Infocommunications Journal

A PUBLICATION OF THE SCIENTIFIC ASSOCIATION FOR INFOCOMMUNICATIONS (HTE)

ISSN 2061-2079

Special Issue

on Applied Informatics

Technically Co-Sponsored by

IEEE ComSoc™ IEEE Communications Society

hte

IEEE HUNGARY SECTION

Indexing information

Infocommunications Journal is covered by Inspec, Compendex and Scopus.
Infocommunications Journal is also included in the Thomson Reuters – Web of ScienceTM Core Collection, Emerging Sources Citation Index (ESCI)

Infocommunications Journal

Technically co-sponsored by IEEE Communications Society and IEEE Hungary Section

Supporters

FERENC VÁGUJHELYI – president, Scientific Association for Infocommunications (HTE)

The publication was produced with the support of the Hungarian Academy of Sciences and the NMHH

National Media and Infocommunications Authority • Hungary

Editorial Office (Subscription and Advertisements)**:**
Scientific Association for Infocommunications
H-1051 Budapest, Bajcsy-Zsilinszky str. 12, Room: 502
Phone: +36 1 353 1027 • E-mail: info@hte.hu • Web: www.hte.hu

Articles can be sent also to the following address:
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
Phone: +36 1 463 4189 • E-mail: pvarga@tmit.bme.hu

Subscription rates for foreign subscribers: 4 issues 10.000 HUF + postage

Publisher: PÉTER NAGY

HU ISSN 2061-2079 • Layout: PLAZMA DS • Printed by: FOM Media

Special Issue on Applied Informatics

Gergely Kovásznai and Imre Varga

IN TODAY'S dynamic and ever-evolving digital landscape, applied informatics plays a pivotal role in shaping the future of technology. From refining algorithms for enhanced data analysis to optimizing communication networks and advancing artificial intelligence, the realm of applied informatics continues to drive innovation and transformation across industries. This current Special Issue features contributions from the 12th International Conference on Applied Informatics (ICAI 2023), which was held in Eger, Hungary on March 2-4, 2023. These research papers explore novel insights, innovative methodologies, and practical applications within the field of computer science and informatics. Each of them represents a valuable contribution to the applied informatics field and offers insights that bridge the gap between theory and practical application. They are a testament to the diversity and dynamism of our field, showcasing a wide range of research topics and applications.

The ICAI Conference series is traditionally held in Eger in every 3 years, and is jointly organized by the Eszterházy Károly Catholic University and the University of Debrecen. The conference is oriented towards professional exchange of ideas in the field of Applied Informatics, covering areas such as Artificial Intelligence, Formal Methods, Computer Networks, Data Visualisation and more. The goal of the conference is to provide a forum for the discussion of academic research and industry.

Gergely Kovásznai is an Associate Professor and Head of the Department of Computational Science at the Eszterházy Károly Catholic University in Eger, Hungary. He received his Ph.D. degree in Formal Methods and Automated Theorem Proving from the University of Debrecen, Hungary, in 2007. Over the years, he worked as a research fellow at the Aristotle University of Thessaloniki, Greece, at the Johannes Kepler University Linz, Austria, and at the Vienna University of Technology, Austria. His research interests include formal methods, formal verification, operations research, and machine learning.

Imre Varga was born in Hajdúböszörmény, Hungary, in 1979. He received M.Sc. degree in physics and informatics teaching from the University of Debrecen, in 2002 and Ph.D. degree in physics also from the University of Debrecen, in 2008.
He has been working at the University of Debrecen since 2002. Now he is an associate professor at the Faculty of Informatics. Imre has been the head of the Department of Informatics Systems and Networks since 2018. He has several journal conference papers in the field of complex systems. His research interests include structure formation, fracture of granular materials, study of complex networks and information spreading on them by the tools of computer simulation.

Special Issue of the Infocommunication Journal

Deep Learning-Based Refactoring with Formally Verified Training Data

Balázs Szalontai, Péter Bereczky and Dániel Horpácsi

***Abstract*—Refactoring source code has always been an active area of research. Since the uprising of various deep learning methods, there have been several attempts to perform source code transformation with the use of neural networks. More specifically, Encoder-Decoder architectures have been used to transform code similarly to a Neural Machine Translation task. In this paper, we present a deep learning-based method to refactor source code, which we have prototyped for Erlang. Our method has two major components: a localizer and a refactoring component. That is, we first localize the snippet to be refactored using a recurrent network, then we generate an alternative with a Sequence-to- Sequence architecture. Our method could be used as an extension for already existing AST-based approaches for refactoring since it is capable of transforming syntactically incomplete code. We train our models on automatically generated data sets, based on formally verified refactoring definitions and by using attribute grammar-based sampling.**

***Index Terms*—Deep learning, Formally verified training data, Neural Machine Translation, Sequence-to-Sequence.**

I. Introduction

BEHAVIOUR-preserving program rephrasing (known as refactoring) is an inevitable step in any software development process. The goal of refactoring is to improve the quality of software source code without altering its observable behaviour [1].

Refactoring is commonly implemented as a transformation on a structured representation (such as a parse tree) of the source code. Admittedly, this approach works well in the typical scenarios with syntactically valid code; furthermore, when defined with syntactic rewriting, simple refactoring steps can be verified for correctness (semantics-preservation) by using formal methods. On the other hand, syntax-based approaches need to hardcode the logic of handling the various combinations of language constructs, and they cannot handle incomplete or ill-formed code fragments. In contrast, deep learning-based methods are inherently adaptive, and they can eliminate the need for hardcoding the vast amount of shapes and combinations syntactic constructs may take in a program. Due to its benefits, there has been an ever-growing interest in using deep neural networks for modifying source code recently. Some of these techniques, in addition to removing the burden of hand-crafting refactoring algorithms, come with the ability of transforming incomplete code fragments as the model is trained to transform code at the lexical level.

Balázs Szalontai, Department of Software Technology and Methodology, Péter Bereczky and Dániel Horpácsi, Department of Programming Languages and Compilers, ELTE Eötvös Loránd University, Budapest, Hungary (E-mail: {bukp00, berpeti}@inf.elte.hu, daniel-h@elte.hu)

Manuscript received April 10, 2023; revised August 21, 2023.

DOI: 10.36244/ICJ.2023.5.1

In this paper, we propose the combination of the above techniques: we apply deep learning for code refactoring and train on datasets generated with syntactic rewriting. We show that processing *Erlang* source code as a sequence of tokens and using deep learning methods to apply changes could serve as a great *extension* to the existing syntax-based methods, because our approach is capable of fixing incomplete or non-compilable code as well, supposing that the parts to be refactored are already complete. Moreover, we train our deep learning model on verified refactorings, that is, the code before and after the transformation are behaviourally indistinguishable. We present the following contributions:

- We formally define refactoring steps as conditional syntactic rewrite rules, and based on previous work [2], [3], [4], we verify the correctness of these steps by means of proving contextual equivalence (based on "CIU" equivalence [5]) between the matching and replacement patterns of the rewrite rules.
- Then we take the rewrite rules and instantiate the metavariables with randomly generated expressions, yielding semantically equivalent expression pairs. We also generate random context around these expressions to ultimately obtain the *formally verified training data*.
- Finally, we train a recurrent neural network to localize the code to be refactored and a Sequence-to-Sequence network with Attention Mechanism to carry out the refactoring steps autonomously. A very similar approach was presented in [6]. By using a similar architecture we show, that this approach is essentially language independent.

The paper is structured as follows. In Section II we discuss the related work, then in Sections III, IV, and V we show the above components of our approach. In Section VI we evaluate our approach, and finally Section VII concludes.

II. Related Work

There have been multiple attempts to transform source code with deep learning. The goal of such methods is generally to fix common errors (such as syntactic errors or semantic bugs) or to refactor code. Although in the current state of research, such techniques are not yet completely reliable, it is nevertheless a very active field.

Gupta et al. [7] aim to fix common C language errors with a Sequence-to-Sequence architecture. Their method is applied iteratively to fix errors one by one. Tufano et al. [8] train a recurrent Encoder-Decoder architecture on a dataset comprising data from pull requests. Their goal is to imitate human code fixing operations. Chen et al. [9] train a Sequence-to-Sequence architecture with Attention Mechanism to repair

programs. In their work, Copy Mechanism is also used to overcome the difficulties of the large number of possible identifiers occurring in code. Jiang et al. [10] first pretrain the model on a general task, then fine-tune an Encoder-Decoder architecture for the task of program repairing. Similarly to these works, we use a Sequence-to-Sequence (Encoder-Decoder) architecture with Attention Mechanism to transform nonidiomatic Erlang functions into their idiomatic alternative. We use formally verified data to train our model. Lutellier et al. [11] use a CNN-based Encoder-Decoder architecture and Ensemble learning techniques to automatically repair programs. Although our proposed Sequence-to-Sequence architecture is a recurrent one, we apply convolution on each chunk of the input source code when attempting to localize the nonidiomatically implemented ones. Chakraborty et al. [12] utilize a tree-based Encoder-Decoder architecture to modify code. First, structural modifications are applied, then the nodes of the modified tree are concretized. Li et al. [13] first train a tree-based recurrent model on the context of the code chunk to be transformed, then they use a tree-based Encoder-Decoder architecture for modifying the chunk. The downside of such tree-based methods is that they require the source code to be completed. A goal of ours is to experiment with incomplete code, which prevents the use of such tree-based approaches.

In previous work [6] we presented a method to localize and refactor nonidiomatic source code snippets in Python to improve code readability and program efficiency. The nonidiomatic snippet is localized using a model that performs sequence tagging: a recurrent model is trained to tag the source code lines with one of four tags (*START*, *IN*, *END*, *OUT*). This tagging indicates whether the line is part of a nonidiomatic snippet or not. The idiomatic alternative is generated based on the nonidiomatic snippet using a Sequence-to-Sequence architecture with attention. This approach of localizing and refactoring Python snippets is very similar to our proposed approach for refactoring Erlang source code.

III. Formally Verified Refactorings

In many cases, we encounter the problem of having a bad-quality dataset, which can decrease the performance of the neural network. Data collected from the web can contain errors, such as incorrect code transformations in the context of refactoring. In this work, we present a method that is trained on correct data. We achieve this by generating data according to formally verified equivalence rules. To the best of our knowledge, using such a verified dataset for training neural networks has never been proposed. Although this is not a common practice, being verified could be a highly desired aspect of training data.

In practice, refactorings are tested thoroughly, but are usually not proven-correct, thus in special circumstances (so-called edge cases) they could introduce errors. Let us motivate formal verification with a seemingly trivial example.[1] In Figure 1 we define the lazy version of conjunction in two ways, where one can be seen as a refactoring of the other. Without familiarity with Erlang, one may think that this two code fragments are equivalent definitions of the same functionality and transforming between them should not affect the program's behaviour.

```
lazy_and(X, Y) ->
  case X of
    true  -> Y;
    false -> false
  end.
```

(a) A correct implementation of lazy conjunction

```
lazy_and(X, Y) ->
  if X    -> Y;
     true -> false
  end.
```

(b) An incorrect implementation of lazy conjunction

Fig. 1: Motivational example for proving program equivalence

However, Erlang is dynamically typed, so `X` and `Y` can take any Erlang values, not just **`true`** or **`false`**—this introduces an extra level of complexity in the behaviour of these functions due to the dynamic type checking. In fact, it can be shown that the two variants of `lazy_and` do not behave the same way: in Figure 1a, if `X` was not a boolean value, we get an exception, while the function in Figure 1b will evaluate to **`false`**. Therefore, we can conclude that the transformation (read in either direction) in Figure 1 is *not* a refactoring.

To correct this inconsistency between the functions above (i.e., make them equivalent), we either need to change Figure 1a to use the wildcard pattern _ instead of **`false`** (in this case it would not be a correct implementation of conjunction any more, though), or we need to include a side condition requiring that `X` can only take boolean values. In either case, such simple, local refactorings are easiest carried out by defining and applying them as term rewriting rules [15], by extracting the irrelevant parts as metavariables denoting arbitrary expressions. This way, the concrete equivalent expression pairs can be obtained by instantiating the metavariables with expressions [16].

```
case e1 of true -> e2;          ⟶   if e1 -> e2;
        _ -> e3                      true -> e3
end                                  end
```

Fig. 2: Expression refactoring example

Figure 2 shows the previous expression refactoring as a rewrite rule with the wildcard pattern solution. As a matter of fact, this refactoring can be shown to be correct for any expressions e_1, e_2 and e_3.

```
f(x) when length(x) == 0 -> e
        ↓ when x ∉ vars(e1)
f([]) -> e
```

Fig. 3: Function clause refactoring example

Figure 3 presents another example, where the refactoring is applied to an Erlang function clause, and it shows how pattern matching can be used instead of a guard expression restricting the length of a list. The rewrite rule defining the refactoring has a side-condition, which needs to be met for the refactoring to be applicable: the parameter variable cannot appear in the

[1]The small refactoring examples we investigate here are inspired by Poór et al. [14].

function body. The variables f, x and e denote a function name, a variable, and an expression, respectively.

Formal semantics and program equivalence: To formally argue about the preservation of behaviour, we need a formal semantics of the language, and a suitable program equivalence definition. If two programs are proved to be equivalent, they are not distinguishable in any program context, that is, they can be exchanged.

Previously, we have defined several formal semantics for Core Erlang [2], [3], [4] which are capable of expressing program equivalence of Core Erlang expressions, and we also implemented these semantics in Coq. Core Erlang is an intermediate language of Erlang in the official implementation; we utilize this by reasoning about equivalence in Erlang via the trusted translation from Erlang to Core Erlang.

We use an (extended) version of the frame-stack semantics we defined previously [4], and the concept of CIU ("closed instances of use") equivalence [5]. The termination relation defined there is denoted by $\langle K, e\rangle \Downarrow$, meaning that expression e terminates in the frame stack K. The frame stacks describe continuations, i.e., K includes what should be evaluated next, once e has terminated. Here, we show a simplified version of the equivalence definition, and refer to [4] and the Coq formalisation [17] for further details.

Definition 1 (CIU equivalence)**.** $e_1 \equiv_{ciu} e_2 \stackrel{\text{def}}{=} (\forall K : \langle K, e_1\rangle \Downarrow \iff \langle K, e_2\rangle \Downarrow)$

We also showed [4] that reasoning about termination is sufficient to ensure the behavioural and contextual equivalence of the expressions (i.e., they evaluate to equivalent values, and equivalent expressions are interchangeable in arbitrary syntactic contexts). Based on the CIU equivalence, we can show the correctness of local refactoring steps.

Definition 2 (Correctness of refactorings)**.** *For all expressions e_1, e_2 and conditions P, $e_1 \to e_2$ when P is a correct refactoring step if P implies $e_1 \equiv_{ciu} e_2$.*

We have already proved the equivalence for the refactoring steps shown in Figure 2 and Figure 3. The proofs are extensive and their presentation is out of the scope of this paper, but we refer to the Coq formalisation [17] for more details.

IV. Generation of Training Data

To produce training data for the neural networks, we require two datasets: one dataset should contain *(nonidiomatic code, snippet location)* pairs to train the localizer network, while the other should consist of *(nonidiomatic snippet, idiomatic snippet)* pairs to train the refactoring network. We have set forth some general expectations for generating these datasets.

- The generated code must be syntactically correct and tokenizable.
- The generated code should share similarity with real-world code, i.e., it should apply functions of the standard library, apply other generated functions, use Erlang-specific values (e.g., `ok`, `false`, `true`), etc.
- The datasets must not only be diverse in terms of code size but also in the internal structure and meaning.
- Generating the idiomatic alternative must be deterministic, to allow the network to grasp the refactoring procedure.
- In the localizer's dataset, each nonidiomatic source code should contain at least one nonidiomatic snippet.
- Sufficient data should be available for training both networks: we believe that a good starting point would be to generate about 50.000 training examples in both datasets.

Based on the verified refactoring rules, we sample loads of concrete program modules including parts where the proven-correct refactoring steps can be applied. At the same time we synthesise the result of the refactoring too, by applying the rewrite rules.[2] With this dual synthesis, we create training data both for the localizing of refactoring candidates as well as for the application of the refactoring.

The program generation is based on a stochastic attribute grammar defining (a subset of) the Erlang programming language. In particular, we randomly generate elements of the language defined by the grammar, where the probabilities associated with the nonterminal symbols, along with some constrains carried in attributes, control the shapes and style of the generated programs; for instance, we can set the maximum number of functions and expressions within clauses, as well as we can fine-tune how deeply, and how likely, expressions get nested. For details about the attribute grammar notation, we refer to [18].

The data generation process consists of the following main steps:

- First a *module context* is generated by a modified variant of the above-mentioned attribute grammar, which produces modules that may contain so-called *holes* (holes mark the designated locations where code to be refactored will be emplaced). Holes encode information about their context (e.g. variable and function names in their scope) so that the refactoring candidates are generated context-sensitively.
- Then a refactoring rule is instantiated with the metavariables replaced by randomly generated names and subexpressions — each such instance will embody a correct rewrite step applied locally. Then the code snippets representing the target and the result of the refactoring are emplaced into the context that was generated in the previous step.
- Finally, generated code is dumped into a data file that annotates/labels each module with the location of the refactoring candidates and the type of refactoring applicable at each candidate.

This grammar-based method for generating the refactoring dataset has some clear advantages. First of all, the generated modules are syntactically and static semantically valid; thus, when the strings are tokenized, the token stream surely represents a well-formed program. Secondly, due to the controlled random sampling, the more cases we synthesise, the wider the variety in size and structure the generated programs expose.

[2]Since the equivalence relation is a congruence, the modules containing the refactored chunks are also equivalent.

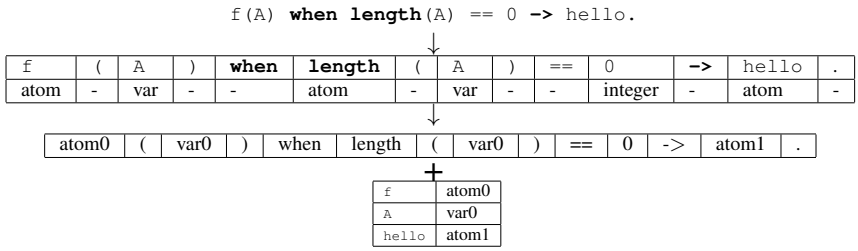


Fig. 4: The approach for tokenizing and creating the corresponding variable dictionary. The tokens `length` and `0` are kept in their original form, since they hold valuable information.

This guarantees that the contexts will teach various syntactic ways a refactoring candidate may be happen to be part of a program, and the refactoring instances themselves show a wide spectrum of syntactic variety. The generated programs are fully random, i.e., including programs that implement no useful behaviour. This is based on the fact that QuickCheck generators are implemented based on Erlang's pseudo-random seeder. We expect this not to cause an issue because the current rewriting approach is lexical, little to do with the actual semantics of the program under transformation.

V. Approach of Refactoring with Deep Learning

In this section, we offer a concise summary of the approach we took to localize and refactor function definitions in Erlang source code. First, we explain how we transformed a given piece of source code into a sequence of tokens that served as input to the models. Second, we introduce the neural network architecture that we trained to localize nonidiomatic code chunks. Finally, we describe the method for generating an alternative for the localized chunk using a Sequence-to-Sequence architecture with Attention Mechanism.

A. Preprocessing Source Code

To transform source code into a sequence of tokens, we undertake multiple steps. We utilize the Erlang module *tok* to tokenize the source code. Using this module, we obtain not only the tokens themselves, but also their types, such as *atom*, *integer*, *variable*, etc. To tackle the challenge of handling the unlimited number of valid identifiers, numbers, variables, etc., present in code, we replace these tokens with their type and a unique index. There are some exceptions to this process: the tokens that hold valuable information are not replaced, such as `length`, `0`, `true`, etc. We drew inspiration for this approach from the work of Chirkova et al. [19]. When generating an idiomatic alternative, we invert the changes made to these tokens that appear in the model's output. The process is visualized on Figures 4 and 5: Figure 4 shows how a function implementation is turned into a sequence of tokens and its corresponding dictionary. Figure 5 shows how the changes are inverted for an idiomatic alternative generated by the refactoring model.

B. Localizing Nonidiomatic Functions

Localizing nonidiomatic patterns (i.e., refactoring candidates) in source code is solved as a sequence tagging task.

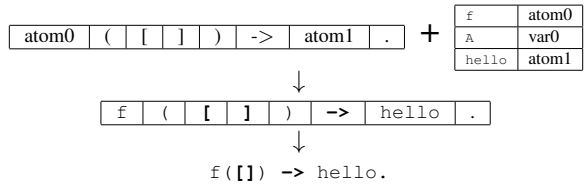


Fig. 5: The approach for turning the output of the refactoring model back to a code string.

The idea behind this approach is that we tag each chunk of the source code with one of two tags: *OUT* or *IN*. The used tag indicates whether a certain chunk of code is to be refactored (*IN*) or not (*OUT*). Using this approach, we gather candidates for refactoring, while the rest of the code remains unchanged. The source code gets splitted by '.' characters. This way, the function definitions get separated and are ready to be tagged.

The proposed neural network architecture (Figure 6) consists of convolutional, recurrent, and feedforward components. Firstly, the preprocessed (tokenized, splitted) code is provided as input to the network. Secondly, the tokens of each chunk are embedded into a 64-dimensional vector space. Thirdly, a one-dimensional convolution is applied to each code chunk using 128 filters and a kernel size of 5. Fourthly, two pooling operators are applied to the convolutional outputs: average and minmax. The average pooling calculates the element-wise average, while the minmax pooling is a unique pooling operator that calculates the element-wise maximum or minimum depending on which value is further away from the average. These pooling operations yield two intermediate representations for each chunk of the source code.

Having the intermediate representations provided by the two pooling operators, the following step is to obtain context-dependent aspects for the chunks as well. To achieve this, we feed the intermediate representations into two separate Bidirectional LSTM[3] (BiLSTM) layers with 32 units and two fully connected layers with 64 units. The BiLSTM outputs are fed into another set of two BiLSTM layers. The activation function used for both the BiLSTM and fully connected layers is *tanh*, and 20% dropout is applied after each. As a result of these steps, we obtain four vectors that represent each chunk of code. These four vectors are concatenated to create a final hidden representation, which is then passed

[3]The LSTM layers used in our method all return the whole sequence of outputs.

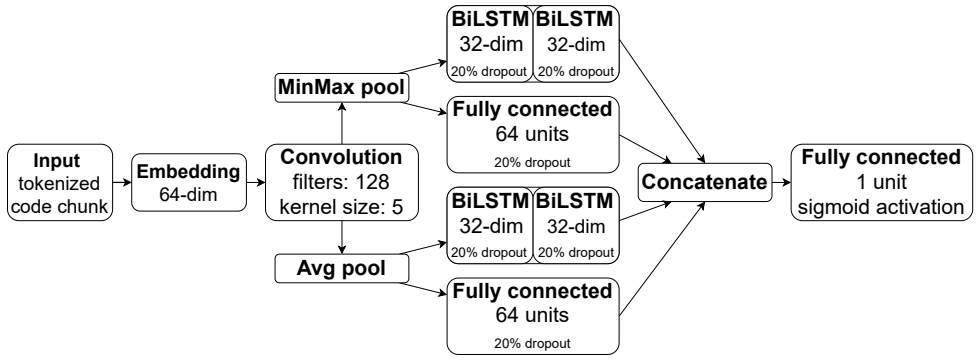


Fig. 6: Proposed neural network architecture to localize refactoring candidates

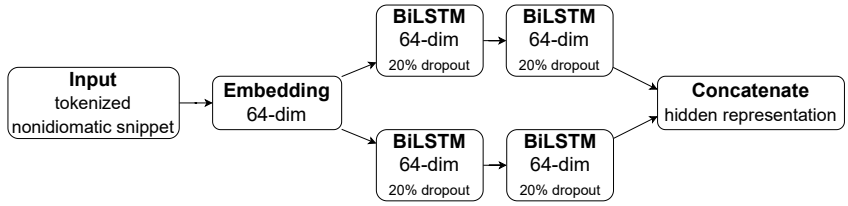


Fig. 7: Proposed architecture of the encoder of the refactoring model

to a fully connected layer to obtain the final output, where the fully connected layer has one unit and uses the *sigmoid* activation function. This result indicates which chunks are to be refactored (1 for *IN*) and which are not (0 for *OUT*). The Adam optimizer is used with a learning rate of 0.001, and binary cross-entropy is the loss function.

We based the selection of hyperparameters on previous work [6] and on intuition. This is the case for both the localizer and refactoring component. Since our current goal is to prove that using deep learning for refactoring nonidiomatic Erlang code is a viable option, we did not focus on optimizing the hyperparameters too much.

C. Generating Idiomatic Alternative

The idiomatic alternative is generated using a recurrent Sequence-to-Sequence architecture with Attention Mechanism. That is, we first feed the tokenized nonidiomatic code to the encoder, and then expect the decoder to generate the idiomatic alternative token-by-token.

The encoder component (Figure 7) aims to produce a hidden representation of the input sequence. This is achieved through the use of a recurrent neural network consisting of four BiLSTM layers with 64 units each. First, the tokens of the input sequence are embedded into a 64-dimensional vector space. Next, the sequence of tokens is fed into the four BiLSTM layers in the following way: the input sequence is fed into two BiLSTM layers, then the outputs of these layers are fed into another BiLSTM layer each. The resulting outputs of the two latter BiLSTM layers are concatenated to produce the final output of the encoder, which serves the hidden representation of the code chunk. Additionally, the BiLSTM layers are followed by 20% of dropout. This architecture is designed with the aims of (1) obtaining two independent representations of the same code chunk by using parallel BiLSTM layers, and (2) acquiring a higher-level representation by using such a stacked architecture.

The decoder uses a single LSTM layer with 256 units to generate an output sequence element-by-element. The LSTM layer takes as input the fixed-length vector representation produced by the encoder, along with the previously generated output token (which is of the idiomatic code). For the first element of the output sequence, a special $START$ token is used in place of the previous output. An attention layer is used to compute the attention weights between the encoder output and the decoder outputs. The attention weights are used to compute the context vector, which then gets concatenated with the decoder outputs. The concatenated vector is passed through a fully connected layer with *softmax* activation to obtain the final decoder outputs, which determines the next token of the idiomatic code. We use the Adam optimizer with the learning rate of 0.001 and categorical crossentropy as the loss function.

VI. Evaluation and Experiments

In this section, we first present the measured accuracy of the two main components of our approach: the localizer and refactoring models. Then we showcase experiments on our method's capability to perform refactoring steps on Erlang code. Finally, we make notes about the usability of our method on real-world codes.

Both the localizer and refactoring models were evaluated on a test set that was separated from the training data before the training process: 10% for the localizer and 4% for the refactoring model. The accuracy of the localizer model is the ratio of the correctly classified code chunks divided by the total number of chunks in the test set: this ratio turned out to be **99.09%**. For the refactoring component, we measured

TABLE I
EXAMPLE OF REFACTORED CODE SNIPPET

Original (nonidiomatic) code		Refactored code
`f(L) when length(L) == 0 -> error;` `f(L) -> lists:max(L).`	→	`f([]) -> error;` `f(L) -> lists:max(L).`
`f(L) when length(L) == 0 ->` `  case X of 0 -> true; 1 ->`	→	`f([]) ->` `  case X of 0 -> true; 1 ->`

TABLE II
EXAMPLE OF REFACTORED SOURCE CODE

Original source code	Refactored source code
`-module(mod).`	`-module(mod).`
`-compile(export_all).`	`-compile(export_all).`
`fact(0) -> 1;`	`fact(0) -> 1;`
`fact(N) -> N * fact(N - 1).`	`fact(N) -> N * fact(N - 1).`
`f(L) when length(L) == 0 -> error;`	`f([]) -> error;`
`f(L) -> double(lists:max(L)).`	`f(L) -> double(lists:max(L)).`
`double(N) -> N * 2.`	`double(N) -> N * 2.`

the ratio of error-free transformations against the total number of attempted transformations. The resulting accuracy of this evaluation was **99.46%**. These results indicate that our models have been trained succesfully and are capable to perform refactorings that are similar to the ones in the training datasets.

We now present some of our experiments with refactoring various kinds of nonidiomatic programs, including complete and incomplete programs. First, we focus only on the refactoring component by experimenting with refactoring nonidiomatic snippets without their surrounding context. After running our model on some nonidiomatic snippets, we compared the output of the model with the original code and checked whether the output is more idiomatic after the changes have been applied. An example of such a refactoring is shown in the first row of Table I: here the original version used a guard to handle empty lists, which was replaced by pattern matching syntax. The refactored code is an equivalent and more idiomatic compared to the original code chunk.

As mentioned earlier, we also experimented with refactoring incomplete code. Of course this only makes sense if the parts that make the code nonidiomatic are present. In such a scenario, our model attempts to refactor the nonidiomatic part(s) only and leave the rest unchanged to allow for completion. The second row of Table I shows such a refactoring. The transformation was successful - that is, the incomplete part was left unaltered - in spite of the fact that the model was not trained on incomplete code.

Next, we performed further experiments on the entire method, including the localizer component. That is, we applied our method on a full Erlang code that contained a nonidiomatic function implementation. The expected result of the method is of course the modified code that only varies in the originally nonidiomatic section. A transformation is correct if the generated alternative is the properly refactored version of the original nonidiomatic snippet, while the behavior of the entire program is preserved. An example of such a refactoring performed by our method is shown in Table II.

While our proposed approach shows promising results in refactoring nonidiomatic Erlang code, it represents an initial proof of concept. At the current stage of research and development, our method cannot be used on real-world codes, because it too often classifies code chunks as refactoring candidates even when they are not. Since incorrectly localized snippets cannot be refactored, our idiomatizer network obviously fails to refactor correctly, which leads to broken code.

We are planning to address this issue by considering multiple approaches. Firstly, we will investigate ways to generate training data that is more representative and closer to real-world code. Secondly, we will explore other approaches for finding the best way to split the original source code: it could be the case that too much unnecessary information is given to the network at once, caused by splitting code by '.' characters. This might make it harder to decide whether or not to refactor. Thirdly, we will experiment with some architectural modifications for the localizer component. Possible modifications include introducing different types of layers (such as GRU or Convolutional LSTM) and optimizing the hyperparameters. Lastly, it is possible to perform an extra pass on the output of our method to filter out some incorrect refactorings, for example if the code before the refactoring was compilable, it should be compilable after it too. We note that although it is also theoretically possible to filter out incorrect outputs by post-verifying each refactoring instance and proving equivalence, it would be very costly as the formal verification is manual.

VII. CONCLUSION

In this paper, we have presented a novel approach to refactor source code using deep learning techniques. We prototyped this approach for Erlang. Our method includes a localizer and a refactoring component, which enable the localization and refactoring of nonidiomatic code patterns into their idiomatic counterparts. Our method processes the source code as a sequence of tokens, making it capable of transforming even incomplete or non-compilable code.

To ensure that the neural networks learn how to correctly refactor, we used formally verified data, which we obtained by instantiating conditional term rewrite rules whose behaviour preservation is formally proven. We do not aim to change already existing AST-based approaches, but rather propose our deep learning-based approach as an *extension* to these.

Finally, we highlight some areas for possible future work:

- Proving the correctness of a larger number of local refactorings, and including them into our approach.

- Extending the generator component to emit code including more language constructs (e.g., strings), standard library functions (e.g., calls to higher-order functions).
- Applying different tagging approaches for the localizer component in order to be able to identify refactoring candidates more precisely and limiting number of false positive localizations.
- Investigating other Sequence-to-Sequence architectures for refactoring, such as CNN-based [20] or Transformer [21].
- Performing a comprehensive analysis to evaluate the performance of the presented method on real-world code.

Acknowledgements

Supported by the ÚNKP-22-3 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. "Application Domain Specific Highly Reliable IT Solutions" project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

References

[1] M. Fowler, *Refactoring: improving the design of existing code*. USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN: 0201485672.

[2] P. Bereczky, D. Horpácsi, and S. Thompson, "A proof assistant based formalisation of a subset of sequential Core Erlang," in *Trends in Functional Programming*, A. Byrski and J. Hughes, Eds. Cham: Springer, 2020, pp. 139–158, DOI: 10.1007/978-3-030-57761-2_7.

[3] P. Bereczky, D. Horpácsi, and S. Thompson, "Machine-checked natural semantics for Core Erlang: exceptions and side effects," in *Erlang'20*. ACM, 2020, pp. 1–13, DOI: 10.1145/3406085.3409008.

[4] D. Horpácsi, P. Bereczky, and S. Thompson, "Program equivalence in an untyped, call-by-value functional language with uncurried functions," *Journal of Logical and Algebraic Methods in Programming*, vol. 132, p. 100 857, 2023, DOI: 10.1016/j.jlamp.2023.100857.

[5] I. Mason and C. Talcott, "Equivalence in functional languages with effects," *Journal of Functional Programming*, vol. 1, no. 3, pp. 287–327, 1991, DOI: 10.1017/S0956796800000125.

[6] B. Szalontai, Á. Kukucska, A. Vadász, B. Pintér, and T. Gregorics, "Localizing and idiomatizing nonidiomatic python code with deep learning," in *Proceedings of the Computing Conference 2023*, June 2023, to appear.

[7] R. Gupta et al., "DeepFix: Fixing common C language errors by deep learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017, DOI: 10.1609/aaai.v31i1.10742.

[8] M. Tufano, J. Pantiuchina, C. Watson, G. Bavota, and D. Poshyvanyk, "On learning meaningful code changes via neural machine translation," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 25–36, DOI: 10.1109/ICSE.2019.00021.

[9] Z. Chen et al., "Sequencer: Sequence-to-sequence learning for end-to- end program repair," *IEEE*, vol. 47, no. 9, pp. 1943–1959, 2019, DOI: 10.1109/TSE.2019.2940179.

[10] N. Jiang, T. Lutellier, and L. Tan, "CURE: Code-aware neural machine translation for automatic program repair," in *2021 IEEE/ACM 43rd International Conferenceon Software Engineering (ICSE)*. IEEE, 2021, pp. 1161–1173, DOI: 10.48550/arXiv.2103.00073.

[11] T. Lutellier, H. V. Pham, L. Pang, Y. Li, M. Wei, and L. Tan, "CoCoNut: combining context-aware neural translation models using ensemble for program repair," in *Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis*, 2020, pp. 101–114, DOI: 10.1145/3395363.3397369.

[12] S. Chakraborty, Y. Ding, M. Allamanis, and B. Ray, "CODIT: Code editing with tree-based neural models," *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1385–1399, 2020, DOI: 10.48550/arXiv.1810.00314.

[13] Y. Li, S. Wang, and T. N. Nguyen, "Dlfix: Context-based code transformation learning for automated program repair," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 602–614, DOI: 10.1145/3377811.3380345.

[14] B. Poór, M. Toth, and I. Bozó, "Transformations towards clean functional code," in *Proceedings of the 19th ACM SIGPLAN International Workshop on Erlang*, ser. Erlang 2020. New York, NY, USA: Association for Computing Machinery, 2020, pp. 24–30, DOI: 10.1145/3406085.3409010.

[15] F. Baader and T. Nipkow, *Term rewriting and all that*. Cambridge University Press, 1998, DOI: 10.1017/CBO9781139172752.

[16] D. Horpácsi, J. Kőszegi, and S. Thompson, "Towards trustworthy refactoring in Erlang," Electronic Proceedings in *Theoretical Computer Science*, vol. 216, pp. 83–103, jul 2016, DOI: 10.4204/eptcs.216.5.

[17] H.-A. R. Project. (2023) Core Erlang formalization. Accessed on 3rd of August, 2023. [Online]. Available: https://github.com/harp-project/Core-Erlang-Formalization/releases/tag/v1.0.3

[18] D. Drienyovszky, D. Horpácsi, and S. Thompson, "Quickchecking refactoring tools," in *Proceedings of the 9th ACM SIGPLAN Workshop on Erlang*, ser. Erlang '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 75–80, DOI: 10.1145/1863509.1863521.

[19] N. Chirkova and S. Troshin, "A simple approach for handling out-of-vocabulary identifiers in deep learning for source code," 2020, DOI: 10.48550/arXiv.2010.12663.

[20] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International conference on machine learning*. PMLR, 2017, pp. 1243–1252, DOI: 10.48550/arXiv.1705.03122.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017, DOI: 10.48550/arXiv.1706.03762.

Balázs Szalontai has completed his bachelor's degree in computer science at Eötvös Loránd University in 2020, followed by a master's degree in the same field at the same university in 2022. He is currently a Ph.D. student. His research focuses on source code transformation with Deep Learning. He also gives classes for university students in the fields of classical Artificial Intelligence, Object Oriented Programming and Logic Programming.

Péter Bereczky is a Ph.D. student at the Faculty of Informatics, Eötvös Loránd University. He received his master's degree in computer science at the same university, in 2020. His research interest include formal verification, formal semantics of programming languages, formal logics, functional programming, and interactive theorem proving.

Dániel Horpácsi is an assistant professor at Eötvös Loránd University. He received his Ph.D. for developing methods for verification and application of program transformations in functional programming languages, especially refactoring in Erlang. He keeps exploring the pragmatics of using formal methods in practical software tools, making them more reliable whilst maintaining their flexibility and accessibility.

Deep Learning from Noisy Labels with Some Adjustments of a Recent Method

István Fazekas[1], László Fórián[2], and Attila Barta[2]

***Abstract*—In this paper we have used JoCoR, a fairly recent method for learning with label noise, that makes use of two neural networks with a joint loss function using an additional contrastive loss to increase the agreement between them. This method can be extended to more than two networks in a straightforward way. We have carried out experiments on the CIFAR-10 and CIFAR-100 datasets (contaminated by synthetic label noise) with this kind of extension using several contrastive losses. We have concluded that it makes a significant improvement if we use a third network, especially when we use Kullback-Leibler terms for all possible pairs of softmax outputs. Further extension also means some kind of improvement, but in the case of the CIFAR datasets, those were not so significant, maybe except the cases with lower ratio of label noise.**

***Index Terms*—Deep Learning, Noisy Labels, Classification, Neural Networks, Supervised Learning**

I. Introduction

DEEP neural networks have excellent performance in image classifications tasks, but they are in need of large sets of training data with correct labels. This is a drawback, since labeling is difficult or too expensive in many cases. The available datasets are often contaminated by label noise, that is why the challenge of learning with noisy labels has become an important research topic with several directions [1], [5]. Even though deep neural networks tend to learn the simple, consistent patterns first, they can easily overfit to noisy labels [2]. If we are able to prevent this overfitting and treat the label noise during the training process, we can obtain models with good generalization ability.

In this work, we have investigated the possibilities of the improvement of a recent method in the topic of learning with label noise. We have applied some modifications to the training process, evaluated those adjusted models and drawn conclusions from the results.

JoCoR [6] is one of the recent state-of-the-art techniques for learning with label noise. It uses the idea of the selection of small-loss samples along with the utilization of two neural networks and it gradually increases the agreement between them. This model is trained with two classifiers in the background and a joint loss function which contains an additional term to reduce the divergence of the two networks, they are forced to make similar predictions. This scheme has a regularization effect during the training, it plays an important role in preventing overfitting. The parameters of the networks are updated simultaneously by the joint loss function, which is a weighted sum of the supervised losses and the contrastive loss term. JoCoR shows very impressive performance on several datasets with label noise, including CIFAR-10 and CIFAR-100 with symmetric and asymmetric label noise.

The method of JoCoR can be considered as a special ensemble of the two classifiers. Unlike the techniques using a disagreement strategy ([3], [4], [7]), JoCoR can be naturally extended to more than two networks. This raises the question: is it worth to use JoCoR with three neural networks if we have the computational capacity?

One of our results is that the answer for the above question is yes; we were able to make a significant improvement in the considered symmetric and asymmetric noise cases on CIFAR-10 and CIFAR-100 using three networks and totally six Kullback-Leibler terms (for every possible pair of softmax outputs). Similar results were obtained by using only three KL terms in a circular manner, but the improvement of the model over the training process was slightly slower and the test performance seemed to have a larger variance. Cross-Entropy contrastive losses were also applied, however they led to moderately weaker performance with larger variance as well.

We have also experimented with the utilization of more networks. They made a slight improvement, too, but the further increase comes with the cost of larger computational needs and the benefits are not as significant as in the case of three networks. However, the improvement is relatively larger when we have a lower amount of label noise.

II. CIFAR-10 and CIFAR-100 with Synthetic Label Noise

The dataset CIFAR-10 consists of images from 10 classes with 32×32 RGB pixels. The size of the training set is 50000 examples and the test set has 10000 samples. For CIFAR-100, the size and quantity of the images are the same, but the number of classes equals 100. We also have 20 superclasses, each of them contains 5 classes.

The CIFAR-10 and CIFAR-100 datasets are used with synthetic label noise of two types: symmetric and asymmetric. We have experimented with two types of synthetic label noise:

- **Symmetric label noise**: a given proportion of the labels is flipped to one of the other classes according to a discrete uniform distribution.
- **Asymmetric label noise**: it is generated by taking pairs of classes (which are similar to each other, for which humans make some mistakes, too), and a proportion of the data labels are flipped between these class pairs.

[1] University of Debrecen, Faculty of Informatics, Debrecen, Hungary (E-mail: fazekas.istvan@inf.unideb.hu)

[2] University of Debrecen, Faculty of Informatics and Doctoral School of Informatics, Debrecen, Hungary (E-mail: forian.laszlo@inf.unideb.hu, barta.attila@inf.unideb.hu)

DOI: 10.36244/ICJ.2023.5.2

III. A Recent Method for Learning with Noisy Labels: JoCoR

JoCoR utilizes the idea of small-loss selection and uses two neural networks. The agreement between those networks is gradually increased during the training process, this was inspired by some semi-supervised learning methods. The model is trained using two classifiers and a joint loss function which contains an additional regularization term to reduce the divergence of the two networks, so they are forced to agree with each other. This setup also has a regularization effect during the training, and it helps to prevent overfitting, too.

JoCoR uses convolutional neural networks (CNNs) with several convolutional and batch-normalization layers in the background, but it can be changed to any other neural network. This backbone CNN can be seen on Fig. 1, the source is [6].

CNN on *CIFAR-10*& *CIFAR-100*
32×32 RGB Image
3×3, 64 BN, ReLU
3×3, 64 BN, ReLU
2×2 Max-pool
3×3, 128 BN, ReLU
3×3, 128 BN, ReLU
2×2 Max-pool
3×3, 196 BN, ReLU
3×3, 196 BN, ReLU
2×2 Max-pool
Dense $256 \longrightarrow 100$

Fig. 1. The network in the background of JoCoR

The loss function of JoCoR is a weighted sum of the supervised loss of the two networks (two Cross-Entropy terms) and a contrastive loss term. The latter quantity is a symmetric Kullback-Leibler divergence (the sum of two KL terms). Here the dataset is given with N samples from M classes as $D = \{\mathbf{x_i}, y_i\}_{i=1}^{N}$. $\mathbf{x}_i$ is the i-th instance with its observed label $y_i \in \{1, \ldots, M\}$. The formulas:

$$L(\mathbf{x}_i) = (1-\lambda) * L_{sup}(\mathbf{x_i}, y_i) + \lambda * L_{con}(\mathbf{x}_i),$$

$$L_{sup}(\mathbf{x_i}, y_i) = L_{C_1}(\mathbf{x_i}, y_i) + L_{C_2}(\mathbf{x_i}, y_i),$$

$$L_{con} = D_{KL}(p||q) + D_{KL}(q||p),$$

if p and q are the two discrete probability distributions obtained from the softmax outputs.

Images considered as clean are selected with the small loss criterion using this joint loss function. At the start, the whole training dataset is used, then fewer training examples are selected in the upcoming epochs until it gradually reaches the ratio $1-\tau$, where τ is the known or estimated ratio of the noisy labels in the training dataset.

IV. Extending the method to three networks

Since JoCoR can be considered as a special ensemble of the two classifiers and it can be extended to more than two networks in a natural way, we wanted to investigate whether it is worth to use three neural networks instead of the original two.

We have investigated the performance of JoCoR with three networks (we have used copies of the same CNN as JoCoR) and several types of contrastive loss (where p_1, p_2, p_3 are the softmax outputs). We have carried out experiments with the following contrastive loss setups:

Totally six KL-terms (for every possible pair of softmax outputs):

$$L_{con}(\mathbf{x}_i) = D_{KL}(p_1||p_2) + D_{KL}(p_2||p_1) + D_{KL}(p_1||p_3) + D_{KL}(p_3||p_1) + D_{KL}(p_2||p_3) + D_{KL}(p_3||p_2).$$

This can be considered the extension of JoCoR's contrastive loss to three networks and the force is quite strong for the classifiers to predict similarly.

Using three KL-terms in a circular manner:

$$L_{con} = D_{KL}(p_1||p_2) + D_{KL}(p_2||p_3) + D_{KL}(p_3||p_1).$$

This function came into consideration because the effect of using these three terms only, may lead to the same situation in the long run: the predictions of the classifiers should be quite similar at the end of the training process.

Three Cross-Entropy terms:

$$L_{con} = L_{CE}(p_1||p_2) + L_{CE}(p_1||p_3) + L_{CE}(p_2||p_3).$$

This loss function came up as an idea since the Cross-Entropy can also be considered as a distance between two discrete probability distributions.

A. Results on CIFAR-10

Table I contains our results on CIFAR-10 (and the results of the original JoCoR). We have implemented our experiments using PyTorch as the authors of JoCoR. We have used the Adam optimizer with momentum 0.8. The initial learning rate was 0.001 and the mini-batch size was set to 128. The number of epochs was 200 and the learning rate has started to decrease from the 80-th epoch and it was linearly decreased to 0 until the end of the training. The parameter λ was set to 0.5 in the case of 6 Kullback-Leibler terms and 0.7 for the setups with 3 Kullback-Leibler and 3 Cross-Entropy terms.

The models were evaluated on the 10000-element test dataset and these values are the averages (and standard deviations) of test accuracies of 10 inependent runs. We can see that it is worth using 3 networks with 6 Kullback-Leibler divergence terms, because we got higher accuracies and lower standard deviations. If we have the capacity, it may also be worth using the 3 Kullback-Leibler version: the results are similar, just the standard deviatons are slightly higher. We are also able to improve JoCoR's results with Cross-Entropy regularization terms, but the difference is smaller in that case.

TABLE I
THE CIFAR-10 AND CIFAR-100 TEST PERFORMANCE OF JOCOR WITH 2 NETWORKS AND WITH 3 NETWORKS USING THE MODIFIED CONTRASTIVE LOSSES

CIFAR-10				
	JoCoR	6 Kullback-Leibler	3 Kullback-Leibler	3 Cross-Entropy
Symm. 20%	85.73 ± 0.19%	86.95 ± 0.19%	86.75 ± 0.19%	85.90 ± 0.27%
Symm. 40%	79.41 ± 0.25%	80.49 ± 0.21%	80.46 ± 0.33%	79.96 ± 0.29%
Symm. 80%	27.78 ± 3.16%	29.02 ± 3.11%	28.10 ± 3.19%	28.02 ± 3.39%
Asymm. 40%	76.36 ± 0.49%	77.27 ± 0.41%	77.43 ± 0.57%	77.28 ± 0.57%
CIFAR-100				
	JoCoR	6 Kullback-Leibler	3 Kullback-Leibler	3 Cross-Entropy
Symm. 20%	53.01 ± 0.44%	54.15 ± 0.39%	54.13 ± 0.43%	53.18 ± 0.51%
Symm. 40%	43.49 ± 0.46%	44.01 ± 0.37%	44.01 ± 0.40%	43.51 ± 0.54%
Symm. 80%	15.49 ± 0.98%	16.23 ± 0.91%	16.15 ± 0.96%	16.07 ± 1.05%
Asymm. 40%	32.70 ± 0.35%	33.35 ± 0.27%	33.34 ± 0.33%	32.71 ± 0.37%

B. Results on CIFAR-100

Our results using three networks on CIFAR-100 are also summarized in Table I. JoCoR's original results are also present for comparison. The value of the hyper-parameters were the same as for the models using CIFAR-10 previously.

The models were evaluated on the 10000-element CIFAR-100 test dataset and these values are the averages (and standard deviations) of test accuracies of 10 runs. The 6 Kullback-Leibler version makes a significant improvement again, but the circular 3-term KL contrastive loss is not far behind, especially in the cases with lower noise ratio. The Cross-Entropy-type loss also means an improvement, but it results in slightly higher standard deviation and it seems to be more suitable for symmetric noise than asymmetric.

V. FURTHER INCREASE OF THE NUMBER OF NETWORKS

We have also investigated the possibilities of improvement by using more than 3 networks and contrastive losses with Kullback-Leibler divergence for all possible pairs of the softmax outputs. In table II we only report the averages of test accuracies of 10 runs. The parameters of the training were the same as before except the value of λ. It was set to 0.3 in the case of the 4-network model. Its value was 0.2 and 0.1 for 5 and 6 classifiers, respectively.

We can see that we were able to make a significant improvement with the third network, and the fourth classifier also makes an improvement, but the difference is not so large. The fifth network could also improve our results for smaller noise ratios, but these accuracies seem to be almost constant for larger models. It is also important to note that there was no significant decrease despite the complexity of computations.

Table II contains the results for more than 3 networks and Kullback-Leibler terms for all possible softmax output pairs, using the CIFAR-100 test dataset. We have used the same set of hyper-parameters that were used for CIFAR-10.

We can observe similar results on the CIFAR-100 dataset, too. The third network gave a significant improvement, but for more than 3 networks, the upgrade was not so significant (maybe except for the 4-part version with smaller amount of noise). The difference between the performance of the models is generally smaller for this dataset since this classification is a more difficult task.

The increase of the computational needs, execution costs of the larger models are approximately linear, so is has to be taken it into consideration when trying to increase the performance of the model.

VI. SOME CONSIDERATIONS ON THE λ HYPER-PARAMETER

The λ hyper-parameter is the weight of the contrastive loss in the overall loss function, hence it controls the force that pulls the predictions together. It also provides the regularization effect in our models. The larger the λ is, the less the divergence of the softmax outputs of the networks. However, if we set it too high, the classifiers make almost the same predictions which is not favourable, especially if we have several neural nets. The best λ depends on the dataset and the model as well. When obtaining the results in our previous tables, we have used the most suitable λ values out of $0.1, 0.2, \ldots 0.9$. As an illustration on CIFAR-10, we present the case of the model with 3 networks and 6 Kullback-Leibler terms and 20% symmetric label noise in Table III. That table also contains the average of the test performance of 10 runs.

VII. CONCLUSIONS

We have carried out experiments with JoCoR, a recent technique for learning with label noise, which can be naturally extended to more than the two networks it originally uses.

We were able to make a significant improvement in accuracy by utilizing a third network with a 6-term Kullback-Leibler contrastive loss. Despite the other types of used regularization losses had some drawbacks, they could also improve the original JoCoR model, especially in the scenarios with lower proportion of label noise.

If we use a contrastive loss with all the possible pairs of softmax outputs, we can further improve our results by increasing the number of neural networks on both CIFAR datasets, but those improvements can be considered not as significant as the case of the 3-classifier model.

ACKNOWLEDGEMENT

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

TABLE II
THE CIFAR-10 AND CIFAR-100 TEST PERFORMANCE OF JOCOR WITH 2-6 NETWORKS (AVERAGE OF 10 INDEPENDENT RUNS)

CIFAR-10					
	JoCoR	3 networks	4 networks	5 networks	6 networks
Symm. 20%	85.73%	86.95%	87.19%	87.26%	87.29%
Symm. 40%	79.41%	80.49%	80.80%	80.82%	80.86%
Symm. 80%	27.78%	29.02%	29.11%	29.11%	29.12%
Asymm. 40%	76.36%	77.27%	77.49%	77.49%	77.48%
	JoCoR	3 networks	4 networks	5 networks	6 networks
Symm. 20%	53.01%	54.15%	54.34%	54.38%	54.41%
Symm. 40%	43.49%	44.01%	44.15%	44.18%	44.20%
Symm. 80%	15.49%	16.23%	16.26%	16.29%	16.30%
Asymm. 40%	32.70%	33.35%	33.42%	33.44%	33.44%

TABLE III
THE CIFAR-10 TEST PERFORMANCE OF JOCOR WITH 3 NETWORKS AND 6 KULLBACK-LEIBLER TERMS FOR DIFFERENT λ VALUES, IN THE CASE OF 20% SYMMETRIC LABEL NOISE

λ	Performance
0.1	75.92%
0.2	76.21%
0.3	81.84%
0.4	84.97%
0.5	86.95%
0.6	86.43%
0.7	85.75%
0.8	84.24%
0.9	83.56%

REFERENCES

[1] G. Algan and I. Ulusoy, Image Classification with Deep Learning in the Presence of Noisy Labels: A Survey, *Knowledge-Based Systems* 215, 106771, 2021, DOI: 10.1016/j.knosys.2021.106771

[2] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio and S. Lacoste-Julien, A Closer Look at Memorization in Deep Networks *Proceedings of Machine Learning Research* vol. 70, 2017, pp. 233–242.

[3] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang and M. Sugiyama, Co-teaching: Robust training of deep neural networks with extremely noisy labels, *NeurIPS*, 2018, 8535–8545.

[4] E. Malach and S. Shalev-Shwartz, Decoupling "when to update" from "how to update", Advances in *Neural Information Processing Systems*, 2017, pp. 960–970.

[5] H. Song, M. Kim, D. Park, Y. Shin and J. Lee, Learning from Noisy Labels with Deep Neural Networks: A Survey *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] H. Wei, L. Feng, X. Chen and B. An, Combating Noisy Labels by Agreement: A Joint Training Method with Co-Regularization, *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2020, 13723–13732, DOI: 10.1109/CVPR42600.2020.01374

[7] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang and M. Sugiyama, How does Disagreement Help Generalization against Label Corruption?, *International Conference on Machine Learning*, 2019, 7164–7173.

István Fazekas graduated from Kossuth Lajos University, Debrecen, Hungary in 1978. He is currently a full professor at Faculty of Informatics, University of Debrecen, Hungary. He had been head of the Department of Applied Mathematics and Probability Theory. His main research interests are asymptotic theorems of probability theory and mathematical statistics, network theory and machine learning.

László Fórián graduated from University of Debrecen in 2019 as a mathematician. He is currently a PhD student at the Doctoral School of Informatics and an assistant lecturer at the University of Debrecen, Hungary. His main research areas consist of neural networks and random graphs.

Attila Barta is an assistant lecturer at University of Debrecen, Hungary. He holds an MSc degree in applied mathematics from the University of Debrecen and he is also a PhD candidate at the Doctoral School of Informatics in the institution. His main research fields are network science and neural networks.

Application of Neural Network Tools in Process Mining

László Kovács, Erika Baksáné Varga, and Péter Mileff

Abstract—**Dominant current technologies in process mining use schema induction approaches based on graph and au- tomaton methods. The paper investigates the application of neural network approaches in schema induction focusing on three alternative architectures: MLP, CNN and LSTM networks. The proposed neural network models can be used to discover XOR, loop and parallel execution templates. In the case of loop detection, the performed test analyses show the dominance of CNN approach where the string is represented with a two- dimensional similarity matrix. The usability of the proposed approach is demonstrated with test examples.**

Index Terms—**process mining, convolutional neural network, graph schema induction.**

I. Introduction

PROCESS mining is a technique to discover, analyze, and monitor processes in an objective manner. It uses log data from corporate information systems and turns them into insights and actions. In this sense process mining is a subfield of data science: it requires the availability of data and aims at improving processes [1]. The induction of complex schema or grammar models is a key challenge in knowledge engineering. The input for schema mining is a set of sequences (traces) and the engine determines the general schema graph covering the input set. The input for schema mining is given with an event log, which is a list of event traces. The traces of the event log are generalized, merged into a schema graph.

The first software systems for schema induction usually used a pattern matching approach to determine the common section in the sentences. In most cases, the induction was based on a set of transformation rules defined by human experts. Current toolsets for process mining are dominated by the graph or automaton oriented algorithms [2] usually starting with the construction of the Direct Follower Graph which is a graphical representation of a process. Traditional frequent pattern mining techniques run into their limits when dealing with massive datasets [3]. Therefore, more advanced algorithms apply tree-based representation [4]. These incremental methods derive relevant patterns recursively. One of the most widely used industrial approaches is the inductive mining algorithm [5], which uses a top-down discovery algorithm. The top-down method recursively decomposes the event log into smaller event logs. The method first converts the event log into a corresponding DFG (direct follow graph), then it simplifies the initial graph into a compact schema graph. Beside the graph-based standard approaches, we can find some recent approaches on application of neural networks in process mining [6]. We can consider graph schema induction as a special type of classification problem, where the final output category corresponds to a complete schema graph. The schema graph is usually constructed in an incremental way by selecting the next winner event or events and the edges in the graph denote the adjacency relation. Considering simple event sequences, the dominating approach for next event prediction is to use Recurrent Neural Networks (RNN), where the output signal of a processing step is used as input component for the prediction of the next element in the sequence. In [7], the RNN network model was applied for the process discovery task. Current solutions are able to detect XOR branches in the trace log, but these methods still miss some crucial functionality required by industrial problems. The main goal of this paper is to show our proposals on the adaptation of neural networks in process mining to cover some missing functionalities, like AND branches and loop detection [8].

The authors are with the Department of Information Technology, University of Miskolc, Miskolc, Egyetemváros, Hungary. E-mail: {laszlo.kovacs, erika.b.varga, peter.mileff}@uni-miskolc.hu

DOI: 10.36244/ICJ.2023.5.3

II. Background

A. Process Mining with Graph Methods

Today's era is about automation which also affects complex business and administrative processes. For the automation of processes, the most important prerequisite is to have process models which can be transformed into running systems. Thanks to process mining methods, the required models can be discovered by extracting knowledge from event logs recorded by information systems. The general process discovery problem is defined in [1] as finding an algorithm that maps an event log (L) into a process model so that the model is representative for the behavior seen in the event log. This task is highly challenging, since an input event log contains a collection of historical cases of a given process, called traces, which are sequences of actions while the expected output model should be compact and meet the requirements of fitness, precision, generalization and simplicity. In order to produce a generalized model, the control-flow structures underlying in the data sequences should be discovered.

The control-flow patterns used in process modelling are presented in [9] and [10]. The identified 43 patterns are classified into 8 classes, amongst which the basic patterns that can be explored in the event log are:

- sequence,
- parallel split,
- synchronization,
- exclusive choice, and
- simple merge.

In sequential routing, a task in a process is enabled after the completion of the preceding task. There can be two different types of splitting nodes in a process model. Parallel split is applied when a single thread of execution is split into two or more branches which are triggered concurrently (AND node). Exclusive choice corresponds to conditional routing. In this case, as soon as the incoming branch is enabled, the thread of control is passed to exactly one of the outgoing branches (XOR node). The splitted branches need to be merged after a while, i.e. an AND or XOR node are generally succeeded by a synchronization or simple merge node, respectively. The difference between these nodes is that a synchronizing node means that all the activated incoming branches should be completed before the thread of control is passed forward; while a simple merge node waits for only one branch to be completed [11].

Graphical process models usually have special notations for the elements that influence the flow of control. These nodes interconnect activities in the conceptual model, but there is no sign of them in the event log. This makes the process discovery problem challenging, since the developed algorithms have to explore the control structures hidden in the event sequences.

B. Standard Tools in Process Mining

Process mining tools utilize existing data from corporate information systems to provide dynamic visualization of the processes and to perform process mining tasks such as process discovery and conformance checking.

The first framework that has been designed to combine different process mining algorithms, is ProM [12]. This framework is flexible regarding the input and output format, and allows for easy implementation of process mining methods. Three types of graph-based algorithms can be plugged in. One can explore the process perspective to find a general characterization of all possible paths, expressed for example in terms of a Petri net or Event-driven Process Chain (EPC). With respect to the organizational perspective, the mining method either structures the organization by classifying people in terms of roles and organizational units, or maps the relations between them. Thirdly, cases can also be investigated by characterizing them with their path in the process, by the agents working on them, or by the values of the corresponding data elements.

A more sophisticated tool is the open source PM4PY developed by the Fraunhofer Institute using Python [13]. It supports several input and output formats, and a variety of process discovery algorithms creating procedural process models (such as the Alpha Miner, the Inductive Miner and the Heuristics Miner algorithms) or methods producing descriptive models. It also allows for conformance checking, object-centric process mining, organizational network analysis and provides some features useful for the application of machine learning techniques. Most recently, PM4PY offers some integrations with OpenAI (e.g. with ChatGPT) for getting insights automatically.

However, a great disadvantage of traditional graph-based process mining methods is the use of recursion which results in high memory consumption and long execution time. In our approach, we apply approximation in discovering a process model by means of neural networks.

C. Process Mining with Neural Network Methods

Neural networks can be utilized in process mining to enhance various aspects of the analysis and optimization of business processes. One of their applications is when they are used to predict future events in a business process. Compared to traditional process mining techniques, such as Petri nets and the Business Process Model Notation (BPMN), deep learning methods have proven to achieve better performance in terms of accuracy and generalization power.

In [14], Hanga et al. propose a method that combines the benefits of visually explainable graph-based methods with more accurate deep learning methods. Among neural networks, the RNN architecture has the capability to provide the context for each following prediction that helps in preserving the state in which the decision was made. Therefore, in this approach, an LSTM model is employed first to find probabilities for each known event to appear in the process next. These probabilities are then used to generate a graphical process model graph.

Obodoekwe et al. [15] used convolutional neural network (CNN) for predicting the next activity in an event trace. The method first detects the spatial structure within the order of historical event sequences and then transforms them into 2D images. The images are then trained using the CNN network to generate a deep learning model that can predict the next activity in an ongoing process. The feasibility of the approach was evaluated using Helpdesk event logs and the results show that the proposed CNN-based method provides highly accurate next activity prediction and is faster in training and inference than the LSTM-based approach.

The most widely studied problem of process mining is automatic process discovery. Several approaches have been proposed, but the applicability and effectiveness of these approaches depend on event log features and the structure of the processes. When applied to real-life event logs, the majority of traditional process discovery methods produce broad and spaghetti-like models, or models with poor fitness and precision [1].

Shunin et al. [7] try to find patterns in event logs using a neural network. The algorithm extracts an RNN's internal state as the desired transition system that describes the behaviour present in the log. One of the main advantages of using this architecture is its natural ability to detect and merge common behavioural parts that are scattered across the log. Another benefit is that the models derived by the approach absolutely fit to the event log.

Sommers et al. [16] applied graph neural networks in process discovery. They encode the discovery problem as a graph of three parts. The first part of the graph is the trace graph representing the event log. The second part of the graph is a candidate Petri-net, which is a possible result model. The third part of the graph are the links from the event nodes of the trace graph to transitions in the candidate model. Their approach is to utilize a collection of neural networks, each of

which takes a segment of the graph as input and simulates a distinct stage in the gradual construction of a Petri-net model from an event log. The method was evaluated on synthetic and real-life data and compared to other methods, achieving the highest simplicity while competing with Inductive Miner, Heuristic Miner, and Split Miner methods in terms of F-score.

In this paper, we present a neural network architecture for the detection of AND (parallel) branches in the event log. With the help of this approach, the engine can determine the parallel processes in the traces and it builds up a process schema graph related to more actors. The proposed engine uses a two-level representation approach where the bottom level corresponds to the single actor level activity chain. These homogeneous segments are merged by using synchronization nodes. The top level contains the synchronization graph of the agent level segments. The second neural network architecture presents a novel loop detection approach which can be used to discover tandem repeat sections. The proposed method first converts the sequence into an image matrix format and this matrix will be sent to a CNN convolutional network. The category labels correspond to the different loop kernel positions in the sequence. The next model integrates these two neural network models to have a more general schema induction engine for process mining. The presented novel network models were implemented in Python Tensorflow/Keras framework. In order to test the induction engines on event logs of different complexity levels, a test set generator application was developed in the research project. The target schema is constructed with a visual editor, and the engine generates the related event log of given size. In the test experiments, we compared the efficiency of our proposed models with some standard graph based engines. In the paper, we analyze the test results showing both the benefits and limitations of the neural network approach.

In the tests we used three main neural network architectures:

- multi-layer perceptron model (MLP), which is a standard NN tool for both general classification and regression,
- recurrent neural model (LSTM), and
- convolutional neural model (CNN).

The detection of branching nodes of the schema graph can be implemented either with the standard MLP network or with the LSTM recurrent model. The main goal of the neural network is to predict the next element of the investigated sequence. The prediction is based on the previous elements, thus the input vector is given as the description vector of the preceding elements. In the standard approach, the engine outputs only the winner category, in this way generating only one next element. According to the literature [17], the standard way of generating sequence branching is to consider not only a single winner category, but a group of best candidates. The size of the winner group is usually an input parameter in the algorithm.

III. Discovery of Parallel Branches

Parallelism denotes parallel workflows of different actors and resources, and the split and join control nodes denote an artifact level dependency among the different branches. For example, we consider a workflow to produce a mobile phone. In this process the production of the different components can be executed in a parallel way. A synchronization join node denotes the case when the next assembly step requires the availability of all components produced in the preceding steps. These control nodes can be automatically discovered if the event log contains an artifact attribute as well. The artifact attribute identifies the target, i.e. the object of the given action. Using this parameter we can discover the artifact level dependency between the different actions of the event log.

Beside the actor events, the extended input event graph for the training process contains also synchronization control nodes which describe the adjacency relationship among the event sequences. We assume that every control node has an input set of event sequences and an output set of event sequences. Similarly to Petri-nets, the join control node is triggered only when all of the input sequences are finished. If the transition is triggered, all output sequences will start the execution.

The event graph structure is defined as

$$\sigma = (N_\sigma, \rightarrow_\sigma)$$

where

- $e \in W \times A$ is an actor event where $A(e)$ denotes the actor of the event, $T(e)$ denotes the timestamp of the event and W denotes the set of event types (activities)
- $c \in C$ is a control event, every c has a timestamp denoted by $T(c)$;
- $E_\sigma = (e_1 e_2 ... e_k : e_i \in W \times A, \forall i, j : A(e_i) = A(e_j))$: the actor event sequence node;
- $C(c)$: the control event nodes;
- $N_\sigma = E_\sigma \cup C_\sigma$: the nodes in the graph instance;
- $\rightarrow_E \subseteq E_\sigma \times C_\sigma$: the edges from actor events to control events;
- $\rightarrow_C \subseteq C_\sigma \times E_\sigma$: the edges from control events to actor events;
- $\rightarrow_\sigma = \rightarrow_E \cup \rightarrow_C$: the edges in the graph.

In the preprocessing phase of the proposed method, the engine determines the related synchronization nodes first. Node mining is based on the following considerations:

1) Event nodes are processed in temporal order, and the current event is denoted by e_i.
2) The set of output artifacts of e_i are stored in out_i.
3) The set of adjacent events $E_j = \{e_j\}$ are determined, where the input artifacts correspond to some elements of out_i, and there is no other intermediate event processing these artifacts.
4) If the actors of the matching e_i and e_j event pair are different, a synchronization node e_s is needed between them.
5) The actor of e_i (a_i) is added to the input-actors of e_s, and the output actor set of e_s is extended with a_j.
6) If an actor is not present in the actor set of e_s while being active at e_s through an artifact-level dependency with some events in the output-actor list of e_s, then it will be included into the input-actor set of e_s. A similar method can be used to extend the output-actor list of e_s.
7) The previous two steps are repeated until a closure of the input-output artifact relationships is achieved at e_s.

TABLE I
ACCURACY COMPARISON OF SEQUENCE PREDICTION NETWORKS

Dataset	LSTM	MLP	NH-MLP	BE-MLP	BE_LSTM
pdc_2016_1.xes	63.5%	63.4%	63.4%	63.9%	64.1%
load_random	58.2%	57.5%	56.6%	58.7%	58.0%
pdc_2016_9.xes	82%	82.5%	81.2%	81.8%	82.6%
pdc_2017_5.xes	62.4%	62.8%	63.7%	64.8%	64.2%
pdc_2019_2.xes	64.6%	65.2%	63.9%	66.27%	65.6%

Considering the efficiency of the proposed models, we performed a comparison test on benchmark sequences. The investigated network models are:

- MLP : baseline MLP,
- LSTM : baseline LSTM,
- BE-MLP : MLP with union-based reduction,
- BE-LSTM : LSTM with union-based reduction,
- HN-MLP : MLP with NN-based reduction.

The first architecture is based on the standard multi layer perceptron neural network model (MLP) which is suitable to perform feature vector based value prediction. This model uses a relatively simple architecture, thus the training has a lower cost. The second version uses a recurrent neural network architecture (LSTM) that is used for prediction on value sequences of arbitrary length. It contains a more complex architecture that enables remembering of previous events in an efficient way. Due to the higher complexity, it is usually harder to find the optimal network parameters. The three other versions apply a sequence reduction preprocessing step. In this reduction phase, the long event sequence is reduced to a shorter one, where the reduction can be based on the value aggregation (MLP with union or LSTM with union) or an extra MLP neural network is trained to perform the reduction step (MLP with NN-based reduction).

Regarding the implementation parameters, the model consists of beside the output or hidden Dense layers, also an LSTM layer. For the hidden layers, we have used the relu activation function, while the output layer used the softmax activation function. Regarding the optimisation parameters, we applied the Adam optimizer with the categorical crossentropy loss function with the accuracy metrics. For the test evaluation of the different methods, we used the benchmark datasets for the Process Discovery Contest events. These competitions are organized by the IEEE Task Force on Process Mining Group . The name of the data file refers to the year of the contest and to the index of the data file. The dataset can be downloaded from the homepage of the contest (*https://www.tf-pm.org/competitions-awards/discovery-contest*). The datafiles contain the event logs in XES standard format.

The results of the comparison tests are presented in Table I. The accuracy values are given in percentage unit. The main conclusion is that the BE_MLP network type is a good choice for our architecture as

- it provides the best accuracy for most of the benchmark datasets, and
- it has the lowest execution cost.

IV. LOOP DETECTION

Loop detection is a key task in schema mining of event logs. The basic operation in loop detection is the discovery of tandem substrings. The task of tandem substring detection can be given with the following formal description. Having an alphabet A, strings are the final sequences based on A:

$$s = a_1, a_2, ..., a_m, a_i \in A.$$

A substring of s is

$$s' = a'_1, a'_2, .., a'_k$$

if

$$\exists i : a_i = a'_1, ..., a_{i+j-1} = a'_j, ..., a_{i+k-1} = a'_k.$$

A substring s' is a tandem substring if

$$\exists i : a_i = a_{i+k} = a'_1, ..., a_{i+j-1} = a_{i+k+j-1} = a'_j, ..., a_{i+k-1} = a_{i+2*k-1} = a'_k.$$

In order to show the ability of a Neural Network architecture to detect loop structures in sequences, we have developed a special network type using the CNN convolutional architecture. The main drawback of the baseline MLP approach is that it is very sensitive to the specific values at each position of the sequence. On the other hand, loop detection should be insensitive to the actual character values. Thus, for example, the sequences ”abcbch” and ”bagagb” need to be equivalent as both contain a loop at the second position. Otherwise it would take a long effort to generate a suitable training set with a good covering for cases of repetition.

In the initial tests, we have compared three candidate neural network architectures: multi-layer perceptron, convolutional CNN and recurrent LSTM. In the case of MLP, the input vector is the one-hot encoded version of the investigated string. For each position, a one-hot encoded vector of size M is given where each position denotes an element of the alphabet. The output vector describes all possible repetition positions, where each position corresponds to a (start position, end position) pair.

The input vector for CNN is given by a two-dimensional similarity matrix. The position (i, j) contains the similarity value of the symbols at the position i and at the position j. Thus, the string is converted into a two-dimensional image matrix, where the repeat sections can be characterized by a set of special lines in the image. The output format is similar to the vectors used in the MLP network architecture.

Considering the applied neural network architecture for loop detection, we used a CNN model with 14 layers including the core convolutional layers and the maxpooling layers. The architecture model is presented in Fig 1.

In the case of LSTM, the input vector contains the one-hot encoded format of the string, just like in the MLP network. Here, the output is a number for each position, denoting the size of the repeat section at the given position. Thus, the processing of the string will generate a sequence of numbers where a value greater than 1 denotes a repetition.

The results of the performed accuracy tests are summarized in Table II.

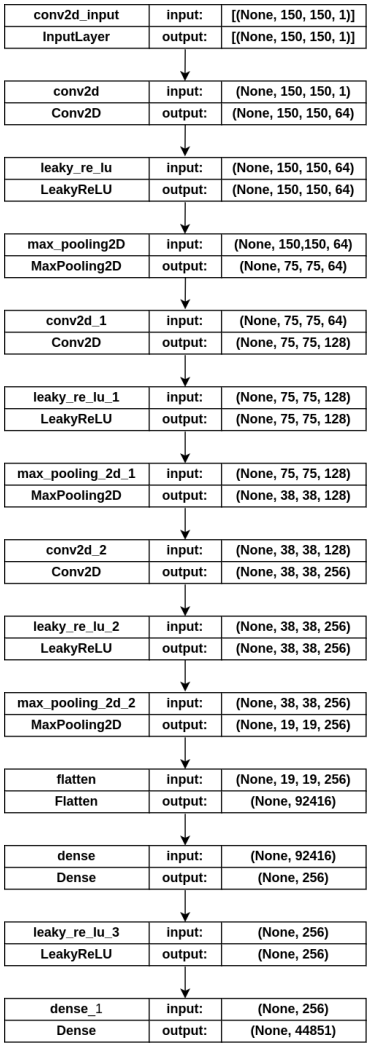


Fig. 1. Architecture of the CNN neural network for loop detection.

TABLE II
EFFICIENCY COMPARISON OF THE EXAMINED NN ARCHITECTURES

Method	N parameters	Time [s]	V accuracy	T accuracy
MLP	27,590,861	26	96%	8%
CNN	175,631,337	288	100%	99%
LSTM	2,527,212	359	96%	94%

In Table II the columns denote the following measures:

- N parameters: the complexity of the network, i.e. the number of graph parameters.
- Time: execution time of an epoch in seconds.
- V accuracy: validation accuracy.
- T accuracy: test accuracy.

Based on the preformed tests, we can see that the CNN network is the winner of the neural network approach. One important remark on the results is the very large differences between the test and verification accuracy values for the MLP architecture. This shows that there is significant overfitting in this case.

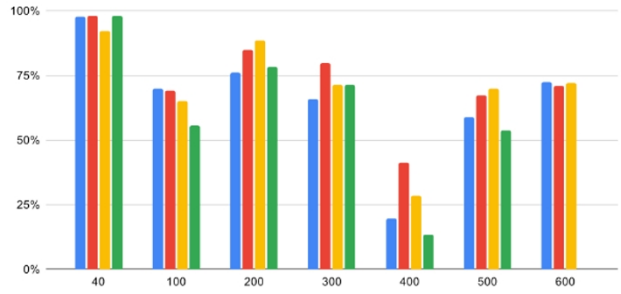


Fig. 2. Accuracy test of the different reduction methods.

Considering the winner CNN variant, repetition appears in the image as a line of pixels parallel to the main diagonal. Using this representation format, CNN can be used to recognize repetitions. One key issue in the CNN approach is the high memory cost, as for a string of length m the number of input neurons of the network will be m^2. Thus, we can reach our hardware limits relatively quickly during CNN training or search for repetitions. In order to overcome this problem, we have applied the following reduction methods:

- Bitmap-based reduction,
- Average aggregator, and
- Adjusted reduction.

In the tests, we have compared three reduction methods. In the first one, (bitmap reduction), in the initial bit matrix, the submatrices are replaced with a single cell where the value is calculated with the max operator. In the second version, the content of the submatrix considered as a matrix of float values and the average value will be stored in the reduced matrix. The third variant applies a special calculation which is sensitive to the diagonal directions as the loops generate specific lines (parallel to the diagonal) in the input matrix.

The test results for comparing the efficiency of the different reduction methods are presented in Fig. 2. In this figure, the X axis shows the length of the pattern window, and the Y axis denotes the achieved classification accuracy. The leftmost bar is for the Adjusted reduction method, the second is for the Average aggregator, the next is for the Bitmap-based reduction method, and the last one is for the baseline method without reduction. Based on the performed tests, the winner approach is the Average aggregation method.

Another way for reducing memory usage is to apply a different representation format. We have tested two additional variants, namely

- A: Neural network to detect whether the word contains repetition or not. In this case, the output vector contains only two dimensions.
- B: Neural network to detect the start (or end) position of the repeat section, where the size of the input vector is of linear cost.

Based on the performed tests, we can say that the CNN method significantly dominates the other variants in accuracy parameters. The test results are summarized in Table III.

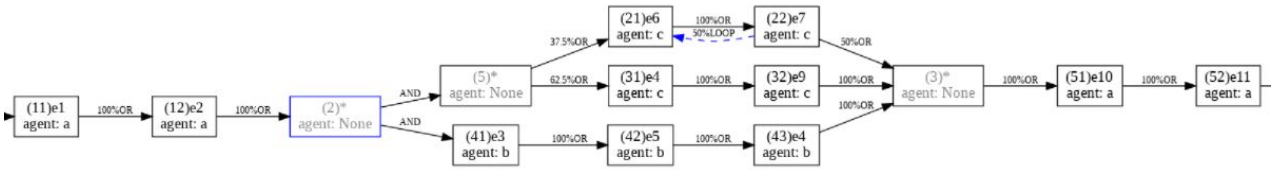


Fig. 3. Sample workflow schema B.

TABLE III
EFFICIENCY COMPARISON OF THE EXAMINED NN ARCHITECTURES USING ALTERNATIVE REPRESENTATION FORMATS

Task	Method	Time [s]	V accuracy	T accuracy
A	MLP	17	100%	87%
A	CNN	12	100%	100%
B	MLP	30	100%	6%
B	CNN	50	100%	100%

V. Implementation and Tests

In the tested model, three actors are defined: a,b and c. The list of available events, which are given by their IDs, actors and the related execution time intervals are as follows:

$$e1(a),(1,2) \quad e2(a),(3,5) \quad e3(b),(1,4) \quad e4(c),(2,3)$$
$$e5(b),(4,5) \quad e4(b),(5,6) \quad e6(c),(1,3) \quad e7(c),(2,4)$$
$$e9(c),(1,4) \quad e10(a),(3,4) \quad e11(a),(2,4)$$

The graph structure of the schema is presented in Fig. 3. For the training phase, 1000 cases were generated for schema B.

In the first processing phase, the engine determined the occurrences of synchronization events. The module then processed the traces in the log, and generated a list of synchronization events for each trace.

Based on the generated synchronization event sequences, as input training set, the constructed neural network model M will predict the following synchronization sequence:

```
['EMPTY', 'C2', 'C3', 'EOS']
```

In the next phase, the engine predicts the agent-level sequences in the following form:

```
a : ['e1', 'e2', 'EOS']
b : ['e3', 'e5', 'e4', 'EOS']
c : ['e4', 'e9', 'EOS']
a : ['e10', 'e11', 'EOS']
```

After performing more trace generation experiments, the action list of agent c may vary. A typical output is the action chain containing actions 'e6' and 'e7':

```
c : ['e6', 'e7', 'e6', 'e7', 'e6', 'e7',
     'e6', 'e7', 'e6', 'e7', 'e6', 'e7',
     'e6', 'e7', 'EOS']
```

During the prediction, the neural network outputs the prediction weights for the different event categories. The proposed measure weight ratio expresses the relation of these weights,

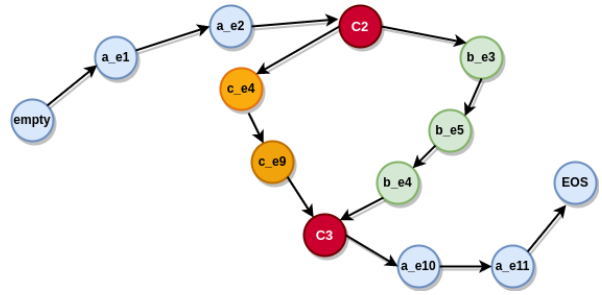


Fig. 4. Generated schema graph for single selection case.

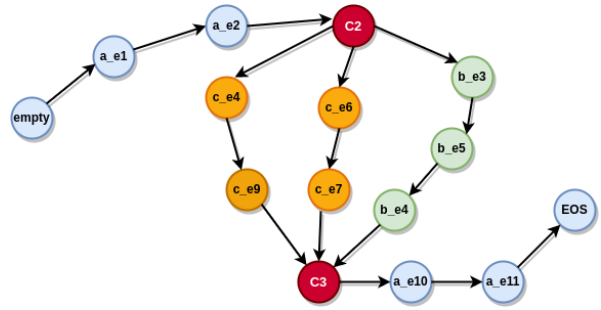


Fig. 5. Generated schema graph for multi-selection case.

the higher is this ratio the better is the prediction quality. If we consider the weight ratio between the best and the second best event sequences for agent c, we can see that most of the steps are unambiguous, except for the starting event which has a very low weight ratio. Namely, the higher the ratio, the stronger is the unambiguity. The next list shows these ratio values at the different steps for the best event sequence of agent c:

```
y e4  , weight ratio: 1.464207
y e9  , weight ratio: 234.37918
y EOS , weight ratio: 237.77512
```

Considering the resulting graph for a single selection case (only the best candidate is selected as next event), we can see that it contains a parallel execution section, where both agents b and c are active (Fig. 4). If we use a multi-selection prediction approach, where alternative routes are allowed, we get the graph presented in Fig. 5.

If we compare these graph results with the standard inductive miner approaches, we see that the proposed variant can manage more functionality and provides a more accurate result for the tested example than the standard solution (Fig. 6).

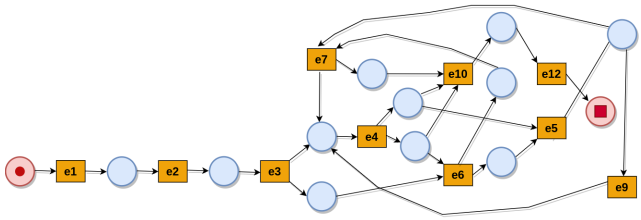


Fig. 6. Schema generated by the Inductive Miner.

VI. Conclusion

Beside the standard automaton oriented approaches, the neural network architectures are good alternatives for automated process schema mining. The analysis presented in this paper shows that the proposed neural network architectures are suitable to perform the same schema induction task as the standard graph-based tools. The proposed methods build up the correct schema graphs. The key benefit of the proposed NN approach is that it can discover also parallel sequences related to different agents in the schema graph, unlike the usual schema induction methods. The further research is aimed at optimisation of the neural network architecture for larger graph schema.

References

[1] W. Van der Aalst, *Process mining: Data science in action*. Springer: Heidelberg, 2016. https://doi.org/10.1007/978-3-662-49851-4

[2] J. Liu, S. Yan, Y. Wang, and J. Ren, "Incremental mining algorithm of sequential patterns based on sequence tree," *Advances in Intelligent Systems,* pp. 61–67, 2012. DOI: 10.1007/978-3-642-27869-3_8

[3] T. Truong-Chi and P. Fournier-Viger, "High-utility pattern mining: Theory, algorithms and applications," *A Survey of High Utility Sequential Pattern Mining*, pp. 97–129, 2019. DOI: 10.1007/978-3-030-04921-8

[4] X. Liu, L. Zheng, W. Zhang, J. Zhou, S. Cao, and S. Yu, "An evolutive frequent pattern tree-based incremental knowledge discovery algorithm," *ACM Transactions on Management Information Systems*, pp. 1–20, 2022. DOI: 10.1145/3495213

[5] Y. Lu, Q. Chen, and S. Poon, "A novel approach to discover switch behaviours in process mining," *International Conference on Process Mining*, pp. 57–68, 2021. DOI: 10.1007/978-3-030-72693-5_5

[6] H. Weytjens and J. D. Weerdt, "Process outcome prediction: CNN vs. LSTM (with attention)," *International Conference on Business Process Management*, pp. 321–333, 2020. DOI: 10.1007/978-3-030-66498-5_24

[7] T. Shunin, N. Zubkova, and S. Shershakov, "Neural approach to the discovery problem in process mining," in *Analysis of Images, Social Networks and Texts*, 07 2018, pp. 261–273. DOI: 10.1007/978-3-030-11027-7_25

[8] M. Kirchmer and P. Franz, "Value-driven robotic process automation (RPA)," *Business Modeling and Software Design*, vol. 356, pp. 31–46, 2019. DOI: 10.1007/978-3-030-24854-3_3

[9] W. Van der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003. DOI: 10.1023/A:1022883727209

[10] N. Russell, A. H. M. Hofstede, W. van der Aalst, and N. Mulyar, "Workflow control-flow patters – A Revised View," *Business*, vol. 2, pp. 06–22, 2006. DOI: 10.1.1.93.6974

[11] N. Russell, W. van der Aalst, and A. Ter Hofstede, *Workflow patterns : the definitive guide*. MIT Press, 2016. DOI: 10.7551/mitpress/8085.001.0001

[12] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The prom framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005*, G. Ciardo and P. Darondeau, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 444–454. DOI: 10.1007/11494744_25

[13] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process mining for Python (pm4py): Bridging the gap between process-and data science," in *Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019)*, 2019. DOI: 10.48550/arXiv.1905.06169

[14] K. M. Hanga, Y. Kovalchuk, and M. M. Gaber, "A graph-based approach to interpreting recurrent neural networks in process mining," *IEEE Access*, vol. 8, pp. 172 923–172 938, 2020. DOI: 10.1109/ACCESS.2020.3025999

[15] E. Obodoekwe, X. Fang, and K. Lu, "Convolutional neural networks in process mining and data analytics for prediction accuracy," *Electronics*, vol. 11, no. 14, 2022. DOI: 10.3390/electronics11142128

[16] D. Sommers, V. Menkovski, and D. Fahland, "Process discovery using graph neural networks," in *IEEE International Conference on Process Mining (ICPM)*, 2021. DOI: 10.48550/arXiv.2109.05835

[17] J. Abonyi, R. Károly, and G. Dörgö, "Event-tree based sequence mining using lstm deep-learning model," *Complexity*, vol. 2021, p. 24, 2021. DOI: 10.1155/2021/7887159

Prof. László Kovács from University of Miskolc, Department of Information Technology is a specialist in knowledge modeling and data mining. He obtained a PhD degree in technical sciences from Uni. Miskolc. Main teaching areas: DB Systems, Data Mining and Ontology Management. His research interests involve soft computing, concept set management, heuristic optimizations, ontology modeling, stat. grammar induction and rough set models. He has about 250 publications with 450 references. He is the Head of the Dept. and leader of the Research Group on Machine Learning at Uni. Miskolc.

Erika Baksáné Varga is an associate professor at the Institute of Informatics, University of Miskolc, Hungary. She received a Ph.D. degree in Computer Science from the University of Miskolc in 2011. She has academic experience in teaching procedural and object oriented programming, and data analysis and data mining. Her research interests include teaching methodologies of programming, data and process modeling, data analysis and data mining, ontological modeling, NLP and text mining.

Péter Mileff is a senior SW developer and an associate professor at the Institute of Informatics, University of Miskolc, Hungary. He obtained his Ph.D. in Computer Science in 2008. He has more than 10 years of teaching experience in software development and the building and design of software systems. He has professional and research experiences in computer visualization and game development; and 15 years of industrial experience in the design and development of digital payment systems and platform-independent technologies.

Special Issue of the Infocommunication Journal

Comparative Study of Interpretable Image Classification Models

Adél Bajcsi*, Anna Bajcsi*, Szabolcs Pável*†, Ábel Portik*, Csanád Sándor* †, Annamária Szenkovits*, Orsolya Vas*, Zalán Bodó*, and Lehel Csató*

***Abstract*—Explainable models in machine learning are increas- ingly popular due to the interpretability-favoring architectural features that help human understanding and interpretation of the decisions made by the model. Although using this type of model – similarly to "robustification" – might degrade prediction accuracy, a better understanding of decisions can greatly aid in the root cause analysis of failures of complex models, like deep neural networks.**

In this work, we experimentally compare three self-explainable image classification models on two datasets – MNIST and BDD100K –, briefly describing their operation and highlighting their characteristics. We evaluate the backbone models to be able to observe the level of deterioration of the prediction accuracy due to the interpretable module introduced, if any. To improve one of the models studied, we propose modifications to the loss function for learning and suggest a framework for automatic assessment of interpretability by examining the linear separability of the prototypes obtained.

***Index Terms*—deep learning, image classification, interpretability, self-explainable models.**

I. Introduction

Explainable artificial intelligence (xAI) is a large set of methods that allows humans to understand the results of a machine learning algorithm [1], [2]. Explainability is defined as the process of making humanly understandable the decision of a machine learning model. Namely, it is the study of relations between the model's decisions and intermediate data representations aimed at better understanding system decisions. Thus, XAI can help us to ensure that the system we built, relying on machine learning algorithms, deep learning, or neural networks, is working as expected. In the literature, there are several techniques for achieving and increasing the explainability of machine learning models, and they differ in their approach and the type of machine learning model used. Among these, the focus will be on self-explainable neural networks, which can be used to increase the transparency of the learning process.

In our work, we studied three self-explainable models: PrototypeDL, ProtoPNet, and BagNet; the primary goal was their evaluation on two datasets: MNIST, which is a dataset of handwritten digits, and BDD100K, which is the largest driving video dataset with 100 000 color images of high resolution. A result of the comparison is the suggestion of possible improvements by examining some components of the models from different aspects, like components of the loss function, and the study of the separability of the prototype vectors that these models use.

In Section I-A we introduce the notion of explainable models in deep learning and in Section I-B we discuss the notion of interpretability in image classification and describe the architecture and operation of our three selected models (PrototypeDL, ProtoPNet, and BagNet). Section I-C contains the experimental comparison of the models, while Section I-D discusses the methods we have used to measure the interpretability of the models and their possible improvements. The concluding Section II enumerates our conclusions from the experiments, as well as the formulation of possible future research directions.

A. Explainable models in deep learning

The increasing popularity of using machine learning models for critical applications like autonomous driving systems or medical diagnosis, suggests an imperative need for methodologies that can help to understand and evaluate the predictions of these models. The main drawback of current state-of-the-art deep neural network models is the lack of reliability and the lack of interpretability of their decisions.

According to a recent overview by [3], *XAI* is a field of AI that aims at providing automated explanations for each decision made by the system. These models can be divided into two types: *post-hoc* and *build-in* methods.

To explain a black-box system, we can start after the training process concluded – in a post-hoc manner: the *linear proxy models* – like e.g. LIME [4] – use local linear models based on the data from the original model (using perturbed inputs). The method can be used to identify the regions of the input that most influence the decision. Decision trees [5] and other rule extraction techniques – like *if-then* rule extraction [6] – increase the transparency of neural networks, however, they are hard to construct.

The concept of *saliency map* was introduced in [7], [8]. The authors created an intensity map illustrating the most/least important pixels or regions used in the computation of the output. Since the training phase is completely independent of the interpretation or explanation stage, the above methods are called post-hoc explainability models.

In this work we focus on self-explainable models for image classification, therefore the system provides visual clues next to the decision. The term self-explainable means that the

* Faculty of Mathematics and Computer Science, Babeş-Bolyai University of Cluj-Napoca, Romania
† Robert Bosch SRL, Cluj-Napoca, Romania

DOI: 10.36244/ICJ.2023.5.4

explainable part is built in during training. As this domain becomes more important, there were different methods used for explainability. Attention mechanism's [9], [10] main idea is to introduce attention weights over the input sequence to prioritize the set of positions where relevant information is present for generating the next output token. Dosovitskiy et al. [11] introduce the Vision Transformers networks (ViT), inspired by the attention mechanisms [12], where a transformer-based model is used in image classification. When pre-trained on large datasets and transferred to smaller image recognition benchmarks, the model outperforms state-of-the-art CNN networks. Disentangled representations have individual dimensions that describe meaningful and independent factors of variation like variational autoencoder [13], Beta-VAE [14], InfoGAN [15]. Disentangled units can be used to create interpretable CNNs with individual units that detect coherent and meaningful patches instead of difficult-to-interpret mixtures of patterns. Deep networks can also be designed to generate human-understandable explanations as part of the explicit training of the system. In this paper, we highlight three such models and examine their performance and explanations.

B. Interpretability in image classification

According to [3], an *interpretable model* in case of classification details the internals of the decision-making process that is understandable. A model is considered *explainable* when it answers the question "Why the output?". The answer is satisfying when – quote: "one could no longer keep asking why". In the field of image classification, it is achieved by highlighting the region that is likely to be responsible for the prediction.

In this section, three models are presented where the self-explainable part was a feature due to the construction of the model.

1) PrototypeDL: presented in [16], is a self-interpretable neural network architecture for image classification aimed at creating a deep learning architecture that "naturally" explains the reasoning behind each prediction. The architecture contains an autoencoder and a prototype classification network. The classifier network has three different layers: a so-called *prototype layer*, a fully connected layer, and a softmax layer. The encoder reduces the dimensionality of the input and allows making comparisons within the latent space. The decoder restores the encoded input allowing to visualize elements from the latent space – e.g. the learned prototypes. Let $\mathcal{D} = [\boldsymbol{X}, \boldsymbol{Y}] = \{(x_n, y_n)\}_{n=1}^{N}$ be the training dataset, where $x_n \in \mathbb{R}^p$ and $y_n \in \{1, \ldots, K\}$ denote the predictor and target variables, respectively. The training objective has four terms:

$$\begin{aligned} L_{PDL}(\mathcal{D}) = &\ \mathrm{CrossEnt}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) \\ &+ \lambda_0\ \mathrm{Rec}(\boldsymbol{X}) \\ &+ \lambda_1\ \mathrm{R}_1(\mathcal{P}, \boldsymbol{X}) + \lambda_2\ \mathrm{R}_2(\mathcal{P}, \boldsymbol{X}), \end{aligned} \tag{1}$$

where $\lambda_0, \lambda_1, \lambda_2$ are hyperparameters of the loss function, $\mathrm{CrossEnt}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}})$ is the cross-entropy – the classification – error function, $\mathrm{Rec}(\boldsymbol{X})$ is the autoencoder's reconstruction error:

$$\mathrm{Rec}(\boldsymbol{X}) = \frac{1}{n} \sum_{i=1}^{n} \|(\boldsymbol{\Phi} \circ \boldsymbol{\Phi}')(x_i) - x_i\|_2^2,$$

with $\boldsymbol{\Phi}(\cdot)$ and $\boldsymbol{\Phi}(\cdot)'$ the encoder and decoder functions respectively, and the pair R_1 and R_2 are costs for the quality of prototypes:

$$\mathrm{R}_1(\mathcal{P}, \boldsymbol{X}) = \frac{1}{M} \sum_{m=1}^{M} \min_{n=1,N} \|\boldsymbol{p}_m - \boldsymbol{\Phi}(x_n)\|_2^2,$$

$$\mathrm{R}_2(\mathcal{P}, \boldsymbol{X}) = \frac{1}{N} \sum_{n=1}^{N} \min_{m=1,M} \|\boldsymbol{\Phi}(x_n) - \boldsymbol{p}_m\|_2^2.$$

In the above formula $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M\}$ is the set of prototype vectors, each vector corresponding to a *prototype unit* in the architecture; M is the number of prototypes and N is the size of the training dataset. These two terms encourage that (1) every prototype to be close to at least one training example ensuring the existence of a meaningful prototype, and that (2) every training example to be close to at least one prototype; yielding a clustering of the training examples around the prototypes.

In contrast to ProtoPNet, the next model to be presented in Section I-B2, PrototypeDL produces full-size prototypes, i.e. of the same size as the encoded images. As a drawback, it fails to produce realistic/interpretable images when trained on natural pictures, due to the autoencoder used as a feature extractor. The decoder will return blurry images like Figure 1, which can not be used as explanations.

2) ProtoPNet: The prototypical part network (ProtoPNet), introduced in [17], is an improvement over the PrototypeDL architecture from Section I-B1. Instead of using an autoencoder to obtain the features, the model includes as a backbone a convolutional network for classification, such as VGG-16, VGG-19, ResNet-152, DenseNet-121, or DenseNet-161 [18]. The "convolutional" part of the above-mentioned backbone networks will serve as prototypes: the system considers the last output layer as a set of features. After extracting the *set* of features is followed by a prototype layer and the fully connected layer that performs multi-class classification. If we assume that the output of the convolution is of shape (H, W, D), then we consider every (H_1, W_1, D) patch a prototype – in practice, we will use $(1, 1, D)$. If we consider the output as a representation of the input image, the prototypes will correspond to a patch (of non-uniform size) from the original image. The prototypes (denoted by $\mathcal{P}$) will have the same size as these convolutional patches, i.e. of (H_1, W_1, D), and each class is assigned a fixed number of prototypes. For a given input image, the j-th prototype unit in the prototype layer computes the Euclidean distance between the j-th prototype vector and all the patches of the convolutional output, resulting in a heatmap that shows which parts of the input image are most similar to the prototype. These maps are reduced to a single similarity score for each prototype vector using global max pooling; the reduced scores can be interpreted as to which extent the given prototype is present in the input image.

The loss function of ProtoPNet can be separated into three parts:

$$\begin{aligned} L_{PPN}(\mathcal{D}) = \; & \mathrm{CrossEnt}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) \\ & +\lambda_1 \mathrm{Clst}(\mathcal{P}, \boldsymbol{X}) \\ & +\lambda_2 \mathrm{Sep}(\mathcal{P}, \boldsymbol{X}), \end{aligned} \tag{2}$$

where $\mathrm{CrossEnt}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}})$ is again cross-entropy, $\mathrm{Clst}(\mathcal{P}, \boldsymbol{X})$ is a clustering term ensuring that images of a given class have at least one latent patch close to a prototype of the same class, while $\mathrm{Sep}(\mathcal{P}, \boldsymbol{X})$ pushes the latent patches apart from the prototypes of other classes:

$$\mathrm{Clst}(\mathcal{P}, \boldsymbol{X}) = \frac{1}{N} \sum_{n=1}^{N} \min_{\boldsymbol{p}_j \in \boldsymbol{P}_n} \min_{\boldsymbol{z} \in \mathrm{patches}(\boldsymbol{\Phi}(\boldsymbol{x}_n))} \|\boldsymbol{z} - \boldsymbol{p}_j\|_2^2,$$

$$\mathrm{Sep}(\mathcal{P}, \boldsymbol{X}) = -\frac{1}{N} \sum_{n=1}^{N} \min_{\boldsymbol{p}_j \notin \boldsymbol{P}_n} \min_{\boldsymbol{z} \in \mathrm{patches}(\boldsymbol{\Phi}(\boldsymbol{x}_n))} \|\boldsymbol{z} - \boldsymbol{p}_j\|_2^2,$$

where $\boldsymbol{P}_n \subset \mathcal{P}$ is the set of prototypes corresponding to class $\boldsymbol{y_n}$ and $\boldsymbol{\Phi}(\cdot)$ denotes the convolutional backbone.

ProtoPNet, as opposed to PrototypeDL, can be used on real-world images. The prototype-to-image is explicit and uses images from the training set, this is the $\mathrm{Clst}(\cdot)$ term in the error function. The anchoring to a given input image and the "localized footprint" of the prototype vector leads to an increased interpretability level – contrasting PrototypeDL, where we have global representations of the inputs.

3) BagNet: these models were introduced by Brendel and Bethge [19], inspired by the *bag-of-words* (BoW), or the *bag-of-features* (BoF) models; these are popular models in information retrieval (IR) and natural language processing (NLP) [20]. BoW or BoF count the occurrence of a feature in an entity, e.g. words in a document, thus the representation of a document becomes a bag data structure collecting the number of word occurrences in a document. The architecture of BagNet resembles BoW: individual features of the input are put together – in this case, averaged – to obtain the global representation of the entire image.

The idea is simple yet effective: we use an FCN to generate features for the input image and use the average of the feature vectors to output the logits. Each feature vector corresponds to a window of the same size as the receptive field of the network (patch). Using logits, the output of the linear classifier will be the same as the average of the logits output for each input image patch.

BagNet is able to generate a heatmap for the input images: for each patch of the input image the model outputs the class logits and these will correspond to one pixel of the heatmap. These heatmaps can be interpreted as explanations for the classification: the part of the image, where the heatmap has a high activation has more importance during classification. Moving the window with stride 1, we are able to generate one heatmap for each class.

This model is built on a simple idea inspired from IR/NLP, yet we found that it produces competitive results for self-explaining image classification tasks.

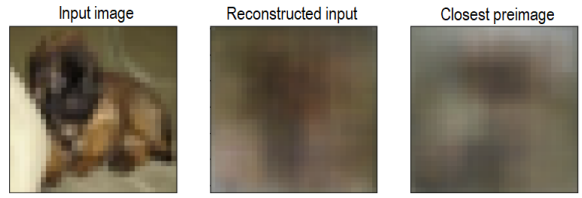


Fig. 1. The PrototypeDL architecture in action on the CIFAR10 data: the input (left), its reconstruction (middle), and the closest pre-image (right). It can be observed that both the reconstruction and the pre-image are blurry, therefore cannot serve as an explanation.

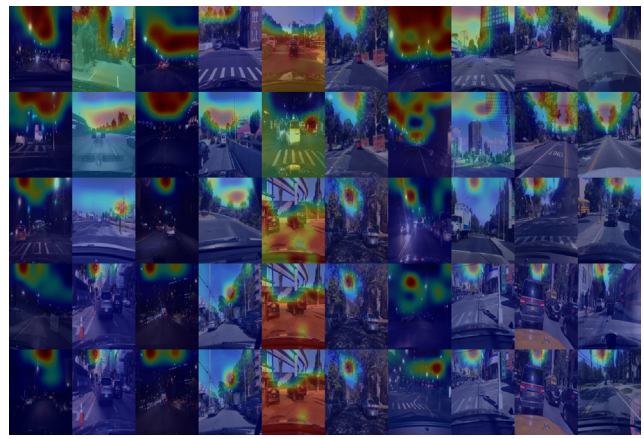


Fig. 2. Evolution of the 10 prototypes assigned to *clear* class – from top to bottom. It is visible that most of the prototypes "focus" on the upper part of the image, the blue sky. There are abnormal behaviours, e.g. in the fifth column the region is not "interpretable" w.r.to the class semantics.

C. Experimental comparison of the models

In our research, we conducted several experiments using different datasets. First, we used a simpler dataset (MNIST – Modified National Institute of Standards and Technology[1] [21]) and then the models were fine-tuned for a more complex one (BDD100K – Berkeley DeepDrive[2] [22]). Working with BDD100K the *weather condition* labels were used, containing 7 classes: *clear*, *foggy*, *overcast*, *partly cloudy*, *rainy*, *snowy*, and *undefined*. This dataset is highly unbalanced, e.g. we have 37 344 images with label *clear* as opposed to 130 images labeled *foggy*.

1) PrototypeDL: In the original paper [16] this method was tested on three datasets: MNIST (99.22% test accuracy), 3D cars [23] (93.5% test accuracy), and Fashion MNIST[3] (89.95% test accuracy), the obtained results being comparable with that of non-interpretable models (within 2.55% margin). We also tested the model on CIFAR10 dataset, but the output prototypes were not interpretable, see Figure 1. Due to its poor interpretability, this model was not tested on BDD100K dataset.

2) ProtoPNet: In our experiments, VGG-19 CNN was used as a feature extractor. The number of prototypes per class was

[1] http://yann.lecun.com/exdb/mnist/
[2] http://bdd-data.berkeley.edu
[3] https://github.com/zalandoresearch/fashion-mnist

TABLE I
COMPARISON OF THE PERFORMANCE OF PROTOPNET AND THE EQUIVALENT BACKBONE MODEL ON BDD100K DATASET.

Metric	ProtoPNet		Backbone model	
	Train	Test	Train	Test
Accuracy	0.8564	0.8264	0.8457	0.8284
F_1 score (micro-avg.)	0.8523	0.8218	0.8457	0.8284
F_1 score (macro-avg.)	0.7082	0.6422	0.6725	0.6510

TABLE II
COMPARISON OF THE PERFORMANCE OF BAGNET-17 AND THE EQUIVALENT RESNET-50 ON THE BDD100K DATASET.

	BagNet-17		Equivalent ResNet-50	
	Train	Test	Train	Test
F_1 score (micro-avg.)	0.7880	0.7953	0.8098	0.8093
F_1 score (macro-avg.)	0.5973	0.6038	0.6494	0.6242
Training epochs	81		84	

set to 10 as in [17]. Besides trying to reproduce the result on CUB-200-2011[4] the model was trained on MNIST and BDD100K datasets. We analyzed the result of our experiments from two aspects. Firstly, the performance of ProtoPNet was compared against its equivalent backbone, here we aimed at assessing the cost of interpretability. Secondly, we performed a subjective analysis of the prototypes and their evolution over the learning epochs.

The backbone model has a feature extractor layer followed by a fully connected one, i.e. the prototype layer was removed from the model. On MNIST dataset the differences were below 0.1%, both interpretable and non-interpretable models had an accuracy of 99.8% on the train and 99.6% on test data. With BDD100K dataset the interpretable model achieved better accuracy on train data, as shown in Table I. We also measured F_1 scores on the BDD100K dataset because of its uneven distribution. We can observe a gap between the models on train data by analyzing these scores. For every class, we collected the images from which the ProtoPNet learned its 10 prototypes in different epochs and aggregated them into an image collage. In the original images, we can see the saliency map of input pixels indicating their contribution to the learned prototype vector. The prototypes were observed form every 10^{th} epoch up to the 50^{th}. Figure 2 shows an example for the class *clear*. As expected, most of the learned prototypes are parts from the blue sky regions. The evolution of prototypes is noticeable (with uninterpretable prototypes at the start). Using visual inspection, we can have a subjective assessment of the prototype, namely the region it focuses on during prediction. This way we may exclude prototypes from the model. For example, in the 5^{th} column, there is a prototype of the road. If we want our model to "focus" only on the sky, we can remove it from the model.

3) BagNet: In [19] the model was tested on ImageNet[5]. Using a 17×17 patch size, the model reached the performance of AlexNet (80.5% top-5 accuracy) [24], while using 33×33 patches a top-5 accuracy of 87.6% was achieved.

[4]https://www.vision.caltech.edu/datasets/cub_200_2011/
[5]https://www.image-net.org/

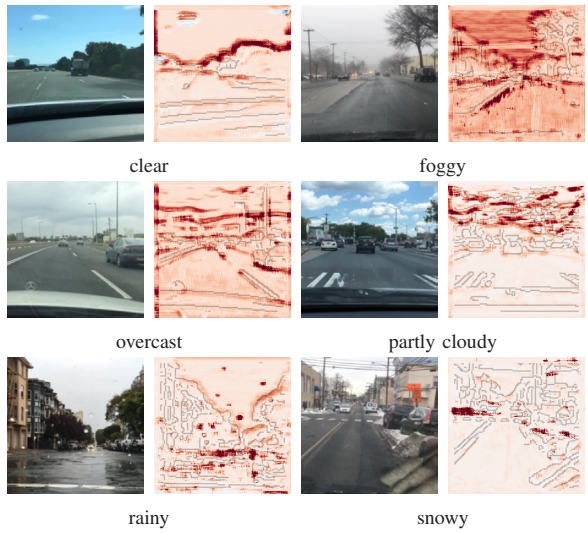


Fig. 3. One sample from each weather class from the BDD100K dataset (left) and their corresponding heatmaps (right) obtained with the BagNet-17 model. The darker the region, the more important it was in assigning the given label to the image.

For conducting the experiments on the BDD100K dataset, we used the initial architecture, based on ResNet-50 [25] by replacing most of the 3×3 by 1×1 convolutions, thus limiting the receptive field size of the topmost convolutional layer to $q\times q$. We experimented with both $q=9$ and $q=17$ and found that $q=17$, that is Bagnet-17, yielded better results.

We compared the results obtained with the BagNet-17 with the corresponding ResNet-50 architecture (backbone model). The model was trained using SGD with momentum (0.9), a batch size of 16, and a learning rate initially set to 0.1 and decayed by a multiplicative factor of 0.35 every 30 epochs. For evaluation, F_1 score was used. The results are summarized in Table II, which shows that BagNet-17 achieved a micro-averaged F_1 score of 0.7880 on the training set and a micro-averaged F_1 score of 0.7953 on the test, respectively, after 81 epochs. Surprisingly, the equivalent ResNet-50 has outperformed the BagNet-17 by only 0.014 on the test set when evaluated with the micro-averaged F_1 score, and by 0.0204 with the macro F_1 score.

To visually assess the interpretability of the BagNet-17 model on BDD100K, we plotted the heatmaps generated by the model, as shown in Figure 3 along with the original images. Our conclusion is that the strong emphasis is in partial agreement with our intuition but it is often not conclusive: e.g it is unclear why in the clear class example – top left – the highest importance is rendered to the line separating the blue sky background from the rest of the image.

D. Measuring interpretability

While measuring prediction accuracy is usually simple, unless labeling data becomes a costly process [26], determining interpretability is much more complicated, however, the assessment of explainability is similarly important to evaluate the method and compare it to other approaches from this perspective as well. Following the works [27]–[29] we define

TABLE III
EVALUATION RESULTS OBTAINED ON MNIST (RESNET-20 BACKBONE) USING DIFFERENT HYPERPARAMETERS.

	λ_1	λ_2	λ_3	Training accuracy	Test accuracy
(a)	0.8	-0.08	0	0.9902	0.9814
(b)	0.8	0	0	0.9951	0.9853
(c)	0.8	-0.08	-0.1	0.9895	0.9826
(d)	0.8	-0.08	-0.15	0.9893	0.9808

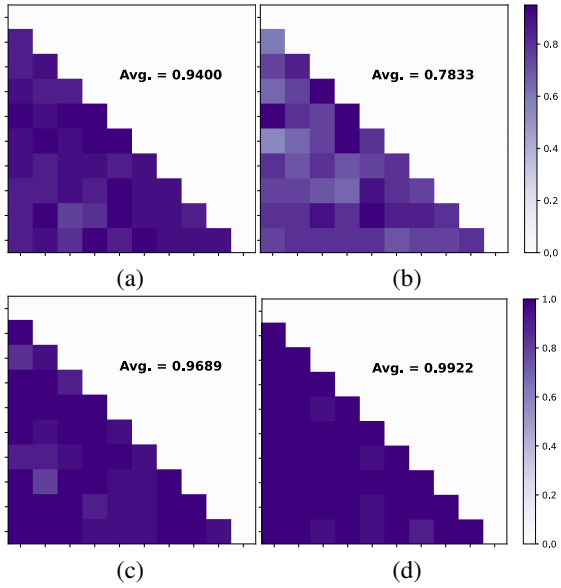


Fig. 4. Matrix of 1-vs-1 average training accuracies obtained using linear SVMs to separate prototype vectors assigned to different classes (the four scenarios correspond to the models displayed in Table III): (a) original ProtoPNet model, (b) omitting the separation cost, (c)–(d) using the additional separation cost.

interpretability as the linear separability of the feature vectors and incorporate a new separation cost to improve on this in the ProtoPNet model[6]. Thus, we measure interpretability by the macro-averaged 1-vs-1 training classification accuracy obtained by a linear classifier considering the prototype vectors assigned to different classes as training examples. In our experiments, linear support vector machines (SVM) [30] were used.

In [17], the separation cost enforces every latent patch of a training image to be distant from the prototypes not of its class, which is a requirement of explainability. If no such condition is imposed on the patches and/or prototypes it may harm the interpretability of the model, since similar prototypes can be obtained for different classes. While discarding separability may improve classification performance [31], it can hurt the visual explanation process.

The additional separation cost introduced in our model pushes away the prototype vectors from each other,

$$\mathrm{Sep}_2(\mathcal{P}) = \frac{-2}{K(K-1)} \sum_{\substack{i,j=1 \\ j>i}}^{K} \min_{\substack{\boldsymbol{p}\in\mathcal{P}_i, \\ \boldsymbol{q}\in\mathcal{P}_j}} \|\boldsymbol{p}-\boldsymbol{q}\|_2^2 \qquad (3)$$

[6]The linear separability of the features increases/should increase monotonically with the depth of the model, explained by the fact that the last layer of deep neural networks is usually a linear classifier.

and thus the total loss function of the optimization problem becomes

$$\begin{aligned} L(\mathcal{D}) = \; & \mathrm{CrossEnt}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) \\ & +\lambda_1 \mathrm{Clst}(\mathcal{P}, \boldsymbol{X}) \\ & +\lambda_2 \mathrm{Sep}(\mathcal{P}, \boldsymbol{X}) + \lambda_3 \mathrm{Sep}_2(\mathcal{P}), \end{aligned} \qquad (4)$$

where CrossEnt, Clst, and Sep represent the same loss components as in the original ProtoPNet model. In Figure 4 the interpretability values are shown as macro-averaged 1-vs-1 training accuracies obtained for separating the prototypes of different classes using a linear SVM. We also display the lower triangular portion of the confusion matrices obtained, where a darker color of a square denotes a better separation. For conducting the experiments, ResNet-20 was used as the backbone of ProtoPNet [25], and the methods were evaluated on the MNIST dataset. The number of epochs was set to 50. Table III shows the hyperparameter settings of the models together with the training and test accuracies obtained. From Table III and Figure 4 one observes that by omitting the separation costs, it is possible to obtain better accuracy scores (model (b)) compared to the base model (model (a)), meanwhile, separability decreases. The introduction of the new separation cost yields slightly better test accuracies and better linear separation of the prototypes (model (c)). Furthermore, setting a higher weight for the newly introduced cost (model (d)) can slightly decrease prediction accuracy but at the same time significantly increase the separability.

II. CONCLUSION

In this article, we presented a comparison of three interpretable image classification models. All models use convolutional neural networks, CNNs, to represent image features via prototype vectors, and the use of prototypes provides a good performance. The interpretability of the models is achieved via the connection of the prototype vectors to the outputs on one hand, and the connection of the prototype vectors to a region in the input image on the other hand; model interpretability is achieved by linking the decision – i.e. the model output – and the region in the input image.

With the measurements on "backbones", we assessed whether the addition of interpretability to the model, similar to extensions towards robustness, impacts accuracy and we conclude that there is no significant drop in performance when interpretability is added to the model. The conclusion is that it was essential to properly adjust model hyperparameters, such as receptive field size or the parameters of the cost function; this fitting of model parameters indicates a potential instability when using the models in real situations.

As a summary, we assessed that the first tested *PrototypeDL* model uses prototypes that are global to the whole image, therefore the pre-images of the prototypes are blurry, as such cannot be used as an explanation. Our second model, *ProtoPNet*, an improved PrototypeDL, provides localized prototypes, and the manual inspection of the results confirms its highly interpretable nature, and this model achieves the best evaluation results on the tested datasets. The third model we tested, the BagNet model, had the simplest architecture of all and it produces good results despite not using prototypes.

An important conclusion was that these models need large datasets to perform well – namely, we emphasize that for ProtoPNet the multiplication of inputs using augmentation had a huge positive impact on the performance.

Since almost all methods studied so far rely on deep features extracted by CNNs, we might ask about the validity of our base assumption about these, namely that similar patches generate similar features. As one-pixel attacks show [32], it suffices to change one pixel in the input image to obtain very dissimilar features. At the same time, other obfuscation techniques, e.g. JPEG compression, resulting in differences imperceptible to the human eye greatly influence the output of the network as well [33]. Therefore, we plan to study the influence of robustness on the predictions and explanations in interpretable methods [34], as well as explore other approaches that could increase interpretability in such self-explainable deep neural network models.

Acknowledgment

The authors acknowledge the financial support through Bosch grant *"Explainable artificial intelligence (xAI) for automated driving systems"*.

References

[1] O. O'Neill, "Linking trust to trustworthiness," *International Journal of Philosophical Studies*, vol. 26, no. 2, pp. 293–300. DOI: 10.1080/09672559.2018.1454637, 2018.

[2] J. A. McDermid, Y. Jia, Z. Porter, and I. Habli, "Artificial intelligence explainability: the technical and ethical dimensions," *Philosophical Transactions; Series A*, vol. 379. DOI: 10.1098/rsta.2020.0363, 2021.

[3] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *DSAA*. IEEE, 2018, pp. 80–89. DOI: 10.1109/DSAA.2018.00018

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier," in *KDD*, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778

[5] T. Hastie, R. Tibshinrani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, 2016.

[6] S. Thrun, "Extracting rules from artificial neural networks with distributed representations," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. MIT Press, 1994, pp. 505–512.

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014. ISBN 978-3-319-10590-1 pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53

[8] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Workshop at International Conference on Learning Representations*, 2014, pp. 1–8. DOI: 10.48550/arXiv.1312.6034

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint *arXiv:1409.0473*, 2015.

[10] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, "An attentive survey of attention models," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 5, pp. 1–32. DOI: 10.1145/3465055, 2021.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint *arXiv:2010.11929*, Tech. Rep., 2020.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, vol. 30, 2017. DOI: 10.48550/arXiv.1706.03762

[13] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013.

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *ICLR*. OpenReview.net, 2017.

[15] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *NeurIPS*, vol. 29, 2016. DOI: 10.48550/arXiv.1606.03657

[16] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *AAAI*, vol. 32, 2018, pp. 3530–3537. DOI: 10.1609/aaai.v32i1.11771

[17] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin, "This looks like that: deep learning for interpretable image recognition," in *NeurIPS*, 2019, pp. 8930–8941. DOI: https://dl.acm.org/doi/10.5555/3454287.3455088

[18] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. ISBN 9780262035613. [Online]. Available: http://www.deeplearningbook.org

[19] W. Brendel and M. Bethge, "Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet," 2019.

[20] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, Cambridge, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[21] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217

[22] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," 2018. [Online]. Available: https://www.arxiv-vanity.com/papers/1805.04687

[23] S. Fidler, S. Dickinson, and R. Urtasun, "3D object detection and viewpoint estimation with a deformable 3D cuboid model," in *NeurIPS*, vol. 25. Curran Associates, Inc., 2012. DOI: 10.13140/2.1.2839.7440

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90. DOI: 10.1145/3065386, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*. IEEE, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90

[26] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep. 1530, 2005.

[27] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," in *ICLR*, 2017. DOI: 10.48550/arXiv.1610.01644

[28] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *CVPR*. IEEE, 2017, pp. 6541–6549. DOI: 10.1109/CVPR.2017.354

[29] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," in *ICML*, 2018, pp. 2668–2677. DOI: 10.48550/arXiv.1711.11279

[30] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT*, 1992, pp. 144–152. DOI: 10.1145/130385.130401

[31] W. Xiao, Z. Ding, and H. Liu, "Learnable visual words for interpretable image recognition," *CoRR*, vol. abs/2205.10724. DOI: 10.48550/arXiv.2205.10724, 2022.

[32] D. V. Vargas and J. Su, "Understanding the one-pixel attack: Propagation maps and locality analysis," 2019.

[33] A. Hoffmann, C. Fanconi, R. Rade, and J. Kohler, "This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks," 2021.

[34] C. Etmann, S. Lunz, P. Maass, and C. Schönlieb, "On the connection between adversarial robustness and saliency map interpretability," in 6 ICML 2019, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. *PMLR*, 2019, pp. 1823–1832. https://doi.org/10.48550/arXiv.1905.04172

Adél Bajcsi is a Ph Dstudent at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University, under the supervision of Camelia Chira. Her current research interest includes image processing and classification, and machine learning. Adél is currently a member of a research team that investigates different self-interpretable deep learning models applied in the field of image recognition.

Anna Bajcsi is currently a member of a research group at the Babeş-Bolyai University investigating self-explainable deep learning models for image classification problems. She obtained her master's degree at the same university in 2022 studying Data Analysis and Modelling.

Szabolcs Pável is an assistant professor at the Faculty of Mathematics and Computer Science of the Babeş-Bolyai University (Cluj, Romania). His research interests include classical computer vision, machine learning (including deep learning), focusing on applications in driver assistance systems and self-driving cars. He also works as a machine learning engineer in the automotive industry, developing video perception systems.

Ábel Portik is a master's student in Data Analysis and Modelling at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University. He currently investigates self-explaining deep neural models applied for image classification as a member of a research group.

Csanád Sándor is an assistant professor at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University, as well as a machine learning engineer in the automotive industry. His main research interest is neural network compression, focusing on structured parameter pruning and post-training quantization.

Annamária Szenkovits is an assistant professor at the Department of Math- ematics and Computer Science of the Hungarian Line of the Babeş-Bolyai University (Cluj, Romania). Her main areas of interest include information retrieval and machine learning. Within machine learning, her work focuses on image recognition and natural text processing. She is currently a member of a research team that investigates different self-explainable deep learning models applied in the field of image recognition.

Orsolya Vas is an assistant professor at the Faculty of Mathematics and Computer Science of the Babeş-Bolyai University of Cluj-Napoca. Her previous research interests and results are related to critical point analysis and differential calculus. She is a member of a research team investigating self-explainable deep learning models for classification.

Zalán Bodó is currently an associate professor at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University, teaching, among other subjects, Information Theory and Natural Language Processing. He obtained his PhD degree from the same faculty in 2010, studying and analyzing kernel methods for semi-supervised learning. Since then, he participated in several research projects, and his main interests include data-efficient machine learning algorithms, natural language processing, information retrieval, and recently deep learning methods.

Lehel Csató is a professor at the Faculty of Mathematics and Computer Science. He holds a PhD from Aston University (UK), his research was the use of probabilistic non-parametrics as latent variables. He is interested in the mathematical aspects of machine learning, in providing approximate solutions for inverse problems, performing sparse inference on large systems with applications to robotics, classification of complex data, and active learning. After the onset of the new deep learning era, he is interested in a better understanding of the working of these systems; the exploration of self-explainable deep learning methods, and simplification possibilities, e.g. pruning.

What Can We Learn from Small Data

Tamas Nyiri and Attila Kiss

Abstract—**Over the past decade, deep learning has profoundly transformed the landscape of science and technology, from refining advertising algorithms to pioneering self-driving vehicles. While advancements in computational capabilities have fueled this evolution, the consistent availability of high quality training data is less of a given. In this work, the authors aim to provide a bird's eye view on topics pertaining to small data scenarios, that is scenarios in which a less than desirable quality and quantity of data is given for supervised learning. We provide an overview for a set of challenges, proposed solution and at the end tie it together by practical guidelines on which techniques are useful in specific real-world scenarios.**

Index Terms—**deep learning, small data, small sample learning, few shot learning.**

I. Introduction

A. Background

Supervised Learning is a type of Machine Learning (ML) which itself is a sub-field of Artificial Intelligence (AI). In supervised learning, the algorithm learns from labeled data to make predictions or decisions about new, unseen data. In this type of learning, the algorithm is trained on a data set that includes both input data and corresponding output labels. The algorithm then uses this training data to learn a mapping function that can predict the output labels for new, unseen input data. [1]

Early works in AI focused on rule-based systems and expert systems, where human experts would define rules and logic for the system to follow. However, these systems were limited by the complexity and variability of real-world data.

In the 1980s, the field of Machine Learning emerged, which focused on algorithms that could automatically learn patterns from data. Early Machine Learning algorithms were primarily based on statistical models, such as linear regression and logistic regression.

In the 1990s, the development of Artificial Neural Networks brought new advances in Supervised Learning. These networks were inspired by the structure of the human brain and were capable of learning complex patterns from data through a process called back-propagation. [2]

In the early 2000s, larger neural network models, such as Convolutional Neural Networks (CNNs) [3] and Recurrent Neural Networks (RNNs) [4], were developed, leading to breakthroughs in areas such as image recognition and Natural Language Processing (NLP).

In recent years, the development of even larger Neural Network models, such as Deep Neural Networks (DNNs), has led to even greater advances in Supervised Learning. These models can learn from vast amounts of training data, and their performance has been shown to improve with increasing amounts of data.

Overall, the history of AI, ML, and Neural Networks has been characterized by a gradual progression towards larger models and more training data, which has enabled breakthroughs in Supervised Learning and other areas of Machine Learning.

This has worked well for the most part on well defined problems where large, good quality data sets are available. We focus on the situations where this assumption does not necessarily hold.

B. Related Work

The topic of learning with limited amounts of data is not a recent one. There does seem to be however many takes on what constitutes "small data" and many techniques developed to be able to achieve competitive results on less than desirable data sets.

There have been several surveys written on this topic, looking at the problem from different directions. One example, "Generalizing from a Few Examples: A Survey on Few-shot Learning" by Wang et al [5] gave a unique taxonomy of Few Shot Learning (FSL) methods, dividing them into three main categories: ones that incorporate prior knowledge into the data, model or the algorithm of the learning system. Another one "Small Sample Learning in Big Data Era" by Shu et al [6] divided Small Sample Learning (SSL) techniques into two main branches: Concept Learning which emphasizes learning new concepts from few related observations, and Experience Learning which focuses on learning with insufficient samples, co-existing with the Large Sample Learning (LSL) manner of conventional machine learning.

Even though there has also been attempts on more theoretical explanations with promising results [7][8], there does still seem to be a large gap to traverse until we see these results used in more practical settings.

Tamas Nyiri and Attila Kiss, Department of Information Systems, ELTE Eötvös Loránd University, Budapest, Hungary. Attila Kiss was also with J. Selye University, Komárno, Slovakia (E-mail: nytuaai@inf.elte.hu, kiss@inf.elte.hu)

DOI: 10.36244/ICJ.2023.5.5

C. Objectives

In subsequent sections, we do not plan to provide an exhaustive overview of the subject, given the vast scope of the topic. Instead, our objective is to highlight key challenges associated with data scarcity and the strategies formulated to tackle them. Furthermore, we delve into select real-world scenarios one could encounter during their data scientific journey, and offer practical solutions to them.

II. Small Data Sources

A. Limited annotations

The lack of annotations, refers to a common challenge faced in Supervised Learning, where the required data lacks the necessary annotations (a.k.a. labels) to train a model. An annotation or label is a piece of information that is associated with each data point that is required for the ML algorithm to learn from it. Without these annotations, the algorithm is unable to differentiate between the correct and incorrect outputs.

An example of this annotation in Computer Vision (CV) can be the names of the objects identified by their bounding boxes used for object detection or in NLP, a sentiment associated with each example sentence, that can be used for Sentiment Analysis.

The lack of annotations can occur for several reasons, including the cost, difficulty, or time-consuming nature of annotation. In many cases, it may simply be impossible to obtain annotations for certain types of data, such as historical archives or rare events. The lack of annotations can also occur in situations where data is unstructured or noisy, making it difficult to label accurately.

B. Limited diversity

Limited diversity of data points refers to a situation in which the data set used to train a machine learning model contains a small number of examples that are not representative of the entire population. This can lead to bias in the model, resulting in poor performance on new, unseen data.

An example of this phenomenon can be seen in facial recognition systems. If the data set used to train the model contains a mostly images of people with lighter skin tones or darker hair color, the model may perform poorly when presented with images of individuals with darker skin tones or lighter hair color. This is because the model has not been trained on a diverse set of data, resulting in a biased model.

The limited diversity of data points can occur for several reasons, including the difficulty in obtaining diverse data or the availability of biased data sources.

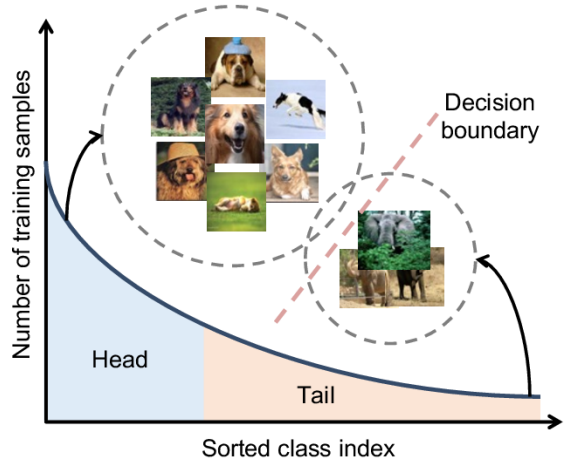


Fig. 1. Example of a long tailed dataset [9]

C. Long tail distribution

Long-tail distribution refers to a situation in which a small number of categories occur frequently, while the vast majority of categories occur infrequently.

An example of this phenomenon can be seen in the recommendation systems that suggest products to users. In many cases, a small number of popular product categories account for the majority of the purchases, while the vast majority of product categories are purchased infrequently. If the recommendation system is trained only on the popular products, it may perform poorly when making recommendations for less popular products.

The long-tail distribution can occur for several reasons, including the inherent nature of the data and the data collection process. In some cases, it may be difficult or expensive to collect data on the less popular data points, resulting in a bias towards the more popular data points.

To understand the origins of this phenomena, it is important to understand that long tail distributions are abundant in nature and thus will naturally show up in randomly sampled data.

One example of this would be the Pareto distribution, which describes the relative wealth distribution in sociology, or Zipf's law which states that in a given corpus of natural language, the frequency of any word is approximately inversely proportional to its rank in the frequency table. [10]

D. Concept drift

Concept drift refers to a situation in which the statistical properties of the target variable in a Machine Learning problem change over time, resulting in a decrease in the performance of the trained model. This can be

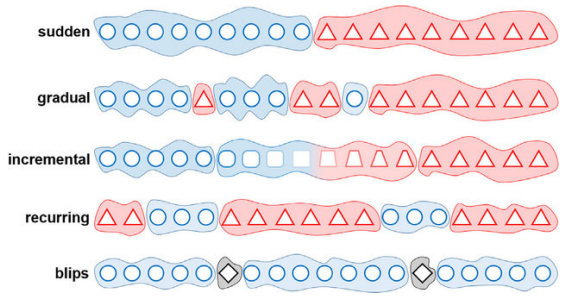


Fig. 2. Five types of Concept Drift according to [11]

a challenge in ML, as the model may become outdated and unable to accurately predict new, unseen data.

An example of this phenomenon can be seen in a spam email classifier. The distribution of spam emails may change over time, with new types of spam emails appearing that are not similar to those seen before. If the model has been trained only on the earlier types of spam emails, it may perform poorly when presented with the new types of spam emails, resulting in increased proportion of false negatives or false positives.

Concept drift can occur for several reasons, including changes in the behavior of users, changes in the environment, and changes in the data generation process. In many cases, the drift is gradual, making it difficult to detect and correct.

III. Small Data Solutions

A. Smart Sampling

The very first step in most practical machine learning is to ensure the data collected is the best quality possible, that is our sample is closest possible to our population. There are several statistical techniques developed over the years. Here we will only show a few that are most useful in a limited data environment.

1) Under-sampling and Over-sampling: When one has to deal with imbalanced dataset, a common approach is to either over-sample the minority class(es) or under-sample the majority class(es) until the desired distribution is reached.

This can be done by randomly removing samples (under-sampling) or adding multiple copies of the same sample (over-sampling) at random.

They both have their disadvantages. Under-sampling can lead to a loss of information since we leave out potentially relevant information from our training dataset. On the other-hand, over-sampling can reinforce existing biases in the over-sampled instances. For these reasons, it's usually better to use a more sophisticated method where possible.

2) Importance Sampling: Importance sampling is particularly useful for catching rare events in long-tail distributions. This involves creating a new distribution where rare events become not-so-rare, sampling from this new distribution, then re-weighing the samples to adjust for the bias introduced.

Let's say we have a target distribution $p(x)$ and an importance distribution $q(x)$. In order to arrive at an approximation of the expectation of a function $f(x)$ under the target distribution:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)\,dx \tag{1}$$

we can sample from the importance distribution (where x_i are samples drawn from $q(x)$) and then re-weight the samples:

$$\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i) \tag{2}$$

3) Active Learning: Active learning is an iterative process where the model selects the most informative samples to be labeled, thus reducing the amount of labeling resources to be used.

An early example for an active learning algorithm introduced by Cohn, Atlas and Ladner [12] is Query by Committee (QBC). The central idea behind QBC is to maintain a committee of models (set of hypotheses) over the data, and to obtain labels for instances about which the committee members disagree the most. This can reduce the amount of labeling by focusing only on the most informative (highest entropy) examples, but at the same time can introduce a large computational overhead. For the detailed algorithm, see Algorithm 1

Algorithm 1 Query By Committee (QBC)

Input: Dataset $\mathcal{D}$, committee of models C
Train each model $c_i \in C$ on $\mathcal{D}$
while stopping criterion not met **do**
  for each unlabeled x_u **do**
    Calculate disagreement:
    $D(x_u) = \sum_{c_i,c_j\in C, i\neq j} \mathbb{I}(c_i(x_u) \neq c_j(x_u))$
  end for
  Query label for instance $x^* = \arg\max_{x_u} D(x_u)$
  Add labeled instance (x^*, y^*) to $\mathcal{D}$
  Re-train each model $c_i \in C$ on $\mathcal{D}$
end while
Output: Labeled dataset $\mathcal{D}$

It has to be mentioned that this is just an early example, since its introduction several other Active Learning techniques have developed, such as Uncertainty Sampling [13] Expected Model Change [14], Expected Error

Reduction [15], Variance Reduction [16], Bayesian Active Learning by Disagreement [17], Diversity Sampling [18], Hierarchical Sampling [19], and Online Active Learning [20], to name a few.

B. Expert knowledge

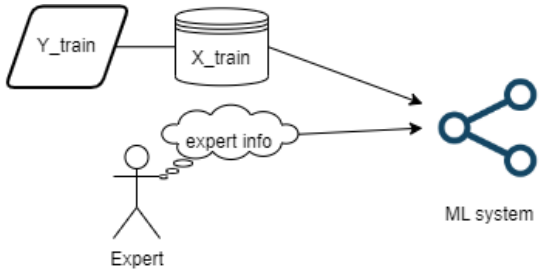


Fig. 3. Schematic drawing of "Expert Knowledge"

1) Taxonomy: One way in which expert knowledge can be incorporated into Deep Learning is through the use of taxonomies. A taxonomy is a hierarchical or otherwise structured organization of data that categorizes items or concepts based on their similarities, differences, and relationships.

In many machine learning tasks, a binary decision can be expanded into a multi-class decision using a taxonomy. This taxonomy divides the initial binary classes into more specific sub-categories, organized hierarchically in a tree-like fashion. Rather than training the model on the initial binary labels, one can use the more detailed labels corresponding to the leaves of this hierarchical structure. This approach allows the model to establish more intricate decision boundaries, capturing subtleties that might be overlooked in a simple binary classification.

Once the model is trained and deployed, predictions made at the leaf-level can be aggregated back to the original binary classification, if necessary.

2) Hand-crafted features: Another way in which expert knowledge can be incorporated into deep learning is through the use of hand-crafted features. Hand-crafted features are manually designed features that can be used as inputs to a neural network. These features are often designed based on domain-specific knowledge or prior research, and can be used to capture important characteristics of the data that may not be captured by the network's automatic feature learning.

These techniques used to be the back-bone of many AI algorithms before Deep Learning came into the picture, but have quickly fallen out of favor due to Deep Neural Networks' ability to learn similar but more complicated features. Examples of such techniques in Computer Vision include Histogram of Oriented Gradients (HOG) [21] and Local Binary Patterns (LBP) [22].

C. Data Augmentation

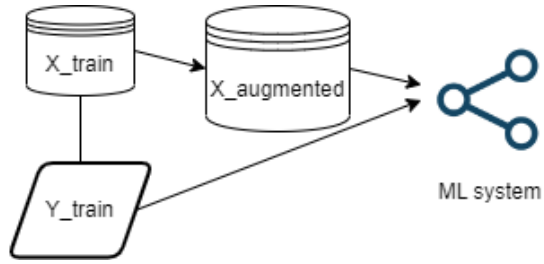


Fig. 4. Schematic drawing of "Data Augmentation"

1) Heuristic-based Methods: Heuristic-driven data augmentation techniques apply specific rules or heuristics to original data, generating new data samples. Designed to imitate natural data variations, these methods produce samples closely resembling, but not identical to, the original ones.

For image data, examples include geometric and color-space adjustments like random cropping, rotation, shifting, and variations in color through flips and jitter.

The same in the case of text-based input can involve: synonym replacement, back-translation, random deletion/insertion, random swap, etc..

It's important to note that these transformations need to be invariant with respect to the labels associated with the input data.

2) Data Generation: Data generation is a data augmentation method in Deep Learning that involves generating new synthetic data from scratch instead of transforming or manipulating existing data samples. This is typically done using generative models, which are deep learning models designed to learn the underlying patterns and structure of the data and generate new samples that are similar to the original data.

One of the most common generative models used for data generation is the generative adversarial network (GAN). GANs consist of two deep neural networks: a generator network and a discriminator network. The generator network takes a random input vector and generates a synthetic data sample, while the discriminator network tries to distinguish between the synthetic data and the real data.

During training, the generator and discriminator networks are trained together in a zero-sum game, where the generator tries to generate synthetic data that fools the discriminator, and the discriminator tries to correctly distinguish between the synthetic and real data. Over time, the generator becomes better at generating realistic data samples, and the discriminator becomes better at distinguishing between the synthetic and real data.

Given a noise variable z drawn from a prior distribution $p(z)$, generator G tries to produce something similar to a sample, $G(z)$.

Given a real sample x, drawn from the observed distribution $p_{data}(x)$ and the fake sample $G(z)$, the discriminator tries to differentiate between the two and outputs a probability associated with its confidence that the generated sample is from the observed distribution.

This way, we get a two-player minimax game, with the value function $V(D,G)$:

$$V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] \quad (3)$$

$$+ \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))] \quad (4)$$

The discriminator tries to maximize $V(D,G)$ with respect to D, while the generator tries to minimize the same.

This results in the following optimization problem:

$$\min_G \max_D V(D,G) \quad (5)$$

Where the generator and discriminator are trained alternatively step-by-step.

D. Semi-supervised Learning

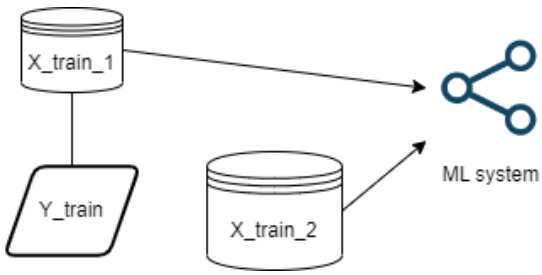


Fig. 5. Schematic drawing of "Semi-supervised Learning"

Semi-supervised learning is a type of machine learning that involves training a model on both labeled and unlabeled data. The core idea is that even though the unlabeled data doesn't provide direct supervision, it still contains valuable information about the underlying data distribution that can assist the learning process. This approach is especially useful when labeled data is limited, expensive, or time-consuming to obtain, but unlabeled data is abundant.

An example of Semi-Supervised Learning is Consistency Regularization, where the model is trained to be robust against different data augmentations by ensuring consistent predictions for different augmented views of the same input, even if the input is unlabeled. [23]

Another is MixMatch, which leverages the MixUp process: a data augmentation technique that creates virtual training examples by linearly interpolating between pairs of examples and their associated labels. MixMatch takes a pair of data points (one from the labeled set and one from the unlabeled set with its guessed label) and applies the MixUp process on them. [24]

Pseudo-labeling is another straightforward way to utilize unlabeled data. The idea is to train a model on all the labeled data and then predict on the rest (unlabeled) data points. If the prediction certainty reaches a certain confidence, we assign the data with the predicted label and use it to retrain the model. [25]

E. Self-supervised Learning

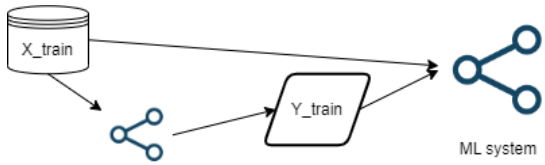


Fig. 6. Schematic drawing of "Self-supervised Learning"

In self-supervised learning, the algorithm learns to generate its own labels or representations from the input data itself, without any explicit supervision. This is usually achieved by defining a "pretext task" or "auxiliary task" that helps the model learn useful features or representations from the data. The idea is that these learned features will be useful for downstream tasks, like classification or regression.

Common examples of self-supervised learning in a text-based context include language model pre-training (e.g., BERT [26], GPT [27]), where the model learns to predict the next word in a sentence, based on huge amounts of unlabeled text data, where the training consists of hiding certain parts of the input and letting the model try to guess the right answer. In this was, the labels are the masked parts of the unlabeled input data, which are hidden from the models during training.

In the realm of computer vision, a famous self-supervised learning method is Contrastive Learning [28]. Here, the model tries to learn an embedding space where similar images are closer to each other, and dissimilar images are farther away. This is done by utilizing concepts such as positive pairs and negative pairs, where positive pairs can be different augmentations of the same data point.

Given:

- $D_w(x_i, x_j)$ as the Euclidean distance between two data point embeddings x_i and x_j, where w represents the parameters of the neural network.
- y as a binary label indicating whether the pair is a positive pair ($y = 1$) or a negative pair ($y = 0$).
- m as a predefined margin to ensure that negative pairs have distances greater than this margin.

The contrastive loss L for a single pair is:

$$L_{ij} = \frac{1}{2} y D_w(x_i, x_j)^2 + \frac{1}{2}(1-y)\max(0, m - D_w(x_i, x_j))^2$$

F. Transfer Learning

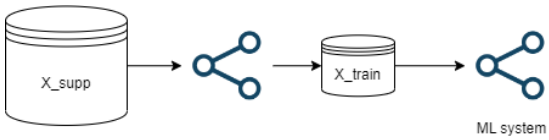


Fig. 7. Schematic drawing of "Transfer Learning"

Transfer learning is a Machine Learning technique that involves using knowledge gained from solving one problem to improve the performance of a model on a different, but related, problem. Transfer learning is based on the intuition that the knowledge and representations learned by a model on one task can be transferred to a different task that shares similar features or structure. The review by Pan et al [29] identified three main subcategories: Inductive, Transductive and Unsupervised Transfer Learning. Below is a quick summary of each:

1) Inductive Transfer Learning: This technique involves transferring knowledge across domains or tasks, where labeled data is available in both the source and the target domain.

2) Transductive Transfer Learning: Here we focus on domain adaptation, where the task remains unchanged, but the data distribution differs, and no labeled data is available in the target domain.

3) Unsupervised Transfer Learning: In this scenario, we attempt to transfer knowledge from the source domain/task to improve the learning of an entirely different task in the target domain, where no labeled data is available.

G. Meta learning

In meta-learning, the goal is to train a model to learn how to learn from a small number of examples, and then use this learned knowledge to rapidly adapt to new, unseen tasks.

There are different approaches to meta-learning, but a common one is to use a "meta-learner" that learns to update the parameters of a "base-learner" model based on a small amount of data from a new task. This process of updating the base-learner parameters based on new tasks is sometimes referred to as "meta-training.". We can identify several categories of Meta Learning, such as Model based, Metric based and Optimization based meta-learning. [30]

1) Model based meta-learning: Here we are training a meta-learner on a set of training tasks, each with limited number of labels. Whenever a new task is presented, the meta-learner adjusts its internal parameters based on the training examples and desired labels for the new task.

One example of a model-based meta-learning algorithm is Memory-Augmented Neural Networks. The core idea is to augment the model architecture (neural network) with an external memory mechanism. This introduces an extra memory component to the training process. Instead of only updating the weights of the network as traditional neural networks do, they can also update the content of their memories to perform better on new tasks. [31]

2) Metric based meta-learning: In metric based meta-learning, we have a distance metric in the space of tasks that can be used to quickly identify similar tasks and generalize to new tasks. The meta-learner can be given new tasks and a few related examples and is trained to be able to identify the similarity between these new tasks and the old ones in its space of tasks.

An example of this is Prototypical Networks. For each class, it computes a prototype (mean representation) from the embedding associated with the examples in that class. For a new data point, its class is determined by its proximity to these prototypes. [32]

3) Optimization based meta-learning: Finally, optimization based meta-learning approaches meta-learning as a bi-level optimization problem. At the inner-level, a base-learner makes task-specifc updates using some optimization strategy (such as gradient descent). At the outer-level, the performance across tasks is optimized.

Here, we can look at Model-Agnostic Meta Learning where the aim is to find a set of model parameters that are not optimal for any single task, but can be quickly adapted to any of the tasks within the desired set of tasks. [31]

IV. Small Data Scenarios

Now let's examine some scenarios a practitioner in the field might encounter in the real-world. For each we will list the most likely problems that can arise and recommended solutions from our list of techniques examined.

A. Diagnosis of Rare Diseases from Medical Images

- Small Data Sources:
  - Limited annotations: Obtaining labels might involve invasive/expensive procedure.
  - Limited diversity: Examples might come from a few specialized hospitals/geographical regions only.

  - Long tail distribution: Many common diseases and a few rare ones.
- Small Data Solutions:
  - Expert knowledge: Incorporate knowledge from medical professionals. [33]
  - Taxonomy: Subdivide diseases based on origin. [33]
  - Data Augmentation: Generate more images through invariant transformations. [33]
  - Transfer Learning: Use models pre-trained on larger dataset. [33]

B. Predicting Customer Churn in a New Market

- Small Data Sources:
  - Limited annotations: Company is still new, so small existing dataset.
  - Concept drift: Customer behavior might change over time, especially in new markets.
- Small Data Solutions:
  - Active Learning: Keep updating the model by querying the most uncertain predictions. [34]
  - Expert knowledge: Incorporate business intelligence and market insights. [35]
  - Semi-supervised Learning: Incorporate information about customer interactions. [36]

C. Sentiment Analysis for a Less Common Language

- Small Data Sources:
  - Limited annotations: Fewer examples in rare languages.
  - Limited diversity: Most examples might come from a limited set of sources (people who like to leave reviews).
  - Concept drift: Words and phrases change their meanings over time, sense of humor might evolve.
- Small Data Solutions:
  - Smart Sampling: Choose diverse examples across different all possible languages.
  - Data Generation: Use translation tools to augment data. [37]
  - Self-supervised Learning: Predict which words are the best sentiment predictors. [38]
  - Transfer Learning: Transfer user biases to textual features. [39]

D. Self-driving in a New Environment

- Small Data Sources:
  - Limited diversity: Training data might not include all types of environments.
  - Concept drift: Environment can change over time (e.g., changes in climate/lighting conditions).
  - Long tail distribution: Certain events in driving happen rarely (e.g., crashes).
- Small Data Solutions:
  - Data Augmentation: Simulate different lighting and object placements. [40]
  - Importance Sampling: Weight experiences that are less frequent but important (like crashes) more heavily.
  - Meta-learning: Use knowledge from common objects to help detect rare ones. [41]

V. Discussion

The era of big data has led to a vast landscape of deep learning techniques, leaving the average practitioner uncertain about which direction to take for unfamiliar challenges. Furthermore, much of the theoretical groundwork is done on unrealistically large and good quality data sources that doesn't take into account the natural shift in the specific domain studied.

Through this paper, our aim is to guide practitioners by offering a concise summary of frequently faced challenges and potential solutions. This is complemented by a curated set of real-world examples. It is our sincere hope that readers find value in our efforts.

References

[1] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.
[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[5] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.
[6] J. Shu, Z. Xu, and D. Meng, "Small sample learning in big data era," *arXiv preprint arXiv:1808.04572*, 2018.
[7] L. Szymanski, B. McCane, and C. Atkinson, "Conceptual capacity and effective complexity of neural networks," *arXiv preprint arXiv:2103.07614*, 2021.
[8] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124020, 2019.
[9] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *arXiv preprint arXiv:2110.04596*, 2021.
[10] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.
[11] B. Krawczyk and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams," *Applied Soft Computing*, vol. 68, pp. 677–692, 2018.
[12] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, pp. 201–221, 1994.
[13] D. D. Lewis, "A sequential algorithm for training text classifiers: Corrigendum and additional data," in *Acm Sigir Forum*, vol. 29, no. 2. ACM New York, NY, USA, 1995, pp. 13–19.

[14] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[15] N. Roy and A. McCallum, "Toward optimal active learning through monte carlo estimation of error reduction," *ICML, Williamstown*, vol. 2, pp. 441–448, 2001.

[16] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *2007 IEEE 11th international conference on computer vision*. IEEE, 2007, pp. 1–8.

[17] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," *arXiv preprint arXiv:1112.5745*, 2011.

[18] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 59–66.

[19] S. Vijayanarasimhan and K. Grauman, "What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 2262–2269.

[20] N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *Advances in neural information processing systems*, vol. 17, 2004.

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[22] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part I 8*. Springer, 2004, pp. 469–481.

[23] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[24] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[25] D.-H. Lee et al., "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2. Atlanta, 2013, p. 896.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[27] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[28] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[29] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[30] M. Huisman, J. N. Van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artificial Intelligence Review*, vol. 54, no. 6, pp. 4483–4541, 2021.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. *PMLR*, 2017, pp. 1126–1135.

[32] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[33] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[34] W. Verbeke, D. Martens, C. Mues, and B. Baesens, "Building comprehensible customer churn prediction models with advanced rule induction techniques," *Expert systems with applications*, vol. 38, no. 3, pp. 2354–2364, 2011.

[35] E. Lima, C. Mues, and B. Baesens, "Domain knowledge integration in data mining using decision tables: case studies in churn prediction," *Journal of the Operational Research Society*, vol. 60, no. 8, pp. 1096–1106, 2009.

[36] X. Liu, M. Xie, X. Wen, R. Chen, Y. Ge, N. Duffield, and N. Wang, "A semi-supervised and inductive embedding model for churn prediction of large-scale mobile games," in *2018 IEEE international conference on data mining (ICDM)*. *IEEE*, 2018, pp. 277–286.

[37] A. Balahur and M. Turchi, "Multilingual sentiment analysis using machine translation?" in *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis*, 2012, pp. 52–60.

[38] J. Tang, Z. Lu, J. Su, Y. Ge, L. Song, L. Sun, and J. Luo, "Progressive self-supervised attention learning for aspect-level sentiment analysis," *arXiv preprint arXiv:1906.01213*, 2019.

[39] P. H. Calais Guerra, A. Veloso, W. Meira Jr, and V. Almeida, "From bias to opinion: a transfer-learning approach to real-time sentiment analysis," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 150–158.

[40] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7230–7240.

[41] Y.-X. Wang, D. Ramanan, and M. Hebert, "Meta-learning to detect rare objects," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9925–9934.

Tamas Nyiri finished him Masters degree in Computer Science in 2019 and started his PhD in 2022, both at Eötvös Loránd University. His research is mainly focused on deep learning scenarios involving data quality and interpretability issues.

Attila Kiss defended his PhD in the field of database theory in 1991. His research is focused on information systems, data mining, and artificial intelligence. He has more than 190 scientific publications. Seven students received their PhD degrees under his supervision. Since 2010, he has been the head of Department of Information Systems at Eötvös Loránd University, Hungary. He is also teaching at J. Selye University, Slovakia.

Survey of Routing Techniques-Based Optimization of Energy Consumption in SD-DCN

Mohammed Nsaif[1], Gergely Kovásznai[2], Ali Malik[3], and Ruairí de Fréin[3]

Abstract—**The increasing power consumption of Data Center Networks (DCN) is becoming a major concern for network operators. The object of this paper is to provide a survey of state-of-the-art methods for reducing energy consumption via (1) enhanced scheduling and (2) enhanced aggregation of traffic flows using Software-Defined Networks (SDN), focusing on the advantages and disadvantages of these approaches. We tackle a gap in the literature for a review of SDN-based energy saving techniques and discuss the limitations of multi-controller solutions in terms of constraints on their performance. The main finding of this survey paper is that the two classes of SDN- based methods, scheduling and flow aggregation, significantly reduce energy consumption in DCNs. We also suggest that Machine Learning has the potential to further improve these classes of solutions and argue that hybrid ML-based solutions are the next frontier for the field. The perspective gained as a consequence of this analysis is that advanced ML-based solutions and multi-controller-based solutions may address the limitations of the state-of-the-art, and should be further explored for energy optimization in DCNs.**

Index Terms—**Data Center, Software-Defined Networking, Integer Programming, Power Consumption, Energy, Routing.**

I. Introduction

The power optimization increasingly attracts many researchers in different sectors of wire and wireless networks [1]–[4]. Specifically, large-scale alternatives to fossil fuels that are secure, affordable, and low-carbon are lacking globally. The connection between access to energy and greenhouse gas emissions is the aspect of energy that receives the most attention. On one hand, Europe's energy grid is facing an unparalleled crisis [5]. Since early 2021, wholesale costs of electricity and gas have increased by as much as 15 times, which has had devastating consequences on both individuals and companies. On the other hand, the huge demand for information services nowadays is causing a dramatic increase in the usage of Data Center Networks (DCN) around the globe. As a consequence, 1 % to 1.5 % of worldwide power consumption is attributed to data center energy usage [6]. According to [7], the increase in power consumption in DCN has been 56% between 2005 and 2010 and it is expected to continue to increase in the future. Current estimates suggest that by 2020, the energy consumption of DCN in the US exceeded 139 billion kWh, and that interconnection devices (switches and links) consumed from 10 % to 20 % of the total energy [8]. The need for building effective network solutions in terms of energy usage and latency has expanded tremendously due to Industry 4.0's and the IoT's rapid development. Alternative techniques are required to address the power consumption issue in DCN. Software Defined Networking (SDN) is a new networking paradigm which can address power consumption. Compared to the legacy network architecture, SDNs have been effectively implemented in a variety of domains to satisfy the needs of the smart industry [9]. It is characterized by physically decoupling the control and data planes. The logically centralized SDN controller orchestrates the policy of the forwarding elements residing on its domain. The advantages of SDN has led to the incorporation of its architecture into a wide range of solutions. For instance, SDNs demonstrated promising results in optimizing the networks power consumption [10]. The term Software-Defined Data Center Networks (SD-DCN) emerged due to the employment of SDN to address various DCN issues such as the energy consumption. There are a number of studies that employed SDN in data centers to enhance the network management in general and to lower the power consumption.

This paper surveys the energy efficiency potential of SDN in enhancing power utilization through the optimization of traffic-aware features of DCNs. SDN has emerged as a critical paradigm for achieving the Network Resource Optimization (NRO) and for dynamically optimizing the network based on load and state. This is the most common carrier application as it optimizes the network using the near-real-time state of traffic, topology, and equipment. NRO uses a variety of southbound protocols (for example, NETCONF, BGP-LS, or OpenFlow) depending on the underlying network [11]. According to the literature, researchers address the problem by considering both hardware and software enhancement. In brief, the energy-saving technologies of hardware focus on frequency scaling techniques (i.e., changing clock frequencies). The motivating idea is that the power is consumed is a function of the working clock rate [12]. In the same context, other researchers optimize power consumption performance by consolidating multiple Virtual Machines (VM) in one physical machine [13]. Quality of Serivce (QoS) is upheld in these approaches by imposing multiple constraints. Routing-aware approaches have appeared in recent years. DCN topologies (i.e., fat-tree, Bcube, etc) come with rich connections and can achieve high network performance by balancing the workload of the DCN, however,

This research was supported by the department of the Information Technology, University of Debrecen. This paper has emanated from research supported in part by a Grant from Science Foundation Ireland under Grant numbers 13/RC/2077 P2 and 15/SIRG/3459.

[1] Department of Information Technology, University of Debrecen, Hungary. E-mail: mohammed.nsaif@(inf.unideb.hu, uokufa.edu.iq);

[2] Department of Computational Science, Eszterházy Károly Catholic University, Eger, Hungary. E-mail: kovasznai.gergely@uni-eszterhazy.hu;

[3] School of Electrical and Electronic Engineering, Technological University Dublin, Ireland. E-mail (ali.malik, ruairi.defrein)@tudublin.ie.

DOI: 10.36244/ICJ.2023.5.6

such structure of the DCN topology wastes energy since traffic volume is not proportional to energy consumption of DCN equipment, especially in the traffic valley time [14].

This paper reviews software-based energy-efficient solutions in the form of subcategories of the traffic-aware approaches. The structure of this paper is organized as follows. We start by classifying data center routing optimization methods and by showing the advantages and disadvantages of each technique in Section II. Then, we discuss the open challenges for the existing implementation approaches along with the potential future directions in Section III. Finally, we provide our conclusions in Section IV.

II. Routing Optimization Techniques

Power consumption of SDN-based DCN routing mechanisms depend on the mode of operation of the set of switches that forward the flows between sources and destinations. Power consumption of DCN switches can be measured in two ways: *dynamic*, which measures the power consumption of active links, and *static*, which measures the sum of power consumed by components such as chassis, fans, and switching fabric. We adopt the following notation conventions. An directed weighted graph, $G = (\mathbb{S}, \mathbb{E})$, models the network topology. The vertex-set is denoted by $\mathbb{S} = \{s_1, s_2, \ldots, s_n\}$, and the edge-set is denoted by $\mathbb{E} \subseteq \{e_{ij} \mid s_i, s_j \in \mathbb{S}\}$. The i-th switch is denoted as s_i and represents an OpenFlow switch. The primary function of each switch is to facilitate the routing of information through the path determined by the network controller. In the graph, G, every edge represents a link, and the link connecting the i-th and j-th switches is identified by e_{ij}. Network links can exist in either an active (ON) or inactive (OFF) state. We use binary variables, denoted by L_{ij}, to indicate the current state of network links. The variable L_{ij} is 1 if the link connecting switches i and j is active. This means that it can transmit packets between two ports. Conversely, L_{ij} is 0 if the link is inactive. In practice, each link consists of two ports, a sending port and a receiving port. Therefore, when designing power-efficient routing the working ports of each link should be considered. Similarly, the variable, ℓ_i, is set to 1 if the switch is active and 0 if the switch is inactive. The Network Power Consumption (NPC) is given in (1).

$$\text{NPC} = S_p \sum_{s_i \in \mathbb{S}} \ell_i + D_p \sum_{e_{ij} \in \mathbb{E}} L_{ij}. \tag{1}$$

Eqn. (1) relates the NPC to ℓ_i and L_{ij}, which denote the state of a corresponding switch, s_i, and link, e_{ij}, i.e., whether they are turned on or off. The variables D_p and S_p denote whether the power consumption is dynamic or static, respectively.

The authors analyzed the traffic of a wide range of DCN network datasets belonging to different layers of DCN topologies in [15]. The results reported in this paper showed that the link utilization was low and varied from one layer to another. The low-utilization links motivated researchers to propose new approaches that were more energy-aware than commonly used routing algorithms (e.g., Equal Cost Multiple Path (ECMP)). To address this problem, two types of methods have been proposed: (1) flow aggregation techniques and (2) flow scheduling techniques.

A. Flow Aggregation Techniques

Flow aggregation techniques consolidate data flows into a smaller set of links and switches that are sufficient to support existing data traffic demands, subject to a tolerance to a certain level of delay, packets loss, etc. To achieve minimum power consumption for a specific traffic matrix, switches and ports that are being used unnecessarily are put into sleep or shutdown mode. Fig. 1a shows how three flows share one link fairly based on the Transmission Control Protocol (TCP) sharing scheme. The disadvantage of these techniques is that using only a subset of the switches and links, a sub-topology, may result in performance degradation, which is typically characterized by a QoS measure. This QoS measure may indicate the significance of increases in delay time (i.e., due to computational complexity of the output solutions), or the extent to which links with higher utilization become overloaded and more susceptible to unplanned failures [16]. Balancing between the level of energy consumption and routing techniques that meet a desired QoS is of great importance. Next, we introduce and discuss the concept of flow aggregation.

1) Elastic-Trees: Need for Correlation-aware Power Optimization: The first analysis on Elastic-Trees (ET) was published in 2010 [17] by researchers from Deutsche Telekom and Stanford University. They considered three optimization techniques: Linear Optimizers (LO), Greedy Optimizers (GO), and ET. All of these optimizers work to consolidate traffic into a small subset of links that can handle the traffic volume. The results showed that LO are the worst due to their high computational complexity and time cost when the number of switches is high, while ET outperformed GO and improved link switch utilization. The authors evaluated ET using both simulations and experiments on a real network. They found that compared to traditional network architectures it could save significant amounts of energy. However, the study did not consider the correlation between flows. To address the high correlation between flows, the CARPO (CoRrelation-aware Power Optimization) algorithm in [18] aiming to reduce energy consumption in a DCN, dynamically consolidated traffic flows on a small set of links and switches, and switched off idle network components. CARPO uses correlation analysis among flows to consolidate traffic flows with low correlation while keeping the QoS at an acceptable level. A heuristic algorithm is used to find a consolidation and rate configuration solution with acceptable runtime overheads. CARPO introduced parameters to represent flow correlation, and the results showed that this extension of the ET, led to a power saving of 46 %.

The study in [12] suggested a platform composed of both software and hardware components, because there was no experimental environment available to test the optimization model. The software part was composed of monitoring (to check the state of the network), an optimizer (to calculate the subset of the topology), power controller (to change the state of the devices on/off), and routing algorithm (to calculate the paths). The hardware parts were composed of a traffic generator and power measuring device implemented using the NetFPGA platform. Experiments investigated the effect

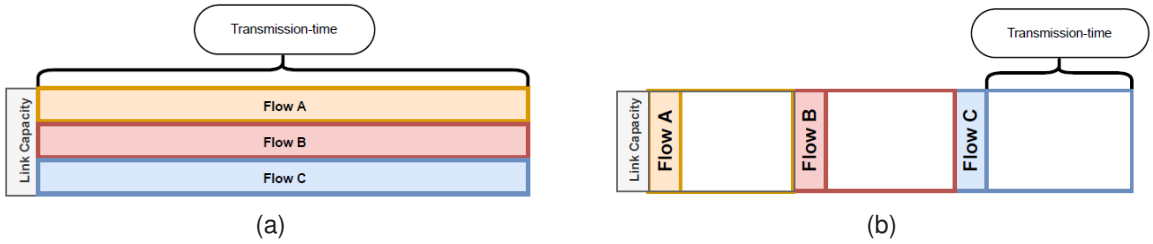


Figure 1. Flow scheduling & flow aggregation routing illustration. (a) Flow aggregation. (b) Flow scheduling.

of changing the frequency of the clock rate and the traffic consolidation on the power consumption in the DCN according to traffic demand. Unfortunately, the authors did not provide sufficient details about the algorithms or the API interfaces that gave access to the components.

The authors reported that the traffic patterns of GEANT networks are generally regular and predictable in [19]. To save power, they divided the traffic into intervals, and implemented link sleeping, which temporarily suspends certain links when the minimum or maximum link utility thresholds were met. They also rerouted the flows among the nodes of the GEANT network topology to maximize flow through the active links. Additionally, the study considered the network's performance under high-traffic conditions and balanced the traffic appropriately. The study described in the paper by Conterato et al. in [20] investigated the effect of using different values of over-subscription factors on reducing power consumption in a DCN, which was using traffic aggregation strategies. Three algorithms, First-Fit, Best-Fit, and Worst-Fit, were evaluated using Python and the Network Simulation Setup (FNSS). Different network workloads (20 %, 50 %, and 80 %) and oversubscription factors ($1:1$, $1:5$, and $1:20$) were tested. Devices that were not enrolled in the current routing state of the network were disconnected. The authors reported that up to a 70.02 % power saving could be achieved when the over-subscription factor was $1:20$ and the topology size was $k = 12$. These results conform with intuition because of the increase in the bandwidth and the number of links and switches in the topology. Motivated by the need for an adaptive routing framework for SD-DCN that can optimize power consumption while maintaining good network performance, the authors of [14] proposed an adaptive routing algorithm that considers factors such as link utilization and switch power consumption to determine the most energy-efficient path for network traffic in real-time. The algorithm used an Integer Linear Programming (ILP) model and a heuristic algorithm. The authors argue that the ILP model is costly and the time for finding a solution increases dramatically when 100 flows are injected simultaneously into the DCN. Conversely, the algorithm is integrated into an SDN controller and uses the OpenFlow protocol to communicate with switches. The proposed algorithms outperform existing routing algorithms such as ECMP in terms of energy consumption and network performance, as measured by the number of dropped packets. The research group built on this work and contributed new findings for the ILP model in [21]. Evaluations were conducted using LINGO [22], which is one of the computationally expensive commercial solvers. The authors of [14] re-implemented the model using solvers such as Gurobi, CP-SAT, and so on. Experiments were implemented in the authors' proposed tool neO-DCN. This comparison revealed that Gurobi outperformed the other solvers by one or two orders of magnitude for different traffic patterns.

OpenFlow switches are increasingly being used in data centers due to their potential to reduce energy consumption. However, the existing OpenFlow protocol does not include any mechanisms to optimize energy consumption. In [23], a power-aware extension to the OpenFlow protocol that does not compromise network performance was proposed. The extension defined new control messages in the OpenFlow standard such as OFPT-ORT-MOD and designed an OpenFlow Switch Controller (OSC) that can turn on and off switches and disable-enable ports of NetFPGA-based OpenFlow switches [12]. The authors conducted experiments to evaluate the energy savings achieved by the proposed extension, using a prototype implementation of the proposed extension on an OpenFlow switch. The experiments were designed to evaluate the energy savings achieved by the proposed extension. However, the authors did not provide sufficient details about the methods and the experiments conducted. Further research is needed to evaluate the effectiveness of these extensions. Table I summarizes the flow aggregation methods according to the adopted evaluation criteria and the main objectives.

2) Formulation: To optimize power consumption in DCNs, we present a general formulation of the objective function and constraints which describe an ILP model. The purpose of this ILP is to determine the optimum flow aggregation technique to reduce energy consumption. In the proposed model, taking inspiration from [14], [21], we introduce a multi-objective function rather than a single function as used in [14], [21]. This is achieved by incorporating the number of active switches, along with the number of active links. The objective is to maximize network utility while considering constraints related to link utilization, bandwidth, and traffic volume on the network links. By considering multiple objectives, our model aims to achieve a more comprehensive optimization of the network power consumption. In this setting network utility is defined as the overall satisfaction of users with respect to the demands they place on the network. Parameters of the DCN model used by the ILP are summarized in Table II:

The optimal configuration of active links and switches, that achieves the desired network utility, while minimizing the

Table I
FLOW AGGREGATION METHODS FOR ENERGY EFFICIENCY.

Study	Experiments				Objective	QoS	SDN
	Environment	Traffic	Topology	Controller			
Elastic Tree [17].	Test-bed	R	Fat-Tree	NOX	Link	-	√
CARPO [18].	Test-bed	R	Fat-Tree	-	Link	√	-
NetFPGA OpenFlow-based Platform [12].	Test-bed	R & A	Fat-Tree	NOX	Switch & Link	-	√
MTSDPFPR [19].	Mininet	R	GEANT	Floodlight	Link	√	√
Aggregation Strategies [20].	FNSS & Python	A	Fat-Tree	-	Link	-	√
OSC [12].	Test-bed	A	-	NOX	switch & Link	-	√
FPLF [14].	Mininet	A	Fat-Tree	POX	Link	√	√

Table II
PARAMETERS OF THE FLOW AGGREGATION MODEL.

Parameters	Definitions
$\mathbb{F}$	Set of flows, where a flow $f = (f.S_r, f.D_s, \lambda_f) \in \mathbb{F}$ is represented by source $f.S_r \in \mathbb{S}$, destination $f.D_s \in \mathbb{S}$, and bit rate $\lambda_f \in \mathbb{N}$
ℓ_i	$\begin{cases} 1, & \text{if switch } s_i \in \mathbb{S} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$
L_{ij}	$\begin{cases} 1, & \text{if link } e_{ij} \in \mathbb{E} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$
$FR(f,i,j)$	$\begin{cases} 1, & \text{if flow } f \in \mathbb{F} \text{ passes through link } e_{ij} \in \mathbb{E} \\ 0, & \text{otherwise} \end{cases}$
$BW_{ij} \in \mathbb{N}$	Bandwidth of link e_{ij}
$T_{ij} \in \mathbb{N}$	Traffic volume over e_{ij}

power consumption is determined by the ILP in (2).

$$\min : \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} + \sum_{i=1}^{n} \ell_i. \tag{2}$$

This objective function is subject to the following constraints:

1) Links and traffic correlation constraint.

$$\frac{T_{ij}}{BW_{ij}} \leq L_{ij}, \quad \forall e_{ij} \in \mathbb{E}.$$

A link is not activated unless at least one flow passes through it.

2) Links and flows correlation constraint.

$$FR(f,i,j) \leq L_{ij}, \quad \forall f \in \mathbb{F},\ \forall e_{ij} \in \mathbb{E}.$$

Flows can only pass through active links.

3) Utility constraint.

$$\sum_{f \in \mathbb{F}} FR(f,i,j) \cdot \lambda_f \leq BW_{ij} - T_{ij}, \quad \forall e_{ij} \in \mathbb{E}$$

The total packet rate of all flows passing through a link does not exceed the available bandwidth of that link.

4) Path conservation constraint.

$$\sum_{i=1}^{n} FR(f, f.S_r, i) = 1,\ \forall f \in \mathbb{F},$$

$$\sum_{i=1}^{n} FR(f, i, f.D_s) = 1,\ \forall f \in \mathbb{F},$$

Every flow has a unique source $f.S_r$ and destination $f.D_s$.

5) Flow conservation constraint.

$$\sum_{\substack{i=1 \\ i \neq f.S_r}}^{n} FR(f,i,j) = \sum_{\substack{i=1 \\ i \neq f.D_s}}^{n} FR(f,j,i), \quad \forall f \in \mathbb{F},\ \forall j \in \mathbb{S}.$$

A flow entering a node is equal to the flow leaving the node for all intermediate nodes.

6) Network connectivity constraint.

$$L_{ij} \leqslant \ell_i,\ L_{ij} \leqslant \ell_j, \quad \forall e_{ij} \in \mathbb{E}.$$

Each active link enforces the activation of the corresponding switches to maintain the connectivity of the network graph.

Solving the ILP model with these constraints results in an optimal flow aggregation policy that minimizes power consumption.

It is worth mentioning that the study assumes switches can turn on or enter sleep mode based on local traffic states, using Wake-on-Arrival (WoA) to wake up the switch when needed for forwarding packets and Sleep-on-Idle (SoI) technique for switches to save power when idle, without considering the transition time in the evaluation [24].

B. Flow Scheduling Techniques

Since the SDN controller maintains a global view of the underlay DCN infrastructure, it can calculate deadlines for flows and their size. This ability has motivated the development of new *scheduling algorithms* to manage the transmission of flows through a sequence of queues. These algorithms send the flows sequentially. This allows the flows to monopolise all the links of the path they traverse, using their full capacity, subject to the deadline and flow-size constraints. Fig. 1b shows how three flows are scheduled in a queue for transmission with a full bandwidth. A disadvantage is that these techniques are not appropriate for time-sensitive traffic (i.e., video streaming and VoIP [25]). Flows with higher priorities can be preemptively routed along the paths of other flows with lower priorities. Coupled with this, some applications in DCNs might not require large bandwidth, and be can slow, and thus not saturate the link capacity. Consequently, the link will be underutilized.

1) Methods: To avoid collisions and to achieve efficient power usage in DCNs, the authors in [26] proposed a technique that combined flow preemption and energy-aware routing to reduce energy consumption. Flows were divided into two lists:

Table III
FLOW SCHEDULING METHODS FOR ENERGY EFFICIENCY.

Study	Experiments				Objective	QoS	SDN
	Environment	Traffic	Topology	Controller			
Preemptive Flow [26].	NS-3	A	Fat-Tree	-	switch	√	-
Green Data Center [27].	-	A	B-Cube & Fat-Tree	-	switch	√	√
BEERS-1 [28].	OMNeT++	A	Fat-Tree & BCube	-	switch	√	√
Willow [24].	-	A	Fat-Tree	-	switch	√	√
BEERS-2 [29].	OMNeT++	A	Fat-Tree & BCube	-	switch	√	√
FLOWP [30].	OPNET	A	Fat-Tree	-	switch	√	√

a sending list and a waiting list. When a new flow entered the DCN, the algorithm checked the flow's priority based on its size. The flow either interrupted an ongoing data transfer and was directly routed to the destination, or was moved to the waiting list based on its assigned priority. Simulations were used to evaluate the performance of their technique and to compare it to other state-of-the-art techniques, such as ECMP. They used the ns-3 network simulator and a Fat-Tree topology. Results suggest that it can be a practical and effective way to make DCNs more energy-efficient.

The authors of [27] were motivated by the importance of implementing exclusion flow routing techniques on different topologies. They proposed a scheduling algorithm based on the priorities of flows, which assigned higher scheduling priority to smaller flow sizes, as in [26]. Experiments were conducted on two types of DCN topologies, namely BCube and Fat-Tree, using OpenFlow as the southbound API. Two metrics were used to evaluate the algorithm's performance: the flow arrival rate and the amount of power saved. The results showed an improvement in the utilization ratio of active links, leading to a reduction in power consumption in both topologies, along with the elimination of traffic congestion on active links.

The authors in [30] were motivated by the need to find an optimal subset to optimize power usage. A convex objective function was used to model energy consumption. Two strategies were pursued, an optimal combinatorial algorithm which was composed of two components (Most-Critical-First and Shortest Path (MCF-SP)) and a Random Scheduler (RS). MCF-SP managed flow scheduling based on the weight and the deadline of the flows. All the flows were sorted according to an Earliest Deadline First (EDF) policy. Power usage was optimized by minimizing the transmission rate of the flows per unit of time. By modeling the problem as a convex optimization problem the authors showed that MCF-SP found the optimal solution to the Deadline-Constrained Flow Scheduling (DCFS) problem under the assumption that flows were routed exclusively through a virtual circuit. Conversely, the RS involved relaxing an NP-hard problem, the Deadline-Constrained Flow Scheduling and Routing (DCFSR) problem. Relaxation consisted of finding an approximation based on a Fractional Multi-Commodity Flow (F-MCF) problem. The power optimization was based on two criteria: a minimum transmission rate criteria for the flows and a usage criteria for the links. The evaluation procedure was conducted using Python, however, the description of the set-up makes comprehensive understanding and reconstruction of the results challenging. Because the aggregation method was not suitable for network-limited flows, the application generated traffic at a high bit rate, and the flow's throughput depended on the network capacity of the routing path. To overcome this challenge a flow scheduling approach named "Willow" was proposed in [24], which took into consideration both the number of switches involved and the durations of their frame working times. A greedy approximate algorithm was designed that scheduled flows in a real-time manner. The proposed algorithm achieved a network energy consumption saving of 60 % of network energy compared to the conventional ECMP scheduling approach.

To overcome the problem of greed in flow selection when finding an energy-efficient path in [30], the authors of [28] proposed an approach that aggregated the flows with a similar deadline into a harmonic flow set, and scheduled them with higher priority, resulting in increased link utilization. The resulting scheduling algorithm was named "BEERS". It ran as a model in the SDN controller. Simulations were run using the OMNeT++ simulator, and results showed improvements in the link utilization and the energy consumption compared with the Exclusive Routing (EXR) algorithm. However, since all the flows competed for links in different periods, it was challenging to form a harmonic flow set at certain points in time, which posed a challenge for the algorithm. The authors extended the BEERs approach and conducted extensive experiments with two types of topology [29]. The results showed that their algorithm could reduce overall energy consumption with respect to traffic volume, and that it could also reduce the average flow completion time.

Conventional methods for reducing power consumption, such as turning off unused switches, can negatively impact network performance. To address this issue, the authors in [31] proposed a dynamic flow scheduling algorithm that balanced the workload on network switches to reduce power consumption. The algorithm considered the flow size, a threshold value that controlled the time delay. The algorithm is evaluated using a simulation-based approach, using OPNET, and the results showed that the algorithm could be an effective solution for reducing power consumption in DCNs without sacrificing performance. Table III summarizes the flow scheduling methods according to the adopted evaluation criteria and the main objectives.

2) Formulation: Similar to the approach in Section II-A2, we present the objective function and constraints that mathematically formulate an ILP model that optimizes power consumption in DCNs using flow scheduling techniques.

The goal we seek to achieve with the objective function

Table IV
PARAMETERS OF THE FLOW SCHEDULING MODEL.

Parameters	Definitions
L_{ijk}	$\begin{cases} 1, & \text{if link } e_{ij} \in \mathbb{E} \text{ is active during timeslot } k \\ 0, & \text{otherwise} \end{cases}$
$FR(f,i,j,k)$	$\begin{cases} 1, & \text{if flow } f \in \mathbb{F} \text{ is scheduled on link } e_{ij} \in \mathbb{E} \\ & \text{during timeslot } k. \\ 0, & \text{otherwise} \end{cases}$
$C_f \in \mathbb{N}$	Bandwidth required by flow f in Mbps
$La_f \in \mathbb{N}$	Latency tolerance of flow f in microseconds
$\alpha_f \in \mathbb{N}$	Starting timeslot of flow f
$\omega_f \in \mathbb{N}$	Ending timeslot of flow f, $\alpha_f \leq \omega_f$
$la_{ijk} \in \mathbb{N}$	Latency of link e_{ij} during timeslot k
$p_{ij} \in \mathbb{N}$	Power consumption of link e_{ij} in watts

in (3) is to minimize power consumption while maintaining network performance metrics. Prior to defining the objective function we define the role of its constituent components in Table IV.

The objective function computes a weighted sum of the variables L_{ijk} for all active time slots and the power consumption p_{ij} for each link.

$$\min : \sum_{i=1}^{n} \sum_{j=1}^{n} \left(p_{ij} \cdot \sum_{k} L_{ijk} \right) \tag{3}$$

The above objective function is subject to the following constraints to guarantee network performance: bandwidth, latency, etc.

1) Bandwidth constraint.

$$\sum_{f \in \mathbb{F}} C_f \cdot FR(f,i,j,k) \leq BW_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \ \forall k.$$

The total bandwidth usage of each link should not exceed its maximum bandwidth capacity BW_{ij} during any timeslot k.

2) Latency constraint.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k} la_{ijk} \cdot FR(f,i,j,k) \leq La_f, \quad \forall f \in \mathbb{F}.$$

The sum of the latencies of all links used by a flow f must not exceed its latency tolerance La_f.

3) Workload constraint.

$$\sum_{f \in \mathbb{F}} FR(f,i,j,k) \leq 1, \quad \forall e_{ij} \in \mathbb{E}, \ \forall k.$$

Each timeslot k can have at most one flow scheduled on a link e_{ij}.

4) Flow activation constraint.

$$FR(f,i,j,k) = 0, \quad \forall f \in \mathbb{F}, \ \forall e_{ij} \in \mathbb{E}, \ \forall k \notin [\alpha_f, \omega_f].$$

The decision variables $FR(f,i,j,k)$ are set to 0 for each flow f outside of its specified time interval between α_f and ω_f. α_f represents the timeslot at which the flow f is allowed to start its transmission. ω_f represents the timeslot by which the flow f must be completely served or finished with its transmission.

5) Links and flows correlation constraint.

$$FR(f,i,j,k) \leq L_{ijk}, \quad \forall f \in \mathbb{F}, \ \forall e_{ij} \in \mathbb{E}, \ \forall k.$$

Flows can only pass through active links, enforcing L_{ijk} to be in the ON state.

Similarly, the Path conservation constraint and the Flow conservation constraint from Section II-A2 have to be adapted, simply by quantifying over the timeslots k.

III. Open Issues and Discussion

Although the SDN paradigm presented a solution to many existing network issues such as those related to power consumption, SDN-based DCN power consumption still needs more investigation and verification under various workload circumstances. It is possible that implementing power optimization approaches based on SDN can result in an increase in the *response time* of the controller. This is because the controller has to continuously monitor the network and to make decisions on how to optimize power consumption while maintaining network performance. This overhead may cause delays in the controller's response time.

Moreover, finding the *optimal subset* of active links and switches can be an expensive process that may require significant *computational resources*. This can lead to a loss of performance in a DCN since the resources used for optimization cannot be used for other critical network functions. Furthermore, some approaches require the use of *heuristics or approximation algorithms*, such as the methods in Sections II-A1 and II-B1, which can typically provide only suboptimal solutions. Therefore, before implementing power optimization approaches based on SDN, it is important to carefully evaluate the potential trade-offs between power consumption, network performance, and computational resources. Further investigation may be required to identify the optimal balance between power optimization and network performance in a particular DCN. Further research which take these variables into account, will need to be undertaken. Moreover, there are several potential advantages when implementing a multiple SDN controllers in DCN, which is discussed in the next section.

A. Multiple Controller SDN

Using multiple controllers can potentially help mitigate the problem of increased response time in power optimization approaches based on SDN. By distributing the workload across multiple controllers, the overall response time can be reduced, improving network performance. Moreover, multiple controllers can potentially improve the efficiency of finding the optimal subset of active links and switches. Different controllers can be responsible for different parts of the network, and they can work together to find the best solution. This can help to reduce the computational demand and improve the speed of optimization. However, the synchronization of multiple controllers in an SDN environment can be achieved through various methods, such as: (1) *Consistency Protocols* ensure that all controllers have the same view of the network. (2) *Event Notifications* are used by controllers in order to communicate with each other. For example, the OpenDaylight (ODL) controller provides a notification service that allows controllers to subscribe to events and to receive notifications

when the state of the network changes. (3) *Replication*: Controllers can replicate/duplicate their state and share this copy with other controllers in the network. For example, the Floodlight controller uses a master-slave replication mechanism to synchronize the state of multiple controllers. By using one or more of these methods, multiple controllers in an SDN environment can synchronize their work and ensure the efficient operation of the network. Therefore, when considering the use of multiple controllers, it is important to carefully evaluate the potential benefits and drawbacks, and design the network architecture and control algorithms accordingly.

B. Machine Learning-Based Approaches

Machine Learning (ML) is a promising approach for solving a wide range of computer networks problems. ML-based techniques can be used to optimize power consumption in SD-DCN. By analyzing network traffic patterns and predicting network demand, ML algorithms can help SDN controllers make real-time adjustments and reconfigure the network to optimize the energy usage. One approach that has been explored in research is using ML to *predict the network demand* for different time periods, such as hourly or daily intervals [32]. Based on these predictions, the SDN controller can adjust the routing of network traffic to reduce power consumption during periods of lower demand. Other ML algorithms can be used to *analyze the behavior of individual network components*, such as switches or servers, to identify patterns that indicate when these components are not being used efficiently [33]. This information can then be used by the SDN controller to adjust network configurations and optimize power usage.

Finally, using ML algorithms to classify network traffic in real-time, SDN controllers can make more informed decisions about network traffic routing and resource allocation, leading to improved network performance and efficiency besides efficient power usage [34]. Overall, ML-based approaches have the potential to significantly reduce power consumption in DCNs, which is an important consideration for organizations looking to improve their energy efficiency and to reduce their carbon footprint.

IV. Conclusion

The escalating power usage in DCNs has become a global concern. Energy optimization techniques that are actively researched are notably scheduling and flow aggregation methods. This paper addresses a literature gap by reviewing cutting-edge SDN-based approaches for traffic scheduling and aggregation in DCNs and analyzing their pros and cons. It highlights the limitations of multi-controller SDN solutions due to performance constraints. Future research avenues include leveraging machine learning to optimize traffic algorithms and exploring hybrid solutions combining advanced scheduling, aggregation techniques, and multi-controller setups.

Acknowledgments

The authors would like to thank the anonymous peer reviewers for their valuable insights and constructive comments. Also, we thank the Department of Information Technology at the University of Debrecen, and the Science Foundation Ireland (SFI) for supporting this work.

References

[1] H. Garmani, D. Ait Omar, M. El Amrani, M. Baslam, and M. Jourhmane, "Joint beacon power and beacon rate control based on game theoretic approach in vehicular ad hoc networks," *Infocommunications Journal*, vol. 13, no. 1, pp. 58–67, 2021. [Online]. Available: DOI: 10.36244/ICJ.2021.1.7

[2] P. Varga, "The metrics of infocommunications journal keep improving," *Infocommunications Journal: A Publication of the Scientific Association for Infocommunications (HTE)*, vol. 14, no. 2, pp. 1–1, 2022.

[3] F. Rabee, A. Al-Haboobi, and M. R. Nsaif, "Parallel three-way handshaking route in mobile crowd sensing (pt-mcs)," *J. Eng. Appl. Sci*, vol. 14, pp. 3200–3209, 2019. [Online]. Available: DOI: 10.36478/jeasci.2019.3200.3209

[4] M. R. Nsaif, A. S. Al-Haboobi, F. Rabee, and F. A. Alasadi, "Reliable compression route protocol for mobile crowd sensing (rcr-msc)." *J. Commun.*, vol. 14, no. 3, pp. 170–178, 2019. [Online]. Available: DOI: 10.12720/jcm.14.3.170-178

[5] J. A. Cámara and V. S. Jiménez, "The european union facing the abyss: legislative review in the face of the energy crisis, 2022," *Journal of Energy & Natural Resources Law*, pp. 1–16, 2023. [Online]. Available: DOI: 10.1080/02646811.2023.2177409

[6] Y. Zhang, K. Shan, X. Li, H. Li, and S. Wang, "Research and technologies for next-generation high-temperature data centers–state-of-the-arts and future perspectives," *Renewable and Sustainable Energy Reviews*, vol. 171, p. 112 991, 2023. [Online]. Available: DOI: 10.1016/j.rser.2022.112991

[7] J. Koomey et al., "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, vol. 9, no. 2011, p. 161, 2011. [Online]. Available: https://alejandrobarros.com/wp-content/uploads/old/Growth_in_Data_Center_Electricity_use_2005_to_2010.pdf

[8] P. Sun, Z. Guo, S. Liu, J. Lan, J. Wang, and Y. Hu, "Smart fct: Improving power-efficiency for data center networks with deep reinforcement learning," *Computer Networks*, vol. 179, p. 107 255, 2020. [Online]. Available: DOI: 10.1016/j.comnet.2020.107255

[9] S. K. Singh, S. K. Sharma, D. Singla, and S. S. Gill, "Evolving requirements and application of sdn and iot in the context of industry 4.0, blockchain and artificial intelligence," *Software Defined Networks: Architecture and Applications*, pp. 427–496, 2022. [Online]. Available: DOI: 10.1002/9781119857921.ch13

[10] S. Shrabanee and A. K. Rath, "Sdn-cloud: A power aware resource management system for efficient energy optimization," *International Journal of Intelligent Unmanned Systems*, vol. 8, no. 4, pp. 321–343, 2020. [Online]. Available: DOI: 10.1108/IJIUS-07-2019-0032

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008. [Online]. Available: DOI: 10.1145/1355734.1355746

[12] N. H. Thanh, P. N. Nam, T.-H. Truong, N. T. Hung, L. K. Doanh, and R. Pries, "Enabling experiments for energy-efficient data center networks on openflow-based platform," in *2012 Fourth International Conference on Communications and Electronics (ICCE)*. IEEE, 2012, pp. 239–244. [Online]. Available: DOI: 10.1109/CCE.2012.6315905

[13] M. Carabaş and P. G. Popescu, "Energy-efficient virtualized clusters," *Future Generation Computer Systems*, vol. 74, pp. 151–157, 2017. [Online]. Available: DOI: 10.1016/j.future.2015.10.018

[14] M. Nsaif, G. Kovásznai, A. Rácz, A. Malik, and R. de Fréin, "An adaptive routing framework for efficient power consumption in software-defined datacenter networks," *Electronics*, vol. 10, no. 23, p. 3027, 2021. [Online]. Available: DOI: 10.3390/electronics10233027

[15] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010. [Online]. Available: DOI: 10.1145/1672308.1672325

[16] A. Malik and R. de Fréin, "A proactive-restoration technique for sdns," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. *IEEE*, 2020, pp. 1–6. [Online]. Available: DOI: 10.1109/ISCC50000.2020.9219598

[17] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks." in *Nsdi*, vol. 10, 2010, pp. 249–264. [Online]. Available: https://www.usenix.org/event/nsdi10/tech/full_papers/heller.pdf

[18] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "Carpo: Correlation-aware power optimization in data center networks," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 1125–1133. [Online]. Available: DOI: 10.1109/INFCOM.2012.6195471

[19] J. Ba, Y. Wang, X. Zhong, S. Feng, X. Qiu, and S. Guo, "An sdn energy saving method based on topology switch and rerouting," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–5. [Online]. Available: DOI: 10.1109/NOMS.2018.8406202

[20] M. d. S. Conterato, T. C. Ferreto, F. Rossi, W. d. S. Marques, and P. S. S. de Souza, "Reducing energy consumption in sdn-based data center networks through flow consolidation strategies," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1384–1391. [Online]. Available: DOI: 10.1145/3297280.3297420

[21] G. Kovásznai and M. Nsaif, "Integer programming based optimization of power consumption for data center networks," 2023. [Online]. Available: DOI: 10.14232/actacyb.299115

[22] "An Overview of LINGO," accessed: 20-03-2023. [Online]. Available: https://www.lindo.com/index.php/products/lingo-and-optimization-modeling

[23] T. H. Vu, P. N. Nam, T. Thanh, L. T. Hung, L. A. Van, N. D. Linh, T. D. Thien, and N. H. Thanh, "Power aware openflow switch extension for energy saving in data centers," in *The 2012 International Conference on Advanced Technologies for Communications*, 2012, pp. 309–313. [Online]. Available: DOI: 10.1109/ATC.2012.6404282

[24] D. Li, Y. Yu, W. He, K. Zheng, and B. He, "Willow: Saving data center network energy for network-limited flows," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2610–2620, 2014. [Online]. Available: DOI: 10.1109/TPDS.2014.2350990

[25] O. Izima, R. de Fréin, and A. Malik, "A survey of machine learning techniques for video quality prediction from quality of delivery metrics," *Electronics*, vol. 10, no. 22, 2021. [Online]. Available: DOI: 10.3390/electronics10222851

[26] Y. Shang, D. Li, and M. Xu, "Greening data center networks with flow preemption and energy-aware routing," in *2013 19th IEEE Workshop on Local&Metropolitan Area Networks (LANMAN)*. IEEE, 2013, pp.1–6. [Online]. Available: DOI: 10.1109/LANMAN.2013.6528281

[27] D. Li, Y. Shang, and C. Chen, "Software defined green data center network with exclusive routing," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 1743–1751. [Online]. Available: DOI: 10.1109/INFOCOM.2014.6848112

[28] G. Xu, B. Dai, B. Huang, and J. Yang, "Bandwidth-aware energy efficient routing with sdn in data center networks," in *2015 IEEE 17th international conference on high performance computing and communications, 2015 IEEE 7th international symposium on cyberspace safety and security, and 2015 IEEE 12th international conference on embedded software and systems*. IEEE, 2015, pp. 766–771. [Online]. Available: DOI: 10.1109/HPCC-CSS-ICESS.2015.12

[29] G. Xu, B. Dai, B. Huang, J. Yang, and S. Wen, "Bandwidth-aware energy efficient flow scheduling with sdn in data center networks," *Future Generation computer systems*, vol. 68, pp. 163–174, 2017. [Online]. Available: DOI: 10.1016/j.future.2016.08.024

[30] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu, "Energy-efficient flow scheduling and routing with hard deadlines in data center networks," in *2014 IEEE 34th International Conference on Distributed Computing Systems*. IEEE, 2014, pp. 248–257. [Online]. Available: DOI: 10.1109/ICDCS.2014.33

[31] J. Luo, S. Zhang, L. Yin, and Y. Guo, "Dynamic flow scheduling for power optimization of data center networks," in *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 2017, pp. 57–62. [Online]. Available: DOI: 10.1109/CBD.2017.18

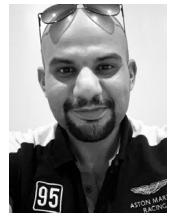
[32] D.-H. Le, H.-A. Tran, S. Souihi, and A. Mellouk, "An AI-based traffic matrix prediction solution for software-defined network," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6. [Online]. Available: DOI: 10.1109/ICC42927.2021.9500331

[33] B. Steiner, C. Cummins, H. He, and H. Leather, "Value learning for throughput optimization of deep learning workloads," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 323–334, 2021. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2021/file/a7e5da037a0afc90fa84386586929a26-Paper.pdf

[34] M. Nsaif, G. Kovásznai, M. Abboosh, A. Malik, and R. de Fréin, "Ml-based online traffic classification for sdns," in *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, 2022, pp. 217–222. [Online]. Available: DOI: 10.1109/CITDS54976.2022.9914138

Mohammed Nsaif is a Ph.D. candidate student at the University of Debrecen, Faculty of Informatics, Department of Information Technology. He received his M.S. in Infocommunication technology and communication systems from Kazan National Research Technical University in the Russian Federation. His research interests include software-defined networks, computer networks, wireless sensor networks and machine learning.

Gergely Kovásznai is an Associate Professor and Head of the Department of Computational Science at the Eszterházy Károly Catholic University in Eger, Hungary. He received his Ph.D. degree in Formal Methods and Automated Theorem Proving from the University of Debrecen, Hungary, in 2007. Over the years, he worked as a research fellow at the Aristotle University of Thessaloniki, Greece, at the Johannes Kepler University Linz, Austria, and at the Vienna University of Technology, Austria. His research interests include formal methods, formal verification, operations research, and machine learning.



Ali Malik is an Assistant Lecturer in computer engineering at the School of Electrical and Electronic Engineering, Technological University Dublin, Ireland. He holds Ph.D. degree in computing from the University of Portsmouth, United Kingdom, in 2019. He worked as a postdoctoral researcher at FOCAS Research Institute, Technological University Dublin, in areas related to data center, monitoring and software-defined networking. His current research interests include software-defined networks, vehicular networks, traffic engineering, machine learning, cybersecurity, microgrids and power networks.

Ruairí de Fréin is a CONNECT Funded Investigator and Lecturer at the School of Electrical and Electronic Engineering, Technological University Dublin, Ireland. He received the B.E. degree in Electronic Engineering in 2004 and Ph.D. degree in Time-Frequency Analysis and Matrix Factorization from University College Dublin (UCD), Ireland, in 2010. He held Marie Skłodowska-Curie fellowships in KTH Royal Institute of Technology, Stockholm and also with Amadeus SAS, Sophia Antipolis, France. Over the past few years he has developed algorithms for predicting quality-of-delivery metrics for network management and monitoring strategies for small cell networks, and monitoring techniques for Internet Protocol TeleVision (IPTV). His research interests include machine learning, sparse signal processing, software-defined networks, vehicular networks, microgrids and power networks.

Decomposition Based Congestion Analysis of the Communication in B5G/6G TeraHertz High-Speed Networks

Djamila Talbi, and Zoltan Gal *Member, IEEE*

***Abstract*—The New MAC mechanism plays a key role in achieving the needed requirements of the B5G/6G radio technology and helps to avoid high-speed frequency issues and limitations. With the help of the ns-3 simulator, we generated 42 different cases for the purpose of analyzing the impact of the network load on the overall effective transmission rate. Therefore, the use of the data-adaptive decomposition method the Empirical Mode Decomposition (EMD) on our non-stationary system benefits in the extraction of the important meaningful components. However, due to the highlighted direction dependency finding of EMD, Ensembled EMD (EEMD) being direction independent shows better performance on our data series. The extracted trend based on the proposed method matches the fitting curve, while the fitting curve parameters can be clusterized into 2 main clusters congested and non-congested cases of the radio channel throughput signal.**

***Index Terms*—Tera-Hertz technology, 6G, Beyond 5G, Empirical Mode Decomposition, Ensemble Empirical Mode Decomposition, Intrinsic Mode Function**

I. INTRODUCTION

WITH the rapid proliferation of the Internet of Things (IoT), an expansive multitude of end devices has emerged, necessitating the advent of a novel wireless generation capable of facilitating seamless connectivity with an exceptionally high bit rate. B5G/6G technology harnesses the potential of Terahertz bands, enabling the attainment of extraordinary data transfer speeds reaching several Tbps, accompanied by an impressively low latency of just 1 ms [5]. However, the effective management of the spectrum allocation encounters formidable challenges attributed to molecular absorption loss as well as the intricate interplay of diverse natural factors, encompassing pressure, relative humidity, and temperature, which profoundly impact the propagation environment [7]. The rapid growth of the Internet of Things (IoT) has resulted in an unprecedented number of connected devices, creating a demand for a new wireless generation that can handle the increasing volume of data and provide seamless connectivity. The current wireless technologies face limitations in terms of capacity and bandwidth, which hinder their ability to support the IoT ecosystem effectively. However, the emergence of B5G/6G and the utilization of Terahertz bands hold great promise in overcoming these limitations. Terahertz frequencies offer a significantly higher data rate potential, enabling transmission speeds in the range of several Tbps. By harnessing the Terahertz bands, the new wireless generation can address the constraints of current technologies, providing the necessary bandwidth and capacity to accommodate the expanding IoT landscape.

The Adaptive Directional Antenna Protocol for THz networks (ADAPT) protocol represents a pioneering Medium Access Control (MAC) mechanism specifically designed for the Terahertz frequency domain. ADAPT has demonstrated remarkable performance improvements, exhibiting a remarkable throughput of approximately 120 Gbps within a single radio cell accommodating 50 Mobile Terminals (MT) [3]. However, it should be noted that ADAPT does encounter certain limitations when operating in a heavily loaded network environment [3, 10]. In scenarios characterized by heightened congestion, the transmission time gradually escalates, thereby adversely affecting the overall channel throughput. Hence, our investigation seeks to make a significant contribution to the advancement of techniques for analyzing non-stationary and nonlinear THz throughput signals. By doing so, we aim to enhance network congestion state detection and overall performance optimization in real-world applications. The main highlights of the paper can be summarized as follows:

- The generation of ADAPT data along with the introduction of the utilized decomposition methods.
- The utilization of diverse decomposition methods with various analyses.
- The data series undergoes decomposition-based trend extraction, followed by the clusterization of the extracted trend parameters.

Chapter two of this work provides an overview of pertinent literature related to the decomposition and the current study. Chapter three explores the characteristics of the decomposition methods and the ADAPT MAC mechanism. Moving on to chapter four, an analysis is conducted on the performance of EMD and EEMD. Lastly, chapter five presents a comprehensive summary and conclusion of the findings derived from the study.

Djamila Talbi is with faculty of informatics, University of Debrecen, Debrecen, Hungary (e-mail: talbi.djamila@inf.unideb.hu)
Dr. Zoltan Gal is with faculty of informatics, University of Debrecen, Debrecen, Hungary (e-mail: gal.zoltan@inf.unideb.hu)

DOI: 10.36244/ICJ.2023.5.7

II. Related Work

Empirical Mode Decomposition (EMD) and Ensemble Empirical Mode Decomposition (EEMD) are two signal processing methods used to decompose non-stationary and nonlinear signals Intrinsic Mode Functions (IMFs). Therefore, the authors in [11] aim to compare those two methods in the analysis of a seismic signal. The results of this work show that the time-frequency spectrum obtained through EEMD more accurately reflects real geological conditions as compared to EMD. Other work has been done on the same topic using EMD, EEMD, and Variational Mode Decomposition (VMD) in [6] for chatter detection in milling. The researchers compared the three methods and found that EEMD and VMD were more effective than EMD. EMD has been widely used for trend extraction in various fields, it is a powerful tool for analyzing non-stationary and extracting meaningful information from them. However, trend extraction methods using the IMFs might differ.

In [4] the authors decompose the IMFs into two parts, the trend one of them. Moreover, in [2] the authors proposed a method for extracting the trend. The method involves decomposing the signal into IMFs, and then fitting the extracted trend component using the least squares method. The proposed method was validated using experimental data and was found to be effective in identifying the trend of the cement-burning zone flame temperature. Research conducted in [12] compared EMD and EEMD and revealed that EMD suffers from mode mixing, leading to inaccurate extraction of signal characteristics. On the other hand, EEMD successfully extracts meaningful components and exhibits superior performance in fault diagnosis for rotating machinery, as demonstrated through simulations and real-world applications. In a related study, another group of researchers investigated EEMD's effectiveness in overcoming mode mixing by introducing white noise [13]. EEMD accurately decomposed signals into distinct IMFs, thereby enhancing time-frequency analysis and providing more realistic time-frequency spectra in geology applications.

The team in [14] introduced a hybrid denoising method that combines thresholded IMFs with data-driven VMD, proving highly suitable for non-stationary seismic signals with reduced noise sensitivity. Additionally, [15] demonstrated that EEMD outperforms EMD and VMD for calculating respiration rates from PPG signals, achieving over 90% accuracy with an average error rate of 1 rate/minute. EEMD shows potential in simplifying sensor devices for accurate RR calculation.

III. Applied Methodology

In our research work, we collected data of the new MAC mechanism ADAPT that is compatible with the first standardization for the THz physical layer defined in IEEE 802.15.3d [3]. For our simulation, we employed the pre-existing example available in the second version of *TeraSim*, a platform designed specifically for simulating extremely high frequencies, integrated into the *ns-3* simulator. In our study, we utilized the ADAPT MAC protocol within the Macroscale scenario to evaluate its performance in the context of THz frequencies. Along with the new proposed parameters [9] the overlapped sectors and the step parameters, we generated 42 different cases. 7 different number of steps based on the properties mentioned in [9] (s = 1, 7, 11, 13, 17, 19, 23), 2 different topologies: the centered topology where the MT are distributed closer to the Access Point (AP), and the random uniform where the MT is distributed uniformly around the AP. The radius of the area under consideration is determined to be 18 meters, and there are 30 sectors within this area. With these values established, it becomes evident to calculate the population density parameter (ρ = n/A $[m^{-2}]$), where n represents the population count and A denotes the area of the circle.

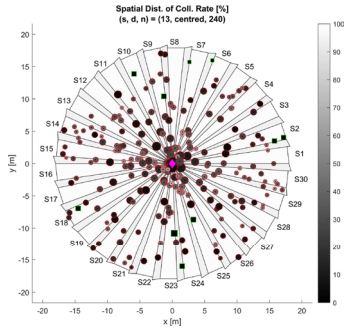


Fig. 1. Spatial distribution of the collision rate (step, d, n) = (13, 1, 240)

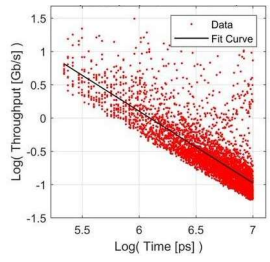


Fig. 2. Throughput vs. time (step, d, n) = (13, 1, 240)

We used 3 different numbers of MT ($n = 60, 240, 960$) having the overlapped ratio fixed $m = 0.3$ (see Fig. 1). The behavior of the throughput is particularly intriguing, as its distribution exhibits a gradual decrease over time, as illustrated in Fig. 2. This trend is further confirmed and supported by the fit curve, which aligns with the decreasing pattern of the throughput as time progresses. The representation declines the nature of the THz throughput, and the fit curve provides a mathematical model that captures and validates this diminishing trend. The combination of empirical evidence from the distribution plot and the fit curve's analytical support strengthens the significance of this observation.

A. Empirical Mode Decomposition

Empirical Mode Decomposition (EMD) can be applied to a nonlinear and non-stationary signal. Although, it is a sophisticated method for features extraction [1]. EMD decomposes the signal into a finite number of IMFs plus the residual, the decomposition process involves identifying all extrema of the signal *X(t)* and connecting them with the help of cubic splines to obtain an upper and a lower envelope [1]. It calculates the mean of the upper and lower envelope *m1*, then subtracted from the original signal to obtain the first component *h1* (1). However, the resulting IMF most of the time is not the right IMF because it does not satisfy the necessary conditions. Therefore, the sifting process is used and repeated to refine the IMF by eliminating riding waves, making it more symmetric and smoothing uneven amplitudes.

$$X(t) - m_1 = h \tag{1}$$

By doing so, the resulting IMF is then subtracted from the original signal to obtain a residue signal (2) where *c1* is the IMF after *j* sifting times.

$$X(t) - c_1 = r_1 \quad (2)$$

IMF1 is then decomposed into the second IMF using the same process. This process is repeated until the last residual. However, like any other processing method, EMD struggles with some limitations that should be considered such as the end effect where the first and the last points most of the time are not the extreme values. Also, the mode mixing limitation occurs when the IMF components overlap and cannot be separated from each other. It can happen when the signal has multiple scales of variation, leading to a difficult interpretation of the IMF components [8].

B. Ensemble Empirical Mode Decomposition

Ensemble Empirical Mode Decomposition (EEMD) is an advanced technique designed to enhance the traditional EMD method, specifically tailored for the analysis of non-stationary and nonlinear signals. One of the key challenges faced by EMD is the presence of the mode mixing problem, which can result in inaccuracies during signal analysis. To overcome these limitations, EEMD introduces a novel approach by incorporating an ensemble of white noise into the original signal before applying the EMD method. This addition of white noise ensures that each iteration of the EMD process produces slightly different results, effectively mitigating the mode mixing problem.

The EEMD process involves several steps: Firstly, the original signal is combined with white noise, leading to the generation of multiple noisy versions of the signal. Subsequently [8], the traditional EMD method is applied independently to each of these noisy signals, extracting a set of IMFs from every iteration. Finally, the IMFs obtained from all iterations are averaged, yielding the final IMFs. Through this ensemble approach, EEMD successfully overcomes the limitations of traditional EMD and improves the accuracy of the extracted IMFs, representing more faithfully the underlying components of the signal. This enhancement proves particularly valuable when dealing with intricate and non-stationary signals, enabling more dependable time-frequency analysis and extraction of signal characteristics.

IV. Measurement Scenario and Analysis of the Mobile ADAPT System

Since non-stationary and nonlinear systems in high-speed wireless communication networks require special signal processing methods, EMD can be a suitable approach. Consequently, we have decided to use EMD to decompose the throughput data of ADAPT and analyze the obtained results.

A. EMD Decomposition-Based Throughput Analysis

To observe the impact of applying the EMD on throughput analysis, we decided to apply EMD in a direct way from $time_{start}$ to $time_{end}$, and then inversely from $time_{end}$ to $time_{start}$. The results of this experiment show the resulting IMFs of the direct and inverse methods plotted versus the time (Fig. 3 and Fig. 4).

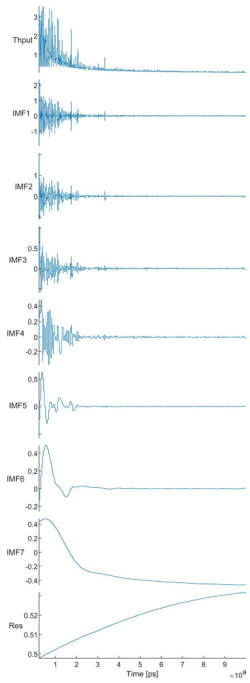


Fig. 3. Direct EMD on Throughput vs. time

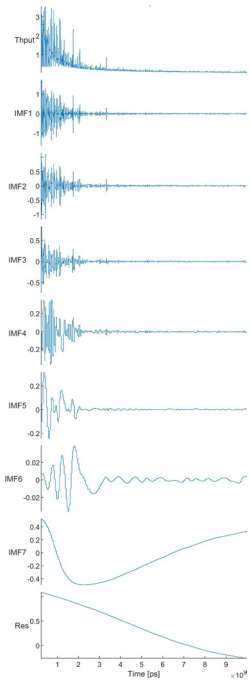


Fig. 4. Inverse EMD on Throughput vs. time (c = 0.48)

The experiment was in the case where the $step = 7$, the topology was centred, and $n = 960$. Our observation from the results was that the IMFs generated by the direct and inverse methods were significantly different, despite having different amplitudes for the same IMFs. This difference was further confirmed by a correlation coefficient of less than *0.5* ($c = 0.48$), which indicates that EMD is direction-dependent.

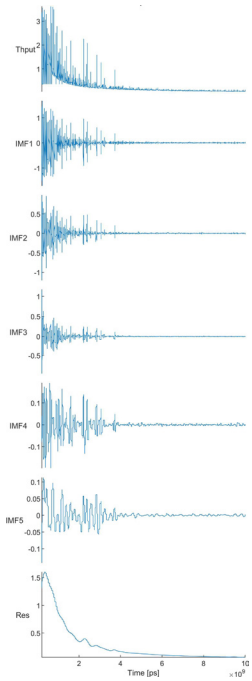


Fig. 5. Direct EEMD on Throughput vs. time (Noise = 0.05)

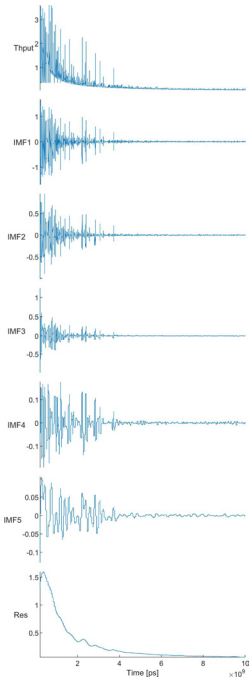


Fig. 6. Inverse EEMD on Throughput vs. time (c = 0.994)

The experiment was repeated using EEMD with 5 percent white noise in cases where we have $step = 23$, the topology centred, and $n = 960$. The IMFs generated by the direct and inverse methods were identical, despite having the same amplitude (Fig. 5 and Fig. 6). This finding was confirmed by a high correlation coefficient $c = 0.994$, which indicates that EEMD is direction-independent.

We came up with the idea of using the fast Fourier transform on the IMFs resulting from EMD to extract the frequency information. It is evident that the FFT gives adjacent sub-frequency bands in log2 of the frequency for adjacent IMFs, which is reminiscent of the dyadic filter bank (See Fig. 7 the throughput signal in case where we have $step = 11$, random uniform topology and $n = 960$, and Fig. 8 in case $step = 7$, centered topology and $n = 960$).

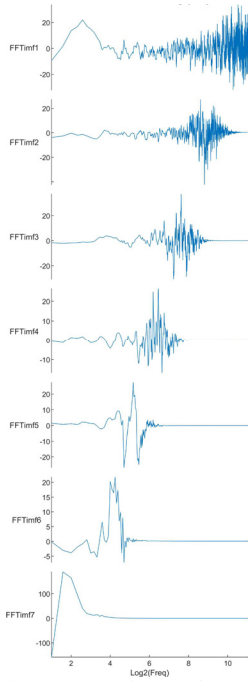


Fig. 7. Fast Fourier transform on IMFs (step, d, n) = (11, 2, 960) vs. frequency

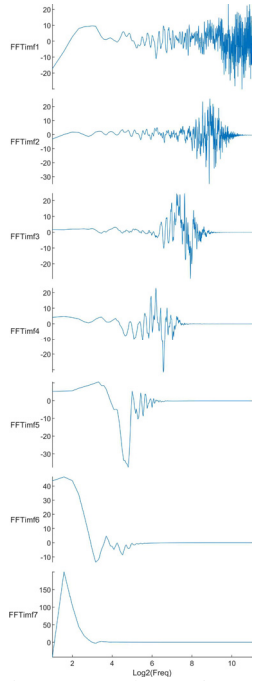


Fig. 8. Fast Fourier transform on IMFs (step, d, n) = (7, 1, 960) vs. frequency.

B. *Analysis of Data and Trend Extraction*

For extracting the trend using the IMFs components, one approach is, to sum up, each k consecutive IMFs together with the residual component (as Fig. 9 shows). It is obvious that if the number of IMFs included in each sum is precisely equal to k, then the resulting trend will exactly match the original throughput signal.

To select the optimal trend among the potential options, we suggest employing a Root Mean Square Error (RMSE) calculation to compare each trend candidate against the original signal. The trend component with the smallest non-zero RMSE will be chosen as the final trend. This approach ensures that the selected trend is as close as possible to the original signal. The RMSE values are plotted versus $trend_k$ in centered topology, $step = 23$, and $n = 960$ (Fig. 10).

By analyzing this graph, it becomes possible to visually identify the trend component with the smallest non-zero RMSE and thus select the optimal trend for the given dataset.

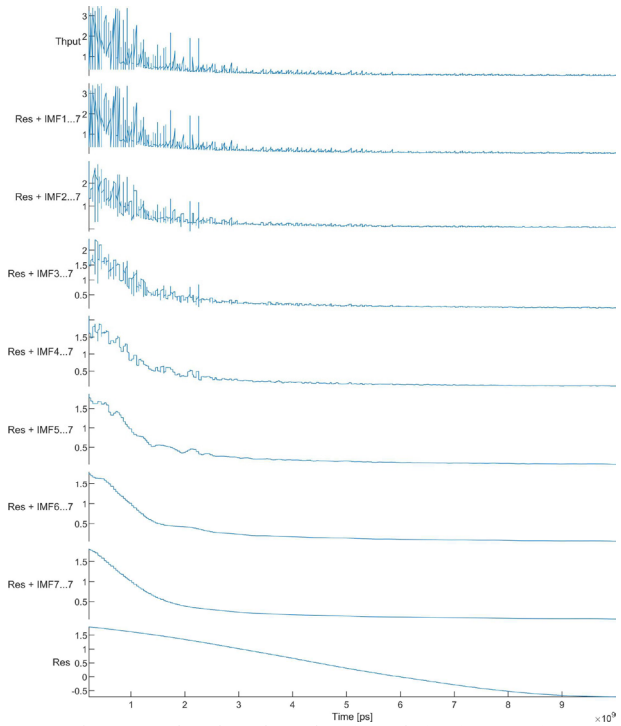


Fig. 9. IMFs-based trends vs. time (step, d, n) = (23, 2, 960)

The visualization of RMSE values versus $trend_k$ for all *42* cases simultaneously allows us to observe that the chosen $trend_k$ differs from case to case (Fig. 11). By examining this graph, we can identify that there is no single trend component that performs optimally across all cases. Instead, the optimal trend varies depending on the specific dataset under consideration.

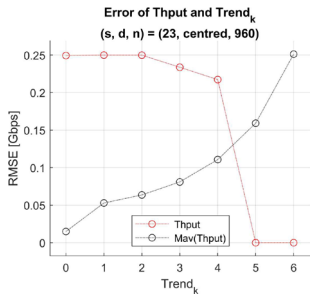


Fig. 10. The error of throughput and trend

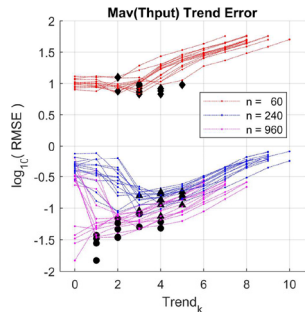


Fig. 11. RMSE of throughput and IMF-based trend

Therefore, it is important to perform an individualized analysis for each dataset to determine the appropriate trend component. This approach ensures that the chosen trend is both accurate and effective for a given dataset, leading to more reliable results and better decision-making.

The ability to observe and understand the variability in RMSE and $trend_k$ across different cases highlight the importance of customizing analysis to specific datasets and avoiding generalizations.

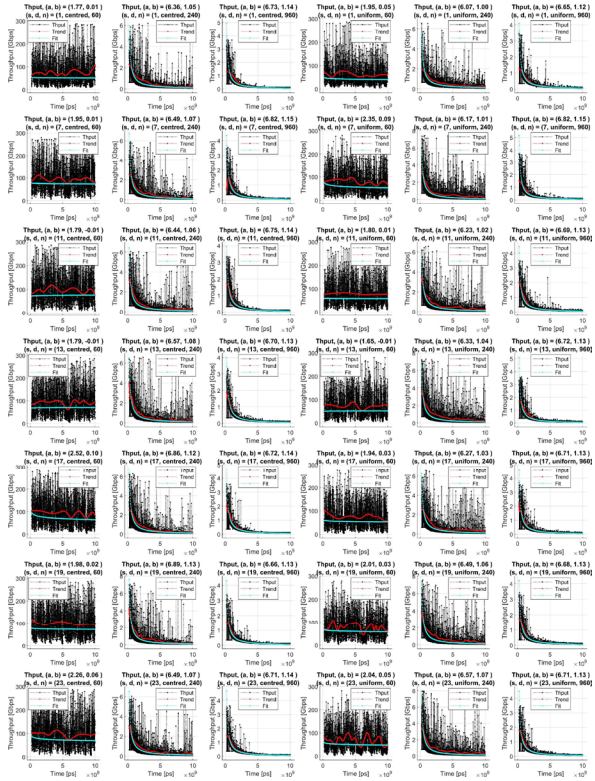


Fig. 12. Dependence of throughput, trend, and fit curve on time

To enhance our analysis, we included a fit curve of the original throughput dataset in addition to the chosen trend and the original data. This fit curve provides a visual representation of the overall trend in the data and facilitates the comparison of the chosen trend with the original signal (Fig. 12 presents a plot showing all 42 cases together).

Upon examining this graph, it is apparent that the chosen trend and the fitting curve are quite similar in the majority of cases. This similarity indicates that the selected trend is an accurate representation of the underlying trend in the data. However, there are some cases where the chosen trend deviates significantly from the fitting curve. These deviations may be the result of outliers or other anomalies in the data. By comparing the chosen trend and the fitting curve in this manner, we can gain a more comprehensive understanding of the data and make more informed decisions based on the analysis results.

Upon applying the fitting curve to the original signal, we meticulously extracted the fitting parameters and subsequently generated a scatter plot to provide visual insight into the outcomes of parameters a and b (as depicted in Fig. 13). Notably, the red-highlighted cluster is associated with instances of lower congestion levels, while the black cluster corresponds to more congested cases.

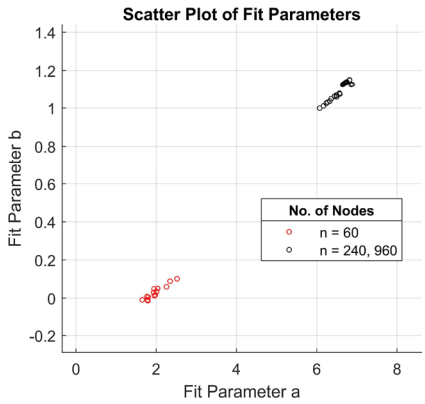


Fig. 13. Scatter plot of the fit parameters

These findings strongly imply that the fitting parameters, specifically parameters a and b, hold promising potential as valuable indicators for discerning and characterizing network congestion states. Such observations signify the scientific relevance and significance of the proposed methodology in understanding and quantifying the complexities of network congestion in our study.

V. Conclusion

The analysis of our throughput data using the EMD method in both left-to-right and right-to-left directions revealed a strong dependence on the processing direction. This dependence is likely attributed to the end effect issue, which affects the EMD's performance. In contrast, when applying the EEMD method, the results indicated its independence from the processing direction, showcasing its superiority in mitigating such issues. Furthermore, the trend extracted through our proposed method (in Section 3, Subsection B) demonstrated a remarkable correspondence with the fitting curve, showcasing the reliability and accuracy of our approach in analyzing radio channel throughput signals. This alignment between the extracted trend and the fitting curve underscores the effectiveness of our proposed method. Additionally, by examining the fitting curve parameters obtained from our method, we observed the emergence of two distinct clusters. These clusters corresponded to the congested and non-congested states of the radio channel throughput signal. This exciting finding implies that our proposed method not only effectively analyzes radio channel throughput signals but also enables precise detection and differentiation of various network congestion states.

Acknowledgment

This work has been supported by QoS-HPC-IoT Laboratory and project TKP2021-NKTA of the University of Debrecen, Hungary. Project no. TKP2021-NKTA-34 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the TKP2021-NKTA funding scheme.

the clock domain switch the synchronization accuracy can be drastically improved.

Regarding future work, the synchronization error of the master and slave device can be further reduced with fine tuning of PI controller or with the implementation of a more sophisticated solution such as Kalman-filtering. Another possible future research direction would be the evaluation and measurements of the different LOS and NLOS scenarios and the characteristics of the distance between the master and slave device. Furthermore, as the UWB technology has significant limitations beyond a certain distance, there is some initial research on a multi-hop UWB PTP system. Such a system can provide clock synchronization on the order of 10 ns over many times the UWB radio range. However, in this case, the synchronization errors are accumulated, offering an exciting research topic.

References

[1] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, H. H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis", Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences 454.1971 (1998), pp. 903–995.

[2] S. Lu, X. Wang, H. Yu, H. Dong, Z. Yang, "Trend extraction and identification method of cement burning zone flame temperature based on EMD and least square", Measurement 111 (2017), pp. 208–215, issn: 0263-2241, DOI: 10.1016/j.measurement.2017.07.047.

[3] D. Morales, J. M. Jornet, "ADAPT: An Adaptive Directional Antenna Protocol for medium access control in Terahertz communication networks", Ad Hoc Networks 119 (2021), p. 102540, issn: 1570-8705. DOI: 10.1016/j.adhoc.2021.102540.

[4] J.-M. Poggi: "Empirical Mode Decomposition for Trend Extraction. Application to Electrical Data", in: 2010.

[5] Z. Qadir, K. N. Le, N. Saeed, H. S. Munawar: "Towards 6G Internet of Things: Recent advances, use cases, and open challenges", ICT Express (2022), issn: 2405–9595,DOI: 10.1016/j.icte.2022.06.006.

[6] P. Seyrek, B. Şener, A. M. Özbayoğlu, H. Ö. Ünver, "An Evaluation Study of EMD, EEMD, and VMD For Chatter Detection in Milling", Procedia Comput. Sci. 200.C (Jan. 2022), pp. 160–174, issn: 1877-0509, DOI: 10.1016/j.procs.2022.01.215.

[7] A. Shafie, N. Yang, C. Han, J. M. Jornet, M. Juntti, T. Kurner, "Terahertz Communications for 6G and Beyond Wireless Networks: Chal lenges, Key Advancements, and Opportunities", IEEE Network (2022), pp. 1–8, DOI: 10.1109/MNET.118.2200057.

[8] U. B. de Souza, J. P. L. Escola, L. da Cunha Brito, "A survey on Hilbert-Huang transform: Evolution, chal lenges and solutions", Digital Signal Processing 120 (2022), pp. 103292, issn: 10512004, DOI: 10.1016/j.dsp.2021.103292,

[9] D. Talbi, Z. Gal, "Impact of Multi-Layer Recurrent Neural Networks in the Congestion Analysis of TeraHertz B5G/6G MAC Mechanism", in: 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2022, pp. 1–6. DOI: 10.23919/SoftCOM55329.2022.9911500.

[10] D. Talbi, M. A. Korteby, Z. Gal, "Neural Network Based Analysis of Terahertz Frequency Signal Propagation for B5G/6G Wireless Networks", in: 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), 2022, pp. 267–272, DOI: 10.1109/CITDS54976.2022.9914236.

[11] T. Wang, M. Zhang, Q. Yu, H. Zhang, "Comparing the applications of EMD and EEMD on time-frequency analysis of seismic signal", Journal of Applied Geophysics 83 (2012), pp. 29–34, issn: 0926-9851, DOI: 10.1016/j.jappgeo.2012.05.002.

[12] Yaguo Lei, Zhengjia He, Yanyang Zi, "Application of the EEMD method to rotor fault diagnosis of rotating machinery", Mechanical Systems and Signal Processing, Volume 23, Issue 4, 2009, pp. 1327–1338, issn 0888-3270. DOI: 10.1016/j.ymssp.2008.11.005.

[13] Tong Wang, Mingcai Zhang, Qihao Yu, Huyuan Zhang, "Comparing the applications of EMD and EEMD on time– frequency analysis of seismic signal", Journal of Applied Geophysics, Volume 83, 2012, pp. 29–34, issn 0926-9851, DOI: 10.1016/j.jappgeo.2012.05.002.

[14] L. Fangyu, Z. Bo, V. Sumit, M. Kurt, "Seismic signal denoising using thresholded variational mode decomposition", Journal of Exploration Geophysics, Volume 49, 2018, issn 0812–3985, DOI: 10.1071/EG17004.

[15] S. Hadiyoso, E. M. Dewi, I. Wijayanto, "Comparison of EMD, VMD and EEMD Methods in Respiration Wave Extraction Based on PPG Waves", Journal of Physics: Conference Series, Volume 1577, 2020, issn 1742–6596, DOI: 10.1088/1742-6596/1577/1/012040.

Djamila Talbi from Laghouat, Algeria, born in 1999. She holds a Bachelor of Science degree in Telecommunication from Amar Telidji University, Laghouat, Algeria, which she earned in 2019. In 2021, she furthered her academic journey, acquiring a Master of Science degree in the Systems of Telecommunication from Amar Telidji University, Laghouat, Algeria.
Currently, she is engrossed in pursuing a Ph.D. degree in the prestigious Faculty of Informatics at the University of Debrecen, Hungary, where she continues to display her dedication to advancing knowledge in her field. Since the beginning of her Ph.D. studies, she has been dedicated to advancing her research in cutting-edge areas of the telecommunications industry, with a primary focus on upcoming 6G/Beyond 5G (B5G) high-speed wireless communication systems and communication in the TeraHertz frequencies. Her research involves conducting thorough analyses and investigations into the limitations and challenges associated with such new frequencies. Through her academic pursuits, she aims to contribute valuable insights to the field and play a significant role in shaping the future of high-speed wireless communication systems in the 6G/B5G era.

Zoltan Gal is currently an associate professor at the Faculty of Informatics, University of Debrecen, Hungary. He earned MSc in electrical engineering and computer science from the Technical University of Timisoara, Romania and PhD in informatics sciences from the University of Debrecen. His scientific interest is focused on distributed processing and communication systems, sensor technologies and services in the Internet of Things. He was the CIO of his institute for 20 years and developed the university-level metropolitan area highs-speed data network and services with over 10k Internet nodes. He is Cisco Certified Network Professional lecturer since 1999 and taught over five hundred network professionals in the field. Starting in 2015 he is head of the Centre of High-Performance Computing at his university. He is an IEEE member since 1996 and published over one hundred twenty scientific conferences and journal papers: He supervises his own R&D&I project called QoS-HPC-IoT Laboratory.

In-network DDoS detection and mitigation using INT data for IoT ecosystem

Gereltsetseg Altangerel* and Máté Tejfel**

***Abstract*—Due to the limited capabilities and diversity of Internet of Things (IoT) devices, it is challenging to implement robust and unified security standards for these devices. Additionally, the fact that vulnerable IoT devices are beyond the network's control makes them susceptible to being compromised and used as bots or part of botnets, leading to a surge in attacks involving these devices in recent times. We proposed a real-time IoT anomaly detection and mitigation solution at the programmable data plane in a Software-Defined Networking (SDN) environment using In-band Network telemetry (INT) data to address this issue. As far as we know, it is the first experiment in which INT data is used to detect IoT attacks in the programmable data plane. Based on our performance evaluation, the detection delay of our proposed approach is much lower than the results of previous Distributed Denial-of-Service (DDoS) research, and the detection accuracy is similarly high.**

***Index Terms*—IoT anomaly detection, data plane, In-band Network Telemetry(INT)**

I. Introduction

With the growth of IoT usage and the ease of exploiting vulnerable IoT devices, the flow of IoT-based attacks has reached unprecedented levels [1], [2]. For example, Mirai is one of the most well-known IoT attacks. It targets insecure IoT devices and turns them into a massive botnet that can be used to launch powerful DDoS attacks. In 2016, Mirai attack on Dyn DNS (Domain name service) provider took down high-profile websites and services such as GitHub, Reddit, Netflix, and Airbnb.

One of the main reasons for the increase in IoT attacks is that organizations may not always have complete control over IoT devices that are located outside of their scope or not directly accessible. This could include situations where the organization operates in a shared office building or public space, where IoT devices may be installed by other tenants or individuals and are not managed by the organization. Despite this challenge, organizations can take measures to reduce the risks associated with these external IoT devices. These measures can be Network Segmentation and Firewalls, Monitoring and Anomaly Detection, Secure IoT Protocols, and Azure IoT Hub (or similar solutions). For example, the last solution enables secure communication between IoT devices and cloud applications while providing the ability to revoke access to unauthorized devices.

In this paper, we proposed an IoT monitoring, anomaly detection and mitigation solution for this issue. Our proposed solution is an in-network (data plane) approach for detecting and mitigating DDoS-like attacks in an IoT environment using INT data. DDoS is a common type of computer network attack that can be easily carried out using insecure IoT devices. INT is a new type of monitoring mechanism that can collect more detailed network information (INT data) in real-time than conventional monitoring, thereby helping to detect not only IoT but also other types of attacks. As far as we know, all previous works on IoT anomaly detection have used datasets based on network traffic. As IoT devices pose challenges to the implementation of standard security solutions, we aimed to create a solution in the SDN environment.

Our proposed method has two main advantages. First, the detection is faster since it is performed directly on the data processing path, and second, it is more efficient since it uses real-time INT data. The main contributions of this work are as follows.

- Generating a new INT dataset under normal and DDoS attack conditions in an SDN simulation environment and making it publicly available to fill research gaps.
- We are the first to propose a P4-based (data plane-based) method for DDoS detection and mitigation in an IoT environment using INT data.
- Evaluating the performance of the proposed method and conducting a comparison with a competing solution [3].

The rest of the paper is organized as follows. Section II is about related works, and Section III describes how we created an experimental network and collected INT data. Section IV and V describe our proposed model, the experimental results, and discussions. Finally, we summarize our work and suggest future directions in Section VI.

II. Related work

Several significant research works [4], [5], [6], [7], [8] have been proposed for IoT anomaly detection, with and without SDN. For instance, Del-IoT [9] is an IoT anomaly detection approach that employs a deep ensemble model to address data imbalance issues in network traffic datasets. It is implemented on an SDN controller. Bhunia and Gurusamy [7] present a machine learning-based anomaly detection and mitigation method for IoT traffic using SDN. They utilize the SVM algorithm to monitor and learn the behavior of IoT devices over time to detect anomalies.

In the current landscape of IoT anomaly detection studies [10], [11] controller-based anomaly detection methods are prevalent in SDN networks. The majority of these solutions leverage network traffic datasets (e.g., NetFlow, Wireshark)

The authors are with the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE), Budapest, Hungary. (e-mail: * gereltsetseg@inf.elte.hu; ** matej@inf.elte.hu).

DOI: 10.36244/ICJ.2023.5.8

with machine learning and deep learning techniques for anomaly detection.

Recent efforts in network applications have focused on reducing processing delays by offloading tasks from the control plane to the data plane or dedicated processors [12], [13]. Implementing anomaly detection directly in the packet processing path or data plane can significantly enhance detection speed compared to control plane solutions.

Since programmable data plane is a relatively new concept, data plane-based anomaly detection solutions are less common compared to controller-based solutions. However, there are some anomaly detection solutions specifically designed for the data plane. For example, Euclid [3] is a data plane-based DDoS detection solution that employs Shannon entropy based on the frequency of source and destination IP addresses. Sanghi et al. [14] focus on identifying potential attacks on data plane systems and present a scalable tool for real-time detection. Kang et al. [15] propose an approach to discover attack vectors in a data plane system and conduct preliminary experiments to demonstrate its feasibility. Although these solutions represent pioneering attempts to detect anomalies in the data plane, none of them utilize INT data.

To the best of our knowledge, there are two experimental solutions that leverage INT data for anomaly detection. Kim et. al. [16] uses a recurrent neural network (RNN), while our earlier work [10] utilizes a one-dimensional Convolutional neural network (1D CNN). However, both of these solutions are implemented on external controllers or servers. The main distinctive feature of our proposed solution in this paper is its ability to detect IoT anomalies directly on the data plane using INT data, offering a unique approach in this domain.

III. INT DATA COLLECTION

A. Overview of the programmable data plane and INT

Packet processing algorithms in network architectures are categorized into control and data planes based on their functions. Data plane algorithms define the packet processing pipeline, while control plane algorithms set the packet processing rules. The interaction between the control plane and the data plane is shown in Fig. 1a.

In traditional network architecture, both types of algorithms are preconfigured on each network device and are not easily customizable. Only device manufacturers have the capability to reprogram them. However, with the advent of the SDN paradigm, the control plane is decoupled from the data plane and operates on dedicated server(s). This separation has led to increased flexibility, manageability, openness, and programmability in computer networks [17].

The concept of the programmable data plane is relatively new compared to the programmable control plane. It enables anyone to quickly design, test, and deploy a variety of applications in the data plane.

P4 is one of the popular domain-specific programming languages used for defining data plane algorithms. The basic architecture of the P4 pipeline, as shown in Fig. 1b, consists of three main parts: the parser, match-action, and deparser.

1) **Parser:** The parser receives incoming packets and extracts header fields from them. This step involves parsing the packet's structure to identify and extract relevant information.
2) **Match-Action:** The match-action section processes the packet headers and metadata. It comprises one or more tables, with each table having a key part and an action part. During table application, the program attempts to find the most suitable key in the table based on the packet's header fields and metadata. Upon finding the appropriate key, the associated action is executed. If there is no matching key, the program either executes the default action if defined or does nothing. An example of a commonly used routing table in P4 involves the match key being the destination IP address. The match type can be the longest prefix match, and the corresponding action depends on the match result, such as forwarding the packet, dropping it, or applying no action.
3) **Deparser:** The deparser assembles the processed header fields and the original payload to build the outgoing packet. This step ensures the proper formatting and structuring of the packet before it leaves the network device [18].

Overall, data plane programmability with P4 or any other language empowers network administrators and developers to define customized data plane behaviors, offering greater flexibility and control over how network packets are processed and forwarded. Moreover, it allows for the development of many interesting applications [12].

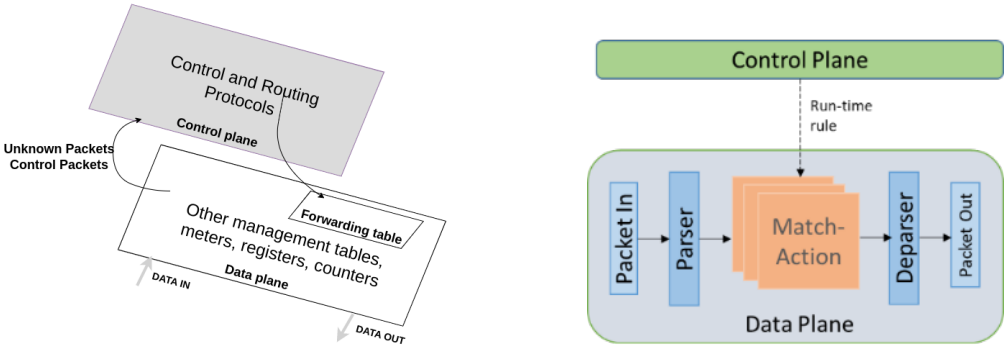
One notable application enabled by data plane programmability is INT. It is a new monitoring system that captures network telemetry information, such as hop latency, flow latency, and queue depth, directly from the data plane. The distinct advantage of this approach is that it bypasses the CPU-driven control plane, resulting in more real-time collection of telemetry data (INT data) compared to traditional monitoring mechanisms [16]. The real-time monitoring capability offered by INT provides valuable insights into network performance. This data enables network administrators to optimize network efficiency and performance while developing effective methods such as network anomaly detection, smart congestion control or routing mechanisms based on these insights.

B. INT data collection on testbed network

We created an INT-enabled SDN network, as shown in Figure 2, on the MININET[1] simulation program. This SDN network consists of a data plane, a control plane, IoT servers, and users. In the data plane, we deployed BMv2 software switches and configured P4 pipeline with INT support. For the control plane, we developed a custom controller using Python. This controller is responsible for configuring the packet processing rules and control rules for the data plane, providing the necessary instructions to the switches.

Within this simulated network, we emulated external IoT devices as attacker nodes and servers as target nodes. This

[1] http://mininet.org/



(a) Control & Data Plane Interaction.

(b) Abstract Packet Forwarding in P4.

Fig. 1. Network packet processing system.

setup allows us to analyze the behavior of the IoT anomalies and test the effectiveness of our proposed INT-based anomaly detection and mitigation approach.

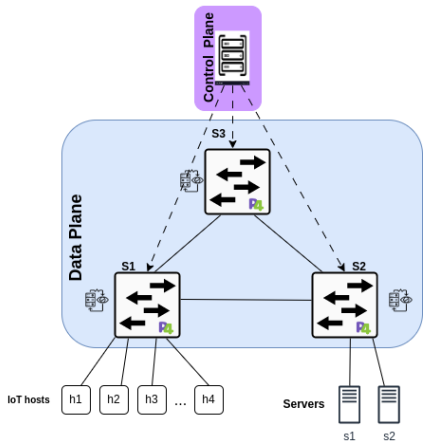


Fig. 2. Testbed SDN network.

After building a testbed network, we collected INT data including the queue depth of the switches' output interface under normal and attack conditions. To collect the INT data, we designed our own probe packet, which is a specialized packet transmitted across the network every millisecond from a specific source to a specific destination. As the probe packet is processed by the switches in the network, each switch appends its queue depth value to the packet. This action allows us to capture real-time queue depth value at each switch in the network. When the probe packet reaches its destination, we extracted queue depth information collected from each switch and stored it in a text file. As mentioned above, we created two kinds of data collection scenarios.

In the first scenario, we generated legitimate traffic, including UDP and ICMP flows using iperf[2] and ping[3] tools from external IoT devices to the target server. During these normal conditions, we captured INT data, including queue depth information, using the probe packet. Over the course of an hour, we accumulated more than 5000 data points, which were saved to a text file for analysis.

In the second scenario, we collected INT data under malicious traffic conditions. Due to hardware limitations, we need lighter attack. Therefore, we chose DDoS attacks with ICMP flooding. We created a DDoS attack using the hping3[4] tool, and legitimate traffic flows with UDP and ICMP messages from the attacker node to the target node. INT data were also collected for one hour under this condition.

C. Characteristics of INT data

After collecting INT data, we conducted a statistical analysis to understand their behavior. Firstly, we performed a t-test to evaluate and compare the difference between the means of the normal and malicious INT datasets, each consisting of 5000 samples. The result is presented in the first column of Table 1. A negative T-statistic (-33.74) suggests that the mean of the normal group is lower than that of the malicious group in our case. The P-value quantifies the probability of obtaining the observed results under the assumption of the null hypothesis. An extremely small P-value (e.g., 3.55e-181) suggests strong statistical significance and it indicates that there is substantial evidence to reject the null hypothesis in favor of the alternative hypothesis, supporting the presence of a meaningful and significant difference between the means of the two compared groups. In summary, based on this t-test result, we can conclude that the normal and malicious datasets were statistically significantly different at a very high probability.

Additionally, we created smaller datasets by randomly selecting 128 records multiple times from the original 5000-sample datasets of both malicious and normal conditions. The t-test results for these smaller datasets are shown in columns 2 to 4 of Table 1. These results were consistent with the previous whole dataset analysis, confirming the significant differences between the two conditions.

Furthermore, we compared the datasets using simple statistical measures like mean, median, mode, etc. Based on these measures, significant differences were observed among the datasets, indicating the potential to detect IoT anomalies

[2] https://iperf.fr/
[3] https://linuxize.com/post/linux-ping-command/
[4] https://www.kali.org/tools/hping3/

TABLE I
T-TEST RESULTS: STATISTICAL COMPARISON OF TWO DATASETS

	T-statistic	P-value
Whole dataset	-33.74	3.55e-181
Sample 1	-9.48	8.63E-18
Sample 2	-7.31	6.61E-12
...	...	...
Sample n	-9.628	3.15E-18

effectively. Since our anomaly detection solution for the IoT ecosystem aims to be simple and fast based on P4-language capabilities, we decided to implement it based on the mean value of queue depth.

IV. PROPOSED P4-BASED ANOMALY DETECTION AND MITIGATION APPROACH

After collecting and analyzing the INT data, we developed a P4-based packet processing pipeline incorporating IoT anomaly detection and mitigation mechanisms. Our proposed P4-based approach, depicted in Fig. 3, operates based on three distinct states: normal, detection, and mitigation state. Each state is designed with specific functionalities, and transitions between these states occur based on predefined conditions. We used global register and metadata in P4 to create a state that can be accessed and manipulated from both the data plane and the control plane. As a result, state transitions are effectively orchestrated by both the data plane and control plane according to the configuration specified in Table II.

In the subsequent text, the block numbers in parentheses provide a reference to how the numbered blocks of the P4 pipeline depicted in Fig. 3 correspond to the descriptions provided.

First of all, the packet counter value is analyzed to determine whether to maintain the current normal state or switch to the anomaly detection state. Packet counters are implemented at the ingress part of the pipeline in the data plane and are only read by the control plane. Based on the configured threshold counter value, the control plane will set either of the two states mentioned above. In order to determine a baseline for the packet counter, we conducted a test to determine the number of packets and bytes transmitted through the input and output interfaces of the S1 switch (which attackers are connected to) over a period of 2 seconds, while transmitting normal and attack packets. We then computed the mean value from 3000 samples of packet counter for each interface of S1, which were subsequently used as baselines to determine whether to initiate the detection state.

In the default (normal) state, the packets are handled according to the white blocks in Fig. 3. The main function in this state is forwarding the packet based on the IP routing table implemented on the ingress side. The state can then go to the detection state based on the condition in Table II.

In the detection state, anomalous traffic will be detected based on the mean of queue depth. Our proposed IoT anomaly detection mechanism is implemented at the egress of the packet processing pipeline on the data plane and it is shown in the red block (9). To implement this mechanism, we utilize

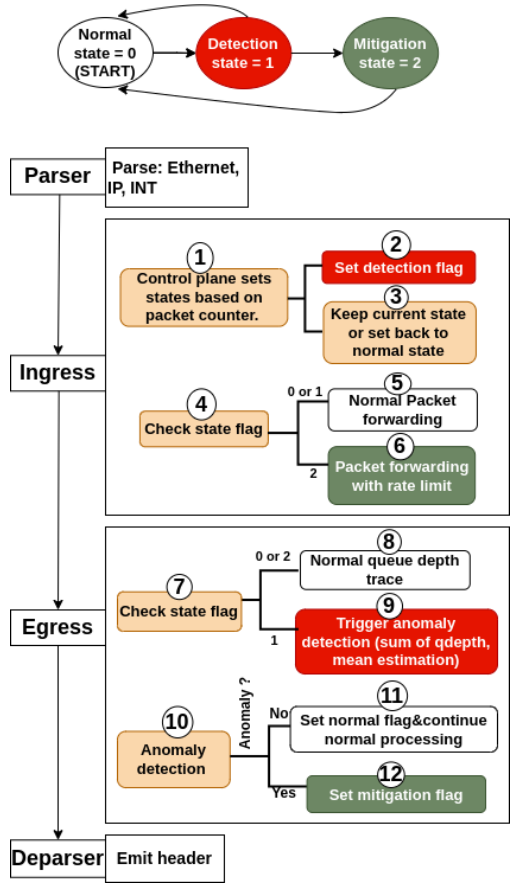


Fig. 3. IoT anomaly detection pipeline in P4.

a stateful register in P4 for storage, in which the queue depth value on the output interface is captured at the moment of transmission of probe packets. For instance, if 128 probe packets are transmitted, we record 128 queue depth values in the storage. After accumulating a certain number of queue depth values, we compute their mean value using the bitshift operator since the modulo and division operators are not available in the P4 language. This mean value is subsequently compared to a threshold. If it surpasses the threshold, the traffic is considered abnormal (10,12). The threshold itself was determined through statistical analysis during normal and malicious traffic scenarios in Section III. Then, the mitigation can be started if this mean value is higher than the baseline mean of the INT data under normal traffic conditions (12).

In the mitigation state, IoT attacks are mitigated by limiting the rate of the packet's incoming interface. To achieve this, rate limiting is implemented using the meter object of P4 at the ingress part of the pipeline on the data plane, as shown in the green block (6) of Fig. 3. P4 supports two types of meters: Indirect and Direct meters. In this implementation, we utilized indirect meters, which can be addressed by index [19]. To configure the rate limit parameters, we set up the corresponding traffic parameters in the control plane. The

TABLE II
STATE TRANSITIONING

	Start	Stop
Normal	Program starts in this state. Data plane set it from the detection state (10,11).	The control plane will switch to Detection (1,2).
Detection	The control plane decides to transition to this state based on packet counters implemented in the data (1,2).	The data plane decides whether to transition to a normal (10, 11) or mitigation state based on queue depth (10,12).
Mitigation	The data plane configures this state based on the detection result (10,12).	Based on the packet counter, the control plane awakens the normal state (1,3).

BMv2 software switch utilizes two-rate three-color meters[5], so we specify the Peak Information Rate (PIR) with Peak Burst Size and the Committed Information Rate (CIR) with Committed Burst Size. The mitigation mechanism may also employ other methods, such as reflecting anomalous traffic to the source port. The control plane decides when to switch back to the normal state based on the packet counter, following the criteria outlined in Table II.

Overall, our proposed P4-based anomaly detection and mitigation approach enhances the security and resilience of IoT networks by providing real-time monitoring, detection, and countermeasures against malicious activities. Its flexibility and customization capabilities empower network administrators to tailor the system to their specific requirements, making it a valuable addition to IoT network security measures.

V. Performance Evaluation

The number of queue depths used to calculate the mean value is a configurable parameter for the detection mechanism of our proposed model. We experimented with different values for this parameter, ranging from 16 to 256, to determine an optimal value that would result in high detection accuracy. Fig. 4 illustrates the relationship between the number of queue depths and the detection accuracy, helping us identify the value that provides the best performance.

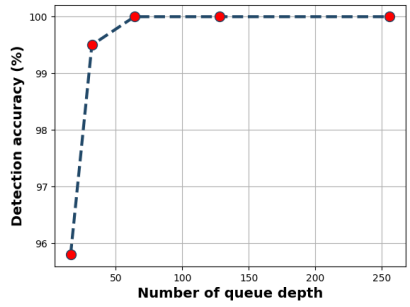


Fig. 4. Detection accuracy.

[5] https://www.rfc-editor.org/rfc/rfc2698

The average anomaly detection time is determined by multiplying the processing time (delay) of a single probe packet by the number of probe packets. A smaller number of probe packets results in faster anomaly detection.

In a real network, the packet processing delay depends on the network protocol, computational power at a node, and the efficiency of network interface cards. However, modern high-speed devices are capable of processing packets almost at line speed. In our case, probe packets can also be sent at line speed, but to clearly see and easily calculate the detection delay, we send 1 probe packet every 1 millisecond. Therefore, the anomaly detection delay is 16 milliseconds if the number of collected queue depth is 16, 32-millisecond if it is 32, and so on.

We have found that the intersection points with the highest accuracy and the acceptable delay happen when the number of queue depths is 64. With this optimal number, our proposed approach's detection delay is four times lower than the results of previous research on DDoS [3], and the detection accuracy is also higher. Additionally, it is evident that the detection delay can be absolutely low in real networks with line speeds such as 10Mbps, 100Mbps, and so on.

VI. Conclusions and Future Work

Our research represents the first experimental solution that employs INT data for detecting IoT attacks on the data plane (in-network). One of the primary advantages of our proposed approach is its lower detection delay, as it is directly implemented in the packet processing path. Additionally, our method leverages real-time INT data, resulting in more efficient and accurate detection capabilities. Compared to previous research, our approach exhibits relatively low detection delays and high accuracy.

Furthermore, by offering real-time monitoring, detection, and countermeasures against malicious activities, our solution provides a valuable tool to secure IoT environments. Its inherent flexibility and customization options enable network administrators to tailor it according to their specific needs, making it a valuable addition to IoT network security measures.

Despite our successful results, we acknowledge the limitations of our work. Our test environment only allowed us to test DDoS attacks with ICMP floods. To further validate our proposed approach, we plan to conduct tests on real hardware with various types of attacks. Moreover, we are aware that some IoT sensors may periodically generate large amounts of data, which we have not yet considered in our approach. Addressing such exceptional scenarios will be a valuable aspect of our future work.

.

Acknowledgment

The authors would like to acknowledge the financial support provided by project no. FK_21 138949 from the National Research Development and Innovation Fund of Hungary, as well as the contribution from Ericsson, which greatly aided the completion of this research paper.

References

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," Computer, vol. 50, no. 7, pp. 80–84, 2017. DOI: 10.1109/MC.2017.201

[2] E. Bertino and N. Islam, "Botnets and Internet of Things security," Computer, vol. 50, pp. 76–79, 2017. DOI: 10.1109/MC.2017.62

[3] A. D. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation," IEEE Transactions on Network and Service Management, vol. 18, no. 3, pp. 3121–3139, 2021. DOI: 10.1109/TNSM.2020.3048265

[4] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," Internet of Things, vol. 7, p. 100059, 2019. DOI: 10.1016/j.iot.2019.100059

[5] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BAIoT—network-based detection of IoT botnet attacks using deep autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018. DOI: 10.1109/MPRV.2018.03367731

[6] V. Timčenko and S. Gajin, "Machine learning based network anomaly detection for IoT environments," in ICIST-2018 conference, 2018.

[7] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in 2017 27th International telecommunication networks and applications conference (ITNAC). IEEE, pp. 1–6, 2017. DOI: 10.1109/ATNAC.2017.8215418

[8] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on IoT devices via SDN-based monitoring of mud activity," in Proceedings of the ACM Symposium on SDN Research, pp. 36–48, 2019. DOI: 10.1145/3314148.3314352

[9] E. Tsogbaatar, M. H. Bhuyan, Y. Taenaka, D. Fall, Kh. Gonchigsumlaa, E. Elmroth, and Y. Kadobayashi, "DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT," Internet of Things, vol. 14, no. March, p. 100391, 2021. DOI: 10.1016/j.iot.2021.100391

[10] G. Altangerel and M. Tejfel, and E. Tsogbaatar, "A 1D CNN-based model for IoT anomaly detection using INT data," in 2022 IEEE 16th International Scientific Conference on Informatics (Informatics). IEEE, pp. 106–113, 2022. DOI: 10.1109/Informatics57926.2022.10083469

[11] Dubem Ezeh, and J. de Oliveira, "An SDN controller-based framework for anomaly detection using a GAN ensemble algorithm", Infocommunications Journal, Vol. XV, No 2,pp. 29–36, June 2023. DOI: 10.36244/ICJ.2023.2.5

[12] G. Altangerel and M. Tejfel, "Study on emerging applications on data plane and optimization possibilities," in International Journal of Distributed and Parallel systems (IJDPS) Vol 13, No. 1, January 2022. DOI: 10.5121/ijdps.2022.13101

[13] L. Sándor, G. Csaba, J. Pető and P. Vörös and G. Szabó "In-Network Velocity Control of Industrial Robot Arms." In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 995–1009. 2022.

[14] A. Sanghi, K. P. Kadiyala, P. Tammana, and S. Joshi, "Anomaly Detection in Data Plane Systems using Packet Execution Paths," in SPIN 2021 - Proceedings of the 2021 ACM SIGCOMM Workshop on Secure Programmable network INfrastructure, no. 1, pp. 9–15, 2021. DOI: 10.1145/3472873.3472880

[15] Q. Kang, J. Xing, and A. Chen, "Automated attack discovery in data plane systems," 12th USENIX Workshop on Cyber Security Experimentation and Test, CSE T 2019, co-located with USENIX Security, 2019.

[16] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, L. J. Wobker, and B. Networks, "In-band Network Telemetry via Programmable Dataplanes," Sosr, no. Figure 2, pp. 2–3, 2015. Available: https://www.cs.princeton.edu/~nkatta/papers/int-demo.pdf

[17] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," IEEE Communications Surveys Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014. DOI: 10.1109/SURV.2014.012214.00180

[18] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," Computer Communication Review, vol. 44, no. 3, pp. 87–95, 2014. DOI: 10.1145/2656877.2656890

[19] L. Vanbever, "Lecture Notes - Advanced Topics in Communication Networks Programming Network Data Planes," 2019.

Máté Tejfel received his B.Sc., M.Sc. and Ph.D. Degree in Computer Science, from ELTE, Budapest Hungary. He is currently working as an Associate Professor in the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE). His research interest includes programming languages, correctness check, SDNs, and network optimization. For more information, visit his database at 0000-0001-8982-1398 orchid-id.

Gereltsetseg Altangerel received her B.Sc. M.Sc. Degree in Information Technology, from the Mongolian University of Science and Technology (MUST). Currently, she is an assistant teacher and a Ph.D. candidate in the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE) under the supervision of Professor Tejfel Máté. Her research interest includes SDNs, deeply programmable networks, and network optimization.
For more information, visit her database at 0000-0002-1594-8158 orchid-id.

The performance of modern centrality measures on different information models and networks

Péter Marjai, Máté Nagy-Sándor, and Attila Kiss

***Abstract*—For the last few years networks became integral parts of our everyday life. They are used in communication, transportation, marketing, and the list goes on. They are also becoming bigger, and more complex and dynamic networks also start to appear more. In light of this, the problem of finding the most influential node in the network remains of high interest however, it is getting more and more difficult to find these nodes. It is hard to grasp the true meaning of what is really being the most influential node means. There are several approaches to define what the most vital nodes are like having the most edges connected to them or having the shortest paths running through them. They can be also identified by calculating the influence of their neighbors, or evaluating how they contribute to the whole of the network. Over recent years various new centrality measures were proposed to order the importance of the nodes of a network.**

In this paper, we evaluate the performance of three modern centrality measures, namely Local Fuzzy Information Centrality (LFIC), Local Clustering H-index Centrality (LCH), and Global Structure Model (GSM) on different information models, and compare them with conventional centrality measures. In our experiments, we investigate the similarity between the top-n ranking nodes of the measures, the influential capacity of these nodes as well as the frequency of the nodes with the same centrality value.

***Index Terms*—Complex networks, centrality, LFIC, LCH, GSM, information diffusion, SIR, Independent cascade, Linear threshold.**

I. Introduction

COMPLEX networks are present in all areas of the modern world, so their investigation is extremely important. Network science includes the theory of real networks and takes other various methodologies into action such as graph theory, statistical physics, geometry, and stochastic processes. Networks can be found anywhere in everyday life. They are used in communication [1], [2] transportation [3], marketing [4], and the list goes on. They are also becoming bigger, and more complex and dynamic networks also start to make an appearance. Finding the most vital nodes has become a fundamental problem in nowadays network science however, it is getting more and more difficult to find these nodes. Determining the most influential node in a network is an important topic with many uses, such as speeding up the spread of information or monitoring and controlling the course of rumors and disease. For example, the Authors of [5] use centrality measures to identify the most influential users in social networks. In [6], the disease control actions are applied to a target group that has been chosen based on centrality, instead of the whole community. Centrality measures are used to identify the source of rumors in [7].

Over time various centrality measures have been proposed, however, each has its drawbacks. Because of this reason, new measures are constantly being developed. Local Fuzzy Information Centrality (LFIC) [8] uses a box for every node that contains the node's closest neighbors. The information that can be found in a node's box is used to evaluate the significance of the node. To calculate the uncertainty of the amount of information in the boxes, and to calculate the contribution of a node's neighbors, an improved Shannon entropy is used. A lot of centrality measures take the whole network into account, but in real life's huge networks, these are not applicable because of their sheer size. Local Clustering H-index Centrality (LCH) [9] only takes the local information into account. While calculating the node's importance, it considers the quality, influence, and topology of first-order and second-order neighbor nodes. Global Structure Model (GSM) [10] not only uses a node's self-influence to rank the nodes but also the node's influence on the whole network. To achieve this, the method utilizes k-shell clusterization.

There are various research that compare the different centrality measures. The Authors of [11] provide a comprehensive summary of how different traditional centrality measures identify the top-k nodes, as well as their extensions, applications, approximation methods, and their connection with dynamic networks. In [12] the Authors investigate the connection between a node's centrality value and its ability to maximize the number of connected components. They found that degree-like centralities are more suitable measures than path-like centralities for the above-mentioned problem. In [13] it is investigated how different centrality measures can be used to mine social network data. The authors of [14] examine how centrality measures that were designed for social networks perform in the case of psychological networks.

The information model that is used in a network can also affect the behavior of the nodes. The SIR model [15] starts with a non-empty array of infected nodes. In each turn, the infected nodes try to infect their neighbors with a fixed probability. They also have a fixed probability to recover. Recovered nodes can not be infected again. Independent Cascade model [16] is a stochastic information diffusion model that uses cascading to flow the information through the network. Each node can have two states, active or inactive. In each step, the active nodes have a fixed probability to activate their passive neighbors. An active node can only try to activate its neighbors once. Another information model is the linear threshold model [17]. It also

Péter Marjai, Máté Nagy-Sándor and Attila Kiss, Department of Information Systems, ELTE Eötvös Loránd University, Budapest, Hungary.
Attila Kiss was also with J. Selye University, Komárno, Slovakia (E-mail: g7tzap@inf.elte.hu, kiss@inf.elte.hu)

DOI: 10.36244/ICJ.2023.5.9

works in iterations. The nodes became active after the ratio of their active and passive exceeded the pre-defined threshold. A survey on current questions and possibilities in information propagation is introduced in [18].

A social network contains a group of people who are connected to each other through social relationships and interactions, such as relationships with family members, friendships, being colleagues or neighbors, and so on. Ties with some social network members can span many years or even a lifetime. The propagation of information is usually interpreted on such networks, such as disease spreads, or gossip. The pages on the web and the hyperlinks connecting them also form a network. The spread of information can also be interpreted on these networks such as the spread of news and fake news.

In this paper, we study three recently proposed centrality algorithms that take different aspects of the nodes into account. Since nodes with the same centrality values are indistinguishable, we examine the frequency of the achieved centrality values. After identifying the most important nodes, and calculating the centrality value of each node with these methods, we examine the propagation of information that has been launched from the vital nodes. We use multiple information diffusion models to inspect the propagation capacity of these nodes. Lastly, the time it takes for the algorithms to rank the nodes is also compared.

The organization of the remainder of this paper is as follows. In Section II the preliminary concepts and the definition of the centrality measures and information diffusion models are presented. The details of the used data and the explanation of the conducted experiments are presented in Section III. Lastly, Section IV contains the conclusions and discusses the different future possibilities to investigate the matter at hand.

II. Concepts and problems

A. Centrality measures

For ease, of reference consider a network as an undirected simple graph, $G = (V, E)$, where V represents the set of vertices, while E is the set of edges that connect the different vertices. $N = |V|$, expresses the number of the nodes, while the number of the edges is represented as $M = |E|$. The traditional centrality measures DC, BC are defined as follows.

Degree centrality (DC) indicates the number of incidents edges upon a node. In case of the risk of catching whatever goes through the network (like infections, a virus, or information) nodes with a higher degree are more likely to be involved. It was proposed by Freeman in [19]. The degree centrality of vertice v, expressed as d_v, is defined as:

$$DC(v) = \sum_{w}^{N} x_{vw}. \quad (1)$$

where w implies the nodes that are connected to v, while x_{vw} represents the link between v and w. The value of x_{vw} is 1 if there is a link between v and w, otherwise 0.

Betweenness centrality (BC) was introduced in [20] and is based on the shortest paths in the network. It enumerates the cases when a vertice acts like a bridge between two other vertices. It is defined as follows:

$$BC(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

Local Fuzzy Information centrality (LFIC) [8] takes a somewhat similar approach to Shannon entropy in information theory which is the following. The more uncertain the message, the larger its Shannon entropy is. Let us consider v node's box (the set of nodes whose shortest distance from v is no longer than a given value) to be the message. The Shannon entropy can be applied to this box to measure node v's importance. The larger the uncertainty in the box, the more vital the node is. The LFIC value can be calculated using the following steps. First, obtain the box size of node v, namely L.

$$L = \lceil \frac{k_v}{2} \rceil$$

where k_v indicates the largest shortest distance from any node to v. Secondly, calculate the weight of the nodes based on their distance from node v.

$$X(l) = exp(-\frac{l^2}{L2})$$

where L is the previously obtained box size and l is the distance from node v. After this, obtain the fuzzy number of the nodes in the box.

$$f_v = n_v(l)X(l)$$

where $n_v(l)$ is the number of nodes that have the shortest distance of l from node v. Next, calculate $F_i(L)$, the total fuzzy number of nodes in the box.

$$F_v(L) = \sum_{l=1}^{L} f_v(l)$$

where $f_v(l)$ is the fuzzy number of nodes with the shortest distance from v being l. Calculate the probability $p_v(l)$ of the nodes in the box

$$p_v(l) = \frac{1}{e} \frac{f_v(l)}{F_v(L)}$$

where $f_v(l)$ and $F_v(L)$ are the fuzzy number and the total fuzzy number that are explained above. Finally, obtain the centrality value of node v as:

$$LFIC(v) = \sum_{l=1}^{L} \frac{-p_v(l) ln(p_v(l))}{l^2}$$

The computational complexity of the algorithm is $O(n\langle k \rangle)$, where n is the number of nodes in the network, while k indicates the average degree.

Local Clustering H-Index centrality (LCH), which were proposed in [9] takes three different characteristics into account. First, to be feasible on large networks, only the nodes that are a maximum of two hops away from the investigated node are taken into account. Secondly, if the investigated node has a high clustering coefficient value [21], it is expected to have a limited propagation ability. Lastly, the H-index is used to improve the value of nodes that are connected to other nodes that themself have a high influence. The H-index (H) was introduced in [22] and is calculated as follows. Let v be a node of the network. The H of v is calculated as

$$H_v = \mathscr{H}(k_1, k_2, \ldots, k_i \ldots, k_n)$$

where k_i indicates the degree of the i-th neighbor of node v. The $\mathscr{H}$ operator returns the maximum integer d such that there are at least d neighbors whose degree is higher or equal than d. The LCH centrality value of node v is then calculated as:

$$LCH(v) = \frac{1}{\langle H \rangle} \times \frac{H_v}{1+C_v} + \sum_{j \in \Gamma_v} \left(\frac{1}{\langle H \rangle} \times \frac{H_j}{1+C_j} + \frac{1}{\langle k \rangle} \times DC_j \right)$$

where C_v and H_v are the clustering coefficient and the H-index of v node respectively. The set that contains the neighbor nodes of v is denoted as Γ_v, while DC_j represents the degree of node j. Lastly, $\langle H \rangle$ and $\langle k \rangle$ indicate the average H value and the average degree in the investigated network. The computational complexity of the algorithm is $O(n\langle k \rangle)$, where n is the number of nodes in the network, while k indicates the average degree.

Global Structure Model centrality (GSM), that were proposed in [10] incorporates the global influence of all of the nodes in the network during the calculation of the centrality values. For calculating both the self and the global influence, finding the subgraph induced by nodes with core number k is necessary. For this purpose, the Improved K-shell Hybrid (IKH) algorithm [23] is used. The self-influence $SI(v)$ of node v is calculated in a way that parameters minimize the overestimation of it.

$$SI(v) = e^{\frac{Ks(v)}{N}}$$

where e is the natural logarithm, $Ks(v)$ is the k-shell of node v and N represents the number of the nodes in the network. The influence of other nodes connected to v also increases its influence, especially if they themself have a high value of k-shell. However, it is important that the contact distance between v and neighbor w cannot be ignored, and it is inversely proportional to the influence. Based on this, the global influence is measured as follows.

$$GI(v) = \sum_{v \neq w} \frac{Ks(v_w)}{d_{vw}}$$

where d_{vw} is the length of the shortest path between nodes v and w. Finally, the influence of a given node v can be expressed as the product of the self and global influence:

$$GSM(v) = e^{\frac{Ks(v)}{N}} \times \sum_{v \neq w} \frac{Ks(v_w)}{d_{vw}}$$

The computational complexity of calculating the self-influence is $O(n)$, while the calculation of the global-influence (due to Dijkstra to find the shortest distance) is $O(n^2)$.

B. Information propagation models

Finding the nodes that have the most control over the network can be interpreted as the *influence maximization problem*. The aim of this problem is to identify a set of nodes that can influence the flow of information in the network the most.

Suspectible-Infected-Removed (SIR) model was introduced by Kermack in [15]. The nodes of the networks can be in either of the three stages, susceptible, infected or removed. In each and every iteration, the infected nodes can try to infect their susceptible neighbors with a fixed probability β. There is also a γ probability that an infected node becomes removed from the network at the end of each iteration.

Linear treshold model was introduced by Granovetter in [17]. It assumes that the number of neighbors already engaging in a behavior influences an individual's decision of taking part in that behavior. A node's individual decision depends on the percentage of its neighbors that have made the same choice, thus imposing a threshold. Each node has its own threshold that can be different from others. The model works as follows. During a generic iteration, every node is observed: if the percentage of its infected neighbors is greater than its threshold it becomes infected as well, otherwise, nothing happens.

Independent cascade model was introduced in [16]. In this model, each node can be in either of two states, active or passive. The diffusion starts with an initial set of active nodes. In each iteration, the diffusive process unfolds in discrete steps according to the following randomized rule. When node v becomes active in iteration t, it has a single chance to activate each of its currently inactive neighbors of $w_1, w_2, \ldots, w_n$. The succession depends on a probability of $p(v, w)$. If node w has multiple newly activated neighbors, their attempts are sequenced in an arbitrary order. If v succeeds, then w will become active in the next iteration, $t+1$. Whether v succeeds or not, it cannot make any further attempts to activate w in the remaining iterations. The diffusion ends when no more activations are possible to be made.

III. Experiments and results

A. Data and experimental analysis

Three real-life networks were employed to investigate the effectiveness of the three modern centrality measures, namely Advogato social network, Hamsterster social network, and Pages network. Advogato is a social community platform where users can explicitly express weighted trust relationships among themselves. Hamsterster is the friendships and family links between users of the website. Pages network represents mutually liked facebook pages. Nodes represent the pages and edges are mutual likes among them. The networks were accessed through [24]. The networks were chosen to be different in size, node-edge ratio, and clustering coefficient. Detailed information on the networks is presented in Table I.

TABLE I
Networks used in the experiments.

Network	$\|E\|$	$\|V\|$	d_{avg}	C	K_{max}
Advogato social	5,2K	47,3K	18	0,2868	32
Hamsterster social	2K	17K	13	0,5375	12
Pages	4K	17K	8	0.3737	57

where $|V|$ and $|E|$ are the number of nodes and edges in the network, d_{avg}, denotes the average degree of a node, C indicates the clustering coefficient, and K_{max} represent the maximum k core number of the networks. The reason these measures were chosen is that they can greatly influence information propagation. In networks where the average degree is higher, it's more likely that information would be passed on to the next neighboring node. The same can be said about having a high clustering coefficient. The maximum k core number was chosen because of its usage by GSM.

The effectiveness of three modern centrality algorithms, namely LFIC, LCH, and GSM are compared with each other

and two traditional centrality measures, degree centrality, and betweenness centrality. We conduct the following three experiments to evaluate the efficiency of the measures.

B. Experiment 1: Investigation of the frequency of the centrality values

Since nodes with the same value cannot be distinguished apart from each other, this kind of behavior can be a disadvantage when we would expect these nodes to give us some kind of answer. Because of this, the frequency of the different centrality values (i.e. the number of nodes that got the same value) can be used to evaluate the performance of a centrality measure. The frequency values achieved by the various centrality measures on the used networks are shown in Figs. 1-3.

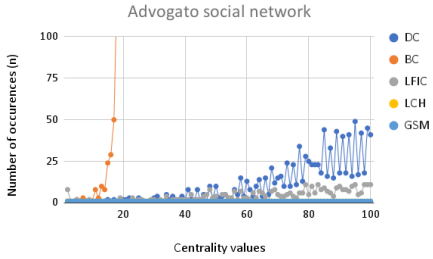


Fig. 1. The frequencies of centrality values on Advogato social network.

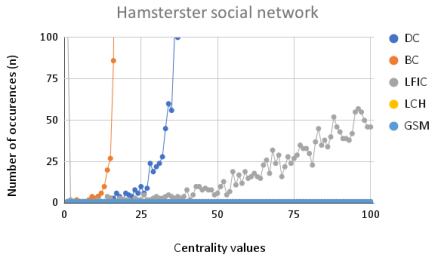


Fig. 2. The frequencies of centrality values on Hamsterster social network.

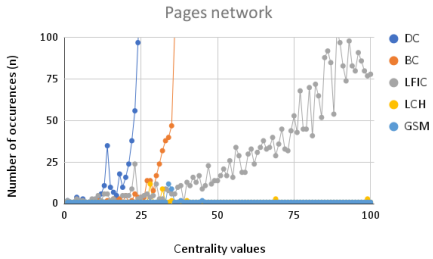


Fig. 3. The frequencies of centrality values on Pages network.

It can be seen that BC and DC are likely to give the same centrality value to nodes. LFIC is capable of achieving better results due to the fact it employs fuzzy numbers and probabilities however, the box of the different nodes is likely to be similar to its neighbors. LCH and GSM have the best performance, due to the fact of employing approaches that result in different values like H-Index and clustering coefficient or the combination of self and global influence.

C. Experiment 2: Evaluating the information propagation ability of the vital nodes

During the identification of vital nodes, the influential capability of the nodes is an important factor. Important nodes are usually capable of influencing a large number of other nodes. In this experiment, we set the five top nodes ranked by the different centrality algorithms as the source of the information propagation. The spreading ability of these nodes can be used to evaluate the performance of the methods. We investigate the spreading ability in three information diffusion models, namely SIR, Independent Cascade, and Linear threshold. The results are shown in Figs. 4-12.

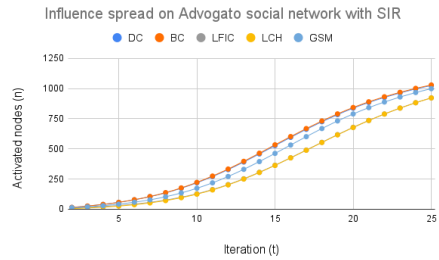


Fig. 4. The influence spread with the SIR model on Advogato social network.

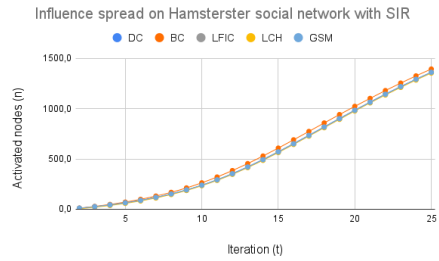


Fig. 5. The influence spread with the SIR model on Hamsterster social network.

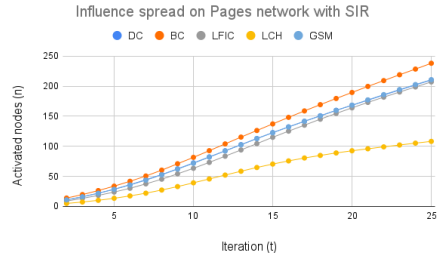


Fig. 6. The influence spread with the SIR model on Pages network.

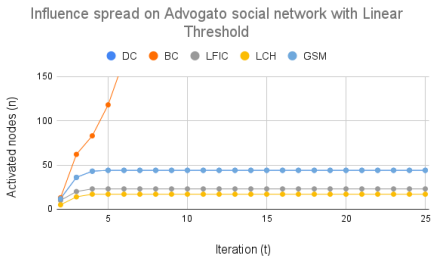


Fig. 7. The influence spread with the Linear threshold model on Advogato social network.

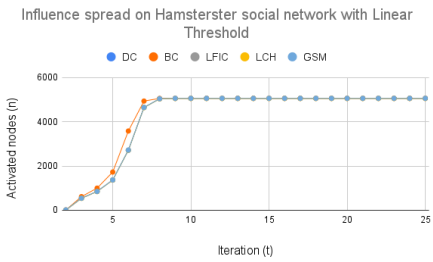


Fig. 8. The influence spread with the Linear threshold model on Hamsterster social network.

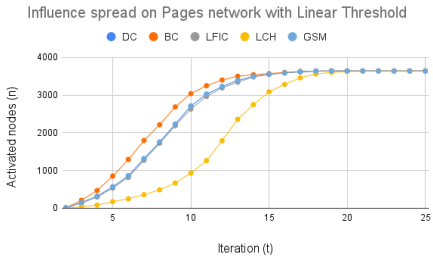


Fig. 9. The influence spread with the Linear threshold model on Pages network.

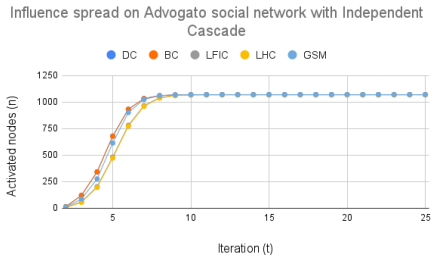


Fig. 10. The influence spread with the Independent Cascade model on Advogato social network.

Based on our experiments it can be said that the nodes considered to be important by the traditional and modern centrality measures have about the same infection-spreading capability. LHC falls back in the case of the Pages network which can be explained by the high triangle count. With numerous triangles, it is likely that the neighbors of a node

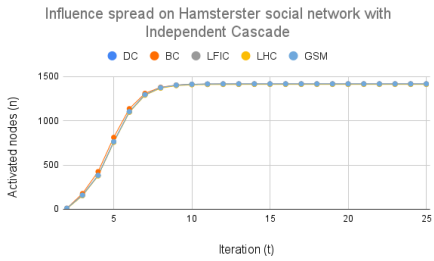


Fig. 11. The influence spread with the Independent Cascade model on Hamsterster social network.

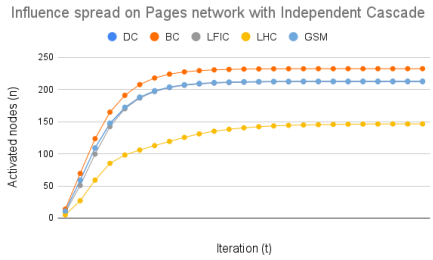


Fig. 12. The influence spread with the Independent Cascade model on Pages network.

have the same H value which can result in similar centrality scores. In the case of Advogato network with the Linear threshold model, all of the investigated centrality measures expect BC have a poor performance. This can be explained by the fact that there are many "bridges" in the network through which the infection does not flow. BC, on the other hand, values these peaks as the most important and initiates the infection from here.

D. Experiment 3: Comparing the speed of the different algorithms

The time it takes for an algorithm to assign the centrality values to the nodes can also be an indicator of performance. The more complex the algorithm is more likely it will result in increased runtime. Figs. 13-15 indicate the time needed by the different algorithms to calculate the centrality measures.

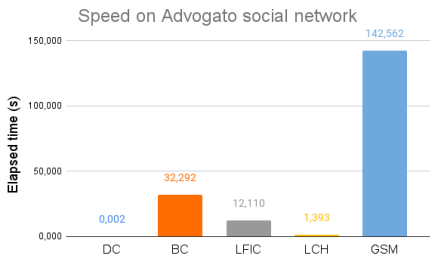


Fig. 13. The elapsed time during the calculation of centrality values on Advogato social network.

The results show that out of all the investigated methods, DC is the fastest. This is no surprise since it only needs to sum

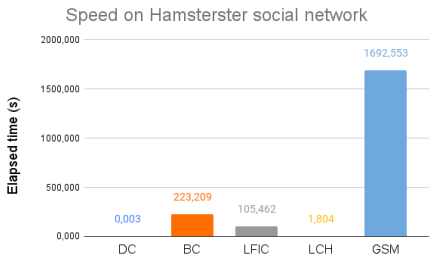


Fig. 14. The elapsed time during the calculation of centrality values on Hamsterster social network.

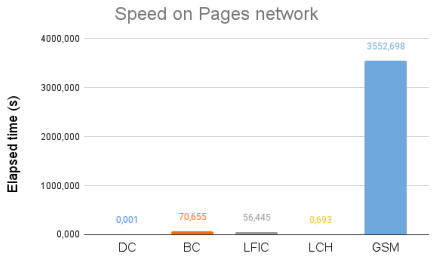


Fig. 15. The elapsed time during the calculation of centrality values on Pages network.

the incident edges on the nodes. LCH achieved the second-best results. Like in the case of DC, this can be also explained by the low complexity of the calculation of the H-Index and the clustering coefficient. LFIC needs an order of magnitude more time to calculate centrality measures. This is explained by the usage of the shortest path between nodes. The same can be said about BC. Lastly, because of the usage of k-shelling, GSM needs the most time to calculate the centrality values of the nodes.

IV. Conclusion

In this paper, we evaluated the performance of three modern centrality measures, namely Local Fuzzy Information centrality (LFIC), Local Clustering H-Index centrality (LCH), and Global Structure Model centrality (GSM). All of these measures take a different approach to centrality measure calculation, like using the uncertainty in a box of nodes around the inspected node, employing the H-Index and clustering coefficient, or taking both the self and global influence of a node into an aspect. For our experiments, we employed three real-life networks with different characteristics. To analyze the performance of modern centrality algorithms, in contrast to the traditional ones, degree centrality (DC) and betweenness centrality (BC) have also been used in the experiments. The experimental results showed that modern algorithms are more capable of assigning a value to a node in a more distinguishable way. The influential capability of the best-rated nodes were about the same as in the case of traditional algorithms in all three of the used information diffusion models. The modern algorithms are capable of calculating the values relatively quickly, the only exception is GSM, which needs magnitudes more time due to the k-shelling that it employs.

In the future, it would be interesting to compare these modern algorithms with other recently proposed methods. It would also be advantageous to investigate if there is any relation between the performance of the algorithms and the characteristics of the networks. Employing dynamic networks can also be profitable.

References

[1] S. Bi, Y. Zeng, and R. Zhang, "Wireless powered communication networks: An overview," *IEEE Wireless Communications*, vol. 23, no. 2, pp. 10–18, 2016, DOI: 10.1109/mwc.2016.7462480.

[2] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. Wang et al., "Towards 6g wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, pp. 1–74, 2021, DOI: 10.1007/s11432-020-2955-6.

[3] W. Liu, X. Li, T. Liu, and B. Liu, "Approximating betweenness centrality to identify key nodes in a weighted urban complex transportation network," *Journal of Advanced Transportation*, vol. 2019, 2019, DOI: 10.1155/2019/9024745.

[4] E. Kauffmann, J. Peral, D. Gil, A. Ferrández, R. Sellers, and H. Mora, "A framework for big data analytics in commercial social networks: A case study on sentiment analysis and fake review detection for marketing decision-making," *Industrial Marketing Management*, vol. 90, pp. 523–537, 2020, DOI: 10.1016/j.indmarman.2019.08.003.

[5] M. Alshahrani, Z. Fuxi, A. Sameh, S. Mekouar, and S. Huang, "Efficient algorithms based on centrality measures for identification of top-k influential users in social networks," *Information Sciences*, vol. 527, pp. 88–107, 2020, DOI: 10.1016/j.ins.2020.03.060.

[6] M. Doostmohammadian, H. R. Rabiee, and U. A. Khan, "Centralitybased epidemic control in complex social networks," *Social Network Analysis and Mining*, vol. 10, pp. 1–11, 2020, DOI: 10.1007/s13278-020-00638-7.

[7] A. Das and A. Biswas, "Rumor source identification on social networks: a combined network centrality approach," in *Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2020*. Springer, 2021, pp. 269–280, DOI: 10.1007/978-981-33-4299-6_22.

[8] H. Zhang, S. Zhong, Y. Deng, and K. H. Cheong, "Lfic: Identifying influential nodes in complex networks by local fuzzy information centrality," *IEEE Transactions on Fuzzy Systems*, 2021, DOI: 10.1109/TFUZZ.2021.3112226.

[9] G.-Q. Xu, L. Meng, D.-Q. Tu, and P.-L. Yang, "Lch: A local clustering h-index centrality measure for identifying and ranking influential nodes in complex networks," *Chinese Physics B*, vol. 30, no. 8, p. 088901, 2021, DOI: 10.1088/1674-1056/abea86.

[10] A. Ullah, B. Wang, J. Sheng, J. Long, N. Khan, and Z. Sun, "Identification of nodes influence based on global structure model in complex networks," *Scientific Reports*, vol. 11, no. 1, pp. 1–11, 2021, DOI: 10.1038/s41598-021-84684-x.

[11] A. Saxena and S. Iyengar, "Centrality measures in complex networks: A survey," *arXiv preprint arXiv:2011.07190*, 2020, DOI: 10.1007/978-3-642-61317-3_5.

[12] O. Ugurlu, "Comparative analysis of centrality measures for identifying critical nodes in complex networks," *Journal of Computational Science*, vol. 62, p. 101738, 2022, DOI: 10.1016/j.jocs.2022.101738.

[13] R. R. Singh, "Centrality measures: a tool to identify key actors in social networks," Principles of Social Networking: *The New Horizon and Emerging Challenges*, pp. 1–27, 2022, DOI: 10.1007/978-981-16-3398-0_1.

[14] L. F. Bringmann, T. Elmer, S. Epskamp, R. W. Krause, D. Schoch, M. Wichers, J. T. Wigman, and E. Snippe, "What do centrality measures measure in psychological networks?" *Journal of abnormal psychology*, vol. 128, no. 8, p. 892, 2019, DOI: 10.1037/abn0000446.

[15] W. O. Kermack and A. G. McKendrick, "A contribution to the mathematical theory of epidemics," *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, vol. 115, no. 772, pp. 700–721, 1927, DOI: 10.1098/rspa.1927.0118.

[16] S. Bikhchandani, D. Hirshleifer, and I. Welch, "A theory of fads, fashion, custom, and cultural change as informational cascades," *Journal of political Economy*, vol. 100, no. 5, pp. 992–1026, 1992, DOI: 10.1086/261849.

[17] M. Granovetter and R. Soong, "Threshold models of diffusion and collective behavior," *Journal of Mathematical sociology*, vol. 9, no. 3, pp. 165–179, 1983, DOI: 10.1080/0022250x.1983.9989941.

[18] T. Rana, P. Meel et al., "Rumor propagation: A state-of-theart survey of current challenges and opportunities," in *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*. IEEE, 2019, pp. 64–69, DOI: 10.1109/ICCT46177.2019.8969023.

[19] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978, DOI: 10.1016/s0378-8733(00)00031-9.

[20] ——, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977, DOI: 10.2307/3033543.

[21] P. W. Holland and S. Leinhardt, "Transitivity in structural models of small groups," *Comparative group studies*, vol. 2, no. 2, pp. 107–124, 1971, DOI: 10.1177/104649647100200201.

[22] Q. Liu, Y.-X. Zhu, Y. Jia, L. Deng, B. Zhou, J.-X. Zhu, and P. Zou, "Leveraging local h-index to identify and rank influential spreaders in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 512, pp. 379–391, 2018, DOI: 10.1016/j.physa.2018.08.053.

[23] G. Maji, A. Namtirtha, A. Dutta, and M. C. Malta, "Influential spreaders identification in complex networks with improved k-shell hybrid method," *Expert Systems with Applications*, vol. 144, p. 113092, 2020, DOI: 10.1016/j.eswa.2019.113092.

[24] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015, [Online]. Available: https://networkrepository.com. DOI: 10.1609/aaai.v29i1.9277.

Marjai Péter received his MSc degree in computer science at Eötvös Loránd University Faculty of Informatics in Budapest, 2021 and currently doing his PhD studies with specialization in information systems. His scientific research is focusing on centrality measures, log processing, parsing and compression.

Máté Nagy-Sándor received his MSc degree in computer science at Eötvös Loránd University Faculty of Informatics in Budapest, 2023.

Attila Kiss defended his PhD in the field of database theory in 1991. His research is focused on information systems, data mining, and artificial intelligence. He has more than 190 scientific publications. Seven students received their PhD degrees under his supervision. Since 2010, he has been the head of Department of Information Systems at Eötvös Loránd University, Hungary. He is also teaching at J. Selye University, Slovakia

Special Issue
of the Infocommunication Journal

Improve Performance of Fine-tuning Language Models with Prompting

Zijian Győző Yang, *Member, HTE* and Noémi Ligeti-Nagy

***Abstract*—This paper explores the effectiveness of prompt programming in the fine-tuning process of a Hungarian language model. The study builds on the prior success of prompt engineering in natural language processing tasks and employs the prompting method to enhance the fine-tuning performance of a huBERT model on several benchmark datasets of HuLU. The experimentation involves testing 45 prompt combinations for the HuCoPA dataset and 15 prompt variations for the HuRTE and HuWNLI datasets. The findings reveal that the addition of an instructional text consistently produces the best results across all winning cases, and that the [CLS] token produces the best results in the separator token experiments. The most significant enhancement was observed in the HuWNLI dataset, with an increase in accuracy from 65% to 85%. These results demon- strate that the addition of instruct text is crucial and sufficient in enabling the language model to effectively interpret and solve the Winograd Schemata problem. These results showcase the potential of prompt programming in enhancing the performance of language models in fine-tuning tasks, and highlight the importance of incorporating task-specific instructions to improve model interpretability and accuracy.**

***Index Terms*—BERT, prompting, fine-tuning**

I. Introduction

PROMPTING is a technique used to guide language models in generating specific types of language. With prompting, a user provides a starting point or "prompt" for the language model, and the model generates text that continues from that point. Prompting can be used to control the topic, style, tone, and other aspects of the generated text.

There are several types of prompting techniques that have been studied in the context of language models. One common technique is prefix-based prompting, where the user provides a few words or a sentence as the starting point for the generated text. Another technique is conditional prompting, where the user specifies a condition or constraint that the generated text must satisfy, such as a certain topic or sentiment. Additionally, prompting can be done through natural language prompts, multiple-choice prompts, or other forms of input.

Prompting can be a powerful tool for controlling the output of language models and making them more useful for specific applications. However, there are also challenges associated with prompting. One challenge is designing effective prompts that achieve the desired result. Another challenge is understanding how the language model is interpreting the prompt and generating the resulting text.

Zijian Győző Yang and Noémi Ligeti-Nagy: Language Technology Research Group, Institute for Language Technology and Applied Linguistics, Hungarian Research Centre for Linguistics, Budapest, Hungary.
E-mail: {yang.zijian.gyozo, ligeti-nagy.noemi}@nytud.hu,
Manuscript received April 19, 2005; revised August 26, 2015.

DOI: 10.36244/ICJ.2023.5.10

There is a growing body of research on prompting in the context of language models [1]. Some studies have focused on improving the effectiveness of prompting [2], [3], while others have explored the ethical implications of using prompts to control the output of language models [4]. Overall, prompting is an important area of research in the field of natural language processing, and it has the potential to shape the future of human-computer interaction and content generation.

Drawing inspiration from the success of prompt engineering, the present study adapted this approach to enhance the fine-tuning performance of a Hungarian language model and associated benchmarks, employing the prompting method. By doing so, our experimentation aimed to build on the prior success of this method in natural language processing tasks.

The currently most performant language model for the Hungarian language is huBERT [5]. Despite being a smaller model, huBERT outperforms HILBERT [6] in available tests, likely due to being trained on more data. Yang et al. [7] have recently introduced three large models trained on large amount of Hungarian data (PULI GPT-3SX, PULI GPT-2, PULI BERT-Large) and evaluated all the above mentioned models on the datasets of HuLU, the Hungarian Language Understanding Benchmark Kit [8], [9].

HuLU was created on the basis of the GLUE [10] and SuperGLUE [11] benchmark databases. The main purpose of HuLU is to enable a standardized evaluation of neural language models in a simple way while also enabling multi-perspective analysis. It also compares the performance of various language models on various tasks. The HuLU comprises 7 corpora containing annotation for various standard language comprehension tasks. As usual, these corpora are divided into training, validation and test sets. The subcorpora of HuLU are either translated datasets (Hungarian Choice of Plausible Alternatives Corpus – translated from CoPA [12] –, Hungarian Recognizing Textual Entailment dataset – translated from the RTE1, RTE2, RTE3 and RTE5 datasets [13], [14], [15], [16] –, Hungarian version of the Stanford Sentiment Treebank – sentences translated from the SST5 dataset [17] –, Anaphora resolution datasets for Hungarian as an inference task [18] – the examples translated from the Winograd schemata and the WNLI dataset [19], [10]) or datasets created from scratch the design of which follows some English datasets (Hungarian CommitmentBank Corpus – designed based on CommitmentBank [20] –, Hungarian Corpus of Linguistic Acceptability – designed based on COLA [21] –, Hungarian Corpus for Reading Comprehension with Commonsense Reasoning [22] – designed based on ReCoRD [23]).

The primary objective of our research is to harness the

power of prompt programming to refine the performance of a Hungarian language model, a step forward that has not been fully explored yet. We chose this area given the promise that prompt engineering holds in enhancing the interpretability and accuracy of language models, especially when dealing with complex language problems. We consider this research vital as it stands to bridge a significant gap in language modeling, potentially setting a new standard for fine-tuning processes in language model development across multiple languages, thus paving the way for better language processing applications.

The structure of our paper is as follows: Section I provides a brief background, discusses previous solutions, and outlines the structure of this paper. In Section II, we present the data and methods used. In Section III, a detailed description of our prompt experiments is provided. Section IV contains the results and evaluations, and Section V presents a concise conclusion.

II. Data and Methods

A. Datasets

In our experiment, we conducted fine-tuning of a huBERT model on several benchmark datasets of HuLU. Our research encompassed experimentation on the HuCoPA, HuRTE, and HuWNLI datasets. The HuCoPA dataset [24] comprises 1,000 instances, each consisting of a premise and two alternatives. The task involves selecting the alternative that describes a situation that stands in a causal relation to the situation described by the premise (example (1)). The train, validation and test sets contain 400, 100 and 500 instances, respectively, following the splits of the original English dataset (as in the GLUE benchmark).

HuRTE [25] is the Hungarian version of the Recognizing Textual Entailment dataset of GLUE, comprising 4,504 instances. Each instance contains a sometimes multi-sentence premise and a one-sentence hypothesis, and the task is to determine whether the former entails the latter or not. The task is a binary classification problem (see example (2)). The train, validation and test sets contain 2 131, 242 and 2 131 instances, respectively.

The HuWNLI dataset [26] comprises the collection of the Hungarian Winograd Schemata [27], extended with the set of sentence pairs of the test set of the WNLI dataset of GLUE, and transformed into a natural language inference task. The NLI format was created by replacing the ambiguous pronoun with each possible referent (see example (3)). The data is distributed among three splits: training set (562), validation set (59) and test set (134).

(1) premise: *A testem árnyékot vetett a fűre.* 'My body cast a shadow over the grass.'
choice 1: *Felkelt a nap.* 'The sun was rising.'
choice 2: *A füvet lenyírták.* 'The grass was cut.'
question: *cause*
label: 1 (the number of the more plausible choice)

(2) premise: *Még nem találtak tömegpusztító fegyvereket Irakban.* 'No weapons of mass destruction have yet been found in Iraq.'
hypothesis: *Tömegpusztító fegyvereket találtak Irakban.* 'Weapons of mass destruction have been found in Iraq.'
label: 0 (1, if the premise entails the hypothesis, 0 otherwise.)

(3) sentence 1: *A férfi nem tudta felemelni a fiát, mert olyan nehéz volt.* 'The man couldn't lift his son because he was so heavy.',
sentence 2: *A fia nehéz volt.* 'His son was heavy.',
label: 1 (1, if sentence 1 entails sentence 2, 0 otherwise.)

B. Fine-tuning process

In our fine-tuning process, we employed identical hyperparameter settings across all cases, and fine-tuned all models for a period of 20 epochs. Our comparison was based on selecting the highest result scores. For the experiments, we used 2 NVIDIA A100 GPUs. The modified hyperparameters are as follows:

- HuCoPA: sequence length: 128; batch size: 8 per GPU; learning rate: 2e-5;
- HuRTE: sequence length: 512; batch size: 32 per GPU; learning rate: 2e-5;
- HuWNLI: sequence length: 256; batch size: 6 per GPU; learning rate: 8e-6.

For fine-tuning our language model, we used the scripts provided by Hugging Face [28]. In the case of HuCoPA, we treated the task as a multiple choice task, while for HuRTE and HuWNLI, we employed the text classification script to train our models. Initially, we used a learning rate value of 2e-5 for all cases. However, further experimentation with HuWNLI revealed that 8e-6 yielded the best results and thus became the preferred choice.

In our experiments, we fine-tuned our models on the training set. Subsequently, we conducted experiments on the validation set and selected the checkpoint that yielded the highest results. Finally, we evaluated this selected checkpoint on the test sets.

III. Prompt experiments

In our current research, we have explored various possible prompt templates, ranging from not using prompts at all, to adding only a separator token between the sentences, and even to utilizing complex prompt templates with multiple sentence-long instructions. In the case of using prompts, we explored several versions of the separator token or text. When using text as a separator, there were instances where it was necessary to modify the syntax of the input sentence, such as converting the original sentence to lowercase. We even experimented with multiple instruction texts, which contain a detailed description of the current task to be solved. All the instructions were in Hungarian. Some examples of the different types of prompts used with the HuCoPA dataset is provided below for your reference:

- Original text from HuCoPA:
  - premise: *A sofőr felkapcsolta az autó fényszóróit.* 'The driver turned on the car's headlights.'
  - choice 1: *Mennydörgést hallott.* 'He/she heard a thunderclap.'
  - question: cause
- Input using a separator token: *A sofőr felkapcsolta az autó fényszóróit.* **[SEP]** *Mennydörgést hallott.* – 'The driver turned on the car's headlights. **[SEP]** He/she heard a thunderclap.'
- Input using text as separator: *A sofőr felkapcsolta az autó fényszóróit.* ***Mert*** *mennydörgést hallott.* – 'The driver turned on the car's headlights. **Because** he/she heard a thunderclap.'
- Input using an instruct text: ***Döntsd el, hogy következik-e az első mondat a második mondatból. Első mondat:*** *A sofőr felkapcsolta az autó fényszóróit.* ***Második mondat:*** *Mennydörgést hallott.* – '**Decide whether the first sentence is entailed by the second one. First sentence:** The driver turned on the car's headlights. **Second sentence:** He/she heard a thunderclap.'

In the samples above, the text/token that is added to the original input text is bolded. Using this additional information or these instructions, we can assist the language models in their fine-tuning training. In some cases, when using text as a separator (as you can see above), we had to modify the syntax of the input sentence.

We conducted experiments in all cases using the following category of prompts:

- $[empty]$: The examined texts were concatenated without using any separator token.
- *Separator token*: Either/both the $[CLS]$ or $[SEP]$ token was inserted as a separator between the two examined texts.
- *Conjunction phrase*: Hungarian conjunction word/phrase was employed as a separator token text (details can be found in Table I).
- *Question sentence*: A question sentence was used as a separator token text (see Table I).
- *Instruct text*: Instruct text was added to the beginning of the input text (see Table II). The *question sentence* can be an instruct text as well.
- *Mix*: Different prompt types were mixed, by combining the use of a separator token with the instruct text, as an example.

In Table I and Table II we provide a comprehensive list of the various prompt texts we experimented with. Specifically, for the HuCoPA dataset, we tested 45 different prompt combinations, whereas for the HuRTE and HuWNLI datasets, we tested 15 prompt variations.

TABLE I
CONJUNCTION PHRASE AND QUESTION SENTENCE AS PROMPTS IN THE CASE OF THE DIFFERENT DATASETS

	Conjunction phrase	Question sentence
HuCoPA	*mert/ezért* *Mert/Ezért* 'because/because of this'	*Oka?/Hatása?* 'Cause of this?/Result of this?' *Mi az oka?/Mi a hatása?* 'What is the cause of this?/ What is the result of this?' *Ez a következtetés helyes?* 'Is this conclusion correct?'
HuRTE	*Tehát* 'Therefore' *Ezért* 'Because of this' *Ebből következik, hogy* 'This implies that'	*Ez a következtetés helyes?* 'Is this conclusion correct?'
HuWNLI	*Tehát* 'Therefore' *Ezért* 'Because of this' *Ebből következik, hogy* 'This implies that'	*Ez a következtetés helyes?* 'Is this conclusion correct?'

IV. RESULTS

In the results section, we have chosen to present only the best scores obtained from each prompt category to enhance the readability of our findings. This decision was made in consideration of the 75 experiment subscores that were obtained, which could otherwise result in excessive data presentation that may obscure the key insights. Figure 1 displays the highest results attained for each category on the test sets. Our experimentation showed that in all corpora, the highest results were achieved through the combination of prompts. Notably, we achieved state-of-the-art results in all three examined benchmarks. To further validate the effectiveness of our prompting method, we submitted our results to the HuLU benchmark competition [29], where our approach outperformed the dedicated three benchmarks, as shown in Table III. The three mixed prompt winners are listed below, the original input texts are marked ({ ... }) as variables:

- HuCoPA: *Ez a következtetés helyes?* 'Is this conclusion correct?' { ... (premise text) } *Mert/Ezért* 'Because/Because of this' { ... (choice sentence text) }
  (To make the sentence grammatically correct, the choice sentence is lowercased.)
- HuRTE: *A következő példákban egy premissza és egy hipotézis található. A premissza több mondatból is állhat. A feladat az, hogy el kell dönteni, a hipotézis következik-e a premisszából: azaz ha a premissza igaz, akkor a hipotézis is igaz.* 'The following examples consist of a premise and a hypothesis. The premise may consist of multiple sentences. The task is to determine whether the hypothesis is entailed by the premise: that is, if the premise is true, then the hypothesis is also true.' [CLS] *premissza:* 'premise' { ... (premise text) } [CLS] *hipotézis:* 'hypothesis' { ... (hypothesis text) }
- HuWNLI: *Az alábbi példákban két mondat látható. El kell dönteni, hogy a második mondat következik-e az első mondatból.* 'The following examples consist of two sentences. The task is to determine whether the second sentence is entailed by the first.' [CLS] *első mondat:* 'first sentence:' { ... (sentence1 text) } [CLS] *második mondat:* 'second sentence:' { ... (sentence2 text) }

As evident from the winning prompts, the addition of an instructional text was consistently observed across all winning cases. This observation aligns with our expectations, as instructional texts typically provide a detailed description of the task at hand, thereby aiding the language models in their training.

TABLE II
INSTRUCT TEXTS USED IN THE EXPERIMENT

	Instruct text
HuCoPA	1.) *Az alábbi példákban van egy mondat, egy kérdés (Ok vagy Hatás), és két lehetséges alternatíva. A feladat az, hogy a két lehetséges alternatíva közül ki kell választani azt, amelyik valószínűbb válasz a kérdésre.* *Mondat: ... Kérdés: ... 1. alternatíva: ... 2. alternatíva: ...* 'The following examples consist of a sentence, a question (Cause or Effect), and two possible alternatives. The task is to select the alternative that is more likely to be the answer to the question. Sentence: ... Question ... 1st alternative: ... 2nd alternative: ...' 2.) *Az alábbi példákban van egy mondat, és két lehetséges folytatás. A feladat az, hogy a két lehetséges folytatás közül ki kell választani azt, amelyik valószínűbb folytatása a mondatnak.* *Mondat: ... 1. folytatás: ... 2. folytatás: ...* 'In the following examples, there is a sentence and two possible continuations. The task is to select the alternative that is more likely to be the continuation of the sentence. Sentence: ... 1st continuation: ... 2nd continuation: ...'
HuRTE	*A következő példákban egy premissza és egy hipotézis található. A premissza több mondatból is állhat. A feladat az, hogy el kell dönteni, a hipotézis következik-e a premisszából: azaz ha a premissza igaz, akkor a hipotézis is igaz.* *Premissza: ... Hipotézis: ...* 'The following examples consist of a premise and a hypothesis. The premise may consist of multiple sentences. The task is to determine whether the hypothesis is entailed by the premise: that is, if the premise is true, then the hypothesis is also true. Premise: ... Hypothesis: ....'
HuWNLI	*Az alábbi példákban két mondat látható. El kell dönteni, hogy a második mondat következik-e az elsőből: azaz ha az első mondat igaz, akkor ebből következik, hogy a második mondat is igaz.* *Első mondat: ... Második mondat: ...* 'The following examples consist of two sentences. The task is to determine whether the second sentence is entailed by the first: that is, if the first sentence is true, then it follows that the second sentence is also true. First sentence: ... Second sentence: ...'

In the separator token experiments, our findings indicate that in all cases, the [CLS] token produced the best results.

The greatest enhancement was attained on the HuWNLI dataset: the results increased from the preceding 65% accuracy to 85%, thereby yielding a markedly superior outcome compared to the previous attempts (see Table III for the comparison of the results on the HuLU datasets). The results indicate that the addition of an instruct text (i.e., the description of the given task) was crucial and sufficient in enabling the language model to effectively interpret and solve the Winograd Schemata problem.

TABLE III
HULU COMPETITION

	HuCoPA (MCC / acc)	HuWNLI (acc)	HuRTE (MCC / acc)
huBERT	56.1 / 78.0	64.93	48.7 / 74.1
PULI BERT-Large	41.4 / 76.6	65.67	51.7 / 75.9
huBERT - Prompt	**56.4 / 78.2**	**85.80**	**53.4 / 76.5**

In Table IV, a snippet of our HuCoPA experiment is presented. The rows represent different prompt sets, while the columns represent epoch numbers (only the first 10 epochs are shown). Upon examining the values in the first epoch, it is evident that the model learned the task at varying speeds depending on the prompt set. The 24th prompt set achieved a precision value of 74.44 in the first epoch, whereas the 17th prompt set only reached 56.99. By the tenth epoch, all models obtained acceptable results; however, there still remains a difference of 7.3 between the highest and lowest values (77.77 – 85.00). An interesting observation is that the 23rd prompt set achieved the highest value in epoch 7.

Due to the nature of our experiments being focused on fine-tuning tasks, we were unable to directly compare our results with large language models and their applications, such as ChatGPT [30].

A. Discussion of the results on the HuWNLI dataset

As highlighted earlier, considerable progress is evident on the HuWNLI dataset. An accuracy of 65% had previously been recorded as the highest, achieved by a BERT-Large model fine-tuned, however, our mixed setting experiment (utilizing an instruct text and the `[CLS]` token) yielded a fine-tuned model with an accuracy score of 85%.

Aside from being an allusion to Turing's imitation game [31], the term "Turing Test" is broadly applied to any test devised to gauge a computer's "intelligence". Winograd schemata are frequently dubbed the new "Turing Test". They consist of sentence pairs as closely related in content as possible (with a difference of one word or phrase), having identical target pronouns that refer back to different precursors (example 4).

(4) The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.
Who [feared/advocated] violence?
a. The city councilmen
b. The demonstrators

Levesque and colleagues [19] suggested a set of Winograd schemata as a fresh AI testing method, inspired by the Turing Test. A Winograd schema must fulfill three conditions to be included in the challenge:

1) it should be easily discernible by a human reader
2) it should not be decipherable by selectional restrictions
3) it should not be searchable on Google

The strength of this new challenge lies in its simplicity: the schemata answer is a binary decision. Furthermore, it's

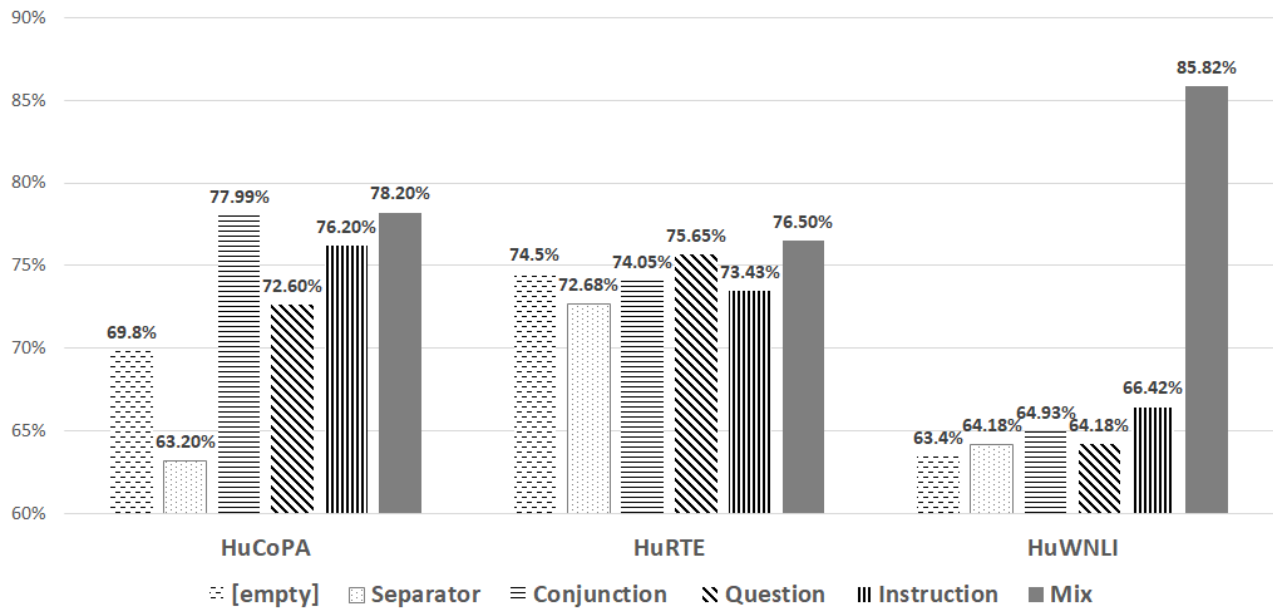


Fig. 1. Performance of the model on three datasets fine-tuned with different prompting strategies

TABLE IV
EXPERIMENTS ON THE HUCOPA VALIDATION SET

id	1 epoch	2 epoch	3 epoch	4 epoch	5 epoch	6 epoch	7 epoch	8 epoch	9 epoch	10 epoch
1	66.55	72.00	79.66	81.77	80.44	80.11	80.33	79.88	80.22	80.22
2	65.55	75.88	81.77	83.77	83.33	82.55	82.99	81.66	81.44	81.66
3	69.99	76.55	81.66	81.99	83.88	82.11	83.66	83.33	82.44	82.77
4	65.44	76.33	82.77	84.55	83.44	84.11	83.66	81.77	81.55	82.11
5	69.11	76.99	82.99	81.77	83.99	82.11	83.55	82.22	82.55	82.55
6	63.88	72.11	78.55	78.66	80.00	78.66	78.66	79.33	79.11	79.22
7	70.44	73.11	80.55	82.88	82.44	80.55	81.33	81.33	81.77	81.33
8	65.22	75.00	79.55	80.44	81.77	79.33	80.88	81.00	80.66	80.77
9	65.77	74.88	80.33	81.99	81.77	81.44	81.00	80.55	80.77	80.77
10	65.88	74.22	81.11	82.88	81.66	81.33	81.88	81.00	81.22	80.66
11	63.33	71.22	77.11	79.77	80.11	80.33	79.55	80.22	80.11	80.11
12	65.33	73.22	76.22	80.33	80.33	78.77	79.00	79.22	79.33	79.22
13	67.44	75.88	80.55	81.33	82.44	82.77	81.77	81.55	81.77	81.99
14	66.66	76.99	78.55	80.66	81.66	81.44	81.55	80.77	80.00	80.33
15	68.66	77.88	81.55	82.99	80.77	82.11	83.33	82.99	81.33	81.77
16	63.44	73.66	79.00	80.66	82.66	81.55	81.99	81.99	80.88	81.00
17	56.99	69.22	70.55	75.55	75.77	77.88	76.99	77.77	77.44	77.77
18	62.33	66.44	72.55	78.11	78.77	79.00	78.11	77.22	77.88	78.00
19	64.77	73.11	76.11	80.55	81.33	79.88	80.33	80.88	80.77	80.44
20	64.11	70.55	79.11	78.55	79.88	81.11	81.00	81.33	79.88	79.77
21	68.11	76.88	80.55	81.77	82.22	81.00	82.66	82.11	81.66	81.99
22	69.11	76.99	84.88	84.88	82.44	82.77	83.44	82.66	83.22	83.33
23	67.77	77.11	83.33	82.99	84.88	84.44	85.11	84.88	84.66	85.00
24	74.44	79.88	84.88	83.33	83.77	84.66	83.77	83.99	84.77	84.78

illuminative: any layperson can deduce that a program that fails to find the right answer lacks sufficient ”intelligence”, i.e., it falls short of human understanding. Lastly, the schemata are demanding: anaphora resolution, while easy for a human, continues to challenge cutting-edge algorithms. This can be attributed to the fact that only world knowledge and reasoning can aid in addressing these issues.

The GLUE and SuperGLUE benchmarks include the WNLI dataset, which features Winograd schemata as sentence pair classification. Here, authors form sentence pairs by substituting the ambiguous pronoun with each possible referent. The task involves predicting whether the sentence, with the pronoun replaced, is implied by the original sentence. In addition, a compact evaluation set composed of new examples taken from fiction books is used alongside the publicly accessible Winograd schemata. This dataset has been shown to be one of GLUE’s most challenging, with ELECTRA first breaking the 90% accuracy barrier in 2019 [32].

Regardless of the impressive accuracy rates that neural models can now achieve on this dataset, commonsense reasoning remains a significant hurdle in AI (for a comprehensive analysis, refer to [33]). Our experiment supports these findings

as it emphasizes the significance of the environment where models are fine-tuned and evaluated. This is one of the major criticisms of the Winograd schemata (and other datasets used for evaluating language model performance) highlighted in [33].

As can be seen in Figure 1, we could only approximate the 90% accuracy with the mix setting, all other prompt structures resulted in an accuracy around 65%. The cause of this may lie in the instruction text itself: we narrow the task with this text, aiding the model to focus on the entailment. The `[CLS]` token also assists in setting the boundaries. These two elements – the instruction text and the separator token – appear to be sufficient for the model to pass this test.

Our findings also concur with the aforementioned results, as we observe a substantial leap with the right configuration (prior to ELECTRA's 91.8%, scores on the WNLI dataset in GLUE were hovering between 65-70%). In order to match up with the English results, further enhancements are required. This could be achieved by experimenting with various neural models or by modifying the fine-tuning process and the prompting environment.

V. Conclusion

In this paper, we investigated the effectiveness of different prompting techniques for fine-tuning huBERT on three datasets of HuLU. We experimented with several types of prompts, including conjunction phrases, question sentences, and instruct texts. Our results demonstrate that prompting can significantly improve the performance of huBERT on these datasets.

Overall, our findings suggest that prompt engineering is a promising area of research for improving the performance of language models on specific tasks. By providing targeted prompts that guide the generation of language, we can achieve better results on tasks such as text classification and natural language inference.

In our experiments, we found that the best results were obtained adding instruction text and separator text as prompts. This suggests that combining different types of prompts can be an effective strategy for improving the performance of fine-tuning language models on specific tasks.

Our study has several limitations that should be addressed in future research. For example, we only investigated a limited set of prompting techniques, and there may be other approaches that are even more effective. Additionally, our study only focused on three datasets of HuLU, and it is unclear how well our findings generalize to other datasets and languages.

In the future, we plan to conduct experiments by fine-tuning huBERT using Parameter-Efficient Fine-tuning techniques (such as Lora [34], Prompt tuning [35], etc.). Additionally, we aim to expand our research to include large language models.

In conclusion, our study highlights the potential of prompting techniques for fine-tuning language models on specific tasks. Further research is needed to explore the effectiveness of different prompting strategies, experiments with fuzzy or voting methods [36] and to investigate the generalizability of our findings to other datasets and languages.

References

[1] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Comput. Surv.*, vol. 55, no. 9, jan 2023. [Online]. Available: DOI: 10.1145/3560815

[2] B. Lester, R. Al-Rfou, and N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3045–3059. [Online]. Available: https://aclanthology.org/2021.emnlp-main.243
DOI: 10.18653/v1/2021.emnlp-main.243

[3] X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353; DOI: 10.18653/v1/2021.acl-long.353

[4] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, Z. Kenton, S. Brown, W. Hawkins, T. Stepleton, C. Biles, A. Birhane, J. Haas, L. Rimell, L. A. Hendricks, W. Isaac, S. Legassick, G. Irving, and I. Gabriel, "Ethical and social risks of harm from Language Models," 2021.

[5] D. M. Nemeskey, "Introducing huBERT," in *XVII. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged, Magyarország: Szegedi Tudományegyetem, Informatikai Intézet, 2021, pp. 3–14.

[6] Á. Feldmann, R. Hajdu, B. Indig, B. Sass, M. Makrai, I. Mittelholcz, D. Halász, Z. Gy. Yang, and T. Váradi, "HILBERT, magyar nyelvű BERT-large modell tanítása felhő környezetben," in *XVII. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged, Magyarország: Szegedi Tudományegyetem, Informatikai Intézet, 2021, pp. 29–36.

[7] Z. Gy. Yang, R. Dodé, G. Ferenczi, E. Héja, K. Jelencsik-Mátyus, Á. Kőrös, L. J. Laki, N. Ligeti-Nagy, N. Vadász, and T. Váradi, "Jönnek a nagyok! BERT-Large, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre," in *XIX. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2023)*. Szeged, Hungary: Szegedi Tudományegyetem, Informatikai Intézet, 2023, pp. 247–262.

[8] N. Ligeti-Nagy, G. Ferenczi, E. Héja, K. Jelencsik-Mátyus, L. J. Laki, N. Vadász, Z. Gy. Yang, and T. Váradi, "HuLU: magyar nyelvű benchmark adatbázis kiépítése a neurális nyelvmodellek kiértékelése céljából," in *XVIII. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged: Szegedi Tudományegyetem, Informatikai Intézet, 2022, pp. 431–446.

[9] N. Ligeti-Nagy, E. Héja, L. J. Laki, D. Takács, Z. Gy. Yang, and T. Váradi, "Hát te mekkorát nőttél! – A HuLU első életéve új adatbázisokkal és webszolgáltatással," in *XIX. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged: JATEPress, 2023, pp. 217–230.

[10] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://aclanthology.org/W18-5446, DOI: 10.18653/v1/W18-5446

[11] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[12] M. Roemmele, C. Bejan, and A. Gordon, "Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning," *AAAI Spring Symposium - Technical Report*, 01 2011.

[13] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL Recognising Textual Entailment Challenge," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 177–190.

[14] R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, "The Second PASCAL Recognising Textual Entailment Challenge," in *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, Venice, Italy, 2006, pp. 1–9.

[15] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, "The Third PASCAL Recognizing Textual Entailment Challenge," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, ser. RTE '07, 2007, pp. 1–9.

[16] L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini, "The Fifth PASCAL Recognizing Textual Entailment Challenge," in *Proceedings of the TAC Workshop*, 2009.

[17] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[18] N. Vadász and N. Ligeti-Nagy, "Winograd schemata and other datasets for anaphora resolution in hungarian," *Acta Linguistica Academica*, vol. 69, no. 4, 2022, in press. DOI: 10.1556/2062.2022.00575

[19] H. J. Levesque, E. Davis, and L. Morgenstern, "The Winograd Schema Challenge," in *Proceedings of the Thitteenth International Conference on Principles of Knowledge Representation and Reasoning*, ser. KR'12. AAAI Press, 2012, pp. 552–561.

[20] M.-C. de Marneffe, M. Simons, and J. Tonhauser, "The commitmentbank: Investigating projection in naturally occurring discourse," *Proceedings of Sinn und Bedeutung*, vol. 23, no. 2, pp. 107–124, Jul. 2019. [Online]. Available: https://ojs.ub.uni-konstanz.de/sub/index.php/sub/article/view/601

[21] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *arXiv preprint arXiv:1805.12471*, 2018. DOI: 10.48550/arXiv.1805.12471

[22] Z. Gy. Yang and N. Ligeti-Nagy, "Building machine reading comprehension model from scratch," *Annales Mathematicae et Informaticae*, pp. 1–17, 2023. [Online]. Available: DOI: 10.33039/ami.2023.03.001

[23] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. V. Durme, "ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension," 2018.

[24] Hungarian Research Centre for Linguistics, "Hungarian Choice of Plausible Alternatives Corpus." [Online]. Available: https://github.com/nytud/HuCoPA

[25] H. R. C. for Linguistics, "Hungarian Recognizing Textual Entailment dataset." [Online]. Available: https://github.com/nytud/HuRTE

[26] H. R. C. for Linguistics, "Anaphora resolution datasets for Hungarian as an inference task." [Online]. Available: https://github.com/nytud/HuWNLI

[27] N. Vadász and N. Ligeti-Nagy, "Winograd schemata and other datasets for anaphora resolution in Hungarian," *Acta Linguistica Academica*, vol. 69, no. 4, pp. 564–580, 2022. [Online]. Available: https://akjournals.com/view/journals/2062/69/4/article-p564.xml DOI: 10.1556/2062.2022.00575

[28] Hugging Face, "Examples." [Online]. Available: https://github.com/huggingface/transformers/tree/main/examples/pytorch

[29] H. R. C. for Linguistics, "Hungarian Language Understanding Benchmark Kit." [Online]. Available: https://hulu.nytud.hu

[30] OpenAI, "Chatgpt." [Online]. Available: https://chat.openai.com

[31] A. Turing, "Computing Machinery and Intelligence," Mind, vol. 59, no. 236, pp. 433–460, 1950.

[32] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1xMH1BtvB

[33] V. Kocijan, E. Davis, T. Lukasiewicz, G. Marcus, and L. Morgenstern, "The Defeat of the Winograd Schema Challenge," 2023. DOI: 10.1016/j.artint.2023.103971

[34] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[35] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, "P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 61–68. [Online]. Available: https://aclanthology.org/2022.acl-short.8, DOI: 10.18653/v1/2022.acl-short.8

[36] T. Tajti, "New voting functions for neural network algorithms," *Annales Mathematicae et Informaticae*, pp. 229–242, 2020. [Online]. Available: DOI: 10.33039/ami.2020.10.003

Zijian Győző Yang is research fellow at Hungarian Research Centre for Linguistics. He obtained his PhD degree in Human Language Technology with summa cum laude in 2019. His research areas include large language models, machine translation and evaluation, text summarization, sentiment analysis and text classification.

Noémi Ligeti-Nagy completed her PhD in Computational Linguistics in 2021 and is currently a Research Fellow at the Hungarian Research Centre for Linguistics. Her academic pursuits revolve around the design and development of language corpora, the benchmarking of language processing systems, and the investigation and evaluation of neural language models.

Challenges in service discovery for microservices deployed in a Kubernetes cluster – a case study

Baasanjargal Erdenebat, Bayarjargal Bud, and Tamás Kozsik

Abstract—**With Kubernetes emerging as one of the most popular infrastructures in the cloud-native era, the utilization of containerization and tools alongside Kubernetes is steadily gaining traction. The main goal of this paper is to evaluate the service discovery mechanisms and DNS management (CoreDNS) of Kubernetes, and to present a general study of an experiment on service discovery challenges. In large scale Kubernetes clusters, running pods, services, requests, and workloads can be substantial. The increased number of HTTP-requests often result in resource utilization concerns, e.g., spikes of errors [24], [25]. This paper investigates potential optimization strategies for enhancing the performance and scalability of CoreDNS in Kubernetes. We propose a solution to address the concerns related to CoreDNS and provide a detailed explanation of how our implementation enhances service discovery functionality. Experimental results in a real-world case show that our solution for the CoreDNS ensures consistency of the workload. Compared with the default CoreDNS configuration, our customized approach achieves better performance in terms of number of errors for requests, average latency of DNS requests, and resource usage rate.**

Index Terms—**Microservice, container, service discovery, CoreDNS, Kubernetes**

I. Introduction

With the continuous development of technology, software systems become larger and more complex, and the architecture of their code must adapt to these changes in order to enable the handling of the increased complexity. The application of a traditional, monolithic architecture is becoming less attractive and less useful in many business scenarios. In a monolithic architecture, any modification made to a small feature necessitates the recompilation and redeployment of the entire application, resulting in long iteration cycles, which is clearly unfavorable. On the contrary, microservices provide an approach to developing a single application as a collection of small services. Each service operates in its own process and communicates through a lightweight mechanism. Transitioning from monolithic applications to microservices requires a significant shift in software design, but it offers numerous advantages, particularly when combined with the introduction of containerization technologies. The microservice-based application code is small, independent, easy to manage, compile and deploy, and allows short development iteration cycles [1], [2]. Due to the small volume of deployment packages, fast service start-up and resource recovery, and easy to achieve flexible scaling, it can well meet application scenarios with high concurrency and large fluctuation in load [3]. The integration of containerization technology and the microservice architecture significantly enhance the performance and efficiency of information systems [4], [5]. We leverage microservice architecture and containerization technology to develop business applications. It allows us to maintain frequent software development and deployment cycles while ensuring consistent and high-quality product delivery. Furthermore, the usage of those technologies facilitates smooth adaptation to the continuous integration and continuous development (CI/CD) methodology and cloud native development [6], allowing for agile adjustments to accommodate evolving business requirements.

In a real world industrial scenario, our team embarked on a Spring Cloud-based DevOps platform project with the objective of gradually transitioning from a monolithic architecture to a microservice-based one. To accomplish this, the entire system needed to be migrated from hypervisor to container virtualization, which allowed us to build a highly available cluster. Over time, we have progressively transitioned our services from a monolithic architecture to a Spring Cloud-based one. In the environment, numerous microservices are constantly being created and perished, while they are making calls to each other. Therefore, a component that specifies the location of a given microservice is needed. This component is called *Service Discovery*. Spring Cloud implements Service Discovery through Eureka [7]. When shifting from VM-based microservice applications to containerized ones, there is a need of using Eureka for service discovery due to the communication between the two different infrastructures: service discovery operates smoothly via Eureka on co-existing infrastructures of VM and Kubernetes. However, it exhibits performance limitations, particularly in the large-scale.

The contributions of this paper are the two-fold. First, we introduce the idea of an easy-to-use dynamic service discovery functionality during the migration from virtual machines to a containerized cluster. Then we provide a detailed evaluation of the service discovery process in container-based clusters. We offer solutions and strategies for service discovery issues and for ensuring smooth operation of Kubernetes clusters.

The rest of the paper is structured as follows. Section II is a brief overview of the background knowledge on service discovery, microservices and containerization technology. Section III introduces the pre-experimental activities including the methodology employed in our study. The detailed investigation on performance overhead is provided in Section IV. In

Baasanjargal Erdenebat and Tamás Kozsik are with Department of Programming Languages and Compilers, ELTE Eötvös Loránd University, Budapest, Hungary (e-mail: baasanjargal@inf.elte.hu, ORCID: 0000-0003-0471-7183; tamas.kozsik@elte.hu, ORCID: 0000-0003-4484-9172.)

Bayarjargal Bud is with National University of Mongolia, (e-mail: bud.bayarjargal@gmail.com)

DOI: 10.36244/ICJ.2023.5.11

Section V, we present our measurement results. Section VI addresses related work. Finally, conclusions are drawn in Section VII.

II. Background

A. Microservices and containerization tools

Quoting from "microservices.io" [8], microservices are an architectural style that organizes an application into a set of services characterized by the following qualities: (1) sufficiently decoupled to ensure high maintainability and testability; (2) loosely coupled from the application glue logic (e.g., orchestration, monitoring, etc.); (3) independently deployable; (4) organized around single business capabilities. Containerization technology plays a crucial role as the key enabler for microservices. From an organizational standpoint, the combination of microservices and containerization empowers the microservice architecture style to facilitate the evolution of an organization's technology and organizational stack in harmony with its architectural structures [9], [10]. In the scope of this work, we focus on Docker as a containerization tool and Kubernetes for the container orchestration. Docker helps to provide a quick and lightweight environment and allows us to build, deploy, run, update and manage container—standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment [11], [5]. On the other hand, Kubernetes is one of the most popular orchestration platforms for automating the deployment of, and managing, containerized workloads and services, as well as for scaling containerized applications across a cluster [12], [13].

B. Service discovery

Service Discovery is a design pattern which can enable clients or API gateways to discover the network information, such as the IP address and port, of microservices [14]. This discovery process relies on a component known as the Service Registry or Discovery Server. The Service Registry is responsible for tracking all individual microservices within the architecture and storing their IP addresses and ports in its database. Whenever a service scales up or down, it sends a message to the discovery service, which updates its database accordingly. The microservices are registered and cancelled through a service registry, with Netflix Eureka serving as an example of such a registry [15], [16].

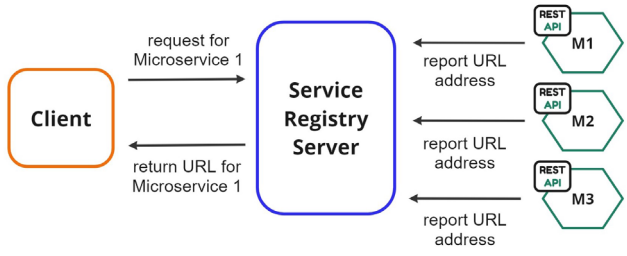


Fig. 1. Client discovery mechanism.

To cater to different business application scenarios, microservices utilize a registration discovery mechanism that combines client discovery and server discovery [17]. The client discovery mechanism can rely on Netflix Eureka technology, where clients query a service registry ("service center") to retrieve a list of available service instances [15], [18] as shown in Figure 1. Using a load balancing algorithm, the client selects one of the available service instances and sends out the request. The registration management and querying of service instances are facilitated through REST API calls provided by Eureka within the application. Eureka's client operates in self-registration mode, requiring it to handle service instance registration, cancellation, and sending regular heartbeats.

III. Migration of microservices from VMs to containers

Two main initiatives were undertaken as part of this study to address service discovery challenges. The first one involved migrating microservices from a virtual machine (VM) based environment to a container-based one. Our microservice system utilizes Spring Cloud development and is deployed on a Docker-based Kubernetes cluster. Over time, we gradually separated services from a monolithic architecture to Spring Cloud. Currently, we are successfully operating approximately 600 instances and 110 application services on DevOps infrastructure and self-hosted Kubernetes cluster. However, to enable service discovery between the VM-based and the container-based infrastructures, a solution was needed. Thus, we designed and implemented a seamless migration process for the microservices, with a primary focus on introducing easy-to-use dynamic service discovery during the transition period.

The second initiative involves leveraging Kubernetes' native service discovery capabilities after the completion of the migration to containerized microservices. The motivation behind this initiative was that we encountered abnormal functionality with Kubernetes DNS and service discovery, particularly in scenarios involving numerous external name services and pods. Therefore, we conducted an investigation to identify the root cause of the problem, to evaluate CoreDNS, and to develop a solution for these challenges.

Performing a migration from legacy system to microservice-based architecture while introducing major architecture changes must be seamless for the product teams and for the end-users. During this study, we performed a gradual migration of front-end applications into containerized microservices deployed within a Kubernetes cluster. However, some of the remaining applications still operate on virtual machines. Until all the applications were completely migrated to a container-based microservice architecture, we need to find a solution to enable service discovery between the Kubernetes cluster and the VM(s). Therefore, we utilized Netflix Eureka as a key component to underpin the service discovery pattern and to provide client-side load balancing. The microservices are able to register themselves with the Eureka server, which stores the microservices' access information, including their respective ports and IP addresses [7].

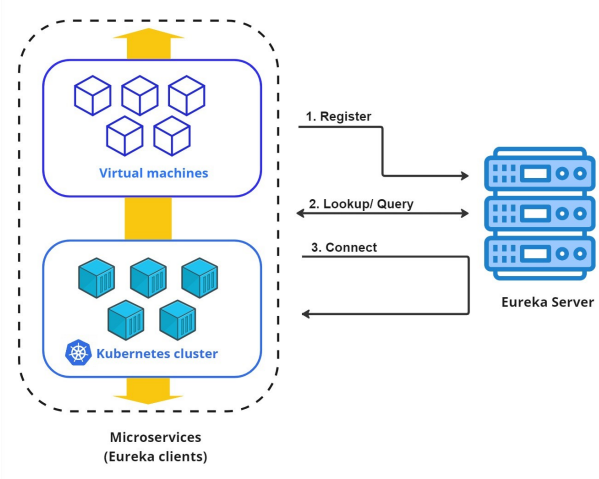


Fig. 2. Service discovery between Kubernetes cluster and VM(s) through Eureka.

For our case, when a microservice running in the Kubernetes cluster needs to invoke another microservice located on a VM, communication completely goes out of the Kubernetes cluster, and reaches out to the Eureka server to discover the location of the called microservice. Then it returns back to the Kubernetes cluster with the desired data as shown in Figure 2. Even when container-based microservices require communication with one another, they rely on Eureka since it is the primary service discovery mechanism within the entire system for now. This approach is not quite efficient from a performance perspective, particularly in large-scale clusters. By transitioning to a container-based architecture, we are able to eliminate certain components from the legacy infrastructure. Once the microservices are fully containerized and deployed in the Kubernetes cluster, we can remove Eureka. This not only enhances performance, but also facilitates a cloud-native deployment. Kubernetes itself can handle both service discovery and load balancing, enabling a more streamlined and efficient approach.

We have identified the necessary steps to smoothly decommission the use of Netflix Eureka during the migration process, with the aim of achieving zero downtime, resolving traffic issues, minimizing effort, and reducing risks and impacts. Our goal was to ensure that microservices should communicate with each other using both Netflix Eureka and Kubernetes service discovery in a seamless manner during the transition phase, simply by switching a "discovery flag". We conducted tests on API endpoints before and after switching the Eureka client flag to verify that the same code base functions properly in both scenarios. The following changes were implemented during the migration process.

A. Code-based changes

- Changes in the `pom.xml` [19] file regarding the dependencies.
- Changes in the main Spring Boot `application.java` file.
- Changes in the `bootstrap.yaml` [20] file of Spring Boot.

B. Adjustments of the cluster configurations

- Modifications in the config-server database – When utilizing Kubernetes discovery, the microservice configuration is automatically updated to include the "Kubernetes" active profile. It is important to note that the default configuration profile is labeled as "development", which differs from the "Kubernetes" profile. To ensure the reliability of microservices, a new profile named "Kubernetes" must be added to the config-server database.
- Add role [21] and role-binding [22] to service account [23] in the Kubernetes cluster – In order for Spring Cloud Kubernetes to retrieve a list of addresses for pods belonging to a specific service, it requires access to the Kubernetes API. To ensure this access, deployment or pod must be assigned to the relevant service accounts, and it is essential to verify that they possess the correct roles.
- Changes in the global configuration file – This modification allows for the convenient switching of the discovery flag between Eureka and Discovery-client, or vice versa.

By implementing these necessary changes in the code base and the configurations, a safe and straightforward transition from Netflix Eureka to native Kubernetes service discovery becomes achievable. After moving dozens of microservices, our implementation proved to be effective and significantly alleviated the burden of the migration process. The focus on simplicity has played a crucial role in enabling a smoother transition while saving effort and time.

IV. Performance Improvements

In large scale Kubernetes clusters, the total number of running pods, services, requests, and workloads can be high, and the increased number of HTTP-requests often result in resource utilization concerns, e.g., spikes of errors [24], [25]. The memory usage of Kubernetes DNS is predominantly affected by the number of pods and services in the cluster [26], [27]. Other factors include the size of the filled DNS answer cache, and the rate of queries received per CoreDNS instance [26].

Upon encountering resource consumption issues and a spike of errors in HTTP-requests, we started to troubleshoot the core pain points, and to solve the issues by fine-tuning the configuration of CoreDNS [24]. Our initial idea was to increase the number of replicas for the application to assess whether it would help enhance performance and mitigate errors. As we delved deeper into the issue with the application developers, we discovered that the majority of failures could be attributed to DNS resolution. This discovery led us to shift our focus towards improving the performance of DNS resolution in Kubernetes.

We have carried out a stress test on service discovery to identify bottlenecks. For our experiment, we used a cluster with one master and 10 worker nodes, which were set up with the default settings of Kubernetes. We executed Java-based

TABLE I
PERFORMANCE BEFORE OPTIMIZATION

Number of pods & services	Max memory (MB)	Max CPU (cores)	Average response time in seconds	Network load: Receive (MB/s)	Network load: Transmit (MB/s)
0	19	0.0001	0.0008	0.0031	0.0047
250	776	2.74	0.67	3.7	6.37
1054	8914	5.27	4.02	8.10	13.98
2000	16664	9.14	8.19	11.25	18.36

front-end applications and microservices as Kubernetes pods and Kubernetes services on that cluster. The figures presented in Table I are based on data collected from the tests using the following setup.

- Master: n1-standard-1 (16 vCPU, 32 GB memory)
- Nodes: n1-standard-2 (32 vCPUs, 125 GB memory)
- Networking: calico-3.19.1
- Kubernetes Version: 1.21.3
- CoreDNS Version: 1.7.0

As shown in Table I, resource consumption and network load drastically increase when the total number of services and pods are raised. We need to emphasize that here the type of most of the services was `externalName` [28], which was one of the reasons of the phenomena that when the total number of services and pods were beyond 1054, the system consumed high amount of resources such as around 16 GB of memory and 9 CPU cores. The average response time for 2000 pods/services was around 8.19 seconds, which resulted in high latency and high error rate for HTTP-requests.

The CoreDNS function with default configuration occasionally crashed when running 500 external services and pods in the Kubernetes cluster. After this incident, we adjusted the memory resource "request/limit" in the CoreDNS deployment up to 8 GB from 170 MB, and increased the total number of instances to four. This high amount of resource consumption indicates that the current implementation may exhibit abnormal behavior, which can lead to malfunctions and failures of the CoreDNS.

According to our experiences, it is insufficient to merely add extra CoreDNS instances or configure Horizontal Pod Autoscaler (HPA) for the cluster based on number of requests, resource consumption, and number of workloads running on the cluster to address the performance and stability problem effectively, especially for large-scale clusters in which numerous projects and environments are being developed simultaneously. Expanding resource utilization continuously in response to increased requests is fruitless, even if the cluster possesses sufficient resources. Therefore, an accurate and appropriate solution is necessary to address these concerns. As we started to investigate more into how the application is making requests to CoreDNS, and troubleshooting the DNS resolution and cluster configurations, we observed most of the outbound requests happening through the application to an external API which leads a high spike of errors. Also, we found out several obstacles that hindered the smooth functioning of the system, including the lack of a logical and well-defined DNS search flow, and misconfigured Kubernetes service objects. These matters had a substantial impact on CoreDNS, leading to failures and excessive load that caused such high resource consumption and latency issues.

Therefore, we developed a dedicated solution at the cluster level, with which organizations can mitigate the previously mentioned service discovery issues and ensure smooth functioning of the Kubernetes clusters.

A. Reconfiguring service object

In our scenario, pods/instances frequently perform external lookups through the `externalName` service object in Kubernetes. We discovered that issues stemmed from unnecessary port definitions in the `externalName` service. However, rectifying this required code modifications on the microservice side to create the `externalName` service without any port definition. As a solution, we made updates to the source code, incorporating the Fully Qualified Domain Name (FQDN) of the client service. This approach prevents invalid DNS lookups resulting from search domains. For example, if the client pod needs to access rate-service, the domain name can be specified as www.rate-service.com. To ensure the effectiveness of these changes, comprehensive end-to-end tests were conducted in the non-production environment.

B. Enabling local DNS Cache

In addition, we implemented a Node Level DNS Cache to enhance the stability and performance of service discovery. This involved optimizing DNS resolution, improving latency, and reducing the burden of CoreDNS. With the current DNS architecture, it is possible that pods with the highest DNS QPS have to reach out to a different node, if there is no local CoreDNS instance [29]. To address this, we deployed local *DNS Caching* agents on nodes as a *Daemonset*, which significantly improved the performance of Cluster DNS. It works as a CoreDNS caching agent and pods will reach out to the agent running on the same node. Thereby it helps to avoid connection tracking and iptables DNAT rules. When the local caching agent encounters cache misses for cluster hostnames, it queries the core-dns service for resolution.

C. Defining search sequence

During our in-depth investigation into how the application sends requests to CoreDNS, we discovered that a significant portion of outbound requests were directed towards an external

API. This led to domain name resolution errors occurring within the cluster. To resolve this issue, we took the necessary steps to optimize the resolv.conf file of the application deployment pod.

```
nameserver 10.10.10.151
search local-namespace.svc.cluster.local \
       global-available.svc.cluster.local \
       svc.cluster.local cluster.local \
       head-zones.local
```

Fig. 3. Defined DNS Search sequence.

We specify the domain name that a client pod needs to access based on the rules as shown in Figure 3. These rules can help minimize the number of attempts for resolving the domain name, and make the DNS resolution service more efficient. When the DNS resolver sends a query to the CoreDNS server, it tries to search the domain considering the search path. If the client pod needs to access a service in the same namespace, the initial search attempt must always be inside a local environment, then the second attempt, in this case, will search in the next specified environment (global available environment). If the domain remains unresolved, the search process proceeds to the current cluster level. If the domain is still not resolved at this stage, the search is expanded to encompass the entire infrastructure of the organization.

If we are looking for a domain `frontapp.io`, the search would make the queries which are presented in Figure 4, and it receives a successful response in the last query.

```
frontapp.io.local-namespace.svc.cluster.local <= NXDomain
frontapp.io.global-available.svc.cluster.local<= NXDomain
frontapp.io.svc.cluster.local <= NXDomain
frontapp.io.cluster.local <= NXDomain
frontapp.io.head-zones.local <= NXDomain
frontapp.io <= NoERROR
```

Fig. 4. DNS query for lookup.

Due to the excessive number of external lookups, an application may receive numerous NXDomain responses for DNS searches. To optimize this, we customized dnsConfig in the Deployment object of the container, which will change `resolve.conf` accordingly on pods. The search is being performed only for an external domain. This reduces the number of queries to DNS servers, and helps mitigate spike errors for the application.

D. Ensuring availability – Optimize resource allocation

During time periods of high DNS query volume and with numerous services/pods in the cluster, CoreDNS tends to consume additional memory and CPU resources. Therefore, the default memory limit can lead to out of memory (OOM) errors. As a result, CoreDNS pods undergo repetitive restart attempts but fail to start successfully. To address this, we fine-tuned CoreDNS and adjusted the resource requirements within the cluster. Specifically, we modified the default values for memory resource "requests and limits" from 170MB to 1GB, and for CPU from 1 milli-core to 500 milli-cores, based on the cluster's status.

E. Enabling scalability – Auto-scale the number of pods - Horizontal Pod Autoscaler

We have implemented the Horizontal Pod Autoscaler (HPA) to dynamically adjust the number of CoreDNS pods. Currently, the cluster is provisioned with five CoreDNS pods. In the event of increased resource utilization leading to overload, we have incorporated a backup configuration within the autoscaler. The HPA is configured with the policy settings illustrated in Figure 5, enabling it to increase the number of CoreDNS pods based on CPU utilization.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: coredns-hpa
  namespace: kube-system
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  minReplicas: 5
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        targetAverageUtilization: 70
```

Fig. 5. HPA configuration based on CPU usage.

V. Results

The combination of the above-mentioned changes and tuning solutions has significantly addressed most of the previously experienced issues, and provided a more optimal resource consumption, minimized the blast radius of CoreDNS crashes, mitigated DNS errors, timeouts, and latency. After the adjustments, the resource utilization of CoreDNS has drastically dropped into a very low amount, as demonstrated in Table II. Even when the total pod and service count reached 2000 (and above), it only consumed 524 MB of memory and 0.2 cores of CPU, which is a remarkable improvement compared to the original state. In the initial measurement data (shown in Table I) we could observe very high latency – around 8 seconds at peak load with 2000 pods/services.

With the new implementation, the average response time has been reduced to less than 2 milliseconds, falling within an acceptable range. This improvement has also helped us minimize spike errors in HTTP-requests. The new measurement proves that the original implementation was functioning in an abnormal way due to design and configuration flaws, resulting in the malfunction and failures of CoreDNS.

VI. Related work

The solution for the increased load and errors of service discovery in Kubernetes requires a multi-faceted approach that addresses both scalability and reliability challenges. To mitigate the issues and ensure smooth operation of the execution environment, a number of techniques can be employed.

TABLE II
MEASUREMENT DATA AFTER THE ADJUSTMENTS.

Number of pods & services	Max Memory (MB)	Max CPU (core)	Average Response time in seconds	Network load: Receive (MB/s)	Network load: Transmit (MB/s)
0	19	0.0001	0.00071	0.0019	0.0025
250	76.480	0.096	0.00104	0.86	0.72
1054	280.72	0.155	0.00125	1.42	1.26
2000	524.09	0.203	0.00172	1.8	1.45

Nguyen et al. [30] proposes the horizontal scaling of the containers using Kubernetes' built-in autoscaling capabilities; in this way resources can be dynamically allocated based on demand, ensuring that DNS pods can adeptly manage surges in traffic and efficiently distribute the workload. We also applied horizontal scaling, but the technique turned out to be insufficient to solve the problems on its own, therefore we had to investigate other solutions as well.

Zhang Wei-guo et al. [31] emphasise that to prevent resource exhaustion and reduce the occurrence of errors or crashes, it is essential to optimize the allocation of resources for DNS pods, including setting appropriate CPU and memory limits. By ensuring accurate resource provisioning, a service discovery pod can gain the necessary capacities to handle the workload efficiently. This approach was still insufficient in our case, thereby we applied it with additional techniques.

Nguyen and Kim [32] argue that implementing a robust load balancing mechanism, either within Kubernetes or by integrating with external load balancers, enables the even distribution of DNS requests across multiple DNS instances. Load balancing helps prevent bottlenecks and ensures high availability. After introducing the changes to our cluster configuration and service discovery, the use of a special load balancer was not necessary. However, it might become so in the future, when the number of system components grow even higher.

Almaraz-Rivera [33] underlines the importance of establishing comprehensive monitoring and alerting systems to track containers' performance, latency, error rates, and resource utilization. Proactive monitoring enables the early detection of potential issues and allows for timely remediation. In align with this approach, we introduced Prometheus and Grafana as monitoring and alerting tools for our Docker-based Kubernetes infrastructure.

Horaleket et al. [34] point out that enabling detailed logging for containers and leveraging logging aggregation solutions can aid in troubleshooting errors and performance issues. Analyzing logs can provide valuable insights into the root causes of problems and guide further optimizations. This technique was a key enabler in our methodology as well. The execution environment and the applications referred to in this paper employ Elasticsearch and Kibana for real-time search, analysis, visualization, and management of massive datasets.

The combination of these techniques can help overcome the challenges related with increased load and errors in CoreDNS within Kubernetes, allowing for the stable and effective operation of DNS resolution. However, they may not be adequate, or sufficient, to handle the arising problems in a variety of scenarios. In our case, the aforementioned approaches were still unsatisfactory to fully address the service discovery challenges. Consequently, we investigated alternative technical and engineering solutions. In this paper, we presented an in-depth explanation of our strategy to tackling the experienced difficulties.

VII. CONCLUSION

As containerization technologies become intensively used, certain challenges and problems arise. This paper proposed a technique to gradually migrate virtual machine based microservices to containerized ones, and solved an issue (which was discovered in a large-scale migration process) in the name service component of a popular cluster management solution.

We introduced a technique to help developers transition from Netflix Eureka based service discovery to a more lightweight native Kubernetes service discovery. This technique is useful when an application is gradually refactored from VM-based to Docker-based microservices, temporarily containing both kinds of components.

We discovered an issue with the default configuration of CoreDNS, the name service of Kubernetes, which causes performance degradation and service failures for high loads. We propose modifications which result improvements in the range of 1–2 orders of magnitude, and drastically increases the stability of CoreDNS.

ACKNOWLEDGMENT

We would like to acknowledge and thank the company and company representatives for their contributions and involvement in the study.

REFERENCES

[1] Blinowski, G., Ojdowska, A. & Przybylek, A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. *IEEE Access*. 10 pp. 1–1 (2022,1)

[2] Thones, J. Microservices. *IEEE Software*. 32 pp. 116–116 (2015,1) DOI: 10.1109/MS.2015.11

[3] Qian, L., Chen, H., Yu, J., Zhu, G., Zhu, J., Ren, C., Mei, Z., Pang, H., Xu, M. & Wang, L. Research on Micro Service Architecture of Power Information System Based on Docker Container. *IOP Conference Series: Earth And Environmental Science*. 440, 032147 (2020,2), DOI: 10.1088/1755-1315/440/3/032147

[4] Shifeng, Z. & Shanliang, P. Application of Docker technology in micro-service. *Electronic Technology And Software Engineerings*. 4, 164 (2019)

[5] Merkel, D. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*., 2 (2014)

[6] Ákos, L., Edina, L., Attila, H., József, V., and László, B., "Closed-loop Orchestration for Cloud-native Mobile IPv6", *Infocommunications Journal*, Vol. XV, No 1, March 2023, pp. 44–54., DOI: 10.36244/ICJ.2023.1.5

[7] Macero, M. Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber. *Learn Microservices With Spring Boot*, 1st ed. Berkeley, CA:Apress, 2017, DOI: 10.1007/978-1-4842-3165-4

[8] Microservices: What are microservices, Available online: https://microservices.io/ (Accessed: 2023)

[9] Li, Z., Kihl, M., Lu, Q. & Andersson, J. Performance overhead comparison between hypervisor and container based virtualization. *In Proceedings Of The 2017 IEEE 31st International Conference On Advanced Information Networking And Applications (AINA)*. pp. 955–962, 2017, DOI: 10.1145/1342250.1342261

[10] Germar, S., Paul, P., Matthias, F., Dirk, R., Feryel, Z., and Jerker, D., "Micro Service based Sensor Integration Efficiency and Feasibility in the Semiconductor Industry", *Infocommunications Journal*, Vol. XIV, No 3, September 2022, pp. 79–85., DOI: 10.36244/ICJ.2022.3.10

[11] IBM: Docker, Available online: https://www.ibm.com/topics/docker (Accessed: 2023)

[12] The Kubernetes Authors. Available online: https://kubernetes.io/ (Accessed: 2023)

[13] Brewer, E. Kubernetes and the path to cloud native. SoCC '15: Proceedings of the Sixth ACM Symposium on Cloud Computing (2015,8), DOI: 10.1145/2806777.2809955

[14] Ranjan R, W., A, L. & A, Q. Peer-to-peer cloud provisioning: Service discovery and load-balancing. *Cloud Computing*. pp. 195–217, 2010.

[15] Christudas, B. *Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud*. Apress (2019,1) DOI: 10.1007/978-1-4842-4501-9

[16] Sharp, T. Deploying the Microservice as a Docker Container. *Introducing Micronaut*. 2022, DOI: 10.1007/978-1-4842-8290-8-9

[17] Md, Varadarajan, A., Mandal, V. & Karan. Efficient Algorithm for Identification and Cache Based Discovery of Cloud Services. *Mobile Networks And Applications*. 24, pp. 1–17, 2019, DOI: 10.1007/s11036-019-01256-0

[18] Suryotrisongko, H., Jayanto, D. & Tjahyanto, A. Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot. *Procedia Computer Science*. 124, pp. 736–743, 2017, DOI: 10.1016/j.procs.2017.12.212

[19] Apache Maven. Introduction to the POM. Available online: https://maven.apache.org/guides/introduction/introduction-to-thepom.html (Accessed: 2023)

[20] Spring Cloud Context: Application Context Services. Available online: https://cloud.spring.io/spring-cloud-commons/multi/multi__spring_cloud_context_application_context_services.html (Accessed: 2023)

[21] The Kubernetes Authors. Using RBAC Authorization. Available online: https://Kubernetes.io/docs/reference/access-authn-authz/rbac/ (Accessed: 2023)

[22] The Kubernetes Authors. RBAC Good Practices. Available online: https://kubernetes.io/docs/concepts/security/rbac-good-practices/ (Accessed: 2023)

[23] The Kubernetes Authors: Service Accounts. Available online: https://kubernetes.io/docs/concepts/security/service-accounts/ (Accessed: 2023)

[24] CoreDNS: DNS and Service Discovery, Available online: https://coredns.io/, (Accessed: 2023)

[25] Kubernetes: Using CoreDNS for Service Discovery. Available online: https://kubernetes.io/docs/tasks/administer-cluster/coredns (Accessed: 2023)

[26] Heidari, A., Navimipour, N. Service discovery mechanisms in cloud computing: a comprehensive and systematic literature review. *Kybernetes*. 51, pp. 952–981, 2022, DOI: 10.1108/K-12-2020-0909

[27] Singh, N., Hamid, Y., Juneja, S., Srivastava, G., Dhiman, G., Gadekallu, T. R., Mohd, A.S. Load balancing and service discovery using Docker Swarm for microservice based big data applications. *Journal of Cloud Computing*. 12:4, 2023, DOI: 10.1186/s13677-022-00358-7

[28] The Kubernetes Authors. Services-External Name. Available online: https://Kubernetes.io/docs/concepts/services-networking/service/ (Accessed: 2023).

[29] Using NodeLocal DNS Cache in Kubernetes Clusters. Available online: https://kubernetes.io/docs/tasks/administer-cluster/nodelocaldns/ (Accessed: 2023)

[30] Nguyen, T.-T., Yeom, Y.-J., Kim, T., Park, D.-H., Kim, S. Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration, *Sensors*, 20(16), p. 4621, Aug. 2020, DOI: 10.3390/s20164621.

[31] Wei-guo, Z., Xi-lin, M., Jin-zhong, Z. Research on Kubernetes' Resource Scheduling Scheme. In Proceedings of the 8th International Conference on Communication and Network Security (ICCNS '18). Association for Computing Machinery, New York, NY, USA, 2018, 144–148. DOI: 10.1145/3290480.3290507

[32] Nguyen, N., Kim, T. Toward Highly Scalable Load Balancing in Kubernetes Clusters. *IEEE Communications Magazine*, 58(7):78–83, July 2020, DOI: 10.1109/MCOM.001.1900660.

[33] Almaraz-Rivera, J.G. An Anomaly-based Detection System for Monitoring Kubernetes Infrastructures, *IEEE Latin America Transactions*, 21(3):457–465, March 2023, DOI: 10.1109/TLA.2023.10068850.

[34] Horalek, J., Urbanik, P., Sobeslav, V., Svoboda, T. "Proposed Solution for Log Collection and Analysis in Kubernetes Environment." in *Nature of Computation and Communication*. ICTCC 2022. vol 473. Springer, Cham, 2023 DOI: y10.1007/978-3-031-28790-9_2

Baasanjargal Erdenebat was born in 1993 in Arkhangai, Mongolia. She earned the bachelor and master degrees from the School of Applied Science and Engineering at the National University of Mongolia, in 2013 and 2016 respectively. Currently, she is pursuing a PhD at Eötvös Loránd University, Dept. Programming Languages and Compilers. Her main research interests are containerization technology (i.e., Docker, Kubernetes), cloud computing, DevOps methodology, as well as the toolchains and implementation related to these areas.

Bayarjargal Bud was born in 1987, Mongolia. He obtained the bachelor's degree from the National University of Mongolia (NUM) in 2010, and master's degree from NUM and Riga Technical University in 2021. Currently, he works as Tech Lead at IT delivery service department of the private sector. His research interests include database system, machine learning, and containerization technology (i.e., Docker, Kubernetes).

Tamás Kozsik is associate professor at Eötvös Loránd University, Dept. Programming Languages and Compilers. His fields of interest are formal verification, programming paradigms (i.e., concurrent programming), static analysis, refactoring, and domain specific programming languages. Currently he focuses on researching programming languages and software technology for quantum computing.

Guidelines for our Authors

Format of the manuscripts

Original manuscripts and final versions of papers should be submitted in IEEE format according to the formatting instructions available on

https://journals.ieeeauthorcenter.ieee.org/
Then click: "IEEE Author Tools for Journals"
- "Article Templates"
- "Templates for Transactions".

Length of the manuscripts

The length of papers in the aforementioned format should be 6-8 journal pages.
Wherever appropriate, include 1-2 figures or tables per journal page.

Paper structure

Papers should follow the standard structure, consisting of *Introduction* (the part of paper numbered by "1"), and *Conclusion* (the last numbered part) and several *Sections* in between.
The Introduction should introduce the topic, tell why the subject of the paper is important, summarize the state of the art with references to existing works and underline the main innovative results of the paper. The Introduction should conclude with outlining the structure of the paper.

Accompanying parts

Papers should be accompanied by an *Abstract* and a few *Index Terms (Keywords)*. For the final version of accepted papers, please send the short cvs and *photos* of the authors as well.

Authors

In the title of the paper, authors are listed in the order given in the submitted manuscript. Their full affiliations and e-mail addresses will be given in a footnote on the first page as shown in the template. No degrees or other titles of the authors are given. Memberships of IEEE, HTE and other professional societies will be indicated so please supply this information. When submitting the manuscript, one of the authors should be indicated as corresponding author providing his/her postal address, fax number and telephone number for eventual correspondence and communication with the Editorial Board.

References

References should be listed at the end of the paper in the IEEE format, see below:

a) Last name of author or authors and first name or initials, or name of organization
b) Title of article in quotation marks
c) Title of periodical in full and set in italics
d) Volume, number, and, if available, part
e) First and last pages of article
f) Date of issue
g) Document Object Identifier (DOI)

[11] Boggs, S.A. and Fujimoto, N., "Techniques and instrumentation for measurement of transients in gas-insulated switchgear," IEEE Transactions on Electrical Installation, vol. ET-19, no. 2, pp.87–92, April 1984. DOI: 10.1109/TEI.1984.298778

Format of a book reference:

[26] Peck, R.B., Hanson, W.E., and Thornburn, T.H., Foundation Engineering, 2nd ed. New York: McGraw-Hill, 1972, pp.230–292.

All references should be referred by the corresponding numbers in the text.

Figures

Figures should be black-and-white, clear, and drawn by the authors. Do not use figures or pictures downloaded from the Internet. Figures and pictures should be submitted also as separate files. Captions are obligatory. Within the text, references should be made by figure numbers, e.g. "see Fig. 2."
When using figures from other printed materials, exact references and note on copyright should be included. Obtaining the copyright is the responsibility of authors.

Contact address

Authors are requested to submit their papers electronically via the following portal address:

https://www.ojs.hte.hu/infocommunications_journal/about/submissions

If you have any question about the journal or the submission process, please do not hesitate to contact us via e-mail:

Editor-in-Chief: Pál Varga – pvarga@tmit.bme.hu

Associate Editor-in-Chief:
Rolland Vida – vida@tmit.bme.hu
László Bacsárdi – bacsardi@hit.bme.hu

Infocommunications Journal

Special Issue

on Applied Informatics

Selected papers of **ICAI 2023**
12th International Conference on Applied Informatics
Eger, Hungary on March 2-4, 2023

IEEE ComSoc
IEEE Communications Society

hte
SCIENTIFIC ASSOCIATION FOR INFOCOMMUNICATIONS HUNGARY
FOUNDED IN 1949

IEEE
HUNGARY SECTION

SCIENTIFIC ASSOCIATION FOR INFOCOMMUNICATIONS

Who we are

Founded in 1949, the Scientific Association for Infocommunications (formerly known as Scientific Society for Telecommunications) is a voluntary and autonomous professional society of engineers and economists, researchers and businessmen, managers and educational, regulatory and other professionals working in the fields of telecommunications, broadcasting, electronics, information and media technologies in Hungary.

Besides its 1000 individual members, the Scientific Association for Infocommunications (in Hungarian: Híradástechnikai Tudományos Egyesület, HTE) has more than 60 corporate members as well. Among them there are large companies and small-and-medium enterprises with industrial, trade, service-providing, research and development activities, as well as educational institutions and research centers.

HTE is a Sister Society of the Institute of Electrical and Electronics Engineers, Inc. (IEEE) and the IEEE Communications Society.

What we do

HTE has a broad range of activities that aim to promote the convergence of information and communication technologies and the deployment of synergic applications and services, to broaden the knowledge and skills of our members, to facilitate the exchange of ideas and experiences, as well as to integrate and harmonize the professional opinions and standpoints derived from various group interests and market dynamics.

To achieve these goals, we…

- contribute to the analysis of technical, economic, and social questions related to our field of competence, and forward the synthesized opinion of our experts to scientific, legislative, industrial and educational organizations and institutions;
- follow the national and international trends and results related to our field of competence, foster the professional and business relations between foreign and Hungarian companies and institutes;
- organize an extensive range of lectures, seminars, debates, conferences, exhibitions, company presentations, and club events in order to transfer and deploy scientific, technical and economic knowledge and skills;
- promote professional secondary and higher education and take active part in the development of professional education, teaching and training;
- establish and maintain relations with other domestic and foreign fellow associations, IEEE sister societies;
- award prizes for outstanding scientific, educational, managerial, commercial and/or societal activities and achievements in the fields of infocommunication.

Contact information

President: **FERENC VÁGUJHELYI** • *elnok@hte.hu*

Secretary-General: **ISTVÁN MARADI** • *istvan.maradi@gmail.com*

Operations Director: **PÉTER NAGY** • *nagy.peter@hte.hu*

International Affairs: **ROLLAND VIDA, PhD** • *vida@tmit.bme.hu*

Address: H-1051 Budapest, Bajcsy-Zsilinszky str. 12, HUNGARY, Room: 502

Phone: +36 1 353 1027

E-mail: *info@hte.hu*, Web: *www.hte.hu*