces, even if they are not publicly accessible, should be secured against forging or sniffing, so that cloud operators or even RAN operators are limited in violating the privacy or availability of the system. This also includes that every stakeholder or component of the RAN is only provided with the least privileges they need to operate.

It is necessary to develop the O-RAN ecosystem using the principle of security-by-design. This leads to the integration of security measures in early releases that reduce the probability of insertion and impact of security issues that often remain in software in later development stages. This is the reason why hardcoded credentials and certificates should always be avoided.

Another aspect of the zero-trust principle is that developers are not trusted. As of now, a single developer could introduce malicious code to the repository without the review of any other developer. The O-RAN SC should introduce forced reviews, that require the review of at least one more party, such as a developer or maintainer.

Furthermore, whenever third party software and protocols are integrated into the O-RAN ecosystem, diligent care should be taken that up-to-date versions are used, the supply-chain is analyzed, patch and vulnerability management are in place and plans for updates made to ensure that end-of-life software is replaced.

xApps and rApps will be developed and maintained by different vendors. To prevent the interference of apps due to overlapping, xApp and rApp vendors must collaborate and adhere to certain standards. Furthermore, means of verifying the authenticity and integrity of xApps must be established. As multiple vendors increase the threat surface, xApps and rApps should be strictly isolated from other components of the RAN.

VI. Security Forensics for O-RAN SC Reference Implementations

From the security evaluation we can derive the necessity to monitor security related events of the O-RAN infrastructure. The O-RAN SC reference implementation, as well as several operational systems, are based on the Kubernetes ecosystem. This requires observation and monitoring capabilities for Kubernetes clusters aside from the traditional analysis of network traffic or logging of server-side events.

The focus in this work is on the cloud infrastructure and deployment of Open RAN systems. The aim of the 5G-FORAN project is to investigate different observability options for attacks on O-RAN systems by different role players: external users, cloud operators, network admins. Typical attacks by users have already been investigated and published in numerous papers e.g. [28,29,30]. Aside typical attacks from user side (UE) or from external networks like the Internet, additional attack potentials are identified with internal cloud operator staff or internal network operator staff. For these attack vectors on Open RAN implementations, especially on the O-RAN SC reference implementation, we see promising first results to detect internal attacks.

One promising technology is the rise of eBPF-based monitoring (extended Berkeley Packet Filter) e.g. [31,32,33], which allows deep insights into systems calls, virtual file systems, protocols, namespaces, storage and network interfaces. One aim of IT forensics in the 5G-FORAN project is to identify traces of attacks and compromises within the O-RAN SC reference implementation.

VII. O-RAN SC Releases and Outlook

The study in this paper is based on the G-Release of the O-RAN SC reference implementation, which was published initially in Dec. 2022. Additional security investigations and findings of the O-RAN Alliance Security Work Group (WG11) were taken into account and the vulnerabilities and threats of the G-Release were investigated.

In Dec. 2023, the I-Release of the O-RAN SC reference implementation was published. In comparison to the G-Release, it includes some additional implementations, like the rApp manager service in the Non-RT RIC or the Y1 interface in the Near-RT RIC. This leads to an extension of our work, as some security leaks were solved but also new functions have been implemented which need to be monitored and observed. Results based on the I-Release of the O-RAN SC reference implementation are expected during 2024.

VIII. Acknowledgements

The work in this paper was funded by the Federal Office for Information Security (BSI), Germany, under the funding code 01MO23020 in the program "Cyber security and digital sovereignty in the communication technologies 5G/6G". The responsibility for the content of this publication lies with the authors.

References

[1] Telecom Infra Project (TIP) OpenRAN MoS Group, URL: https://telecominfraproject.com/openran/

[2] O-RAN Alliance, URL: https://www.o-ran.org/

[3] Forschungsprojekt 5G-FORAN - IT-Forensik und Behandlung von IT-Sicherheitsvorfällen im Open RAN, URL: https://5g-foran.de/

[4] O-RAN Alliance e.V., https://www.o-ran.org/

[5] O-RAN ALLIANCE e.V. WG1, “O-RAN Architecture Description 10.0 (O-RAN.WG1.OAD-R003-v10.00).” O-RAN Alliance e.V., Alfter, Germany, Oct. 2023

[6] Federal Office for Information Security (BSI), Germany: Open-RAN Risik Analysis 5GRANR, Feb. 2022

[7] O-RAN ALLIANCE e.V. WG11, “O-RAN Work Group 11 (Security Work Group) Security Protocols Specifications (O-RAN.WG11.Security-Protocols-Specification.O-R003-v08.00).” O-RAN Alliance e.V., Alfter, Germany, Feb. 2024

[8] O-RAN Alliance e.V. WG11, “O-RAN Study on Security for Near Real Time RIC and xApps 4.0 (O-RAN.WG11.Security-Near-RT-RIC-xApps-TR.0-R003-v04.00).” O-RAN Alliance e.V., Alfter, Germany, Oct. 2023

[9] J. S. Boswell and S. Poretsky, “Security considerations of Open RAN.” Telefonaktiebolaget LM Ericsson, Stockholm, Sweden, Aug. 2020

[10] Cybersecurity and Infrastructure Security Agency (CISA) and National Security Agency (NSA) Cybersecurity Directorate Endpoint Security, “Kubernetes Hardening Guide (CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829) .” CISA and NSA, Aug. 2022

[11] Open Infrastructure Foundation community, “Learn about the kata containers project,” Kata Containers - Open Source Container Runtime Software, https://katacontainers.io/learn/ (accessed Feb. 17, 2024)

[12] The gVisor Authors, “What is gVisor?,” gVisor, https://gvisor.dev/docs/ (accessed Feb. 17, 2024)

[13] L. Rice, _Container security: Fundamental technology concepts that protect containerized applications_. ‘ O’Reilly Media, Inc.’, 2020

[14] MITRE, “CVE-2019-5736 Detail ,” NIST - National Vulnerability Database, https://nvd.nist.gov/vuln/detail/CVE-2019-5736 (accessed Feb. 12, 2024)

[15] J. Smith, “Leaky vessels: Docker and Runc container breakout vulnerabilities - January 2024,” Snyk, https://snyk.io/blog/leaky-vessels-docker-runc-container-breakout-vulnerabilities/ (accessed Feb. 8, 2024)

[16] O-RAN Working Group 11 (Security Working Group), “O-RAN Security Threat Modeling and Remediation Analysis (O-RAN.WG11.Threat-Model.O-R003-v06.00).” O-RAN ALLIANCE e.V., Alfter, Germany, Jun. 2023

[17] NIST (U.S. Department of Commerce), “National Vulnerability Database,” NIST - National Vulnerability Database, https://nvd.nist.gov/ (accessed Feb. 17, 2024)

[18] The Kubernetes Authors , “Patch releases,” Kubernetes, https://kubernetes.io/releases/patch-releases/#support-period (accessed Feb. 20, 2024)

[19] National Vulnerability Database, “Vulnerability Metrics,” NIST - National Vulnerability Database, https://nvd.nist.gov/vuln-metrics/cvss (accessed Feb. 14, 2024)

[20] National Vulnerability Database, “Kubernetes Kubernetes Version 1.16.0 : Security vulnerabilities, CVES,” Kubernetes Kubernetes version 1.16.0 : Security vulnerabilities, CVEs, https://www.cvedetails.com/vulnerability-list/vendor_id-15867/product_id-34016/version_id-1162678/Kubernetes-Kubernetes-1.16.0.html (accessed Feb. 17, 2024).

[21] MITRE, “CVE-1999-0667 Detail,” NIST - National Vulnerability Database, https://nvd.nist.gov/vuln/detail/CVE-1999-0667 (accessed Feb. 14, 2024)

[22] Cloud Security Alliance (CSA), “Top Threats to Cloud Computing.” CSA, 2022

[23] Trufflesecurity, “Trufflesecurity/Trufflehog: Find and verify credentials,” GitHub, https://github.com/trufflesecurity/trufflehog (accessed Feb. 13, 2024).

[24] SonarSource SA, “SonarCloud documentation,” SonarQube, https://docs.sonarsource.com/sonarcloud/#what-is-sonarcloud (accessed Feb. 13, 2024).

[25] “Oom hardcoded passwords list,” OOM Hardcoded Passwords List - Developer Wiki - Confluence, https://wiki.onap.org/display/DW/OOM+Hardcoded+Passwords+List (accessed Feb. 15, 2024)

[26] “Common weakness enumeration,” CWE, https://cwe.mitre.org/ (accessed Feb. 27, 2024).

[27] S. Poretsky and J. Jardal, “Zero Trust Architecture for evolving Radio Access Networks.” Telefonaktiebolaget LM Ericsson, Stockholm, Sweden, Nov. 2023

[28] C. T. Shen *et al*., "Security Threat Analysis and Treatment Strategy for ORAN," *2022 24th International Conference on Advanced Communication Technology (ICACT)*, PyeongChang Kwangwoon_Do, Korea, Republic of, 2022

[29] M. Polese, L. Bonati, S. D’Oro, S. Basagni and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376-1411, 2023

[30] M. Liyanage, A. Braeken, S. Shahabuddin, P. Ranaweera: Open RAN security: Challenges and opportunities, Journal of Network and Computer Applications, Volume 214, 2023

[31] D. Soldani *et al*., "eBPF: A New Approach to Cloud-Native Observability, Networking and Security for Current (5G) and Future Mobile Networks (6G and Beyond)," in *IEEE Access*, vol. 11, pp. 57174-57202, 2023

[32] J. Kawasaki, D. Koyama, T. Miyasaka and T. Otani, "Failure Prediction in Cloud Native 5G Core With eBPF-based Observability," *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Florence, Italy, 2023

[33] C. Cassagnes, L. Trestioreanu, C. Joly and R. State, "The rise of eBPF for non-intrusive performance monitoring," *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, 2020