

# Journal Of Computer And Information Sciences



SAKARYA UNIVERSITY

e-ISSN 2636-8129

VOLUME 5

ISSUE 2

AUGUST 2022

*Malware Detection Method Based on File and Registry Operations Using Machine Learning*

***Sentiment Analysis on the Metaverse: Twitter Data***

*Effects of Neighborhood-Based Collaborative Filtering Parameters on Their Blockbuster Bias Performances*

***Application with Deep Learning Models for COVID-19 Diagnosis***

***An Approach for Audio-Visual Content Understanding of Video using Multimodal Deep Learning Methodology***



www.saucis.sakarya.edu.tr

***Transfer of Analogies in Traditional Programming Languages to Teaching VHDL***

***An Implementation of Traffic Signs and Road Objects Detection Using Faster R-CNN***

***A Relational Database Design for The Compounds Cytotoxically Active on Breast Cancer Cells***

***The Effect of Ambient Temperature On Device Classification Based On Radio Frequency Fingerprint Recognition***

***A Software Defined Networking-based Routing Algorithm for Flying Ad Hoc Networks***



# SAUCIS

**Sakarya University Journal of Computer and Information  
Sciences Volume: 5 – Issue No: 2 (August 2022)**  
<http://saucis.sakarya.edu.tr/issue/72246>

## Editor in Chief

Nejat Yumuşak, Sakarya University, nyumusak@sakarya.edu.tr

## Associate Editors

İhsan Hakan Selvi, Sakarya University, Turkey, ihselvi@sakarya.edu.tr

Muhammed Fatih Adak, Sakarya University, Turkey, fatihadak@sakarya.edu.tr

Mustafa Akpınar, Sakarya University, Turkey, akpınar@sakarya.edu.tr

Unal Cavusoglu, Sakarya University, Turkey, unalc@sakarya.edu.tr

Veysel Harun Sahin, Sakarya University, Turkey, vsahin@sakarya.edu.tr

## Editorial Assistants - Secretary

Deniz Balta, Sakarya University, Turkey, dduural@sakarya.edu.tr

Fatma Akalin, Sakarya University, Turkey, fatmaakalin@sakarya.edu.tr

Gozde Yolcu Oztel, Sakarya University, Turkey, gyolcu@sakarya.edu.tr

Ibrahim Delibasoglu, Sakarya University, Turkey, ibrahimdelibasoglu@sakarya.edu.tr

Muhammed Kotan, Sakarya University, Turkey, mkotan@sakarya.edu.tr

Sumeyye Kaynak, Sakarya University, Turkey, sumeyye@sakarya.edu.tr

Ahmet Erhan Tanyeri, Sakarya University, Turkey, tanyeri@sakarya.edu.tr

## Editorial Board

Ahmet Ozmen, Sakarya University, Turkey, ozmen@sakarya.edu.tr

Aref Yelghi, Istanbul Ayvansaray University, ar.yelqi@gmail.com

Ayhan Istanbulu, Balikesir University, Turkey, iayhan@balikesir.edu.tr

Aysegul Alaybeyoglu, Izmir Katip Celebi University, Turkey, alaybeyoglu@gmail.com

Bahadir Karasulu, Canakkale Onsekiz Mart University, bahadirkarasulu@comu.edu.tr

Celal Ceken, Sakarya University, Turkey, celalceken@sakarya.edu.tr

Cihan Karakuzu, Bilecik Seyh Edebali University, cihan.karakuzu@bilecik.edu.tr

Fahri Vatansever, Bursa Uludag University, fahriv@uludag.edu.tr

Ibrahim Turkoglu, Fırat University, Turkey, iturkoglu@firat.edu.tr

Levent Alhan, Sakarya University, Turkey, leventalhan@sakarya.edu.tr

Kamal Z Zamli, Malaysia Pahang University, Malaysia, kamalz@ump.edu.my

Muhammed Fatih Adak, Sakarya University, Turkey, fatihadak@sakarya.edu.tr

Mustafa Akpınar, Sakarya University, Turkey, akpınar@sakarya.edu.tr



# SAUJCIS



## Editorial Board (Cont.)

Nuri Yilmazer, Texas A&M University, US, [nuri.yilmazer@tamuk.edu](mailto:nuri.yilmazer@tamuk.edu)

Nejat Yumuşak, Sakarya University, [nyumusak@sakarya.edu.tr](mailto:nyumusak@sakarya.edu.tr)

Orhan Er, Bozok University, Turkey, [orhan.er@bozok.edu.tr](mailto:orhan.er@bozok.edu.tr)

Priyadip Ray, Lawrence Livermore National Laboratory, [priyadipr@gmail.com](mailto:priyadipr@gmail.com)

Resul Das, Firat University, Turkey, [rdas@firat.edu.tr](mailto:rdas@firat.edu.tr)

Veysel Harun Sahin, Sakarya University, Turkey, [vsahin@sakarya.edu.tr](mailto:vsahin@sakarya.edu.tr)





# SAUCIS

Sakarya University Journal of Computer and Information Sciences  
Volume: 5 – Issue No: 2 (August 2022)  
<http://saucis.sakarya.edu.tr/issue/72246>

## Contents

Author(s), Paper Title	Pages
<i>Ömer ASLAN, Erdal Akin</i> Malware Detection Method Based on File and Registry Operations Using Machine Learning	134-146
<i>Gulsum Akkuzukaya</i> Sentiment Analysis on the Metaverse: Twitter Data	147-156
<i>Emre Yalçın</i> Effects of Neighborhood-Based Collaborative Filtering Parameters on Their Blockbuster Bias Performances	157-168
<i>Fuat Türk, Yunus Kökver</i> Application with deep learning models for COVID-19 diagnosis	169-180
<i>Emre Beray Boztepe, Bedirhan Karakaya, Bahadır Karasulu, İsmet Ünlü</i> An Approach for Audio-Visual Content Understanding of Video using Multimodal Deep Learning Methodology	181-207
<i>Halit Öztekin, Ali Gülbağ</i> Transfer of Analogies in Traditional Programming Languages to Teaching VHDL	208-215
<i>Emin Güney, Cüneyt Bayılmış</i> An Implementation of Traffic Signs and Road Objects Detection Using Faster R-CNN	216-224
<i>Zeynep Oktay, Çiğdem Erol, Nazlı Arda</i> A Relational Database Design for The Compounds Cytotoxically Active on Breast Cancer Cells	225-232
<i>Özkan Yılmaz, Mehmet Akif Yazıcı</i> The Effect of Ambient Temperature On Device Classification Based On Radio Frequency Fingerprint Recognition	233-245
<i>Mehmet Akif Yazıcı, Berat Erdemkılıç</i> A Software Defined Networking-based Routing Algorithm for Flying Ad Hoc Networks	246-256

# Malware Detection Method Based on File and Registry Operations Using Machine Learning

 Ömer Aslan<sup>1</sup>,  Erdal Akın<sup>2</sup>

<sup>1</sup>Corresponding Author; 17 Eylül University, Department of Software Engineering, Bandırma/Balıkesir Turkey; omer.aslan.bisoft@gmail.com

<sup>2</sup>Bitlis Eren University, Department of Computer Engineering, Bitlis, Turkey; erdalakin1985@hotmail.com

Received 28 December 2021; Revised 18 April 2022; Accepted 25 May 2022; Published online 31 August 2022

## Abstract

Malware (Malicious Software) is any software which performs malicious activities on computer-based systems without the user's consent. The number, severity, and complexity of malware have been increasing recently. The detection of malware becomes challenging because new malware variants are using obfuscation techniques to hide themselves from the malware detection systems. In this paper, a new behavioral-based malware detection method is proposed based on file-registry operations. When malware features are generated, only the operations which are performed on specific file and registry locations are considered. The file-registry operations are divided into five groups: autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs (Dynamic link libraries) related registry locations. Based on the file-registry operations and where they performed, the malware features are generated. These features are seen in malware samples with high frequencies, while rarely seen in benign samples. The proposed method is tested on malware and benign samples in a virtual environment, and a dataset is created. Well-known machine learning algorithms including C4.5 (J48), RF (Random Forest), SLR (Simple Logistic Regression), AdaBoost (Adaptive Boosting), SMO (Sequential Minimal Optimization), and KNN (K-Nearest Neighbors) are used for classification. In the best case, we obtained 98.8% true positive rate, 0% false positive rate, 100% precision and 99.05% accuracy which is quite high when compared with leading methods in the literature.

**Keywords:** Cybersecurity, malware detection, behavior-based detection, file-registry behaviors, machine learning

## 1. Introduction

In simple terms, malware can be defined as a set of symbols which performs undesirable changes to the computer hardware as well as operating system resources. There are various types of malware including virus, worm, rootkit, Trojan horse, backdoor, spyware, and so on. The number, complexity and damage of malware to the world economy is increasing everyday. According to scientific reports, the cost of cyber-based attacks to the world economy is estimated in trillions of dollars, and most of these damages come from malware.

To protect the computer based systems from malware, malware needs to be detected before entering the victim system or during the infection. Thus, malware samples need to be examined by using relevant tools. There are two common ways to analyze the malware: static and dynamic analysis [1]. In static analysis, the malware samples are analyzed without running the actual code. Program structures, used strings, imported and exported functions are obtained during the static analysis. However, in dynamic analysis, the codes of malware are performed under the protected environment (Virtual machines or sandboxes), and execution traces which represent the behaviors of the malware are collected. After the malware execution traces are collected, the features are generated. There are several approaches to detect malware which use static and dynamic analysis, namely, signature-based, behavioral-based, heuristic-based, model checking-based, deep learning-based, cloud-based, and mobile and IoT (Internet of Things)-based [2]. The names of the approaches vary according to the platform on which the detection algorithm is proposed and the method used.

At the beginning, the signature-based detection approach was widely used. However, due to the increasing complexity of malicious software in recent years, it has been insufficient to detect new

generation malware [3]. Therefore, over time, researchers have developed behavioral-based, heuristic-based and model checking-based detection approaches. In these approaches, models are created by determining the behaviors or static features of malware samples, and malicious software is detected by using these models [2]. These approaches can detect some malware that has not been seen before. Furthermore, deep learning-, cloud-, mobile- and IoT-based detection approaches have also been used especially in the last few years.

In this paper, a behavioral-based malware detection approach is used which identifies the specific file and registry operations. We divided file-registry operations into five groups including autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs (Dynamic link libraries) related registry locations. After behaviors are created by using specified locations, the features and their frequencies are calculated. The most significant features are selected by using information gain measures. After features are selected, the machine learning classifiers including C4.5 (J48 version), RF (Random Forest), SLR (Simple Logistic Regression), AdaBoost (Adaptive Boosting), SMO (Sequential Minimal Optimization), and KNN (K-Nearest Neighbors) are used for classification.

In practice, current malware detectors cannot effectively recognize the zero-day malware variants. The proposed approach decreased the deficiencies that current studies have on malicious software detection and improved the performances. In the proposed method, only specific file directories and registry locations are considered when detecting malware. This is because the majority of malware strains show similar behaviors on specific file and registry locations, which cannot be seen on benign samples. To increase the model performance while decreasing the time complexity of the proposed method, the same behaviors on different instances of the same resources match into the same feature, but the frequency of the feature is increased.

The rest of the paper is organized as follows: Section 2 provides background information about malware types, malware analysis, and detection processes. Section 3 reviewed the literature studies on malware detection methods. A proposed method is presented in section 4, and an implementation is given in section section 5. Results and discussion is explained in section 6 and conclusion is given in section 7.

## 2. Background Information

In this section, definition and types of malware as well as malware analysis process, and detection approaches are explained in detail.

### 2.1 Malware Definitions and Types of Malware

Any program which performs malicious payloads on victim machines can be defined as malware. There are several malware variants created everyday including virus, worm, Trojan Horse, rootkit, backdoor, ransomware, and many more. The detection of those malware variants becomes challenging because one malware type can present other malware features. Besides, for complicated and targeted attacks, various malware strains work together to increase the impact of the cyber attacks. In addition, new devices and technologies are connected to computer networks everyday, and most of those devices and technologies have several vulnerabilities for hackers to exploit. Code obfuscation techniques are hiding malicious codes from the malware detection system [2]. All of these reasons make the malware detection process more difficult. To effectively detect the different malware variants, the main definitions and features that malware present needs to be examined deeply. Thus, the well-known malware variants are explained shortly as follows:

- **Virus:** It is a malicious software which injects its code into other files to reproduce. During the propagation, it changes its appearance to become undetectable by antivirus scanners [4].
- **Worm:** Unlike the virus, worms do not need other programs to reproduce, instead worms use computer networks to spread. It identifies the vulnerable machines on the network and then copies

itself into those vulnerable systems [5]. Most worms open backdoors in the victim system as well as enable unauthorized access. They delete their own files to hide themselves in the infected system.

- **Trojan Horse:** It appears to be useful benign software, however, it contains malicious code blocks. The attachment program needs to be performed to infect the victim system. It can create backdoors, cause unauthorized access, and reveal sensitive information to the third parties.
- **Rootkit:** A set of programs that conceal its existence from the operating system. It is a kind of man in the middle attack because it intercept and change the communications between the interfaces and several os components [6]. Rootkits are generally used by hackers to hide their existence in the system and provide root level access privileges. Rootkits are also combined with other malware strains in order to perform more sophisticated attacks. It is almost impossible to detect kernel rootkits since they run in kernel mode.
- **Backdoor:** It is a program that bypasses the traditional security mechanisms and opens the system to remote access for attackers [4]. The created backdoors are used by hackers and other malicious softwares to launch more complicated cyber attacks. Backdoors are mostly installed on victim systems by using Worms and Trojan horses.
- **Ransomware:** Ransomware is one of the most destructive types of malware which is designed to prevent or limit access to a computer system [7] until some amount of ransom is paid as a cryptocurrency. Recently, the number and impact of ransomware attacks on big companies have been increasing rapidly.

## 2.2 Malware Analysis and Detection Processes

To effectively detect malware from celanware, malware needs to be analyzed by relevant tools automatically. During the analysis process, program structures or behaviors are obtained. Malware analysis process divided into two subcategories: static and dynamic analysis. In static analysis, the content of the malware is analyzed without performing the actual codes, on the other hand, in dynamic analysis, the code of the malware is analyzed under dynamic analysis tools and malicious activities are collected [1]. There are a variety of static and dynamic tools including BinText, Md5deep, PEiD, PEview, IDA Pro, API Monitor, Regshot, Process Explorer, Process Monitor, Wireshark and Sandboxes. After malware execution traces are collected by relevant tools, feature generation as well as selection process take place. During the feature creation and selection processes, data mining (DM) and machine learning (ML) techniques are used. Furthermore, ML classifiers are used for the detection process as well. The relationship between malware analysis method, detection approaches, and machine learning techniques can be seen in figure 1.

There are several malware detection approaches which mostly use ML-based detection techniques. The name of the malware detection approaches change on the feature generation and selection processes as well as the platform that is used. The malware detection approaches are broadly divided into signature-based, heuristic-based, behavioral-based and model checking-based detection, and also can be divided further as deep learning-based, cloud-based, mobile devices-based, and IoT-based detection [2]. In each approach, the feature extraction method is different from one another. One detection approach certainly cannot be said to work better than the others, as each approach has its own advantages and disadvantages. The way the approach is used, the feature extraction and classification method affect its success. Traditional malware variants can be detected with high accuracy using the signature-based approach, but signature-based approach cannot demonstrate the same success when detecting new malware strains. A considerable amount of unknown malware samples are detected using behavioral-, heuristic-, and model-based detection approaches. However, they are insufficient to detect some new generation malware. Similarly, deep learning-, cloud- and mobile and IoT-based approaches fall short of detecting malicious software that constantly changes itself [2]. Different approaches can be combined to detect more complex as well as unknown malware. In this study, dynamic analysis is used for malware feature creation, and behavioral-based approach is used for detection.

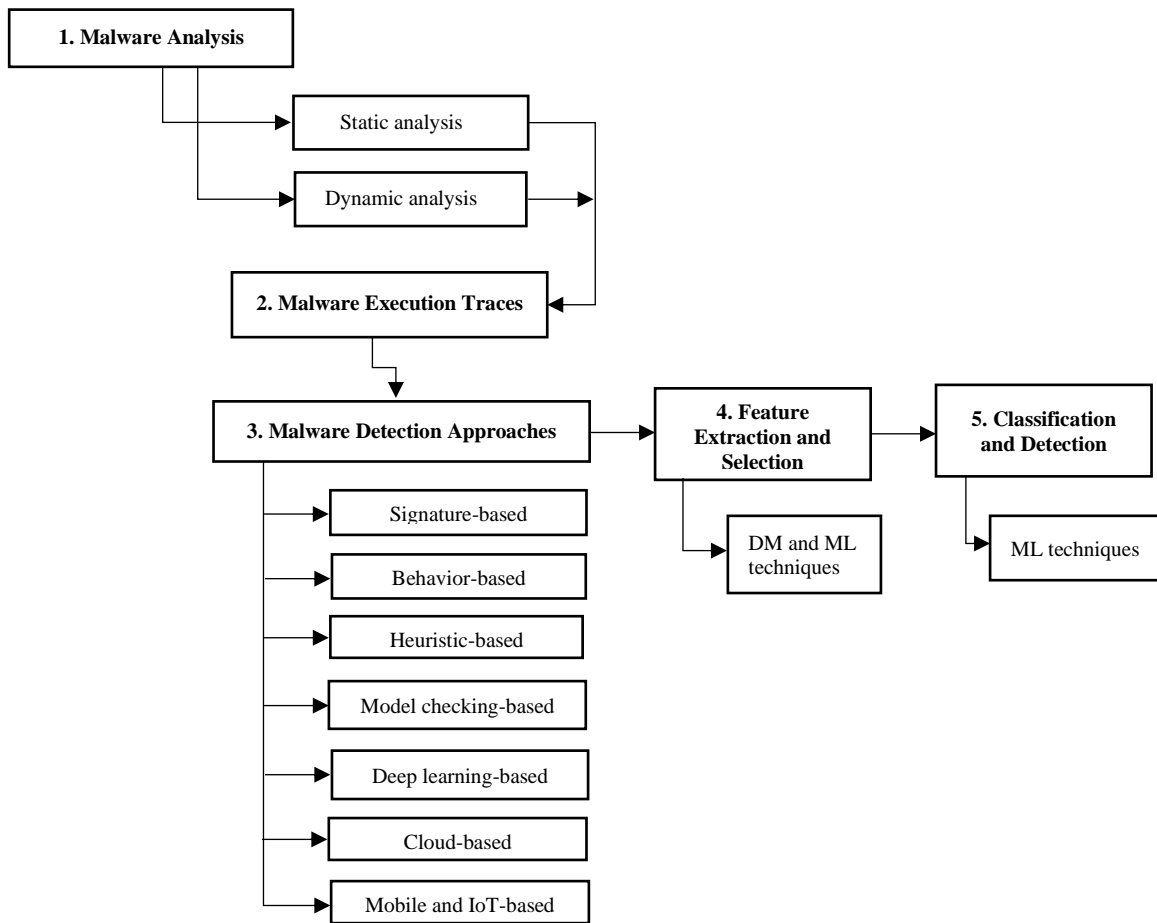


Figure 1 Malware analysis methods and detection approaches

### 3. Leading Methods in the Literature

There are several scientific papers written about malware detection. In each paper, the method used to distinguish malware from benign is different from one another. In this section, related studies are summarized based on proposed methods as well as measured performances.

Luckett *et al.* proposed a neural network based rootkit detection by using system call timing behaviors [8]. They collected system calls as well as corresponding execution times from the infected and uninfected system. They used KBeast rootkit for analysis. For training and detection, the zero is assigned for infected, 1 is assigned for uninfected. MATLAB is used to evaluate the collected features. They used two neural network architectures, namely, a static feedforward architecture, and recurrent nonlinear auto-regressive architecture, for experiments. According to the authors, the best experiment results obtained when combining a radial base and hyperbolic tangent transfer functions.

A static malware analysis technique, which uses static tools such as Bintext, PEID, PEview, MD5deep, Dependency walker, and IDA Pro as well as antivirus scanners, suggested in our previous study [9]. In the proposed method, collected malware and benign samples were analyzed under the static analysis tools with domain experts. For analyzed samples, important strings, reverse compiling results, hashes, and imported and exported functions were used to separate malware from the cleanware. The test case was performed on different versions of Windows virtual machines. As stated in the paper that antivirus scanners were fast and preferable for known malware, while static analysis tools performed better for zero-day malware

A malware detection method, which uses virtual memory access patterns was proposed by Xu *et al.* [10]. The study utilized hardware-assisted technique which monitored and classified memory access



patterns by using ML. According to the authors, malware may modify the control flow or data structures, which leaves fingerprint traits on program memory accesses. They used three markers including system calls, function calls, and the complete program run to collect the memory accesses. After the execution traces were collected, the feature selection was performed to specify the most relevant features. Finally, LR (Logistic Regression), SVM (Support Vector Machines) and RF classifiers were used for classification. As stated in the paper, the analysis rootkits were detected with 99% DR and less than 5% FPR.

Rosli *et al.* proposed a behavior-based detection method which uses registry data [11]. The suggested behavioral-based method used *k*-means clustering technique to separate malware based on the registry activities. According to the authors, the unsupervised clustering techniques are crucial in order to group the similar malware behaviors. In the proposed method, first, malware properties were extracted and chosen from the computer registry. Then, the *k*-means clustering detection model was used to separate suspicious behaviors from the normal ones. The experiment test results indicated that the proposed method clustered the normal and suspicious behaviors with more than 90% accuracy.

Bahador *et al.* presented a hardware level method which uses behavioral signatures to distinguish malware from the benign ones [12]. The proposed method used performance counter traces in order to detect and disable the malicious program samples immediately at the beginning of the execution. The paper stated that each behavior signature consists of a few number of singular values obtained from the hardware performance counter traces of known malware variants. When two performance counter traces are similar, the corresponding values should be proportional to each other. The collected malware and benign samples were analyzed under virtual machines. The test results indicated that the proposed method achieved the performance of 95.19%, 89.96%, and 92.50% for precision, recall, and f-measure, respectively.

Zhang *et al.* proposed a malware detection method which uses behavior chains [13]. Initially, the proposed method monitored the behavior points by using API calls. Then, the behavior chain was constructed by using the calling sequence of those behavior points. In the end, a LSTM (Long short-term memory) was used to specify malicious behaviors from the behavior chains. For experiment, 54.324 malware and 53.361 benign samples were collected and tested on Windows operating systems. The proposed method achieved 98.64% accuracy with less than 2% FPR.

The malware detection method for Industrial Internet of Things (IIoT) based on the behavior graph is proposed by Sun *et al.* [14]. API calls were obtained by running the malware and benign samples under Cuckoo sandbox. Then, parameter normalization and *n*-gram were applied to decrease the number of API traces. CBG (classified behavior graph) was created by selected APIs. The CBGs are the original program features which are generated for each sample. Similar behaviors were removed by using CNSG (crucial *n*-order subgraph). Then, CBGs were optimized to the key CBGs. Finally, NB, SVM, and AdaBoost classifiers were applied to key CBGs to separate malware from the benign samples. According to the papers, the proposed method achieved a high detection accuracy for tested malware and benign samples.

Azeez *et al.* proposed an ensemble learning approach to detect Windows PE malware strains [15]. The proposed approach consisted of fully connected and CNNs (Convolutional neural networks) which used the ExtraTrees classifier as a meta-learner. The base phase classification was performed by using a stacked ensemble of fully-connected and one-dimensional CNNs. In this phase for end-stage classification, machine learning (ML) classifiers were applied. Fifteen ML algorithms were used for meta-learner, and five ML algorithms including RF, NB, DT (decision tree), AdaBoost, and gradient boost were used for comparison. For experiments, Windows PE files were used. As stated in the paper, the more satisfactory results were obtained when an ensemble of seven neural networks as well as the ExtraTrees classifier for final-phase was used.

Rey *et al.* presented a federated learning at detecting malicious software samples in IoT devices [16]. They applied supervised as well as unsupervised federated methods to recognize malware variants for IoT devices. For this purpose, N-BaIoT dataset which consists of many real IoT devices' network traffic that were affected by malicious software were used. In addition, the obtained performances were

compared with participants which train a model locally (does not share the data) and participants which train a model globally (share data with the central). As mentioned in the paper, using the divergent data increases the performance of the federated models. To test the strength of the federated methods, adversarial attacks were used. According to the test results, the basic federated learning models are prone to adversarial attacks. However, their federated model is more robust to adversarial attacks. The robustness of the used federated learning model against adversarial attacks should be improved in the future.

A dynamic malware detection method which used IP (Internet protocol) reputation and ML techniques was explained by Usman *et al.* [17]. The suggested method computed the malicious behaviors of IP addresses' at run time. For this purpose, big data forensic was used to calculate the risk score of IP addresses. They used weighted risk score to identify the re-attempt of the causing damage, and confidence level to specify the degree of maliciousness. Paper stated that the experiments were carried out with a few ML classifiers including NB, SVM, MBK (mini batch k-means), DT, and best results were obtained when DT was used. The false alarm rate was high in the proposed method, in the future the false alarm rate should be reduced to improve the model performance.

Vu *et al.* suggested CNN-on-matrix technique to classify Android malware variants [18]. Each Android application was used as an image. First, for each application, the adjacency matrix was constructed. Then, constructed matrices were used as input images, and given to the CNN (Convolutional Neural Network) model. Finally, CNN model recognized each image sample as malware or benign. In this phase, the family of each malware apps were specified as well. According to the paper, the proposed approach could detect the Android apps with 94.3% when the drebin dataset was used and 97% accuracy is obtained for different malware families. The proposed method is limited with Android apps, and needs to be extended to support other mobile platforms.

In the related works, several malware detection methods were reviewed based on the suggested methods, used ML techniques and measured performances. Most of the studies used static and dynamic tools to analyze the malware. After the malware analysis stage was completed, data mining (DM) and ML techniques were applied to create malware features from the execution traces. Most of the papers were performed on specific malware variants or only a few malware samples which cannot be generalized to detect all malware types. In practice, current malware detection methods cannot detect unknown malware variants efficiently as well. The proposed method decreased the deficiencies that current studies have on malware detection and improved the detection and accuracy rates. Even though some of the existing studies results are close to the proposed method performances, the time complexity of the proposed method is lower. Besides, the proposed method can detect high percentage of the zero-day malware as well as different variants of malware including virus, worm, rootkit, ransomware, and obfuscated malware.

#### **4. Proposed Method**

We proposed a malware detection method which can be categorized in a behavioral-based approach that uses file and registry operations. In general, most of the malware performs operations on specific file directories and registry locations. This is because the majority of malware samples display similar behaviors on specific file and registry locations, which cannot be seen on benign samples. The proposed malware detection architecture seen in figure 2.

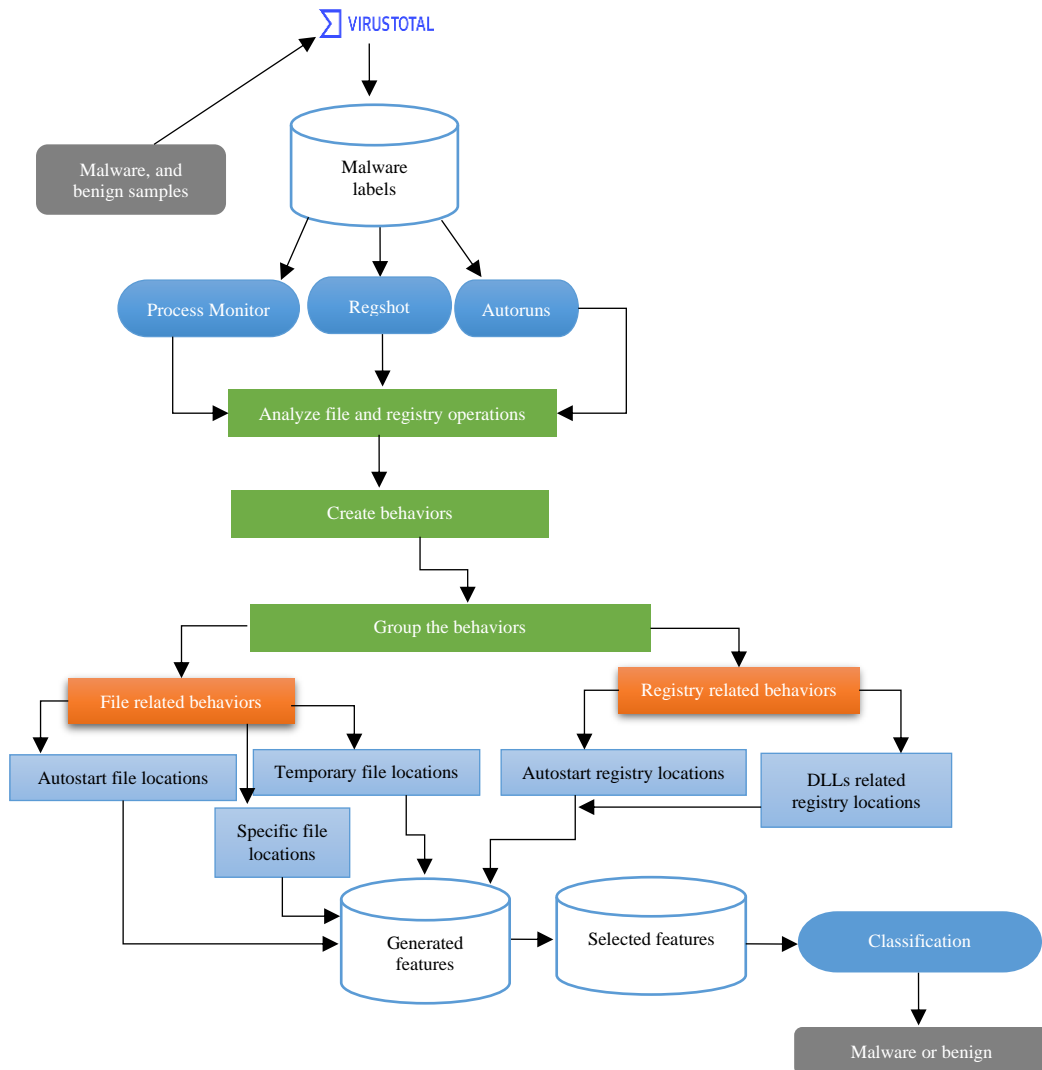


Figure 2 Malware detection architecture

Instead of all file and registry related operations, we only consider the file and registry operations which can rarely be seen in benign samples, however, mostly seen in malware samples. For instance, most of the malware types perform read and write operations to spread or inject itself into system processes or commonly used DLLs. In addition, most of the malware variants perform on specific folder locations including temporary file locations, as well as specific file and registry automatic startup (autostart) locations. When execution traces are collected following file-registry locations are considered:

1. Autostart file locations:
  - Shell:startup
  - Shell:common startup
  - %appdata%\Microsoft\Windows\Start Menu\Programs\Startup
  - C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
  - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
2. Temporary file locations:
  - %system%\Windows\Temp
  - C:\Windows\Temp
  - %userprofiles%\AppData\Local\Temp
3. Specific system file locations:
  - \Windows\System32

## 4. Autostart registry locations:

HKLM\Software\Microsoft\Windows\Currentversion\Run  
 HKLM \Software\Microsoft\Windows\Currentversion\Runonce  
 HKEY\_Local\_Machine\Software\Microsoft\Windows\Currentversion\Run  
 HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run  
 HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run  
 HKCU\Control Panel\Desktop\Scrnsave.exe  
 HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices  
 HKLM\ Software \Wow6432Node\Microsoft\Active Setup\Installed  
 HKLM\ Software \Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects  
 HKLM\ Software \Microsoft\Windows\CurrentVersion\Explorer\Shell Folders  
 HKLM\ Software\ Microsoft\Windows NT\ CurrentVersion\Winlogon\UserIni  
 HKLM\Software\Microsoft\Windows NT\CurrentVersion\Drivers32

## 5. DLLs related registry locations:

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs  
 HKLM\Software\Microsoft\Windows NT\ CurrentVersion\ Windows\AppInit\_DLLs

After the execution traces are collected based upon specific file-registry locations, behaviors and features are created. When creating behaviors from the file-registry operations, one or more operations can create behaviors on the same file and registry instance. When creating features from the behaviors, ten consecutive orders are placed and behaviors are grouped based on file and registry operations. The same behaviors on different instances of the same resources match into the same feature, but the frequency of the feature is increased. Even if the behaviors are performed on different resources including file and registry, they can create features when relation is observed based on used locations. After the feature generation process is completed, the frequency of each feature is computed.

When the feature generation process is completed, most significant features are selected by using information gain. The information gain selects the features with maximum gain which decrease the information needed for the next split. Most of the time, if the dataset is properly constructed, the information gain is chosen for the most significant features in the dataset. We can compute the information gain as follows:

$$\text{Information gain}(A) = \text{Information}(D) - \text{Information}_A(D) \quad (1)$$

$$\text{Information}(D) = -\sum_{j=1}^v p_j \log_2(p_j) \quad (2)$$

$$\text{Information}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right) \quad (3)$$

$\text{Information}(D)$  shows the average amount of information needed to identify the class labels in the dataset, while  $\text{Information}_A(D)$  indicates the amount of information needed after each partitioning during classification for features. After the most significant features are selected by using information gain feature selection criteria, the learning and testing phases are performed. For classification, well-known ML algorithms are used including C4.5 (J48), RF, SLR, AdaBoost, SMO, and KNN. We present the effectiveness of our proposed method by applying and comparing the outcomes of these ML algorithms.

## 5. Implementation

For the experiments, Windows 10 was used as a host machine with Oracle VirtualBox installed. Windows 7 and Windows 8 are installed on Oracle VirtualBox as guest machines. The collected malware and benign samples are performed on virtual machines Windows 7, 8, and 10. The malware samples are collected from several sources including ViruSign, Malshare, and Tekdefense [19, 20, 21]. The collected malware samples are from different malware types such as virus, worm, rootkit, backdoor, ransomware, spyware, and so on. The collected malware samples are labeled by using VirusTotal.

VirusTotal is a website which contains several antivirus scanners. The tested benign files are different system and third party' softwares. After each malware sample performed, the execution traces of file-registry operations are collected by using Process Monitor as well as Regshot, and Autoruns. Each time, the clean version of the guest machine is used. The collected execution file-registry traces are analyzed by using Python language in Windows 10 environment.

Totally, 582 malware and 300 benign samples are tested. After the file-registry traces are collected, behaviors are formed. Only five types of file and registry operations are used for behavior creation: autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs related registry locations. After behaviors are created, features and their frequencies are computed. During the feature creation, ten consecutive orders are used. Then, most significant features are chosen by using information gain measures. That way, the most distinctive file and registry based features are generated. To correctly classify the most distinctive features, which are mostly seen in malware but rarely seen in benign samples, ML classifiers including C4.5 (J48), RF, SLR, AdaBoost, SMO, and KNN are used.

Our dataset consists of different types of malware including virus, worm, rootkit, backdoor, Trojan, ransomware and packed malware. To label the malware samples, the VirusTotal is used which contains several antivirus scanners. To identify each malware sample, the kind of file signature MD5 (message-digest algorithm) hashes are used. For each feature, the frequency of the properties is written into the dataset. If the related feature is not given, 0 is written as a frequency.

To measure the feasibility and efficiency of the proposed method: TPR, FPR, precision, and accuracy are used on our created dataset. Confusion matrix is used to calculate these values (Table 1).

Table 1 Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

In the confusion matrix, TP shows the number of malware samples correctly labeled as malware, TN shows the number of benign samples correctly labeled as benign, FP represents the number of benign samples being accidentally labeled as malware, and FN represents the number of malware mistakenly labeled as benign. These values are used to compute the TPR, FPR, precision, and accuracy as follows:

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN}) \quad (4)$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN}) \quad (5)$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (6)$$

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (7)$$

## 6. Results and Discussion

This section shows the experiment results and evaluates the model performances. Learning and testing phases are performed on classifiers by using 10-fold cross-validation and holdout 75%, 25% split. The results are summarized in table 2, figure 3, table 3, and table 4.

Table 2 presents the various ML algorithms' performances on created malware dataset. The performances of classifiers based on TPR, FPR, and precision metrics are quite high. For example, when RF is selected as a classifier TPR, FPR, and precision measured as 98.6%, 0%, and 100%, respectively. In the same way, AdaBoost performance measures as 98.8%, 3.7%, and 98.1%, respectively. Similar performances are obtained when SLR, and J48 are used. The performances of KNN and SMO are lower

than other classifiers. Our performance results present that the proposed method can effectively separate malware from benign samples.

Table 2 Proposed method performances on different ML classifiers

Classifier	TPR (%)	FPR (%)	Precision
<b>RF</b>	<b>98.6</b>	<b>0</b>	<b>100</b>
<b>AdaBoost</b>	98.8	3.7	98.1
<b>SLR</b>	98.1	0.9	99.5
<b>J48</b>	98.1	1.8	99
<b>KNN</b>	92.6	4.2	97.9
<b>SMO</b>	88	13.3	92.8

Figure 3 shows the accuracy results on several ML algorithms. As it can be seen from the figure 3 that the best accuracies are obtained in order of RF, SLR, J48, AdaBoost, KNN, and SMO. Since, information gain is used as a feature selection, RF (99.05% accuracy), SLR (98.42% accuracy), J48 (98.11% accuracy), and AdaBoost (97.95% accuracy) classifiers performed pretty well. However, SMO classifier accuracy is measured as 87.52% which is lower than other classifiers. When we use correlation coefficients for the feature selection process, the performance of the SMO is increased up to a certain degree.

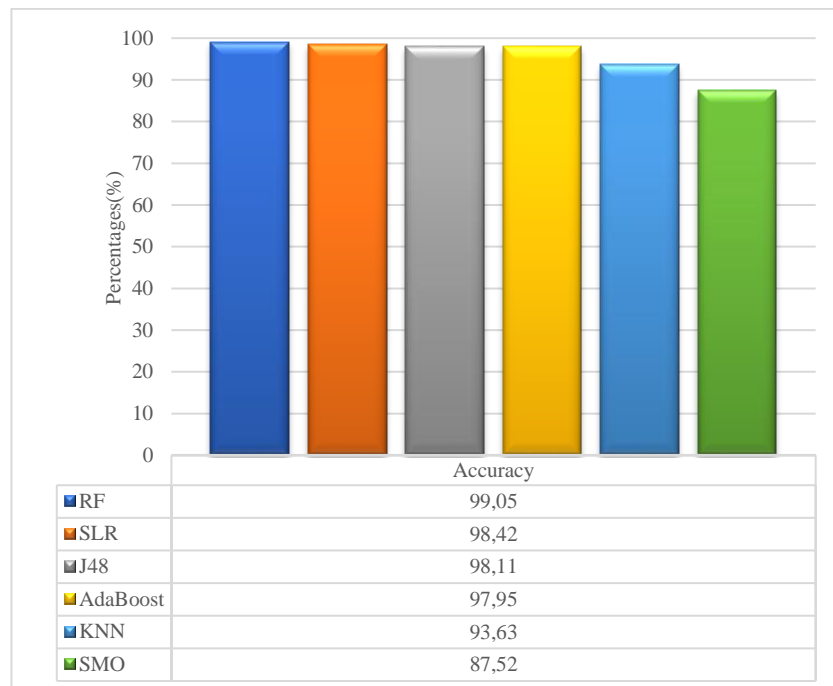


Figure 3 Various ML Classifiers accuracy on created dataset

Our dataset consists of 45 features. The table 3 shows the 27 file-registry related properties (which are generally seen in autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs related registry locations) mostly seen in malware files with high frequency and rarely seen in cleanware. Even if a few of the listed features may be seen in cleanware, the frequency of those properties are very low when compared to the frequency of properties counted in malware samples. CreateFileReadFile, ReadFileWriteFile, WriteFile, SetBasicInformationFile, CreateFileWriteFile, CreateFileMapping, QueryStandardInformationFile, QuerySecurityFile, RegOpenKey, RegQueryValue, RegSetInfoKey, RegCreateKey, RegSetInfoKeyRegEnumKey, and RegDeleteValue properties are mostly seen in malware samples with high frequencies in substantial file and registry locations.

Table 3 The list of file and registry related properties which mostly seen in malware seldomly seen in cleanware files (The order of the features are not preserved)

ReadFileLoadImage
ReadFileWriteFile
WriteFileCreateFile
WriteFile
CreateFileSetBasicInformationFile
SetBasicInformationFile
CreateFileWriteFile
WriteFileCreateFileMapping
CreateFileReadFile
QueryBasicInformationFileReadFile
QueryBasicInformationFileCreateFileMapping
CreateFileMapping
CreateFileMappingLoadImage
LoadImageReadFile
QueryBasicInformationFileQueryDirectory
CreateFileMappingCreateFile
QueryStandardInformationFile
QuerySecurityFile
RegOpenKey
RegQueryValue
RegSetInfoKey
RegSetInfoKeyRegQueryKey
RegCreateKey
RegSetValue
RegQueryKeyRegSetInfoKey
RegSetInfoKeyRegEnumKey
RegDeleteValue

Table 4 shows the comparison results of the proposed method against the state-of-the-art methods in the literature. The proposed method generated remarkable results among the other methods. For instance, the proposed behavioral-based file-registry operations performance was measured as 99.05% when RF was used as a classification algorithm (Table 4). On the other hand, system call timing behaviors performance was 82.8% and memory access patterns performance was 88.4%. The best performance obtained from the sequence of behavioral points (behavioral chains) by 98.64% which was still lower than the proposed method performance (99.05%). Besides, the proposed method's feature space is much lower than the other methods that are mentioned in table 4.

Table 4. Shows the proposed method performances against leading methods in the literature

Paper	Year	Feature Representation	ML Algorithm	Performance(%)
Lockett <i>et al.</i> [8]	2016	System call timing behaviors	Neural Network	82.8
Xu <i>et al.</i> [10]	2017	Memory access patterns	RF	88.4
Rosli <i>et al.</i> [11]	2019	Behavior-based registry data	K-Means	90
Bahador <i>et al.</i> [12]	2019	Behavioral signatures on performance counter traces	RF	82.65
Zhang <i>et al.</i> [13]	2020	Sequence of behavioral points	LSTM	98.64
Sun <i>et al.</i> [14]	2021	Classified behavior graphs	SVM	97
<b>Proposed Method</b>	<b>2022</b>	<b>Behavioral-based file-registry operations</b>	<b>RF</b>	<b>99.05</b>

## 7. Conclusion

The number, severity, and complexity of malware have been increasing rapidly. To protect the computer based systems from malware, malware needs to be detected whenever it infect the victim system. However, the detection of malware becomes harder since new malware variants are more intelligent and use obfuscation techniques to hide themselves from the malware detection systems.

This paper proposed a new behavioral-based malware detection method based on file-registry operations. Only the operations which are performed on specific file and registry locations are considered during the features generation. These features are seen frequently in malware samples rarely

seen in benign samples. Hence, most of the malware variants are detected with our approach. After features are generated, information gain is used for feature selection. Finally, several machine learning classifiers including RF, J48, AdaBoost, SLR, SMO, and KNN are used for classification. The test case is performed on Windows virtual machines 7, 8, and 10. Proposed method could effectively detect the malware with high percentages. For instance, 98.8% true positive rate, 0% false positive rate, and 99.05% accuracy are obtained to distinguish malware from cleanware. As a future study, we aim to analyze more malware and benign samples, and also examine network related features by using Wireshark.

## References

- [1] Ö. Aslan, R. Samet, "Investigation of possibilities to detect malware using existing tools," *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)* pp. 1277-1284, October 2017.
- [2] Ö. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, 8, 6249-6271, 2020.
- [3] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, 8(1), 1-22, 2018.
- [4] Ö. Aslan, R. Samet and Ö.Ö. Tanrıöver, "Using a Subtractive Center Behavioral Model to Detect Malware," *Security and Communication Networks*, 2020.
- [5] J. Nazari, "Defense and Detection Strategies against Internet Worms," Artech House, 2004.
- [6] S. Sparks and J. Butler. "Shadow walker: Raising the bar for rootkit detection," *Black Hat Japan*, 11(63), 504-533, 2005.
- [7] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware," Symantec report, August 2015, available at: <https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware.pdf>.
- [8] P. Lockett, J. T. McDonald and J. Dawson, "Neural network analysis of system call timing for rootkit detection," *Cybersecurity Symposium (CYBERSEC)* (pp. 1-6), April 2016.
- [9] Ö. Aslan, "Performance comparison of static malware analysis tools versus antivirus scanners to detect malware," In *International Multidisciplinary Studies Congress (IMSC)*, 2017.
- [10] Z. Xu, S. Ray, P. Subramanyan and S. Malik. "Malware detection using machine learning based analysis of virtual memory access patterns," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017 (pp. 169-174), March 2017.
- [11] N.A. Rosli, W. Yassin, M. A. Faizal and S. R. Selamat. "Clustering Analysis for Malware Behavior Detection using Registry Data," *International Journal of Advanced Computer Science and Applications (IJACSA)*, 10, 12, 2019.
- [12] M. B. Bahador, M. Abadi and A. Tajoddin, "HLMD: a signature-based approach to hardware-level behavioral malware detection and classification," *The Journal of Supercomputing*, 75(8), 5551-5582, 2019.
- [13] H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang and N. Chilamkurti. MALDC: "A depth detection method for malware based on behavior chains," *World Wide Web*, 23(2), 991-1010, 2020.
- [14] Y. Sun, A. K. Bashir, U. Tariq and F. Xiao, "Effective malware detection scheme based on classified behavior graph in IIoT," *Ad Hoc Networks*, 102558, 2021.
- [15] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti and R. Damaševičius, "Windows PE malware detection using ensemble learning," In *Informatics*, (Vol. 8, No. 1, p. 10). Multidisciplinary Digital Publishing Institute, 2021.
- [16] V. Rey, P. M. Sánchez, A. H. Celdrán and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, 108693, 2022.



- [17] N. Usman, S. Usman, F. Khan, M. A. Jan, A. Sajid, M. Alazab and P. Watters, "Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics," *Future Generation Computer Systems*, 118, 124-141, 2021.
- [18] L. N. Vu, and S. Jung, "AdMat: A CNN-on-matrix approach to Android malware detection and classification," *IEEE Access*, 9, 39680-39694, 2021.
- [19] [Online]. Available: <https://www.virusign.com/> [Accessed in November 2021].
- [20] [Online]. Available: <https://malshare.com/> [Accessed in November 2021].
- [21] [Online]. Available: <http://www.tekdefense.com/> [Accessed in November 2021].

## Sentiment Analysis on the Metaverse: Twitter Data

 Gulsum Akkuzu Kaya<sup>1</sup>

<sup>1</sup>Corresponding Author; Kirsehir Ahi Evran Universitesi; gulsum.akkuzukaya@ahievran.edu.tr; 5432704681

Received 15 March 2022; Revised 06 Jun 2022, Accepted 07 Jun 2022; Published online 05 January 2019

### Abstract

In recent days, the metaverse which is defined as virtual-reality space in which people can interact with each other in a computer-generated environment, has attracted people's attention. People have posted their opinions about the metaverse on social media platforms. Twitter is one of those platforms in which people have tweeted about the metaverse. Tweets help researchers to understand public attitudes on a subject. This research focuses on two analysis: The first one used sentiment analysis on Turkish tweets about various alternative words of the metaverse, such as karma evren, misal evren, ote evren, sahte dunya, sanal evren and internet otesi. The second focus of our research was on an analysis of a questionnaire that aimed to understand whether people are aware of the metaverse and willing to experience it or not. The result of sentiment analysis showed that the most of the collected tweets were positive about the collected tweets. The questionnaire analysis showed that the majority of participants were aware of the metaverse and would like to experience that virtual space.

**Keywords:** metaverse, sentiment analysis, social network, virtual reality, twitter

### 1. Introduction

Recently, a new concept of the Internet has been taking people's attention; 'Metaverse'. It is a combination of various aspects of technology such as social media, augmented reality (AR), virtual reality (VR), cryptocurrencies, and online gaming. The word 'metaverse' was first coined in a piece of speculative fiction named Snow Crash, written by Neal Stephenson in 1992 [1]. Stephenson defined the metaverse as a virtual-reality space in which people can interact with each other in a computer-generated environment. The metaverse provides an environment to its users' where they live mentally in a virtual world while being any physical environment. The connection between the real world and the metaverse is done through the Internet, in other words the signals received from people's bodies from the real world with physical equipment and transferred to the virtual world.

In this virtual world (i.e. metaverse), people have their avatars to experience to alternate life in a virtuality that is a metaphor of people's real worlds. This point of the metaverse could be looked at from two perspectives. On the one hand, the metaverse is the next version of the Internet and it is a great development for humanity. On the other hand, the metaverse has a scary point because it might cause serious issues to the future life and might be the reason for ending humanity. Based on these two aspects of the metaverse, we construct our hypothesis;

- H1: We can analyze Twitter users' opinions on the alternative words of metaverse and can help us to understand what they feel and think about it.

Carrying out surveys is a traditional way to assess public opinions and attitudes; however, this method has limitations, such as closed questions, sample sizes in which restricted number of observations used for determining the estimations of a given population, and spatio-temporal granularity. In order to overcome these limitations social media data has increasingly been used for understanding and analysing the public's viewpoint [2]. Twitter is one of the most commonly utilised social media platforms with 206 million daily active users globally. It provides real-time public discussions, attitudes, and reflections of different opinions from all over the world [3]. It is a popular social media platform not only for its users to express their actual opinions but also widely drawn upon by researchers to analyze these data.

Agrali et al. have focused on analysing Twitter data related to the metaverse by capturing users' opinions using sentiment analysis[4]. They evaluated tweets before and after Mark Zuckerberg's speech about the metaverse. Their research showed that the negative and neutral tweets increased while the positive tweets decreased.

## **2. Related Works**

An interaction technique in the metaverse was proposed by Young et al. [5], the technique made synchronised high five gestures between physical and virtual environments. Another interactive system for the metaverse and physical-world was proposed by Ariel et al. [6], use of tablets and smart-wearables was introduced. To customize virtual characters in the metaverse, user interfaces were made by Wei et al [7]. Orgaz et al. [8] used clustering approaches to understand virtual characters and their behaviors.

Above studies mainly focused on the techniques that introduce the ways to make the communication between the physical world and the virtual world (i.e. metaverse). To propose the above techniques, analysis of user feedback and user interactivity about the metaverse were used [9]. One of the analysis techniques to understand users' opinions on a subject is the sentiment analysis technique [10]. According to Cambria sentiment analysis are the key factors to develop Artificial Intelligence [3]. Because sentiment analysis incorporates not only the text but also visual contents [11]. Public opinions and sentiments are very relevant to our daily lives therefore it is a need to analyze users' opinions to understand public opinion and make decisions on a subject [12]. Twitter data is the most used data for sentiment analysis [13].

Agrali and Aydin carried out research to analyze metaverse related data in Twitter [14]. It is the only research that analyzes the metaverse related tweets. Their research showed that most of their collected tweets were classified into the positive sentiment class. They collected tweets which only include the text "metaverse" however Turkish citizens suggested words which could be alternative words to metaverse. We therefore collected tweets include the alternative words and provide a sentiment analysis on them.

## **3. Material and Method**

### **3.1. Questionnaire Analysis**

We disseminated a questionnaire over the internet. The questionnaire was designed to understand whether people are aware of the metaverse. Five hundred people respond to the questionnaire, two hundred twenty females and two hundred and two hundred eighty males. We gave a very general definition of the metaverse then we asked the respondents whether they heard about it or not. There were only fifteen of them who did not have an idea about the metaverse. Two questions were about willingness and feelings about the metaverse. We asked them how they would feel if they were able to visit distant locations in the metaverse. Seventy eight respondents found it scary while others chose the options happy or excited. The last question was about their willingness to have experiences in the metaverse. Four hundred two respondents answered these questions with the option "Yes".

The interesting point was respondents who chose "Scared" for the question "People will be able to 'visit' distant locations in The Metaverse. How would you feel about it?" and "No" for the question "Would you like to have experiences in the Metaverse?" were male.

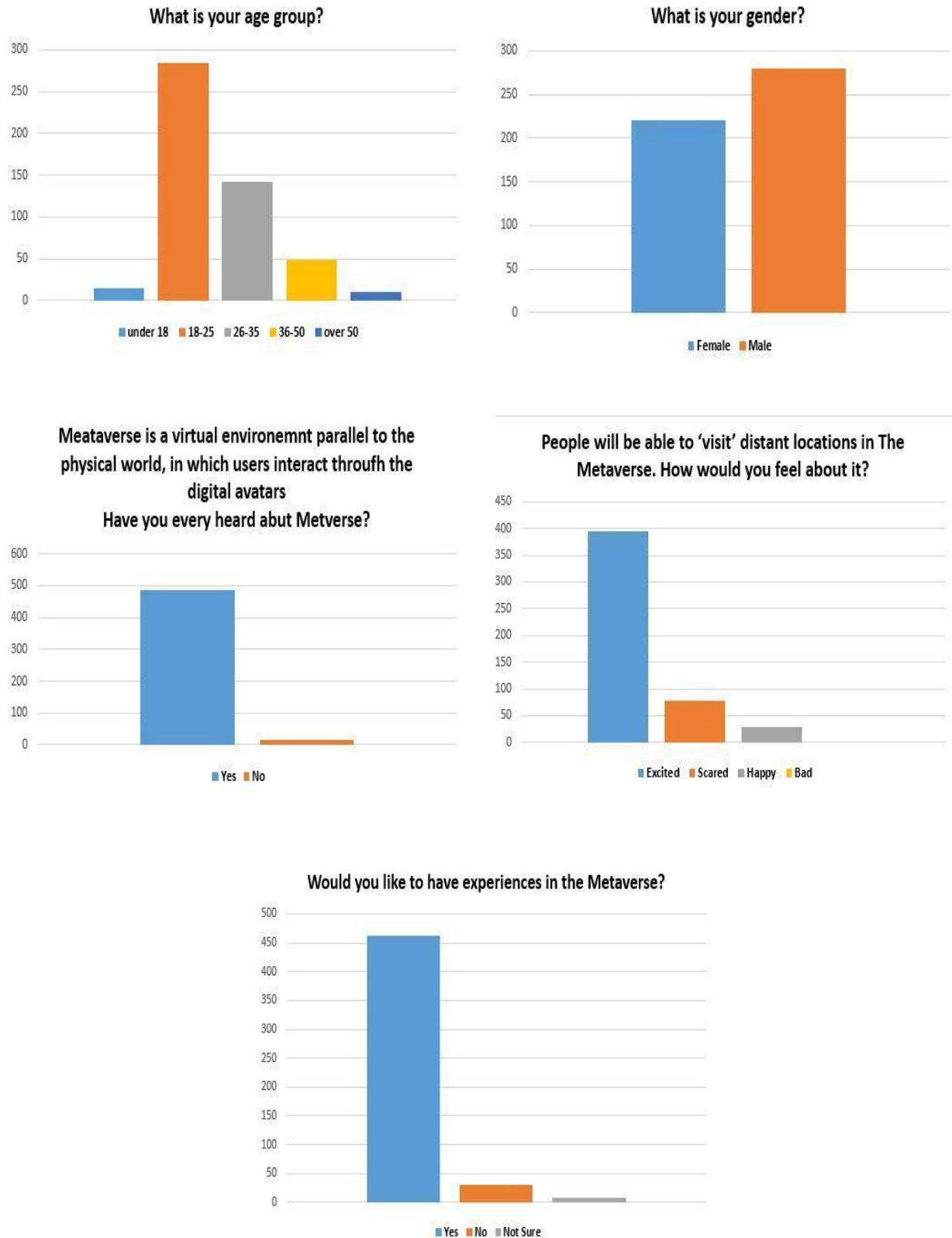


Figure 1. Results of the Questionnaire

### 3.2. Sentiment Analysis Method and Materials

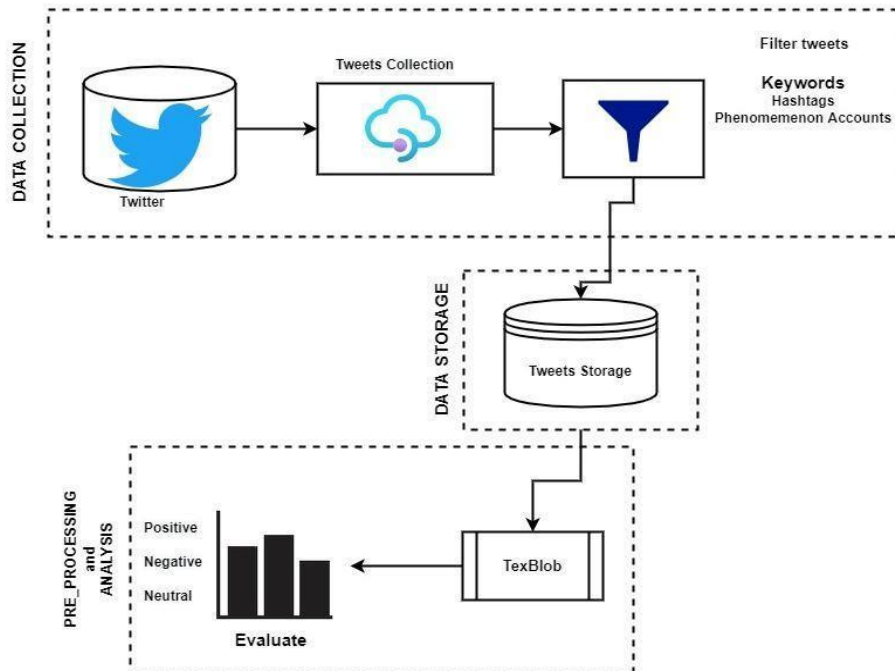


Figure 2. Structure of the methodological steps

#### Data Collection

Twitter API was used to collect metaverse related data. There were various words used to create our data set. Turkish citizens on the Twitter social media suggested words which could be alternatively used instead of metaverse. Words used to do our search on Twitter are given Table 1.

Table 1: Used Turkish words and English meanings

WORD	ENGLISH MEANING
internet ötesi	
karma evren	mixed universe
meta evren	meta universe
misal evren	instance universe
öte evren	other universe
sahte dünya	fake world
sanal evren	virtual universe

#### Pre-processing

Raw data are mostly redundant and consistent. This is because data may mean various expressions uploaded by different users on Twitter. We cleaned our data by taking following steps;

Customize pre-processing to recognise Turkish alphabet characters

Remove all URLs, hashtags, and targets

Remove all symbols, numbers, and punctuations

Remove stop words

Convert collected tweets to lower case

Remove repeated characters in a word if there is any

Perform the stemming and lemmatization processes.

Removal steps were not taken while word-clouds were created but the polarities of tweets were calculated. We used the Python library Textblob which uses Natural Language Processing (NLP) and also machine learning principles for analyzing every word in the dataset.

#### 4. THE EXPERIMENTAL STUDY

Each tweet on the dataset was labelled with one of three values; positive, negative, and neutral.

We determined the sentiment polarity of each tweet by setting the threshold value at 0.6 which determined the direction of the sentiment. Figure 3 and Figure 4 illustrate the number of sentiments on each keyword

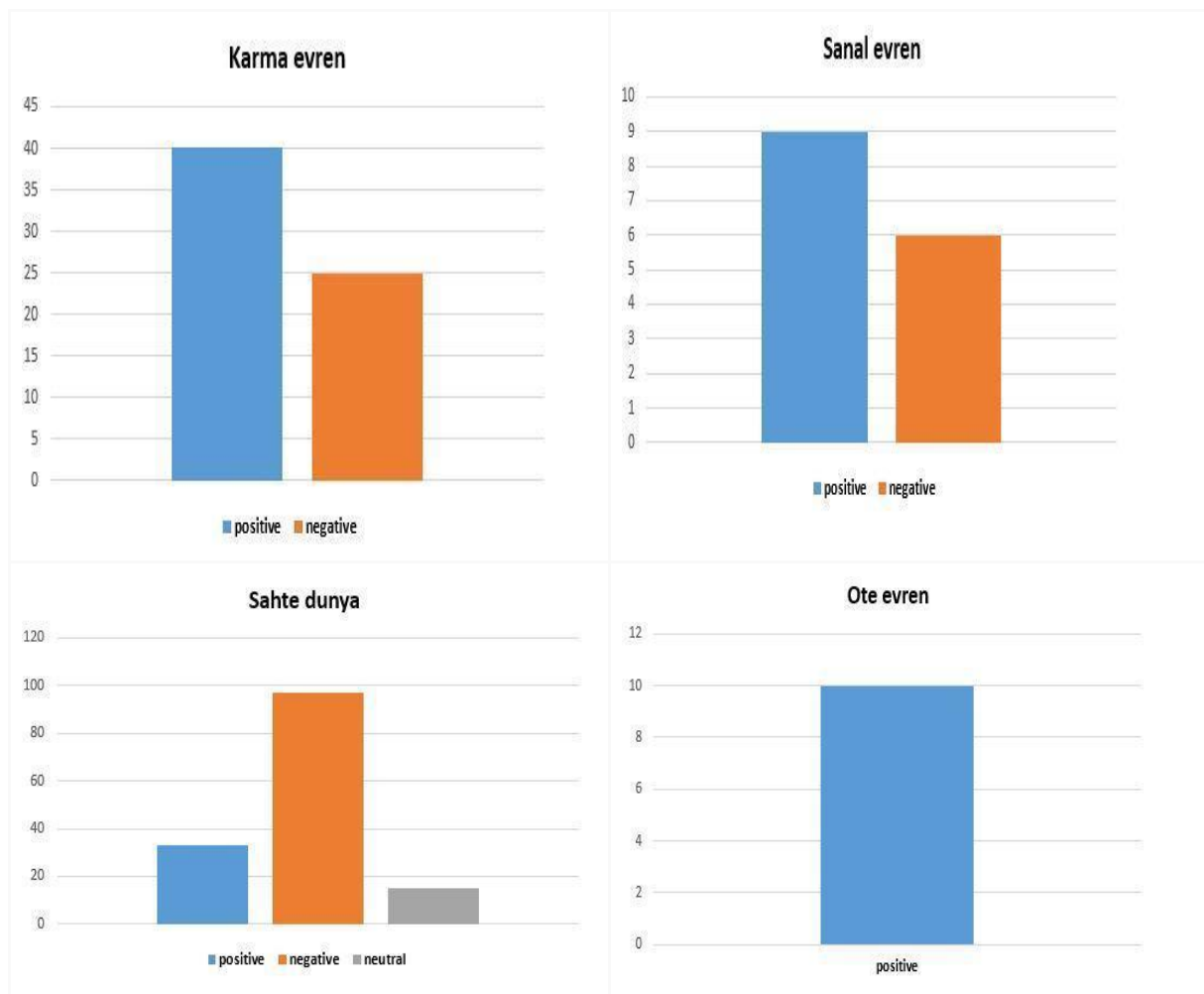


Figure 3. Sentiment analysis distribution of tweets on Karma evren and Sanal evren

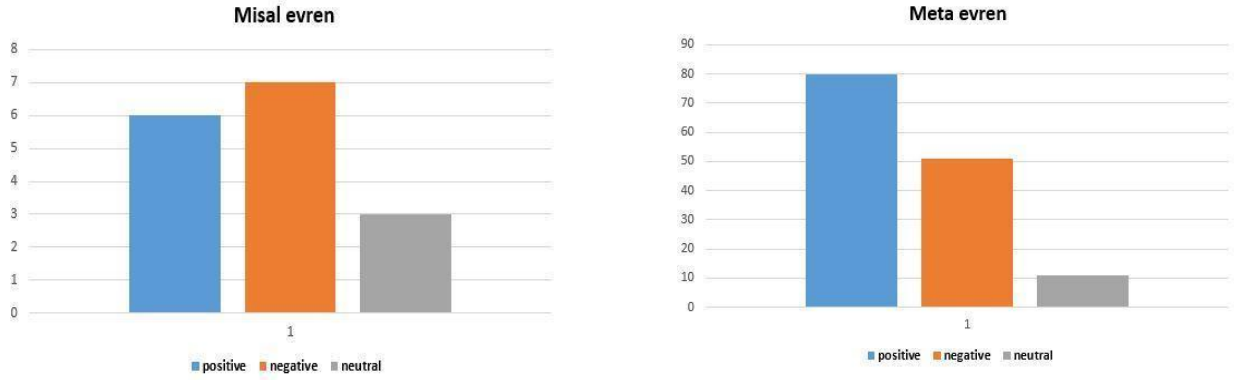


Figure 4. Sentiment analysis distribution of tweets on Sahte dünya, Ote evren, Misal evren, and MEta evren keywords

Figure 5 and Figure 6 depict the word distribution of the keywords.

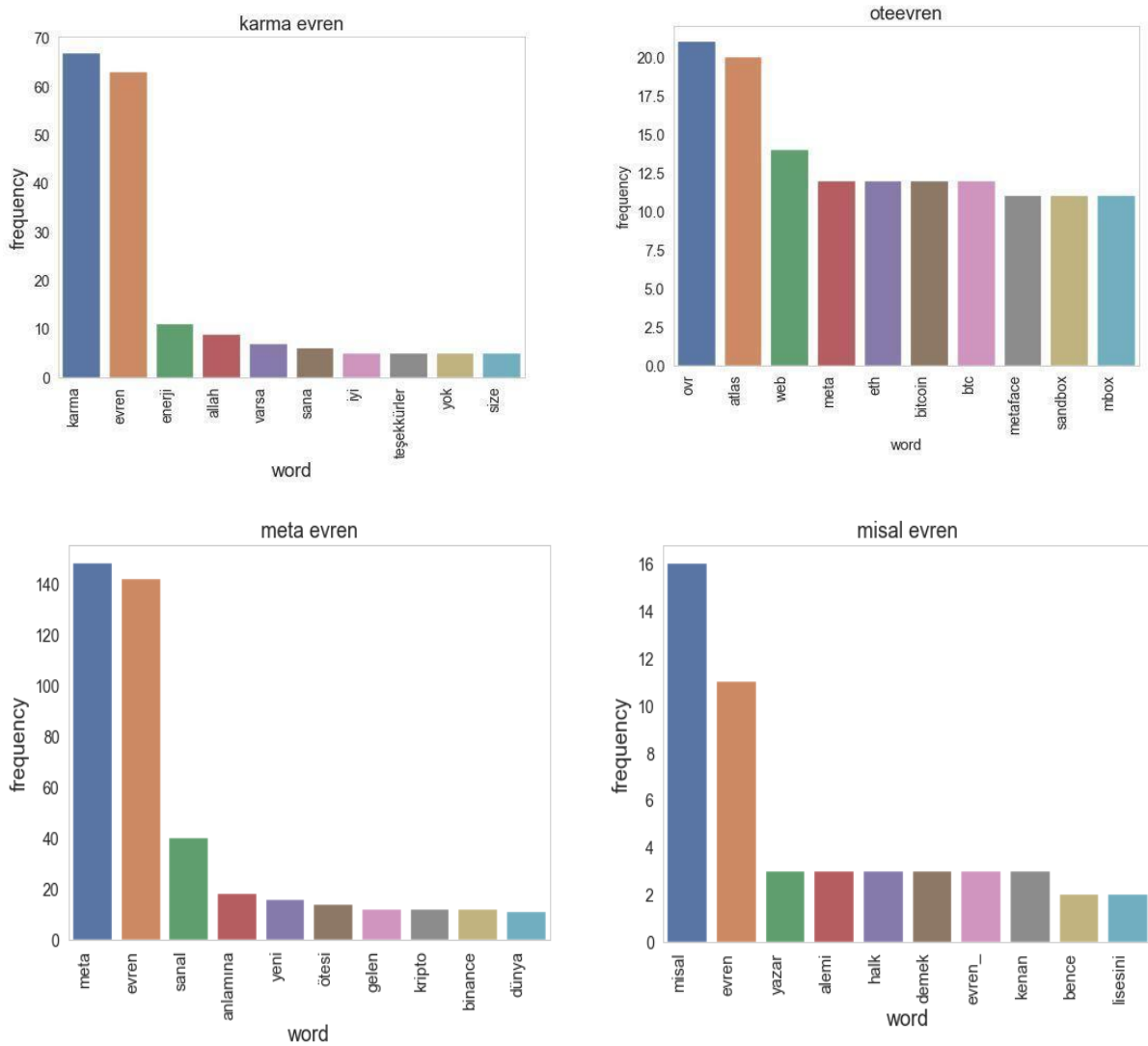


Figure 5. Word Distribution of karma evren and ote evren Keywords







Table 2: An example of Texts with Their Polarity and Polarity Score

Text	Polarity	Score
sanal gerçeklik internetin geleceği metaverse metaverse sanal gerçeklik sanal sanal evren dijitalleşme	positive	0.70
misal alemi tüm varlık için ayna görüntüsü taşıyan ve yansıtan alem maddi alemdeki evren gözle görünen her şeyin olayın şekillerini ve görüntülerini içeriyor insan hayal aracılığıyla orada gezinebiliyor	positive	0.98
gelecek zamanda hayatımızın kesin meta evren olacak teke tek bilim metaverse meta dünya	positive	0.98
gelecek zamanda hayatımızın kesin meta evren olacak teke tek bilim metaverse meta dünya	positive	0.98
infaz koruma cezaevi memuruna ek zam sanal evren olan metaverse kadar ses getirmede	negative	0.99
alemi misal sanal evren olursa kadim alemi misal lafzı bence alemi ervah olmalı dersem bir ticaret liseli olarak haddimi aşmış olur muyum ortaksoz ducane	negative	0.61
insanlık yol ayırımına doğru gidiyor sanal evren kölesi mi olacak yoksa allah'ın kulu olarak mı kalacak metaverse meta sanal	negative	0.82
evren mevren enerji enerjisi karma marma ne varsa işte belki cidden işe yarıyordur diye ne kadar yurtdışıyla ilgili tweet varsa beğeniyorum belki bir şeyler olur kim bilir kötü bir şeyler olmasın diye de tweetleri seçiyorum ona göre beğeniyorum	negative	0.66
bu uygulamalar meta evren olmasa da bir dereceye kadar benzerdir meta evren henüz hayata geçmiş değildir	neutral	0.58
ayırt edemiyorlar meta evrende bunun olmayacağı ne malum anonim kullanıcıları söylemiyorum bile bu kullanıcı tacizleri nasıl engellenecek bi çok şüphe var ama görünen o ki bi matrix evreni yaratılmak isteniyor önce coin para sonra meta evren insanoğlu evrimini tamamlıyor	neutral	0.59

## 6. Conclusion

This study presented two analysis about the metaverse. The first analysis was on the questionnaire that focused on understanding people's opinions about the metaverse and their willingness on having experiences on the metaverse events. The second analysis was on the tweets collected from Twitter. The second analysis mostly focused on using sentiment analysis methods on the Turkish tweets, which included the alternative words of the metaverse. Both analysis showed that people have positive thoughts about the metaverse. Besides that they also have concerns about it.


Whilst this work aimed at capturing Turkish people's opinions on the metaverse, it was subject to some limitations;

- The scope of keywords we defined in this study could be extended and this case, the results of analysis might well differ
- Not all Turkish people use the Twitter platform and hence the findings could differ if the same analysis was undertaken with different datasets.
- The questionnaire did not have large scope for disseminating it to different people
- The approach taken by this study might have missed some posts. There might have been posts with different hashtags.

## References

- [1] J. Judy. "Information Bodies: Computational Anxiety in Neal Stephenson's Snow Crash." *Interdisciplinary Literary Studies* 19.1 (2017): 17-47.
- [2] S. Sumit, et al. "Health monitoring on social media over time." *IEEE transactions on Knowledge and Data Engineering* 30.8 (2018): 1467-1480.
- [3] G. Akkuzu Kaya, "Sentiment Analysis of Users' Reactions to Deadly Disasters Posts in Turkey: Facebook Data" . *International Journal of Multidisciplinary Studies and Innovative Technologies* , 5 (2) , 167-172 . (2021).
- [4] Ö. Agrali, & Ö. Aydin. "Tweet Classification and Sentiment Analysis on Metaverse Related Messages". *Journal of Metaverse*, 1(1), 25-30. (2021).
- [5] K. Young Mary., J. J. Rieser, and B. Bodenheimer. "Dyadic interactions with avatars in immersive virtual environments: High fiving." *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception*. 2015.
- [6] A. Vernaza, V. Ivan Armuelles, and I. Ruiz. "Towards to an open and interoperable virtual learning environment using Metaverse at University of Panama". In 2012 Technologies Applied to Electronics Teaching (TAEE), pages 320–325. (2012).
- [7] W., Yungang, et al. "The Design of a Visual Tool for the Quick Customization of Virtual Characters in OSSSL." *2015 International Conference on Cyberworlds (CW)*. IEEE, 2015.
- [8] O., Gema Bello, et al. "Clustering avatars behaviors from virtual worlds interactions." *Proceedings of the 4th International Workshop on Web Intelligence & Communities*. 2012.
- [9] D., Haihan, et al. "Metaverse for social good: A university campus prototype." *Proceedings of the 29th ACM International Conference on Multimedia*. 2021.
- [10] C., Iti, et al. "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges." *Information Fusion* 44 (2018): 65-77.
- [11] S. Rada, J. Fernando, and C. A. Iglesias. "Social context in sentiment analysis: Formal definition, overview of current trends and framework for comparison." *Information Fusion* 52 (2019): 344-356.
- [12] L., Lizhen, X. Nie, and H. Wang. "Toward a fuzzy domain sentiment ontology tree for sentiment analysis." *2012 5th International Congress on Image and Signal Processing*. IEEE, 2012.
- [13] R., Abdur, M. Sadat, and S. Siddik. "Sentiment Analysis on Twitter Data: Comparative Study on Different Approaches." *International Journal of Intelligent Systems & Applications* 13.4 (2021).
- [14] A., Waseem, et al. "Topics, Sentiments, and Emotions Triggered by COVID-19-Related Tweets from IRAN and Turkey Official News Agencies." *SN Computer Science* 2.5 (2021): 1-19.
- [15] J., Hyun-joo, et al. "Blockchain and AI Meet in the Metaverse." *Advances in the Convergence of Blockchain and Artificial Intelligence* (2022): 73.

# Effects of Neighborhood-based Collaborative Filtering Parameters on Their Blockbuster Bias Performances

 Emre Yalcin<sup>1</sup>

<sup>1</sup>Computer Engineering Department, Sivas Cumhuriyet University; eyalcin@cumhuriyet.edu.tr

Received 31 January 2022; Revised 26 April 2022; Accepted 15 June 2022; Published online 31 August 2022

## Abstract

Collaborative filtering algorithms are efficient tools for providing recommendations with reasonable accuracy performances to individuals. However, the previous research has realized that these algorithms propagate an undesirable bias in favor of blockbuster items in their recommendations, resulting in recommendation lists dominated by such items. As one most prominent types of collaborative filtering approaches, neighborhood-based algorithms aim to produce recommendations based on neighborhoods constructed by considering similarities between users/items. Therefore, the utilized similarity function and the size of the neighborhoods are critical parameters for their recommendation performances. This study considers three well-known similarity functions, i.e., Pearson, Cosine, and Mean Squared Difference, and varying neighborhood sizes and observes how they affect the algorithms' blockbuster bias and accuracy performances. The extensive experiments conducted on two benchmark data collections conclude that as the size of neighborhoods decreases, these algorithms generally become more vulnerable to blockbuster bias while their accuracy increases. The experimental works also show that using the Cosine metric is superior to other similarity functions in producing recommendations where blockbuster bias is treated more. However, it leads to having unqualified recommendations in terms of predictive accuracy as they are usually conflicting goals.

**Keywords:** Recommender systems, neighborhood-based collaborative filtering, blockbuster bias, similarity function, neighborhood size.

## 1. Introduction

With the increasing Internet usage in recent years, individuals are inevitably faced with a vast amount of available information, making their decision-making process more complicated as they cannot find relevant services/products. Recommender systems (RSs) are highly-effective intelligent devices to cope with such an information overload problem [1]; since they aim to guide individuals by suggesting a list of preferable contents that are filtered out based on their preferences in the past [2]. Due to their significant advantages for both business and user sides, they have become more widespread in many digital systems on the Internet in different areas such as music<sup>1</sup>, e-commerce<sup>2</sup>, hotel accommodation<sup>3</sup>, movies<sup>4</sup>, etc.

In a typical recommendation scenario, a user (also called the active user) requests from the RS a numerical prediction for an item (also called the target item) untasted by himself or a ranked recommendation list containing preferable items. Researchers have recently introduced several methods for these recommendation tasks, such as collaborative filtering (CF) [3], content- or demographic-oriented filtering [4], or hybrid strategies [5]. CF techniques are the most prevalent among these methods as they are highly effective in achieving accurate recommendations. Also, studies have long been focusing on enhancing these algorithms in quantitative terms such as scalability, coverage, and accuracy. Finally, according to the following mechanism, CF techniques are commonly classified as memory- or model-based. While former methods usually provide recommendations based on similarities between users/items, the latter construct a model of the preference data for producing recommendations [2].

---

<sup>1</sup> <https://spotify.com/>

<sup>2</sup> <https://www.ebay.com/>

<sup>3</sup> <https://www.booking.com/>

<sup>4</sup> <https://www.netflix.com/>

As a prominent type of memory-based CF approaches, k-nearest neighbor (kNN) CF algorithms assume that people who have similar tastes in the past will show similar behaviors in the future [6]. Based on this assumption, they provide recommendations based on the neighborhoods constructed with the most similar users (also known as the user-based kNN) or items (also known as the item-based kNN) by performing a user-item rating matrix that includes the past choices of individuals on items [3]. The recommendation process of kNN CF algorithms is usually a two-step. Initially, it is located like-minded individuals called neighbors, and then a prediction score is computed based on the past preferences of neighbors on the target item. Also, it is a known phenomenon that their general success is firmly bound to the phase of neighborhood formation [7]. Therefore, the previous research has verified that the utilized similarity function and considered neighborhood size become vital parameters in properly locating neighborhoods, and the accuracy performance of the kNN algorithms is strongly related to how the tuning of such parameters [8]–[11].

CF algorithms are generally evaluated based on their accuracy performances. However, recent research on RSs has realized that CF algorithms are strongly biased towards popular and highly-liked items, also called the blockbuster items [12], [13], in their produced referrals due to their internal mechanism or imbalances in the rating data. This issue leads to having ranked lists where such a few blockbuster items in the catalog have appeared too often, while other vast items can not get the deserved chance even when they might be desirable for users. Unfortunately, such blockbuster bias propagation of the CF algorithms leads to low-qualified recommendations for beyond-accuracy dimensions like coverage and diversity [12]. In addition, this bias leads to having a system where unfair competition occurs, as the products of different providers are not equally treated. Moreover, this issue makes the system more unguarded to shilling attacks or social bots of malicious stakeholders to increase the visibility of their products and thus sale rates. Therefore, recent research on RSs has aimed to profoundly investigate the impacts of such a bias against blockbuster or popular items in recommendations and develop beneficial treatment approaches to counteract its adverse effects [13]–[15].

The presented study comprehensively evaluates how the blockbuster bias propagated by the kNN algorithms differentiates based on their parameter tuning. In the following, we summarize the main contributions of our study.

1. We consider three famous similarity functions: Pearson Correlation Coefficient, Cosine Similarity, and Mean Squared Difference Similarity. We observe how they affect the blockbuster bias of two prominent kNN CF methods, i.e., user- and item-based kNN, via an adopted efficient blockbuster bias evaluation protocol on two real-world datasets.
2. We also consider varying neighborhood sizes when applying kNN algorithms and investigate how they are related to the blockbuster bias in produced recommendations.
3. In addition, we analyze how these parameters of kNN algorithms affect the quality of the provided recommendation lists in terms of predictive accuracy.

We organize the remaining of this paper as follows: Section 2 gives a literature review on bias issues, especially those towards blockbuster items, in RSs. Section 3 gives some background information about our study, including the working mechanism of the kNN CF algorithms and blockbuster items. Section 4 presents the experimental studies realized to analyze how blockbuster bias performance of the kNN CF algorithms changes based on their parameter-tuning. Finally, Section 5 concludes the presented work and introduces our future directions.

## 2. Related work

In recent years, one of the main concerns of RSs has been exploring bias issues in recommendations, such as position [16], selection [17], conformity [18], and popularity [19], and treating their adverse effects on recommendation quality [14], [20], [21].

Popularity bias is the most prominent among such bias types, and it is known as the intrinsic tendency of recommendation algorithms to recommend popular items too frequently while not giving the unpopular ones enough chance [19]. Therefore, previous research has primarily aimed at exploring the

degree of popularity bias induced by different recommendation strategies for different areas like music [22], movies [23], and online education [24]. Also, several studies attempt to explore how the parameter-tuning of some CF algorithms affects their popularity bias performance [19], [24]. Besides, several previous research attempts to develop efficient procedures to achieve more qualified referrals by treating this problem. The existing popularity bias treatment approaches are usually classified as pre-processing, in-processing, and post-processing [21], according to how they are involved in the phase of recommendation generation. More specifically, pre-processing methods aim to decrease the degree of imbalances in the original rating matrix where algorithms are trained [25]. The methods of in-processing try to modify the mechanism of the recommendation algorithms for achieving recommendations where popularity bias is treated [26]. Finally, post-processing methods create new recommendation lists or resort products in the produced ranked lists [21], [27].

In a recent study [12], the authors have evaluated item popularity from a different perspective and hypothesized that the popularity of a product does not always mean that it is strongly preferable for users or vice versa. Therefore, they consider blockbuster items, both popular and highly-liked by users, and show that some well-known recommendation algorithms, including neighborhood-based CF ones, are unfortunately biased in favor of such blockbuster items in generated recommendations. In other words, they have introduced a new bias type, referred to as blockbuster bias, in recommendations. To achieve more diverse recommendations by mitigating this bias issue, they have also introduced an efficient post-processing method that motivates re-sorting the produced ranked lists by penalizing blockbuster items [13]. However, more explorative analyses of blockbuster bias in recommendations are required by considering the parameters of the CF algorithms.

Considering neighborhood-based algorithms are the most used CF methods, many previous studies have analyzed the parameterization of these algorithms on the success of recommendations [8], [9], [11]. However, these studies usually consider predictive accuracy and examine how the similarity function and neighborhood size affect the accuracy performances of the algorithms. Such analyses are also performed for different types of RSs, such as multi-criteria [28] and group recommender systems [29].

However, to the best of our knowledge, there is no study investigating how such parameters of neighborhood-based algorithms influence their bias issues, especially those towards blockbuster items. Therefore, this study mainly aims to elaborate on how the blockbuster bias of the neighborhood-based CF methods changes according to their parameter-tuning.

### 3. Preliminaries

This section introduces background information on the kNN CF algorithms, similarity functions, and blockbuster items.

#### 3.1. The kNN CF algorithms

In traditional RSs, the kNN algorithms are the most prominent approaches to providing referrals to individuals. They operate a user-item rating matrix that contains preference information from  $n$  users to  $m$  items. Such preferences are mostly numerical values in a specified rating scale or sometimes binary ratings such as like or dislike, depending on the choices of the service providers. During an online interaction with an RS, an active user ( $\mathbf{a}$ ) requests a prediction value for a target item ( $\mathbf{q}$ ) after sharing her available preferences. The kNN algorithms perform the prediction estimation process in two steps: (i) initially determining neighbors by calculating correlations/similarities between  $\mathbf{a}$  and all other available individuals in the system, and then (ii) computing a prediction value as a weighted average based on neighbors' ratings on  $\mathbf{q}$ . Such correlations between  $\mathbf{a}$  and any user  $\mathbf{u}$  can be computed using various methods referred to as similarity functions [8], [30]. In this study, we consider three famous similarity functions explained in detail in the following.

- **Pearson Correlation Coefficient (Pearson):** This similarity function is a type of correlation coefficient that represents the linear relationships between two variables measured on the same ratio scale [31]. Also, the Pearson measures the strength of the association between two

continuous variables. More formally, in RSs domain, it calculates the similarity ( $w_{au}$ ) between  $\mathbf{a}$  and any user  $\mathbf{u}$ , as in the formula given in Equation 1.

$$w_{au} = \frac{\sum_{i \in I_{au}} (r_{ai} - \bar{r}_a)(r_{ui} - \bar{r}_u)}{\sqrt{\sum_{i \in I_{au}} (r_{ai} - \bar{r}_a)^2} \sqrt{\sum_{i \in I_{au}} (r_{ui} - \bar{r}_u)^2}} \quad (1)$$

where  $r_{ai}$  and  $r_{ui}$  denote the ratings for the item  $\mathbf{i}$  by users  $\mathbf{a}$  and  $\mathbf{u}$ , respectively. Similarly,  $\bar{r}_u$  and  $\bar{r}_a$  demonstrate the average votes of  $\mathbf{u}$  and  $\mathbf{a}$ , respectively, and  $I_{au}$  indicates the set of co-voted items by both  $\mathbf{a}$  and  $\mathbf{u}$ .

To calculate the similarity value ( $w_{qt}$ ) between  $\mathbf{q}$  and any item  $\mathbf{t}$ , the Pearson function can be modified as in the formula given in Equation 2.

$$w_{qt} = \frac{\sum_{u \in U_{qt}} (r_{uq} - \bar{r}_q)(r_{ut} - \bar{r}_t)}{\sqrt{\sum_{u \in U_{qt}} (r_{uq} - \bar{r}_q)^2} \sqrt{\sum_{u \in U_{qt}} (r_{ut} - \bar{r}_t)^2}} \quad (2)$$

where  $r_{uq}$  and  $r_{ut}$  are the votes for  $\mathbf{u}$  on items  $\mathbf{q}$  and  $\mathbf{t}$ , respectively, and  $\bar{r}_q$  and  $\bar{r}_t$  denote the average votes of  $\mathbf{q}$  and  $\mathbf{t}$ , respectively. The  $U_{qt}$  denotes the set of individuals who rated both  $\mathbf{q}$  and  $\mathbf{t}$ .

- **Cosine similarity (Cosine):** In traditional settings, the Cosine metric calculates the cosine of the angle observed between two vectors in n-dimensional [9]. It can be considered the dot product of such vectors divided by the product of their magnitudes or lengths. Therefore, the Pearson explained above can be considered as centered cosine similarity. More formally, in an RS scenario, it calculates the similarity ( $w_{au}$ ) between  $\mathbf{a}$  and any user  $\mathbf{u}$ , as in the formula given in Equation 3.

$$w_{au} = \frac{\sum_{i \in I_{au}} r_{ai} r_{ui}}{\sqrt{\sum_{i \in I_{au}} r_{ai}^2} \sqrt{\sum_{i \in I_{au}} r_{ui}^2}} \quad (3)$$

or the similarity value ( $w_{qt}$ ) between  $\mathbf{q}$  and any item  $\mathbf{t}$ , as in the formula given in Equation 4.

$$w_{qt} = \frac{\sum_{u \in U_{qt}} r_{uq} r_{ut}}{\sqrt{\sum_{u \in U_{qt}} r_{uq}^2} \sqrt{\sum_{u \in U_{qt}} r_{ut}^2}} \quad (4)$$

- **Mean Squared Difference (MSD):** It is based on the geometrical principles of the well-known Euclidean distance metric [8] and computes the similarity ( $w_{au}$ ) between  $\mathbf{a}$  and any user  $\mathbf{u}$ , as in the following.

$$w_{au} = \frac{1}{\left(\frac{1}{|I_{au}|} \sum_{i \in I_{au}} (r_{ai} - r_{ui})^2\right) + 1} \quad (5)$$

or the similarity value ( $w_{qt}$ ) between  $\mathbf{q}$  and any item  $\mathbf{t}$ , as in the formula given in Equation 6.

$$w_{qt} = \frac{1}{\left(\frac{1}{|U_{qt}|} \sum_{u \in U_{qt}} (r_{uq} - r_{ut})^2\right) + 1} \quad (6)$$

where  $+1$  is utilized to shirk dividing by zero, which is the case of any co-rated item set not being constructed.

After computing similarities between users (or items), the most similar  $k$  users (or items) are labeled as neighbors. The user-based kNN algorithm produces a prediction for  $\mathbf{a}$  on  $\mathbf{q}$ , denoted with  $\mathbf{p}_{aq}$ , as a weighted average of the ratings of active user's neighbors on  $\mathbf{q}$ , using the formula given in Equation 7.

$$\mathbf{p}_{aq} = \bar{r}_a + \frac{\sum_{u \in k} [(r_{uq} - \bar{r}_u) \times w_{au}]}{\sum_{u \in k} w_{au}} \quad (7)$$

or the item-based kNN estimates the  $\mathbf{p}_{aq}$  value as the weighted average of the ratings of the target item's neighbors on  $\mathbf{q}$ , as in Equation 8.

$$\mathbf{p}_{aq} = \bar{r}_q + \frac{\sum_{j \in k} [(r_{uj} - \bar{r}_j) \times w_{qj}]}{\sum_{j \in k} w_{qj}} \quad (8)$$

where  $w_{au}$  represents the similarity weight between the user  $\mathbf{a}$  and  $\mathbf{u}$  for the user-based kNN algorithm. Similarly, for the item-based kNN algorithm,  $w_{qj}$  indicates the similarity weight between items  $\mathbf{q}$  and  $\mathbf{j}$ .

### 3.1. Definition of blockbuster items

In RSs, the blockbuster term is used to describe items that are both popular (i.e., evaluated by many users) and highly liked (i.e., evaluated with high ratings) at the same time. The literature has verified that recommendation algorithms are biased toward such blockbuster items and therefore produce recommendation lists a few blockbuster items have dominated [12]. On the other hand, many other items in the catalog are under-represented in the produced recommendation lists. Although recommending such blockbuster items seem to be desired for satisfying users, it can negatively affect the system's overall success for some critical aspects. For example, featuring only blockbuster items makes it difficult to discover new items and, therefore, negatively affects getting diverse recommendations. Also, any system subject to blockbuster bias might lack opportunities to discover more obscure items, resulting in a system where a few large service providers or well-known products have dominated. Therefore, such a system becomes more homogeneous; thus, it offers fewer options for creativity and diversification.

Recent studies have analyzed potential bias issues in favor of blockbuster items in generated recommendations [12] and tried to alleviate its adverse effects to achieve more qualified recommendations on beyond-accuracy sides such as coverage, diversity, and novelty [13]. In doing so, their primary concern is how blockbuster items should be defined and formulated. A relevant study has proposed a practical formulation that labels whether a product/item is blockbuster or not based on the characteristics of its received votes. In this study, we have adopted this strategy in analyzing blockbuster bias performances of the kNN algorithms.

This strategy mainly aims to calculate the blockbuster level for the item by incorporating its popularity and liking-degree through harmonic combination. Suppose that  $r_{ui}$  is the rating value of user  $u$  on the item  $i$ , it initially determines a popularity value for the  $i$  referred as to  $P_i$  by considering total number of individuals who rated  $i$ , and a liking-degree value referred as to  $ld_i$  for  $i$ , as in the following.

$$ld_i = \frac{\sum_{u \in U_i} r_{ui}}{|P_i|} \quad (9)$$

where  $U_i$  is the set of users who provided a rating for  $i$ .

Accordingly, the obtained  $P_i$  values inevitably vary a broader interval when compared to  $ld_i$ . For the former, the maximum possible value corresponds to the total number of users, while for the latter, it equals the highest rating score in the used rating scale. Such differences in observed values require a normalization process to scale them on the same interval before operating a combination process. To this end, this strategy firstly transform  $P_i$  and  $ld_i$  values into  $[0, 1]$  interval through well-known min-max normalization, and then incorporates the normalized  $\bar{P}_i$  and  $\bar{ld}_i$  using a harmonic combination procedure to calculate  $B_i$  scores that indicate the blockbuster levels of the items, as in Equation 10. The followed harmonic combination strategy is also helpful in balancing the potential trade-off between such



two properties of items as they might be conflicting properties in some circumstances. In other words, the obtained  $B_i$  scores properly reflect such two features of the products.

$$B_i = \frac{2 \times \overline{P_i} \times \overline{ld_i}}{\overline{P_i} + \overline{ld_i}} \quad (10)$$

#### 4. Evaluating how kNN parameters affect their blockbuster bias performance

This section presents and discusses the observed outcomes of extensive experimental studies performed to observe how the parameters of the kNN algorithms affect the blockbuster bias of these algorithms. The following sections give detailed information about the used data collections, evaluation metrics, experimental findings, and discussions.

##### 4.1. Datasets

We have used two real-world publicly-available benchmark data collections in the conducted trials, namely MovieLens-100K (MLP) and MovieLens-1M (MLM), released by GroupLens Research Team<sup>5</sup>. Both include users' preferences for movies, and the ratings are discrete and on a five-star rating scale. Table 1 presents the properties of the MLP and MLM datasets.

Table 1 Details of used datasets

Dataset	Number of Users	Number of items	Number of Ratings	Density (%)
MLP	943	1,682	100,000	6.3
MLM	6,040	3,952	1,000,209	4.3

##### 4.2. Evaluation protocols

To analyze the blockbuster bias of the kNN algorithms, we have adopted *Blockbuster Recommendation Frequency* (BRF) metric that has recently been proposed to measure how much blockbuster items overwhelm the generated top- $N$  ranked lists [13]. When using the BRF metric, it is required to categorize items in the catalog as the blockbuster or not. To this end, we sort items in descending order according to their blockbuster level scores calculated using the formula given in Equation 10, and split them into two different classes, i.e., *head* and *tail*, through the well-known Pareto principle [32]. Hence, items in the *head* class are the most blockbuster, which have received 20% of all ratings in the dataset. On the other hand, the tail class contains the remaining underappreciated items in the catalog.

After determining *head* and *tail* item classes, the BRF measures the blockbuster bias degree of a recommendation algorithm in a two-step process: (i) It initially counts how many times *head* items have been observed in generated top- $N$  lists, and then (ii) calculates its ratio to the total number of recommended items. Therefore, higher BRF results indicate more unwanted blockbuster bias and worse top- $N$  lists in terms of beyond-accuracy perspectives.

More formally, assume that  $\mathbb{N}$  is the multiset of the recommendations generated by stacking top- $N$  ranked lists provided for each individual by a proper recommendation method. The BRF score of the employed method is estimated as in Equation 11.

$$BRF = \frac{\sum_{i \in \mathbb{N}} \mathbb{1}(i \in H)}{|\mathbb{N}|} \quad (11)$$

where  $H$  is the set of items in the *head* class.

To better understand how the BRF metric works, we provide a toy example in the following. Assume that  $\{i_1, i_2, \dots, i_5\}$  is the set of available items in the system, and  $i_1$  and  $i_5$  are the items classified as the *head*. Also, suppose that there exist two available individuals and generated top-3 lists through any

<sup>5</sup> <http://www.grouplens.org/>

algorithm for them are  $\{i_4, i_2, i_1\}$  and  $\{i_1, i_5, i_2\}$ . For such a recommendation scenario, the multiset of top- $N$  lists, i.e.,  $\mathbb{N}$ , is constructed as  $\{i_4, i_2, i_1, i_1, i_5, i_2\}$ . Thus, the BRF value for the employed algorithm is computed as the ratio of how many times the items in the *head* class, i.e.,  $i_1$  and  $i_5$ , have appeared in the  $\mathbb{N}$  to the size of the  $\mathbb{N}$ , which is equivalent to  $3/6=0.5$ . This calculated BRF value demonstrates that half of the recommended items are in the *head* class, and this observation concludes that an unwanted bias towards blockbuster items has occurred in the recommendations.

In the evaluation phase, we also measure the accuracy performances of the kNN algorithms via the normalized Discounted Cumulative Gain (nDCG) metric [13], [33], which is widely used in previous research on RSs. It is a metric aimed at measuring the quality level of the suggested items by considering their actual ratings and their positions in the produced top- $N$  recommendation lists. Suppose that  $r_{ui}$  is the actual vote of user  $u$  on the item  $i$  and  $\{i_1, i_2, \dots, i_N\}$  is the produced top- $N$  item list for  $u$ , then the Discounted Cumulative Gain (DCG) and nDCG for that user are calculated as in Equations 12 and 13, respectively.

$$DCG_N^u = r_{u,i_1} + \sum_{n=2}^N \frac{r_{u,i_n}}{\log_2(n)} \quad (12)$$

$$nDCG_N^u = \frac{DCG_N^u}{IDCG_N^u} \quad (13)$$

where  $IDCG_N^u$  is the maximum possible gain for user  $u$ , and it is observed by re-sorting  $N$  items to achieve the perfect order for  $u$  based on her actual ratings. Note that higher nDCG scores mean more accurate recommendation lists.

### 4.3. Experimentation strategy

As the experimentation methodology, we have followed the well-known all-but-one strategy. Accordingly, we consider one of the users in the original data as the test user and the remaining ones as the train set. For each item of the test user, we produce a prediction value applying the user- or item-based kNN algorithm on the train set and then select the top-10 items with the highest prediction scores as the recommendation list for the test user. This process is repeatedly performed for each user in the data collection. When applying the user- or item-based kNN algorithm, we also consider different similarity measures and neighborhood sizes to monitor how they affect the recommendation quality in terms of blockbuster bias in recommendations. Finally, we measure the quality of the top-10 lists produced for each user with BRF and nDCG metrics and average them to achieve final BRF and nDCG scores for each considered kNN variant. We employ a well-known Python library named Surprise<sup>6</sup> to implement the kNN algorithms.

### 4.4. Experiment results

In this section, we present the results of the experiments realized to investigate how the parameters of the kNN algorithms affect both their blockbuster bias and accuracy performances. Our experiments consider three similarity functions, i.e., Pearson, Cosine, and MSD, and six maximum neighborhood sizes ( $k$ ) varying from 5 to 100. Accordingly, we first present the BRF results of top-10 recommendations obtained on both MLP and MLM datasets for two variants of kNN algorithms, i.e., UserKNN and ItemKNN, in Figures 1 and 2, respectively.

---

<sup>6</sup> <http://surpriselib.com/>

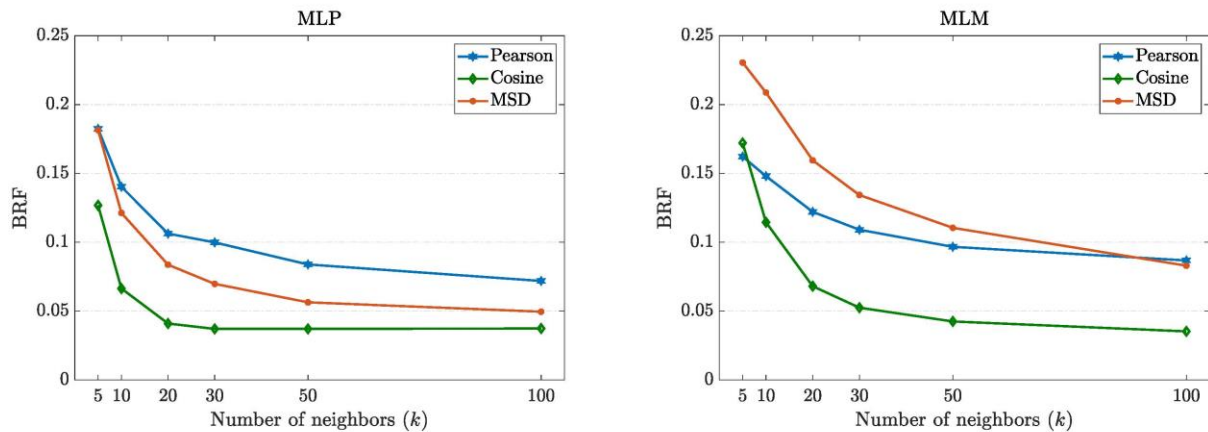


Figure 1 BRF results for the UserKNN algorithm

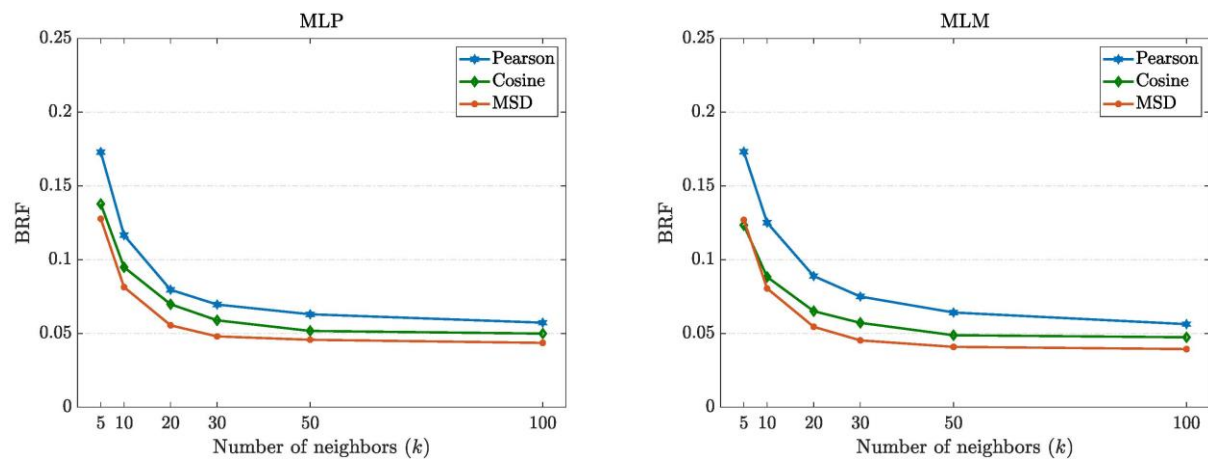


Figure 2 BRF results for the ItemKNN algorithm

As shown in Figures 1 and 2, the highest BRF results, i.e., the most blockbuster biased recommendations, are obtained for both UserKNN and ItemKNN algorithms when the number of neighbors ( $k$ ) is selected as 5. However, the BRF results obtained for both algorithms significantly decrease as the value of  $k$  increases, concluding that they become less biased towards blockbuster items with large neighborhoods. Such BRF trends of the algorithms also hold regardless of the utilized dataset and similarity functions. However, as can be seen from Figures 3 and 4, choosing relatively larger neighborhoods leads to significant decreases in the ranking accuracy performance of both algorithms due to the well-known trade-off between recommendation accuracy and diversification [11]. The obtained results also suggest that both UserKNN and ItemKNN algorithms show similar BRF performances for the MLP. However, the former algorithm seems to be more vulnerable to blockbuster bias than the latter for the MLM dataset. A similar trend is observed for nDCG scores; both algorithms have identical nDCG scores for the MLP, but the UserKNN is superior to the ItemKNN for the MLM dataset.

The experimental results also indicate that the obtained BRF and nDCG results converge at a constant level with neighbors bigger than 50, even if there are slight decrements for the MLM dataset for neighborhood size larger than 50. This finding is because of the size of the utilized dataset, making it challenging to find at least  $k$  numbers of similar individuals who have rated target items or  $k$  numbers of similar products evaluated by the active users. It leads to obtaining identical BRF or nDCG results after a specific number of neighbors, and such threshold value is relatively smaller for MLP than the MLM as the number of available users/items in the MLP is relatively smaller than those in the MLM dataset.

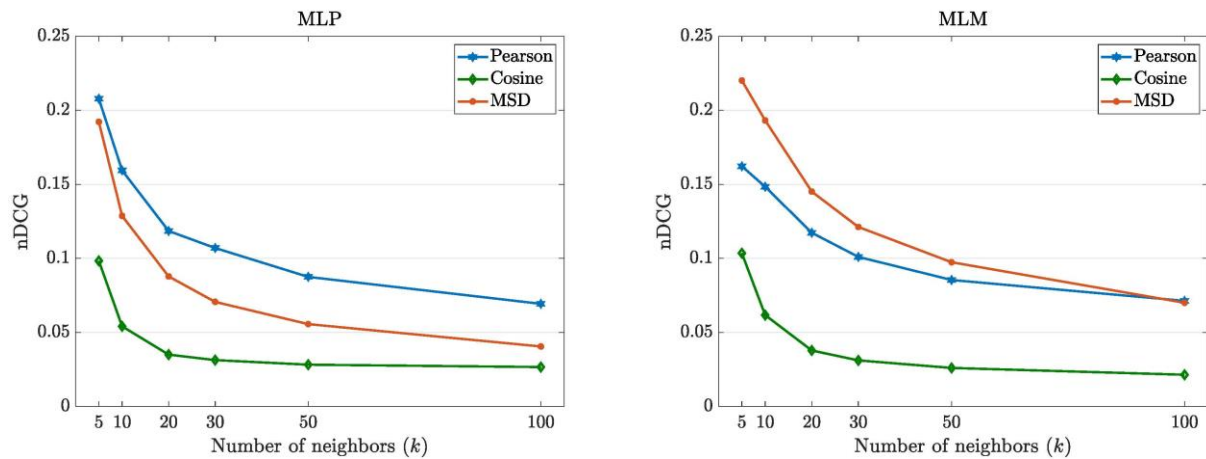


Figure 3 nDCG results for the UserKNN algorithm

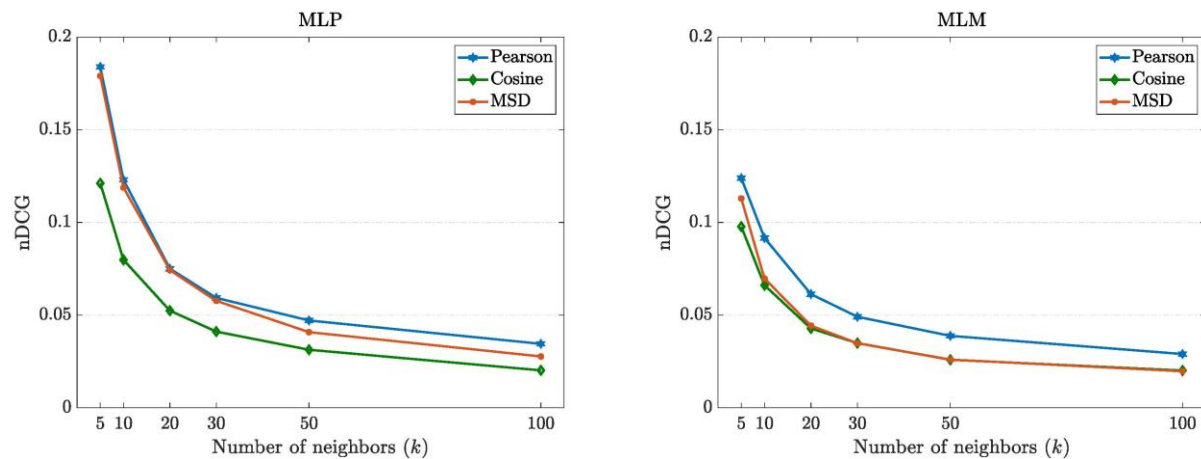


Figure 4 nDCG results for the ItemKNN algorithm

As shown in Figures 3 and 4, the worst recommendations in terms of accuracy are usually obtained when the Cosine similarity function is utilized. However, it seems to be the most robust metric against blockbuster bias, as shown from the BRF results in Figures 1 and 2. On the other hand, as shown in Figures 3 and 4, Pearson is the most prominent similarity function in terms of ranking accuracy, except only for the UserKNN algorithm on the MLM dataset. The underlying reason for the success of the Pearson metric is that it carries out an adjusted normalization step with the average of the ratios of each user since there may be users with an inclination to evaluate very negatively and others with a propensity to evaluate very positively. This finding is also strongly parallel with the outcomes in previous related studies [8]. Even if employing the Pearson metric for kNN algorithms helps produce recommendations with high accuracy, it propagates a significant bias in favor of blockbuster items in the referrals, as depicted in Figures 1 and 2. One explanation for this finding is that the accuracy and beyond-accuracy qualities of the recommendations are usually assumed as conflicting goals.

The experimental results also demonstrate that the kNN algorithms, especially the UserKNN, slightly achieve higher BRF results for the MLM when compared to MLP dataset, as can be followed in Figures 1 and 2. This observation is caused by the sparsity ratio of the data collection; the CF algorithms become more inclined to feature the most blockbuster items in the generated top- $N$  ranked lists since the dataset on which they are trained becomes sparser. However, it positively affects the recommendation accuracy since the algorithms show more successful nDCG performances for the MLP dataset, followed by Figures. 3 and 4.

#### 4.5. Insights and discussion

One of the most important findings of our analysis is that both user- and item-based kNN algorithms become less biased blockbuster items with large neighborhoods. The main reason for this observation is that as the neighborhood size increases, more users (for the UserKNN) or items (for the ItemKNN) contribute to the computed prediction score. Therefore, users' degree of variance is averaged out over the more significant numbers, which improves the chance of including non-popular or not highly-liked items into recommendation lists. This fact enables to produce more diverse and thus less blockbuster-biased ranked lists. On the other hand, selecting relatively larger neighborhoods provides more accurate recommendations for both algorithms. This observation also verifies the trade-off between accuracy and biased performances of the algorithms.

Our study also concludes that the Cosine is the worst similarity metric in predictive accuracy. One explanation for this finding is that this metric considers individuals who rated very different votes as highly similar users. For example, in a [1, 5] rating scale, if an individual rated two items as strongly bad (1, 1) and another one rated them as highly perfect (5, 5), this similarity function becomes ultimately unsuccessful since it outputs the maximum likelihood of similarity between these quite different individuals, by its nature. Even if it is assumed that the probability of maintaining the proportionality in the ratios of two individuals is low for larger datasets, this metric still leads to achieving the worst ranking accuracy performance in our experiments. However, another important finding of our analysis is that such drawback of the Cosine ends with improving beyond-accuracy aspects of recommendations and, as a result, provides having less blockbuster biased ranked lists compared to both Pearson and MSD metrics.

#### 5. Conclusion and future work

As a prominent type of memory-based collaborative filtering (CF) algorithms, neighborhood-based, i.e., also referred to as k-nearest neighbor (kNN) CF methods, are widely used in recommender systems due to their success in providing personalized recommendations. It is a known phenomenon that parameter-tuning, such as similarity function and neighborhood size, significantly impacts their accuracy performances when locating neighborhoods. Also, they are subject to blockbuster bias issues, i.e., they expose blockbuster (i.e., both popular and highly-liked) items more in their recommendations than other ones.

This study mainly focuses on evaluating how the parameters of two well-known kNN algorithms affect their blockbuster bias performances through an efficient evaluation protocol. We consider three different similarity functions, namely Pearson, Cosine, and Mean Squared Difference, and varying neighborhood sizes. Also, we investigate how these parameters influence their recommendation accuracy performances. Experiments conducted on two benchmark datasets demonstrate that as the neighborhood size decreases, the kNN algorithms generally become more vulnerable to blockbuster bias while their accuracy increases. One explanation for this finding is that more users (for the UserKNN) or items (for the ItemKNN) contribute to the computed prediction with larger neighborhoods, achieving more diverse and thus less blockbuster biased recommendations. Also, using the Cosine metric for the kNN algorithms is superior to other similarity functions in producing recommendations where blockbuster bias is treated more; however, it leads to having unqualified recommendations in terms of predictive accuracy as they are usually conflicting goals. On the other hand, we found that the Pearson usually performs better than other functions regarding recommendation accuracy, which is also parallel with the outcomes in previous related studies [11]. In conclusion, our analysis provides inference on how parameter-tuning of the kNN algorithms should be performed according to the purposes of the service providers.

Improving beyond-accuracy quality while maintaining accuracy is essential for recommender systems in several aspects. Therefore, our future direction is to develop a mitigation strategy modifying the prediction calculation step of the kNN algorithms by including a tuning parameter penalizing blockbuster items to alleviate the effects of such blockbuster bias on recommendation quality.

## Acknowledgments

This study is supported by Project No. M-2021-811 from the Sivas Cumhuriyet University.

## References

- [1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*, Springer, pp. 1–35, 2011.
- [2] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, vol. 52, no. 1–37, 2019, doi: 10.1007/s10462-018-9654-y.
- [3] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004, doi: <https://doi.org/10.1145/963770.963772>.
- [4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, pp. 109–132, 2013, doi: 10.1016/j.knosys.2013.03.012.
- [5] R. Chen, Q. Hua, Y.-S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018, [Online]. Available: <https://doi.org/10.1109/ACCESS.2018.2877208>.
- [6] M. Karimi, D. Jannach, and M. Jugovac, "News recommender systems – Survey and roads ahead," *Inf. Process. & Manag.*, vol. 54, no. 6, pp. 1203–1227, 2018, [Online]. Available: <https://doi.org/10.1016/j.ipm.2018.04.008>.
- [7] M. Nilashi, O. Bin Ibrahim, and N. Ithnin, "Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and Neuro-Fuzzy system," *Knowledge-Based Syst.*, 2014, doi: 10.1016/j.knosys.2014.01.006.
- [8] J. L. Sánchez, F. Serradilla, E. Martínez, and J. Bobadilla, "Choice of metrics used in collaborative filtering and their impact on recommender systems," 2008, doi: 10.1109/DEST.2008.4635147.
- [9] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. from Data*, vol. 4, no. 1, pp. 1–24, 2010, [Online]. Available: <https://doi.org/10.1145/1644873.1644874>.
- [10] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Syst.*, vol. 23, no. 6, pp. 520–528, 2010, [Online]. Available: <https://doi.org/10.1016/j.knosys.2010.03.009>.
- [11] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr. Boston.*, 2002, doi: 10.1023/A:1020443909834.
- [12] E. Yalcin, "Blockbuster: A New Perspective on Popularity-bias in Recommender Systems," in *6th International Conference on Computer Science and Engineering (UBMK)*, Sep. 2021, pp. 107–112, doi: 10.1109/UBMK52708.2021.9558877.
- [13] E. Yalcin and A. Bilge, "Treating adverse effects of blockbuster bias on beyond-accuracy quality of personalized recommendations," *Eng. Sci. Technol. an Int. J.*, vol. 33, p. 101083, Sep. 2022, doi: 10.1016/J.JESTCH.2021.101083.
- [14] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and Debias in Recommender System: A Survey and Future Directions," *arXiv Prepr. arXiv2010.03240*, 2020.
- [15] L. Boratto, G. Fenu, and M. Marras, "Connecting user and item perspectives in popularity debiasing for collaborative recommendation," *Inf. Process. & Manag.*, vol. 58, no. 1, p. 102387, 2021, [Online]. Available: <https://doi.org/10.1016/j.ipm.2020.102387>.
- [16] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay, "Accurately Interpreting Clickthrough Data as Implicit Feedback," *ACM SIGIR Forum*, vol. 51, no. 1, pp. 4–11, Aug. 2017, doi: 10.1145/3130332.3130334.
- [17] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *International Conference on Machine Learning*, 2014, pp. 1512–1520.

- [18] S. Krishnan, J. Patel, M. J. Franklin, and K. Goldberg, "A methodology for learning, analyzing, and mitigating social influence bias in recommender systems," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 137–144, doi: <https://doi.org/10.1145/2645710.2645740>.
- [19] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac, "What recommenders recommend: an analysis of recommendation biases and possible countermeasures," *User Model. User-adapt. Interact.*, vol. 25, no. 5, pp. 427–491, Dec. 2015, doi: 10.1007/S11257-015-9165-3.
- [20] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," *Proc. 32nd Int. Florida Artif. Intell. Res. Soc. Conf. FLAIRS 2019*, pp. 413–418, 2019.
- [21] E. Yalcin and A. Bilge, "Investigating and counteracting popularity bias in group recommendations," *Inf. Process. Manag.*, vol. 58, no. 5, Sep. 2021, doi: 10.1016/j.ipm.2021.102608.
- [22] D. Kowald, M. Schedl, and E. Lex, "The unfairness of popularity bias in music recommendation: A reproducibility study," in *European Conference on Information Retrieval*, 2020, pp. 35–42, [Online]. Available: [https://doi.org/10.1007/978-3-030-45442-5\\_5](https://doi.org/10.1007/978-3-030-45442-5_5).
- [23] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation", *arXiv preprint arXiv:1907.13286*, 2019.
- [24] L. Boratto, G. Fenu, and M. Marras, "The effect of algorithmic bias on recommender systems for massive open online courses," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11437 LNCS, pp. 457–472, 2019, doi: 10.1007/978-3-030-15712-8\_30.
- [25] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Missing data modeling with user activity and item popularity in recommendation", in *Asia Information Retrieval Symposium*, 2018, pp. 113–125, [Online]. Available: [https://doi.org/10.1007/978-3-030-03520-4\\_11](https://doi.org/10.1007/978-3-030-03520-4_11).
- [26] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Correcting Popularity Bias by Enhancing Recommendation Neutrality", in *RecSys Posters*, 2014.
- [27] H. Abdollahpouri, R. Burke, and B. Mobasher, "Popularity-Aware Item Weighting for Long-Tail Recommendation", *arXiv preprint arXiv:1802.05382*, 2018.
- [28] G. Adomavicius and Y. Kwon, "Multi-criteria recommender systems," in *Recommender Systems Handbook, Second Edition*, 2015.
- [29] N. A. Najjar and D. C. Wilson, "Differential neighborhood selection in memory-based group recommender systems," in *Twenty-Seventh International Flairs Conference*, 2014.
- [30] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, Jan. 2010, doi: 10.1145/1644873.1644874.
- [31] K. Choi and Y. Suh, "A new similarity function for selecting neighbors for each target item in collaborative filtering," *Knowledge-Based Syst.*, 2013, doi: 10.1016/j.knosys.2012.07.019.
- [32] R. Sanders, "The Pareto principle: its use and abuse," *J. Serv. Mark.*, 1987.
- [33] L. Baltrunas and F. Ricci, "Group Recommendations with Rank Aggregation and," *Proc. fourth ACM Conf. Recomm. Syst. ACM*, 2010.

# Application with deep learning models for COVID-19 diagnosis

 Fuat Türk<sup>1</sup>,  Yunus Kökver<sup>2</sup>

<sup>1</sup>Corresponding Author; Çankırı Karatekin University, Department of Computer Engineering;  
fuatturk@karatekin.edu.tr

<sup>2</sup> Ankara University; Department of Computer Technologies, ykokver@ankara.edu.tr

Received; 10 March 2022 Revised;29 April Accepted;19 June Published online August 2022

## Abstract

COVID-19 is a deadly virus that first appeared in late 2019 and spread rapidly around the world. Understanding and classifying computed tomography images (CT) is extremely important for the diagnosis of COVID-19. Many case classification studies face many problems, especially unbalanced and insufficient data. For this reason, deep learning methods have a great importance for the diagnosis of COVID-19. Therefore, we had the opportunity to study the architectures of NasNet-Mobile, DenseNet and Nasnet-Mobile+DenseNet with the dataset we have merged. The dataset is divided into three separate classes: Normal, COVID-19, and Pneumonia. Recall, Precision, and F-measure are the main metrics which used to measure the performance of classification algorithms. The accuracy is obtained as 87.16%, 93.38% and 93.72% for the NasNet-Mobile, DenseNet and NasNet-Mobile+DenseNet architectures for the classification, respectively. The results once again demonstrate the importance of Deep Learning methods for the diagnosis of COVID-19.

**Keywords:** COVID-19 diagnosis, DenseNet, NasNet-Mobile, deep learning classification

## 1. Introduction

Coronavirus disease (COVID-19), which emerged toward the end of the 2019, is a type of coronavirus known to infect extremely rapidly. The International Committee on Virus Taxonomy has named this virus SARS-COV-2 [1,2].

In cases where coronavirus is not diagnosed and treated quickly, a severe clinical picture can be observed, which can lead to pneumonia and death [3,4]. In addition, the continuous emergence of different variants affects the diagnosis and treatment process negatively. At this point, imaging examination, as a rapid and more convenient examination [5] method, plays a role in reducing the cost of disease worldwide as well as significantly shortening the time for treatment of patients infected with COVID-19 [6,7].

Currently, radiographs and computed tomography (CT) are the main screening methods for diagnosis of COVID-19. These methods can contribute the very rapid screening of an infected person [8].

Recently, Deep Learning applications have become a highly accepted method in the field of image processing for diagnosing many types of diseases [9]. Since the emergence of COVID-19 virus, many researchers have developed Deep Learning based models for the detection of COVID -19 on radiological images and achieved successful results. Therefore, a study is conducted using Deep Learning models for rapid diagnosis of COVID-19. The dataset used in this study consists of three classes. Figure 1 shows the images of these classes.



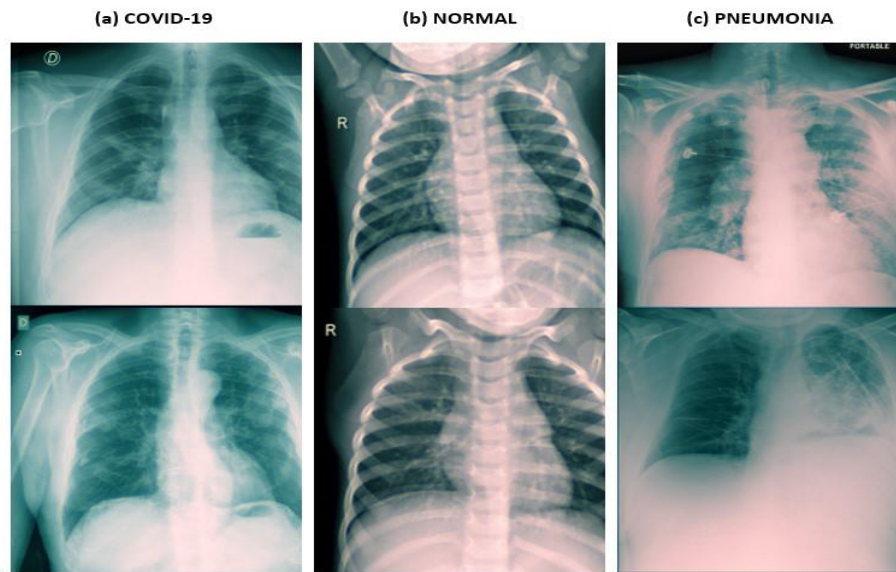


Figure 1 Sample images of COVID-19, Normal and Pneumonia

The main benefits of the study can be explained as follows:

- A balanced dataset is created with three separate class labels from the existing datasets. The dataset is beneficial and reliable for COVID-19 multi-classification problems.
- There was an opportunity to run DenseNet and NasNet-Mobile architectures separately and together on this dataset and examine the results.
- These architectural structures provide a highly accurate framework for the multi-classification problem through the radiographs and can be easily tested in hospitals with a computerized interface.

In the remainder of the paper, a literature review is conducted in the second part. Material and methods are presented in chapter 3, research results and discussion are explained in chapter 4, finally, conclusion is given in chapter 5.

## 2. Related Works

If examining the literature on COVID-19 classification and diagnostic methods, the first thing that stands out is the study conducted by Jia et al. using the dynamic CNN model. In this study, they found that the understanding and classification of Chest X-Ray, Computed Tomography (CT) scans are of great importance in the diagnosis of COVID-19. Existing researches on classification of COVID-19 cases are due to instability of data, insufficient generalizability, lack of studies, etc. They said they faced difficulties. To solve these problems, their study proposed a kind of modified MobileNet to classify COVID-19 X-ray images. A later ResNet block for CT image classification and CNN modification method, they made a design that can improve the performance for solving the problem and classification. The proposed methods obtain a test accuracy of 99.6% for the CXR image dataset with five categories and a test accuracy of 99.3% for the CT image dataset [10].

Fang and Wang conducted a study on a convolution-based COVID-19 deep learning classification architecture and the local evolution of deconvolution. In this study, they stated that COVID-19 is a strain that makes a significant difference between normal patients with asymptomatic infections and patients with coronavirus or pneumonia. In order to effectively improve the accuracy of normal and COVID-19 assessment by manual examination by doctors, this paper proposed a deep learning model based on improving the current situation. Additionally, this paper was prepared using an open access COVID-CT dataset of 143 novel coronaviruses provided by Petuum researchers at the University of California, San Diego. The dataset (original image, organized image) contains 1460 images. For the performance of the model, 70% of this dataset was used for the train and the remaining one was used for testing. It has been

shown that the proposed network has high classification success. The corresponding values for sensitivity, specificity, and precision are %98, %96, %98, respectively [5].

Singh et. al developed a machine learning model to classify images with or without COVID-19. In this study, they presented an alignment-free approximation model to classify COVID-19. The dataset used here consists of a total of 1582 image samples with genome sequences of different lengths from different area obtained from different data and divided into COVID-19 and non-COVID-19 groups. The available dataset was used to test the accuracy and performance of the F-measure-based classifiers by 10-fold cross-validation. In addition, a cross-validation tests with paired "t" tests were used to test the best model with unused dataset. The random forest method was identified as the best model to discriminate COVID-19. It also formed a control group with 97.4% of accuracy, 96.2% of sensitivity and 98.2% of specificity when tested with unknown samples [11].

Gour and Jain performed a study on classification of images with uncertain convolutional networks for COVID-19 radiographs. In this study, they mentioned that deep learning methods were successful during image process analysis detection. They also designed an uncertainty sensitive deep learning model called UA-ConvNet for automatic detection of COVID-19 diseases from Chest X-Ray images. The method they developed was evaluated on three datasets consist of chest x-ray data, COVID-19 x-ray data and Kaggle-based datasets. The UA-ConvNet architecture achieved a G average of 98.02% and a precision of 98.15% for the classification mission on the COVID-19 X-ray dataset. The proposed binary classification model achieved a G-mean of 99.16% and a precision of 99.30% for the radiograph dataset [12].

Hassan et al. conducted a study to review and classify COVID-19 CT imaging models. In their study, they proposed a joint AI-based model for the diagnosis of the COVID-19 model and CT images. After reviewing previous studies, they argued that the COVID-19 literature studies on computer image processing tasks such as classification and segmentation were insufficient. Therefore, studies on COVID-19 deep learning methods based on CT images focused on. They preferred to scan popular websites such as Kaggle, GitHub, Google Scholar, , IEEE Xplore and ScienceDirect in order to align related studies. After extensive research, 114 studies were reviewed and selected research analyzes suggested that artificial intelligence and computer vision have significant potential for rapid COVID-19 virus diagnosis as they can contribute significantly to the automation of the diagnostic process. In addition, they concluded in the study that deep learning methods are extremely important for the diagnosis of COVID-19 [13].

Tuncer et al. presented a new method for classifying covid-19 and pneumonia called "F-transformation". In this study, they stated that COVID-19 is an important disease affecting life all over the world. Since COVID-19 disease is similar to pneumonia, three classes of datasets were used in their study: COVID-19, pneumonia, and normal lung radiographs. Using this dataset, a new machine learning model, called the example model, is presented. First, a fuzzy tree transform was applied to each box image used, and 15 images were obtained from a box image. In the continuation of the study, the pattern splitting process was applied to the existing images. Valuable features were then selected through the iterative neighborhood component (INCA). INCA selects the 616 most prominent features and these features are output to the 16 different classifiers. The best classifier is the cubic support vector machine, which provides a classification accuracy of 97.01% for this dataset [14].

Balaha et al. proposed a Deep Learning-based hybrid system for segmentation of COVID-19 virus. The study proposes a hybrid COVID-19 architecture based on deep learning. In the segmentation phase was proposed using X-ray images to extract the lungs. For the classification phase, a hybrid CNN with an abstract designed CNN model was used. Using the model, which they named HMB-HCF, they achieved a successful result with 99.84% of accuracy [15].

Islam and Nahiduzzaman conducted a study on complex feature extraction for COVID19 detection from CT scan images using a machine learning approach based on the ensemble model. In this study, they tried the Contrast Limited Histogram Equalization method to CT images as a pre-processing step to improve the images. Finally, they developed a new CNN model to extract 100 salient features from a total of 2482 CT s images. This ensemble model achieved 99.73%, 99.46%, and 100% accuracy, precision and recall, respectively [16].

### 3. Material and Methods

In this paper, DenseNet and NasNet-Mobile networks have separated and operated together to show the superiority of the hybrid models.

The block diagram containing the process steps of the applied work is shown in Figure 2.

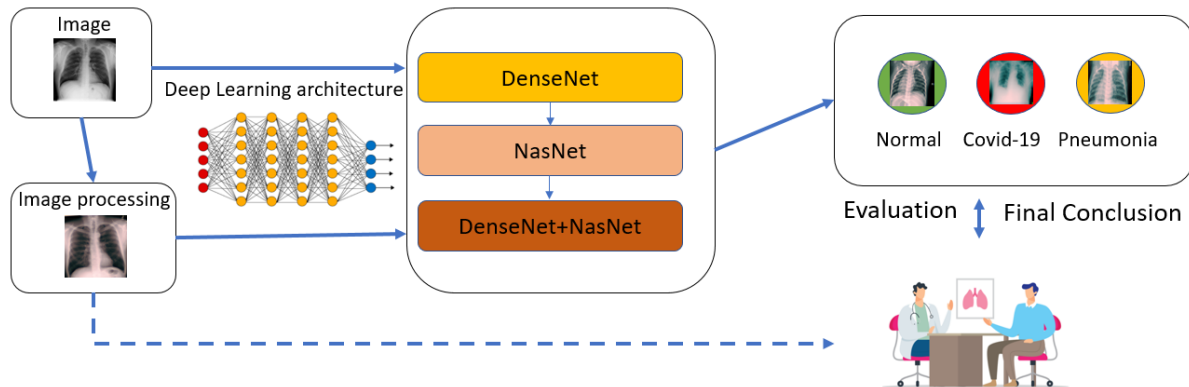


Figure 2 Block diagram of the proposed model

#### 3.1 DenseNet Network Structure

In Figure 3, the architectural structure of DenseNet is explained in a simple way.

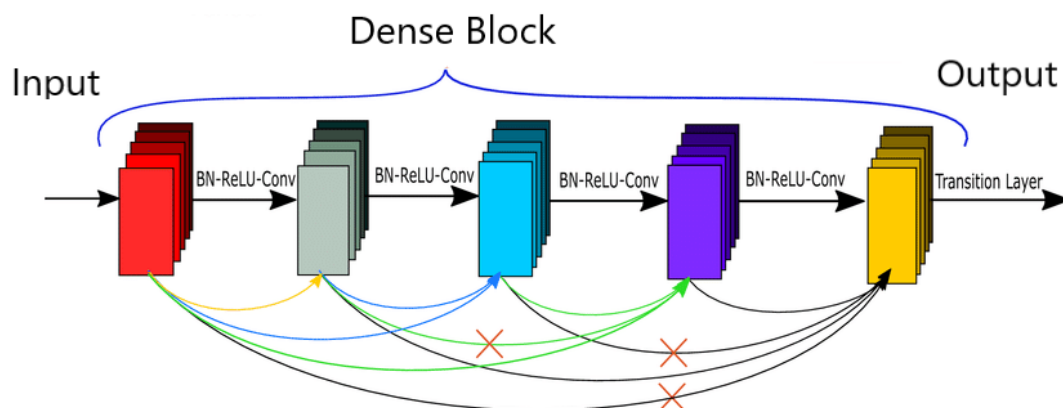


Figure 3 DenseNet Architecture

The neural network model of DenseNet differs from other convolutional neural networks by two basic features. The first feature is the dense connection layer structure. In this structure, the layers are linked to each of the previous layers to allow for feature reuse. Second, due to feature reuse, each layer uses a small number of convolutional kernel subtraction features to reduce redundancy. For this reason, in DenseNet, layers are merged with all previous layers in channel size. The advantages of DenseNet are mainly the following:

- Effective solution to the problem of the disappearance of the gradient
- Propagation of the support feature
- Reuse of the support feature
- Significant reduction of the number of parameters

This model can bypass the pre-training on Image Net dataset and achieve the goal of time saving and efficiency by starting the training directly with the randomly initiated model. In many large-scale studies, there may be significant differences when comparing the actual dataset to the ImageNet dataset. Therefore, it may be a rational alternative to apply the model that does not require prior training in imaging studies [17,18].

### 3.2 NasNet-Mobile Network structure

The NasNet-Mobile architectural structure is shown in Figure 4.

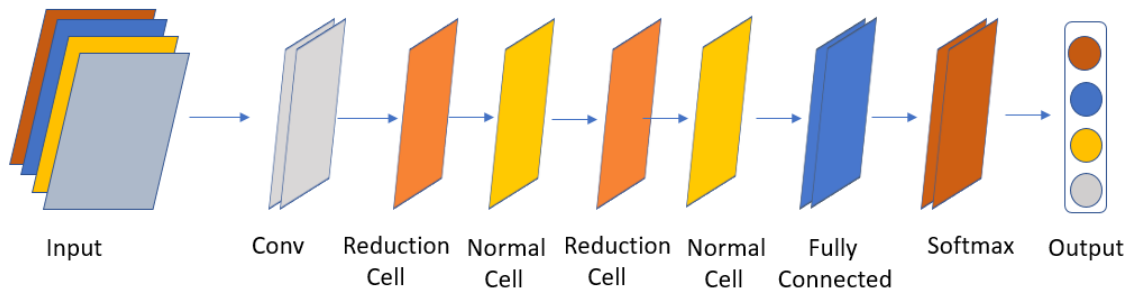


Figure 4 NasNet-Mobile Architecture

NasNet-Mobile Convolutional Neural Network is trained from ImageNet dataset. The network has succeeded in recognizing images using an active feature. The size of the input images is  $224 \times 224$ . NasNet architecture is basically divided into two classes, NasNetMobile and NasNetLarge. The NasNetMobile network is designed for smaller datasets compared to NasNetLarge. It uses the search strategy to search for layers or cells with the best convolutions in relatively small image datasets. Convolutional cells are used to achieve better classification performance and a smaller computational budget. In NasNet, although the architecture is predetermined, the number of cells or blocks to be found by the search method, that is, the number of initial convolution filters, are free parameters. [19].

### 3.3 DenseNet+NasNet-Mobile Architectural Structure

The layered structures, parameter numbers, and combination values of the model created by combining the architectural structures of DenseNet+NasNet-Mobile were arranged. The architectural design was created by starting with the DenseNet block in the input layer and then adding the NasNet-Mobile block. It starts with the input image size  $(224, 224, 3)$  and changes to  $(7, 7, 1024)$  in the concatenation phase. Then, in the output layer, it is converted to the last convolutional block structure of size  $(7, 7, 32)$ . Finally, the model structure is completed by smoothing the output layer with three classes using the softmax function.

### 3.4 Image Pre-processing

Image pre-processing is required in Deep Learning architectures, especially when the dataset is difficult to interpret. At this stage, image pre-processing techniques are used to better explore and interpret existing images. In this study, a series of pre-processing steps are performed using the OpenCV library in Python. First, its value is set to 255 (white) if the pixel density is greater than the specified threshold, and 0 (black) otherwise. Then a noise removal technique is used to keep the edges sharp. Finally, the contrast of the image is increased to expand the intensity range. Figure 5 shows the images before and after image pre-processing.

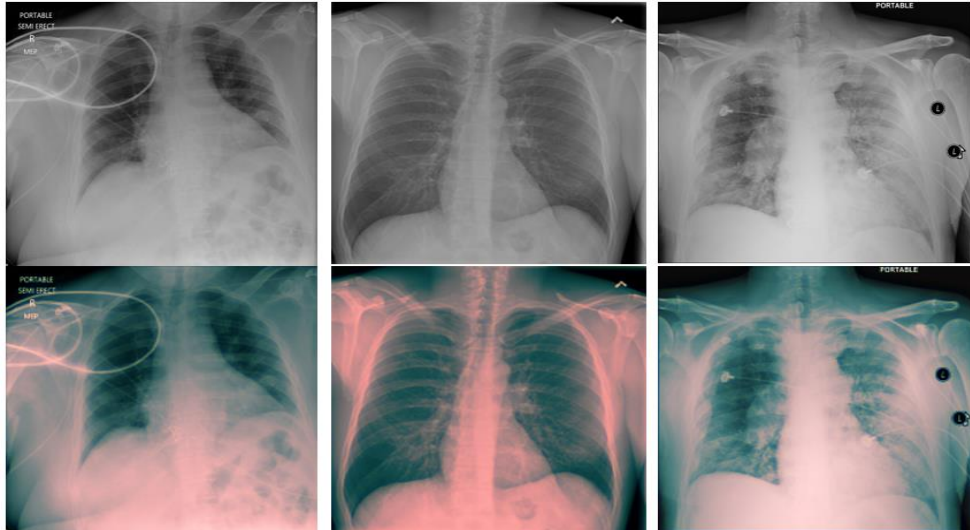


Figure 5 Image pre-processing phase

### 3.5 Dataset

Explanations about the dataset and all labels can be downloaded from the relevant link(for dataset: <https://github.com/turkfuat/covid19-multiclass>). In the study, three different datasets were examined in details. The images in the dataset belong to three different classes:

- Normal: individuals without signs of disease on their chest images.
- COVID -19: Individuals diagnosed by physicians who show the spread of COVID -19 virus on their chest images.
- Pneumonia: Includes individuals who show pneumonia (with subtypes) on chest images and have been diagnosed by physicians.

Firstly, the images in the data set from Deb et al [20] were analyzed and transferred to the data pool. Then the images in the dataset from COVID-19 Grand Challenge were added to this pool with the same labels. Finally, different data with the same labels from the dataset from Cohen et al [21] were imported into data warehouse. The current state of the dataset is shown in Table 1.

Table 1 Merged and Balanced Dataset

	Normal	COVID-19	Pneumonia
<b>Train</b>	7834	8107	7025
<b>Valid</b>	1679	1737	1505
<b>Test</b>	1679	1738	1506
<b>Total</b>	11192	11582	10036

### 3.6 Performance Evaluation Criteria

Accuracy, Sensitivity, Precision and F1-score are the main metrics for measure the performance of the classification algorithms. Accuracy shows the ratio of correct prediction rate. It is calculated as:

$$Accuracy = (TN + TP)/(TP + FP + TN + FN) \quad (1)$$

Precision shows how many of the values predicted as Positive are actually relevant. Precision value is calculated as:

$$Precision = TP / (TP + FP) \quad (2)$$

Sensitivity is a true positive rate that expresses the probability of a positive test provided it is actually positive. Sensitivity value is calculated as:

$$Sensitivity = TP / (TP + FN) \tag{3}$$

F1-Score takes into account both precision and sensitivity as harmonic mean. It is calculated as:

$$F1 - Score = 2 * (Recall * Precision) / (Recall + Precision) \tag{4}$$

Furthermore, the precision, sensitivity and F1-Score values for three models are shown in Table 3.

#### 4. Results and Discussion

The training and testing process is performed on the GTX 1070 TI graphics card for about two hours. The Adam optimizer is used for the model followed by the Leaky Relu activation function.

The learning rate is set as 0.0001 and the stack size as 32. The accuracy values of the three models run at 100 steps are shown in Table 2. It is found that the accuracy value increased to 93.72% with image pre-processing and using both models together.

Table 2 Accuracy Values of The Models

Model name	Accuracy (%)
DenseNet	93.38
NasNet-Mobile	87.16
Dense Net+ NasNet-Mobile	92.46
Dense Net+ NasNet-Mobile (with image pre-processing)	93.72

In Figure 6, the accuracy and loss values for Nasnet mobile are shown, and in Figure 7, the accuracy and loss values for the DenseNet architecture are shown. Figure 8 shows the accuracy and loss values obtained by using both models in combination.

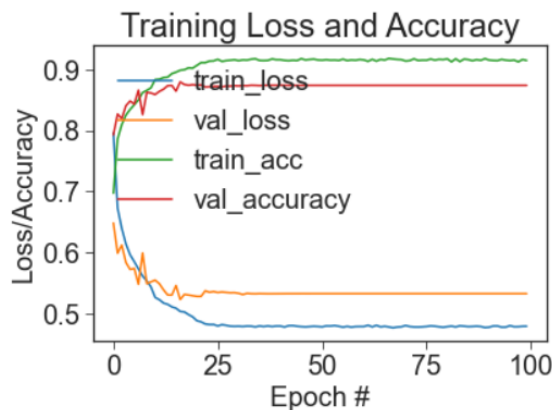


Figure 6 Accuracy/Loss Values for Nasnet-Mobile Model

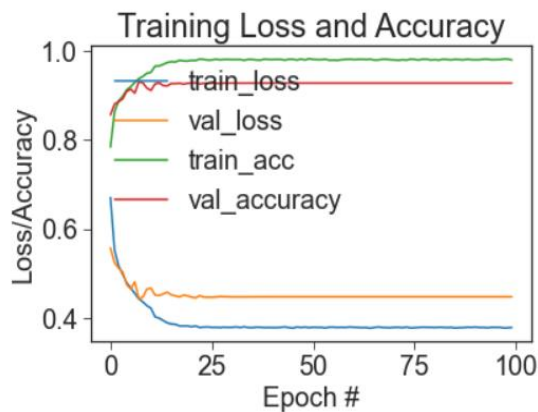


Figure 7 Accuracy/Loss for DenseNet Model

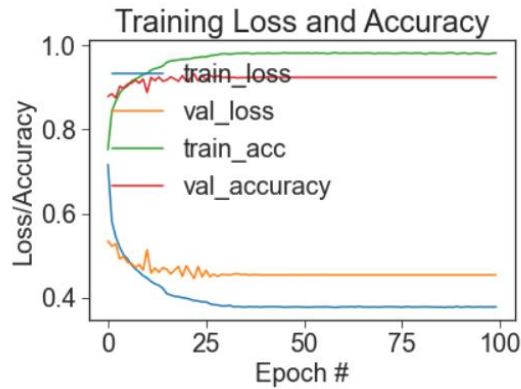


Figure 8 Accuracy/Loss for NasNet-Mobile+DenseNet Model

Table 3 shows the accuracy, recall, and F1 score values for all three models. When the results are compared, it is clear that the DenseNet+NasNet-Mobile model gives more successful results.

Table 3 Precision/Sensitivity/F1-Score values for three models

Class	Accuracy	Precision	Sensitivity	F1-Score	Test Samples
<b>NasNet-Mobile</b>					
COVID-19	0.87	0.87	0.87	0.87	1738
Normal	0.86	0.89	0.86	0.88	1679
Pneumonia	0.88	0.86	0.88	0.87	1506
<b>DenseNet</b>					
COVID-19	0.91	0.97	0.91	0.94	1738
Normal	0.94	0.91	0.94	0.93	1679
Pneumonia	0.95	0.92	0.95	0.93	1506
<b>DenseNet+ NasNet-Mobile</b>					
COVID-19	0.91	0.97	0.91	0.94	1738
Normal	0.96	0.92	0.96	0.94	1679
Pneumonia	0.94	0.93	0.94	0.94	1506

In Figure 9, the values of the confusion matrix for Nasnet mobile are shown. In Figure 10, the values of the confusion matrix for the DenseNet architecture are shown. Figure 11 shows the confusion matrix results from using both models together.

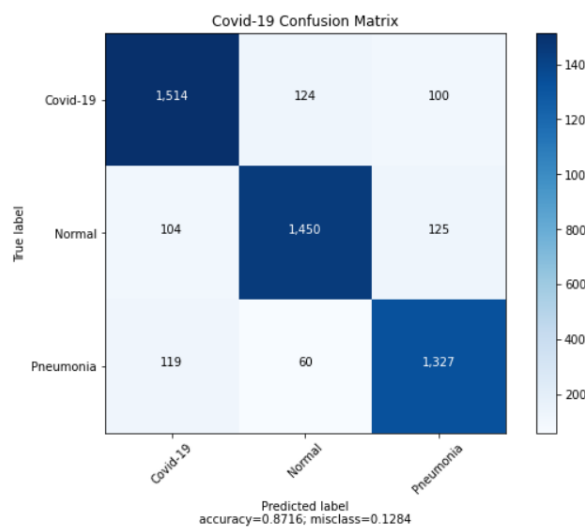


Figure 9 Confusion Matrix for NasNet-Mobile

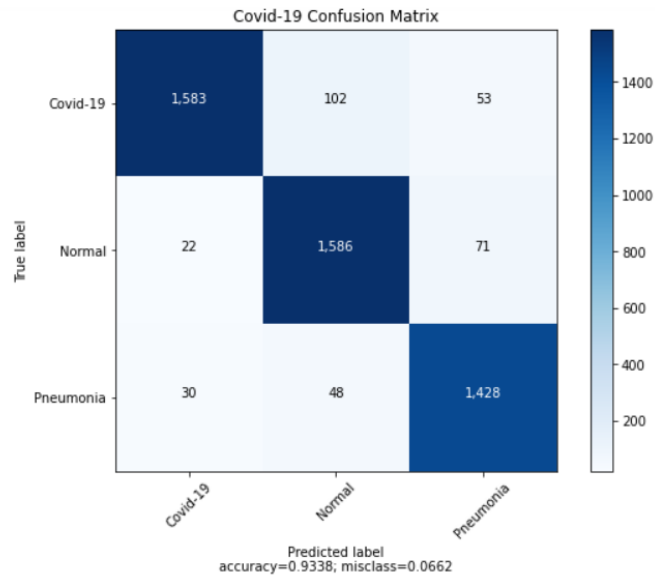


Figure 10 Confusion Matrix for DenseNet

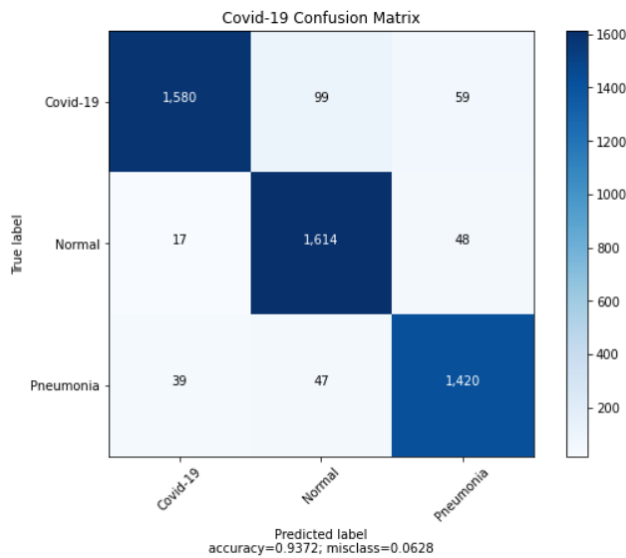


Figure 11 Confusion Matrix for NasNet-Mobile+DenseNet

Figure 12 lists some examples of images that were mislabeled as a result of the testing process of all three models. When the incorrectly labeled results are examined, it is noteworthy that the classes of COVID-19 and pneumonia are often confused.

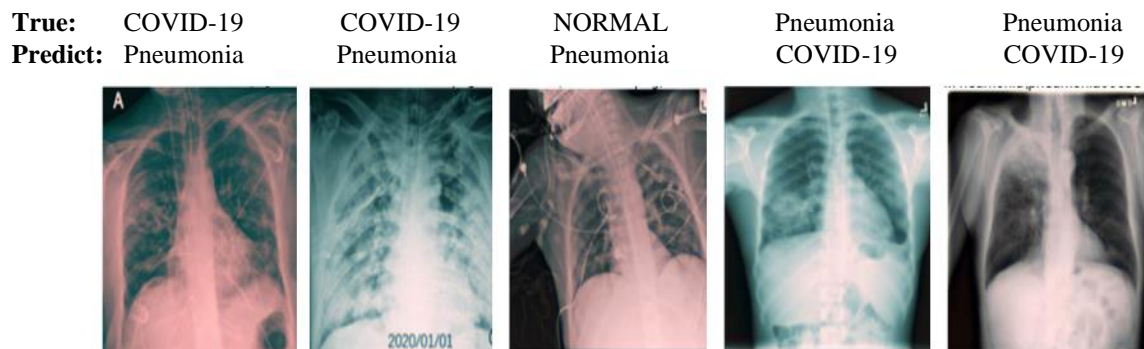


Figure 12 Some images that were mislabeled as a test result



Table 4 shows an overall comparison of the results of other studies in the literature with the results of this study. According to table, MAI-Net architecture was the study that gave the most successful results. It is seen that the developed model gives respectable results when compared with the studies on similar data sets. In addition, it can be said that subclass types are also important in multi-class studies. Even small differences in CT images can lead to incorrect results in classification. Given these details, it can be said that the DenseNet+NasNet-Mobile model has had remarkable success in classification problems.

Table 4 Comparison of the model with the similar studies

Study	Dataset	Architecture	Accuracy (%)
Wang et al. [22]	<a href="https://github.com/lindawang/COVID-Net">https://github.com/lindawang/COVID-Net</a>	Covid-Net	92.4
Lascu [23]	Kaggle Dataset	FC-DenseNet	-
Wang et al [24]	covid-chestxray-dataset	MAI-Net	96.42
Wang et al [25]	covid-chestxray-dataset	CFW-Net	94.35
Asif et. al. [26]	Cohen et al.	Xception	89.6
Ozturk et al. [27]	Cohen et al.	Darknet	87.02
Ioannis et al. [28]	Kaggle Challenge Dataset	VGG19	93.48
Deb et al. [20]	Private Dataset	Ensemble M.	91.39
Rahimzadeh et. al [29]	Covid-chestxray-dataset + Kaggle Challenge Dataset	Xcept- ResNet	91.4
Mahmud et al [30]	Private Dataset	CovXNet	89.6
<b>DenseNet+NasNet</b>	<b>Merged+Balanced</b>	<b>Ensemble M.</b>	<b>92.46</b>
<b>DenseNet+NasNet (with image pre-processing)</b>	<b>Merged+Balanced</b>	<b>Ensemble M.</b>	<b>93.72</b>

This method can primarily guide new users in deep learning applications where the data distribution is balanced but the classification task is difficult. It is clearly seen how the performance measures change when different architectural models are used separately or together. When NasNet-Mobile and DenseNet architectures are used together, it is seen that the accuracy rate reaches 93.72%. The disadvantage of the system is that it needs more time for classification.

Despite this, it can be said that the number of data is more and it is open to experimentation and development in different multi-class applications.

## 5. Conclusion

This research has enabled the application of deep learning models for the diagnosis of COVID-19 with chest radiographs. It also provides the opportunity to show the differences between COVID-19 and pneumonia images. The success of deep learning models in classification problems can be tested when models run together and separately. In addition, the newly created classes provide users with the ability to use the balanced dataset for COVID-19 classification studies. Three separate models we have developed have shown that Deep Learning methods can be successfully used in COVID-19 studies. We are optimistic that we can achieve even better results.





## References

- [1] U. G. Kraemer *et al.*, "Data curation during a pandemic and lessons learned from COVID-19," *Nat. Comput. Sci.*, vol. 1, no. 1, pp. 9–10, 2021.
- [2] H. Panwar, P.K. Gupta, S. M. Khubeb, R.M. Menendez, P. Bhardwaj, V. Singh, "A Deep Learning and Grad-CAM based Color Visualization Approach for Fast Detection of COVID-19 Cases using Chest X-ray and CT-Scan Images," *Chaos, Solitons Fractals*, vol. 140, 2020.
- [3] P. Rai, B. K. Kumar, V. K. Deekshit, I. Karunasagar, "Detection technologies and recent developments in the diagnosis of COVID-19 infection.," *Appl. Microbiol. Biotechnol.*, pp. 1–15, 2021.

- [4] C. C. Nathaniel et al., "Multiplexed detection and quantification of human antibody response to COVID-19 infection using a plasmon enhanced biosensor platform," *Biosens. Bioelectron.*, 171, pp. 112679-112679, 2021.
- [5] L. Fang, X. Wang. "Mathematical modelling of two-axis photovoltaic system with improved efficiency." *Elektronika Ir Elektrotehnika*, vol. 21, no. 4, pp. 40-43, 2015.
- [6] V. Manivel, A. Lesnewski, S. Shamim, G. carbonatto, T. Govindan, "CLUE: COVID-19 lung ultrasound in emergency department," *Emerg. Med. Australasia (EMA)*, vol. 32, no. 4, pp. 694–696, 2020.
- [7] S. Yang, Y. Zhang, J. Shen, "Clinical potential of UTE-MRI for assessing COVID -19: patient- and lesion-based comparative analysis," *Magn. Reson. Imag.*, vol. 52, no. 2, pp. 397–406, 2020.
- [8] A. Narin, C. Kaya, Z. Pamuk, "Automatic Detection of Coronavirus Disease (Covid19) Using X-Ray Images and Deep Convolutional Neural Networks," arXiv preprint arXiv, pp.10849, 2020.
- [9] L. Luo, Z. Luo, Y. Jia, C. Zhou, J. He, J. Lyu, X. Shen, "CT differential diagnosis of COVID-19 and non-COVID-19 in symptomatic suspects: a practical scoring method," *BMC Pulm. Med.*, vol. 20, no. 11, pp. 719–739, 2020.
- [10] G. Jia, H.Keung, L.Y.Xu, "Classification of COVID-19 chest X-Ray and CT images using a type of dynamic CNN modification method," *Computers in Biology and Medicine*, vol. 134, 2021.
- [11] P. Singh, M. Vallejo, I.M. El-Badawy, A. Aysha, J. Madhanagopal, A. Athif, M. Faudzi, "Classification of SARS-CoV-2 and non-SARS-CoV-2 using machine learning algorithms," *Computers in Biology and Medicine*, vol. 136, 2021.
- [12] M. Gour, S. Jain, "Uncertainty-aware convolutional neural network for COVID-19 X-ray images classification," *Computers in Biology and Medicine*, vol. 140, 2022.
- [13] H. Hassan, Z. Ren, H. Zhao, S. Huang, D. Li, S. Xiang, Y. Kang, S. Chen, B. Huang, "Review and classification of AI-enabled COVID-19 CT imaging models based on computer vision tasks," *Computers in Biology and Medicine*, vol. 141, 2022.
- [14] T. Tuncer, F. Ozyurt, S. Dogan, A. Subasi, "A novel Covid-19 and pneumonia classification method based on F-transform," *Chemometrics and Intelligent Laboratory Systems*, vol. 210, 2021.
- [15] H.M. Balaha, M. H. Balaha, H.A. Ali, "Hybrid COVID-19 segmentation and recognition framework (HMB-HCF) using deep learning and genetic algorithms," *Artificial Intelligence in Medicine*, vol. 119, 2021.
- [16] R. Islam, Md. Nahiduzzaman, "Complex features extraction with deep learning model for the detection of COVID19 from CT scan images using ensemble-based machine learning approach," *Expert Systems with Applications*, vol. 195, 2022.
- [17] O. Russakovsky, J. Deng, H. Su, et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no.3, pp. 211–252, 2015.
- [18] H. Li, S. Zhuang, D. Li, J. Zhao, Y. Ma. "Benign and malignant classification of mammogram images based on deep learning," *Biomedical Signal Processing and Control*, vol. 51, pp. 347-354, 2019.
- [19] S. Vallabhajosyula, V. Sistla, V. K. K. Kolli," Transfer learning-based deep ensemble neural network for plant leaf disease detection," *Journal of Plant Diseases and Protection*, pp.545-558, 2021.
- [20] S.D. Deb, R.K. Jha, "Covid-19 detection from chest x-ray images using ensemble of cnn models," *2020 International Conference on Power Instrumentation, Control and Computing (PICC)*, 1–5. 2020.
- [21] J.P. Cohen, P. Morrison, L. Dao, K. Roth, T.Q. Duong, M. Ghassemi, "Covid-19 image data collection: Prospective predictions are the future 2020," arXiv preprint arXiv: 2006.1198, 2020.
- [22] W. Linda, Z. Quiu and A. Wong, "Tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images," *Journal of Network and Computer Applications*, vol. 10, pp. 19549, 2020.
- [23] M.R. Lascu, "Deep Learning in Classification of Covid 19 Coronavirus, Pneumonia and Healthy Lungs on CXR and CT Images," *Journal of Medical and Biological Engineering*, vol. 41, pp.514-522, 2021.

- [24] W. Wang, X. Huang, J. Li, P. Zhang and X. Wang, "Detecting COVID-19 patients in X-ray images based on MAI-nets," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 1607–1616, 2021.
- [25] W. Wang, H. Liu, J. Li, H. Nie and X. Wang, "Using CFW-net deep learning models for X-ray images to detect COVID-19 patients," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 199–207, 2021.
- [26] A.L. Khan, J.L. Shah and M.M. Bhat, "Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images," *Computer Methods and Programs in Biomedicine*, vol. 196, pp. 105581, 2020.
- [27] T. Ozturk, M. Talo, E.A. Yildirim, U.B. Baloglu, O. Yildirim, U.R. Acharya, "Automated detection of covid-19 cases using deep neural networks with x-ray images," *Computers in Biology and Medicine*, vol. 121, pp. 103792, 2020.
- [28] I.D. Apostolopoulos and T.A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, vol. 43, pp. 635–640, 2020.
- [29] M. Rahimzadeh and A. Attar, "Modified deep convolutional neural network for detecting Covid-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2," *Informatics in Medicine Unlocked*, vol.19, pp. 100360, 2020.
- [30] T. Mahmud, M.A. Rahman and S.A. Fattah, "CovXNet: a multi-dilation convolutional neural network for automatic Covid-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization," *Computers in Biology and Medicine*, vol.122, pp. 103869, 2020.

# An Approach for Audio-Visual Content Understanding of Video using Multimodal Deep Learning Methodology

 Emre Beray Boztepe<sup>1</sup>,  Bedirhan Karakaya<sup>2</sup>,  Bahadır Karasulu<sup>3</sup>,  İsmet Ünlü<sup>4</sup>

<sup>1</sup>Corresponding Author; Çanakkale Onsekiz Mart University, Department of Computer Engineering; berayboztepe@outlook.com

<sup>2</sup>Çanakkale Onsekiz Mart University, Department of Computer Engineering; bedirhankrky@gmail.com

<sup>3</sup>Çanakkale Onsekiz Mart University, Department of Computer Engineering; bahadirkarasulu@comu.edu.tr

<sup>4</sup>Çanakkale Onsekiz Mart University, Department of Computer Engineering; ismetyny@gmail.com

Received 02 July 2022; Revised 5 July 2022; Accepted 6 July 2022; Published online August 2022

## Abstract

This study contains an approach for recognizing the sound environment class from a video to understand the spoken content with its sentimental context via some sort of analysis that is achieved by the processing of audio-visual content using multimodal deep learning methodology. This approach begins with cutting the parts of a given video which the most action happened by using deep learning and this cutted parts get concanarated as a new video clip. With the help of a deep learning network model which was trained before for sound recognition, a sound prediction process takes place. The model was trained by using different sound clips of ten different categories to predict sound classes. These categories have been selected by where the action could have happened the most. Then, to strengthen the result of sound recognition if there is a speech in the new video, this speech has been taken. By using Natural Language Processing (NLP) and Named Entity Recognition (NER) this speech has been categorized according to if the word of a speech has connotation of any of the ten categories. Sentiment analysis and Apriori Algorithm from Association Rule Mining (ARM) processes are preceded by identifying the frequent categories in the concanarated video and helps us to define the relationship between the categories owned. According to the highest performance evaluation values from our experiments, the accuracy for sound environment recognition for a given video's processed scene is 70%, average Bilingual Evaluation Understudy (BLEU) score for speech to text with VOSK speech recognition toolkit's English language model is 90% on average and for Turkish language model is 81% on average. Discussion and conclusion based on scientific findings are included in our study.

**Keywords:** Multimodal Deep Learning, Association Rule Mining, Named Entity Recognition, Natural Language Processing

## 1. Introduction

Nowadays, decomposition of various environmental sounds for environment recognition has gained popularity. Various background sounds in videos could be classified with high success with deep learning and machine learning techniques. In this way, semantically enriched video scenes can be depicted. Also, different techniques have been used to strengthen the result from sound recognition to recognize the environment well and increase the accuracy which was gained from trained models for the environment. Sound recognition, which emerged with the development of technology, is a technology based on the analysis of the audio signal [1]. Environmental sound recognition is a technology which is used for detecting sounds that define everything about the environment such as animal sounds, human sounds, car sounds and so on. Technology for developing new techniques for recognizing environmental sounds by using deep learning has been improved so much nowadays. This technology is important to be used for different sectors such as cinema, different departments such as police and fire departments, and etc. It could help to catch the criminals by detecting the environment if there is a video recording with sound. Different fields such as multimedia systems, it is used for detection of sound scene events [2] and such as sentiment analysis techniques, it is used for audio emotional state classification using audio features [3] and so on.

There are different techniques to build a model for speech recognition by using deep learning. Getting a spectrogram from an audio signal and building a Convolutional Neural Network (CNN) model using

spectrograms as inputs is one of the most used techniques [4]. Adding Recurrent Neural Networks (RNN) after CNN to create a Convolutional Recurrent Neural Network (CRNN) model is another technique that could have been used for this problem [5].

In multimodal deep learning methodology, data is obtained from different sources. Furthermore, these data can be used to learn important features over multiple modalities. In this way, a representation would be constructed between different modalities (e.g., audio, visual and possibly textual data) in a shared manner as well. Instead of using single mode data for understanding of audio-visual content, the use of multiple modalities is expected to result in highest performance. In machine learning and deep learning, representation learning is a process that automatically proceeds for a learning system to discover the representations needed for feature detection, selection or classification from given data. With the help of the representation learning, the semantic gap between low level features and semantically rich high level features would be filled to achieve a successful understanding of content [6].

Audio data augmentation techniques can be built with some items such as an attribute type consisting of coefficients in the Mel scale Mel-frequency cepstral coefficient (MFCC) [2] type feature extraction method, Chroma [3] type feature extraction method, adding Gaussian noise to the audio track, shifting the audio track, and so on can be used to get an augmented data [2]. Using different Natural Language Processing (NLP) techniques to strengthen the result of sound recognition prediction can be beneficial to increase the accuracy because the raw sound can not be enough for detecting the environment. Displaying audio signals that are in the same frequencies is easy but building a model to detect sound from them is not easy enough than building image recognition models. These NLP techniques help us to do some processes. For example, the tokenization method which is used to get all the words individually from a sentence [7, 8].

The Part of Speech Tagging (POS tagger) method is another technique for the process of marking up a word in a text (i.e., corpus) as corresponding to a particular part of speech, based on both its definition and its context. A simplified form of this is commonly taught to everyone starting from childhood, in the identification of words as nouns, verbs, adjectives, adverbs, etc. The POS tagger technique is a method to help to define what is the job of the word in the sentence [9]. In addition, the stemming method which is another technique which is used to get stems of a word and some other methods could be given as an example [8]. Implementation of these techniques is different according to language that has been used.

The Named Entity Recognition (NER) is important to find the relation between the word and the category in which the word belongs to the related category [10]. For example, if someone heard some words from a sentence like "*penalty kick*", everyone would think that it is about football. NER helps to categorize these words to its related category. There are a lot of ways to do this process. In this scope, a novel dataset could be created by searching all the words and phrases and collecting them somewhere (in a spreadsheet file, database, and etc.), some libraries from some programming languages could be used or pre trained files could be found. But in order to implement this process, the first thing needed to do is to implement NLP techniques to the words in a sentence. Otherwise, all conjugations of the word have to be added to the created NER file to be checked. But that also means more time needed to search related words and create this NER file and more time to search if the word that is found is in the NER file or not.

The Association Rule Mining (ARM) is a kind of process used to determine the relation between the given categories [11]. It is mostly used in the marketing sector to determine the relation between the items that are shopped. International e-commerce and entertainment companies are using this method a lot and everyone could see that technique like "*this another item is bought with the item you bought, would you prefer to buy this item too?*" or "*this movie is watched by people who watched the movie you watch*". The implementation explained at this section of this work is to determine to make a comment like "*if this category is in this video, the rate of another category to be in this video increases this much*". This way could be used to collect data of categories and suggest a video to users about a video that is like "*the user's video is about this category and the user may also want to view this category too*". In our study, sound environment recognition for a given video's processed scene is

achieved with high accuracy. In addition, the speech to text process of our approach based on the VOSK offline speech recognition [12] toolkit's English language model and Turkish language model have highest performance results that prove our approach's superiority. Of the three English VOSK models, the second model was chosen by us to achieve optimized performance. This model is more accurate than the first, but faster than the most complex. There is just one VOSK model for the Turkish language. This model is used for getting the Turkish speech.

Our main contribution to the literature is to be able to realize and prove the content understanding mechanism for determination of the sentiment from spoken content based on the multimodal information obtained from multilingual audio-visual content extracted using a deep learning approach without any other manual intervention.

In addition, the second contribution of our study is applying sound scene classification and recognition to reach a refined sentiment analysis from spoken content that related sentences are taken from the processed video clip using speech recognition, natural language processing (NLP) and ARM techniques. In the literature, there are several analysis methods for sentimental context that are pre-trained and used with some published studies [13, 14]. Everyone could choose any one of this kind of pre-trained model by respect to its performance, by its language type, by its emotional representation styles (negative/positive speech, angry/happy speech etc.) and by its model size. These kinds of models are used to determine the given sentence's sentiment by its determination of any positive sentence, negative sentence or neutral sentence. For example, when the video is related to the football category and its spoken content covers news about a team called "*Galatasaray*" from the Turkish Super League a loss they have taken from yesterday's match then sentiment analysis model helps to determine the speech from the news. Furthermore, the model can say that "*this sentence is a negative sentence related to the football category*". It actually is not about finding the category or strengthening the found category from sound recognition, but it is a technique to find out if the model has negativity or positivity. With this way, it can be said about the example that is given, this actioned part from a video has a negativity. At a glance, it is a challenging issue for real world problems.

The remaining sections are organized as follows. Literature review is given as the second section. In the third section, materials and methods are presented with the details of video summarization, sound event detection and speech recognition, and also, multimodal deep learning methodology, system integration and Graphical User Interface (GUI) are given in the section as well. Experimental results are introduced in the fourth section which covers datasets used in the experiments, performance evaluation methodology, our experimental results and proof of concept of our approach. In the last section, the discussion and conclusion of scientific findings of our study are included to emphasize the contributions of our work.

## 2. Related Work

At a glance, there is a huge potential to extract useful information from raw data in contemporary multimedia systems. Multimedia is a term that covers text, audio, image and video as given as a processable object. In the literature, deep learning is used to process and extract these kinds of information for building a consequent knowledge about the solution of a real world problem using the above mentioned multimedia components. Nowadays, sound event recognition, speech recognition, automatic speaker recognition and speech to text translation are especially popular issues in the deep learning and machine learning research areas. With its well constructed paradigms. Deep learning provides an automatized feature extraction and selection scope belonging to its layered structure which proves a refined learning called representative learning [15, 16].

In this scope, video is an integrated multimedia component that involves audio and consecutive images as video frames. These frames mostly include some important information about actions or events. Since these actions or events may be grouped into one or more frame groups (i.e., video shot), a deep learning system can be easily used in these frames to detect and classify them from certain mechanisms. Audio features based sound event recognition is relatively less complex than visual feature based action recognition. For sound event detection and speaker recognition, the sound

characteristics obtained from sound samples assigned to the same sound class are compared with each other. If possible, the audio features of the sound samples assigned to a different class are desired to be much different than each other. Audio data representation is the audio signal content consisting of numerical values expressed as features such as MFCC, audio spectrum, and etc. These features are based on displaying the signal energy, frequency distribution, and signal change over time [2].

In the literature, the process of audio event recognition in a given audio scene is taken from a couple of features which feed the deep learning mechanism as low level features. Since the semantic gap, this kind of pure low level feature usage remains quite limited as a tool to interpret the content for complete understanding. Shel et al. [17], proposed a model which is one of the examples for multimodal-based video prediction using deep learning. The idea behind their work is to create CNN [4] features from the frames of the video and give them to generated Long Short Term Memory (LSTM) [14-16] layers and to create MFCC features from the audio tracks of these video frames and give them to a generated different LSTM layers. Therefore, multimodal feature learning is applied into these two outputs and the classification process is done at the end. Jiang's dataset [18] is used for this model and 78.6% of accuracy is gained for the test dataset. Agrawal et al., proposed a CNN model using the ESC-50 dataset [19]. Only augmenting the data by using MFCC feature extraction, 68.40% of accuracy gained. Mushtaq et al., used different data augmentation techniques and combined them to build multiple models [20]. One of the data augmentation techniques that is used is Chroma [3] with Short Time Fourier Transform (STFT). Using a model which is pre-trained for deep learning is called Dense161 [21] and freezing/unfreezing layers method with using only Chroma feature based data augmentation technique, 46.80% of accuracy is gained for ESC-50 dataset and 70.50% of accuracy is gained for ESC-10 dataset. The ESC-10 dataset is created by using ten different audio classes selected from the ESC-50 dataset. To give more example to proposed model about environmental sound recognition with using ESC-50 or ESC-10 dataset, Khamparia et al. [22], proposed a model by getting spectrograms with log-mel features and giving them into a model constructed with Tensor Deep Stacking Network (TDSN) [22], 56% of accuracy gained for ESC-10 dataset [22]. The ESC-50 dataset's creator Piczak proposed a model by getting log-mel features and giving them to a CNN model in a related study, thus 81.5% of accuracy was gained from the trained model by using the ESC-10 dataset which is a reduced version of ESC-50 [23].

In order to bridge the semantic gap between low level features and the high level semantically rich features as human perception of action/event, recent works have introduced an intermediate representation between the audio features and the video's affective content. These methods construct representations based on audio features and NLP based on the speech to text translated transcription which employ these representations for affective content analysis and understanding of videos. Khalil et al. [24], proposed a model to recognize the emotion from input speech signals. Both machine learning and deep learning techniques are used but a model which is constructed with the Deep Convolutional Neural Network (DCNN) [15, 16] is performed much better than other machine learning algorithms.

In the literature, video summarization is a process which selects representative video frames to summarize key events present in the entire video. In our study, video summarization is used to localize the positions of most important actions and/or events in full length video. This process is similar to a combinatorial optimization problem which selects video frames due to action-basis to group them into meaningful segments as video clip. These kinds of video clips are a shorter form of original full length video, but it is more refined to represent a more compact and concise form of action/events as well. As told above, video summarization is addressed as an optimization problem in many contemporary studies [25-28] as well.

As told above, the accuracy from the proposed model of Piczak [23] was gained 81% of accuracy by only training the model using the ESC-10 dataset. At this point of view, unlike Piczak's proposed model, our model is trained with seven categories from the ESC-50 dataset which could be different categories from the ESC-10 dataset, and three extra categories which are designed and collected by us with new samples. Our gathered dataset is more complex than the ESC-10 dataset. Our work is proposed with a quite different performance evaluation scope than Piczak. Therefore, unlike all the

unimodal (e.g., audio signal, or spectrogram as still image) works that are explained about this section, our model strengthens the result of environmental sound classification with the support of NLP [7] techniques and different sound classes are used in the experiments.

The remaining sections explain the details about our methodology, and also, the different parts than other studies or superior sides of our study with related experiments.

### 3. Materials and Method

In order to build the multimodal deep learning infrastructure with audio, video and image processing, that has been described in this work, multiple libraries have been used that are created to use with the Python language [29]. These libraries have been selected by their properties, ease of use and some other criterias. First of all, to build all deep learning models, Tensorflow [30] and Keras [31] libraries have been used. And also, a pre-trained model which is ready to use for everyone in Keras library to build a deep learning architectural structure, MobileNetV2 [32] has been used as a base model for sound event recognition. Some libraries that allowed us to do some operating systems operations like creating a folder, deleting a folder, copying and pasting files, reading images and videos from folder and so on such as OS [29], Glob [29] and some libraries that help to do machine learning operations, mathematical operations, plotting operations, NLP operations, video operations and so on such as Pandas [33], Matplotlib [34], Seaborn [35], Numpy [36], SpaCy [37] and Pydub [38] have been used which their detailed information can be seen from Python website [29]. The OpenCV [39] library has been used to do reshaping, resizing and some other operations for frames of video clip as given as image sequence. To generate audio and video files, Moviepy [40] and Pydub [38] libraries have been used. The VOSK [12] library and a model from this library have been used to get the part of speech from a given video. For data augmentation methods, along with Tensorflow, OpenCV libraries, Librosa [41] library also have been used. For NLP and sentiment analysis operations, NLTK [42], SpaCy [37] and re [29] (Regular Expression Operations) libraries have been used to prepare text data for NLP methods and Transformers [43] library has been used to use Bidirectional Encoder Representations from Transformers (BERT) [43] models to do sentiment analysis. For NER operations, to score the similarity between the word in the sentence and the word in NER dataset, DiffliB [44], Zeyrek [45] and Jellyfish [46] libraries have been used. By using all these libraries, a suitable model and its infrastructure has been generated. To create Graphical User Interface (GUI), Gradio [47] library has been used.

The underlying methodology of our approach is described as follows. First of all, a video summarization technique has been generated to summarize the video to get scored action points from extracted video shots with related key frames of given video. After that, the sound event detection technique has been applied to do speech recognition from the above mentioned summarized video which is given as a short video clip with sound. The multimodal deep learning approach has been constructed to get speech from the video and do some NLP operations onto that speech to strengthen the result of sound recognition and to perform sentiment analysis. In the end, video summarization, sound event detection, NLP and sentiment analysis techniques have been integrated into a compact generated GUI that is presented for use by end users with online interaction capabilities as well.

#### 3.1 Dataset

In order to train the sound recognition model, data from ten different categories have been used. Seven of these categories have been chosen from a dataset that has been created using environmental sounds. The dataset that has been used to train the model is the Environmental Sound Classification called ESC-50 [48] dataset which is a labeled collection of 2000 environmental audio recordings. The dataset has five different main topic categories and a total fifty sound classes. These main categories are “Animals”, “Natural soundscapes & water sounds”, “Human, non-speech sounds”, “Interior/Domestic Sounds”, “Exterior/Urban noises”. Among these main topic categories, seven different categories from these main subject categories were chosen according to their evocation of places. These chosen sound categories are “Chirping Birds”, “Rain”, “Sea Waves” from “Natural soundscapes & water



sounds” main category, “Clapping”, “Laughing”, “Footsteps” from “Human, non-speech sounds” and “Car Horn” sound from “Exterior/Urban Noises” main category. For example, hearing chirping birds sounds may evoke natural forest to the people, hearing sea waves sounds may evoke seaside to people, hearing clapping may evoke somewhere where people are celebrating something in a place such as theater, concert etc. All of the sound clips are made of 5000 milliseconds and all of these sound categories have forty different sound clips. To expand the sound event categories of ESC-50, we collected some new audio data to generate three new sound event categories which contain different audio samples from ESC-50 original data. These three categories that have been used to train the model are “Football”, “Racing” and “Human Speech”. In order to collect data for “Football”, different sound clips have been cut such as fan chants, goal celebrating, missing goal reaction from fans, hit the ball sound, referee whistling and some other things. By considering the collection of samples of fan chants, the model finds a similarity between the “Human Speech” and “Football” categories. In order to collect “Racing” sounds, different sound clips have been cut as it was done for “Football” such as different engine sounds from different cars from racings, car sound from a long distance and short distance, sounds from pit stops and so on. And last, for the “Human Speech” sounds have been collected from a dataset called Librispeech [49]. Librispeech has samples that are 1000 hours total. Forty different english speeches have been selected randomly from this dataset. All of these categories have forty samples, and all of the samples are 5000 milliseconds long. In total, there are 400 counts and 2000 seconds of samples. Each sound sample is a single channel (i.e., mono) audio clip. These data were used in the training process of our audio-visual content understanding processes.

For testing our audio-visual content understanding methodology, we used a real world dataset called CPSM “sports minute” dataset [50] which has 74 videos collected from Youtube [51] in total with 10 different sports activities involving two years worth news highlights about college sports. Also, the proportions of each activity and number of labels per clip are variable for this CPSM dataset [50]. For “Turkish” videos to test the Turkish VOSK [12] model, “MediaSpeech” dataset has been used [52]. This dataset is generated by using short speech videos from Youtube [51] and some other websites for media videos by using news videos which are mostly about politics. The dataset has 4 different languages and Turkish is one of them. There are a total of 10 hours of speech for each language.

Above mentioned datasets are used in our experiments to achieve a reliable sound event recognition for analyzing the action topics in audio-visual content of given video which is based on our deep learning approach to understand the spoken content with its emotional scope.

### 3.2 Video Summarization

By considering the optimization methodology, the video summarization process is treated as an abstraction method which selects representative video frames to summarize key events present in the entire video. In the literature, recent studies on video summarization have learned how to select informative video subsets (i.e., video clips rearranged with selected video frames) close to summaries generated barehand by humans [53]. In the last few years, several studies have been presented [54,57] to solve the video summarization problem, and also, this problem is treated as a combinatorial and constrained optimization problem with the use of labeled full length videos. By detecting changes in visual features of video frames, Kernel Temporal Segmentation (KTS) produces segment boundaries. Video segment is built up with a bunch of video frames. Video shot involves above mentioned video frames grouped as an action related video segment. In this way, a video segment tends to be long if visual features do not change considerably [58, 59].

In our study with deep learning scope, in order to produce video summarization, it has been firstly done to get every frame from the video. These video frames have been reshaped as a tensor as (224, 224, 3) to be processed. Therefore, with the help of a pre-trained deep learning model called VGG19 [60], related visual features extracted from these video frames have been used to determine the change points in the whole video as treated as indicators of the variations of action or event locations [61]. The KTS process [62] has been applied to these obtained features for temporal segmentation. While ensuring that the summary length does not exceed a defined limit, video shots that involve actions are selected to generate a summary by maximizing the total scores. The summary length limit is circa 10%

to 15% of the full video length. Knapsack problem is a combinatorial optimization problem in computer science, which is known as NP-hard. The 0/1 Knapsack problem's dynamic programming solution is a more preferred solution in the literature for optimized video summarization as well. Above mentioned maximization step in the video summarization process is based on the 0/1 Knapsack problem. A near-optimal solution via dynamic programming can usually be obtained [63]. In the manner of combinatorics, dynamic programming requires an optimal substructure and overlapping sub-problems. These are present in the 0/1 Knapsack problem [64].

To ensure the selection reliability, the trained models' group with two parts as an integrated meta model was used to compute the necessary scores. The first part of the above mentioned model was used to extract visual features to comply with temporal segmentation as change points. The second part as a frame-relation learner part was to predict the frame selection probabilities as importance scores. The video shot level scores are computed by averaging frame level scores within the same video shot as well. By using both obtained importance scores and detected change points (i.e., temporal segmentation) within the scope of KTS process, thus related video segments needed for video summarization have been chosen. For long-term temporal dependencies between video frames and shots located far away each other, the above mentioned model's frame-relation learner part is used to build data for dynamic programming (i.e., Knapsack problem) that this part of the model is based on the bidirectional Long-Short Term Memory (Bi-LSTM) [14-16] and its structure was built on two Bi-LSTM layers which have 256 and 128 neural nodes, respectively. These layers work as a processor for revealing the forward hidden states and backward hidden states represented in a sequence of frames of video. Thus, this part learns the frame relationship with scores. At the end of this structure, a dense layer with 1024 neural nodes was used to classify the frames into some frame groups with frame probabilities. The results form these frame groups and video segments feed the dynamic programming infrastructure for solving the problem of Knapsack such as those given as input for Knapsack part. Thus, the whole video is summarized with the use of results as selected frames as "one" or not selected as "zero" by the dynamic programming solution of Knapsack.

In order to construct the video summarization in dynamic memory allocations, full length video has been split into smaller pieces. The reason for it was to split video into pieces to get optimal summaries separately, and thus, these summaries are concatenated to get a novel summarized video as a video clip as well. For the testing of video summarization problem solving processes, there are a bunch of video dataset with annotations in the literature. The SumMe is a video summarization dataset in which some unedited or minimally edited videos are presented as seen as 25 personal videos obtained from YouTube [51]. There are 15 to 18 reference summaries for each video which are individually annotated by human annotators [65]. The TVSum is another video summarization dataset in the literature. It has 50 YouTube videos with metadata that each one of videos is presented with a title and a category label. For every two seconds of each video, human annotated importance scores are provided to construct the reference summaries with a predefined length are generated. Therefore, as in the generated summary, these videos are separated into short video segments. To obtain a segment-level importance score, the importance scores within a video segment are averaged. At the end, a reference summary is generated by finding a subset of segments that maximizes the total importance score in the summary [61, 63].

In our study, we used the above mentioned video summarization meta model to test and validate our approach for summarizing the full (long) length videos that a summarized short video clip presents the important parts involving action/event related to the topic of spoken content. With these video summarization datasets, the performance of summarization was tested and validated that the overall success is used to show as indicator of optimal selection of frames for video clip involving salient action or events. The performance evaluation is based on the comparison between automatic summaries and ground truth summaries. In our study, an average performance score (i.e., *F-metric*) [3] was obtained as equals to 51% with the summarization tests on above mentioned TvSum and SumMe datasets with the integrated meta model's VGG19 [60], Bi-LSTM [14-16] and dynamic programming parts as told above. This performance score is the same as given in Zhou et al. [61] study that they obtained this kind of average score in their study as equal to 50% (i.e., for SumMe as 42.1% and TVSum as 58.1%). The detail of this metric is given in the following sections. Since the experiments

show that our video summarization stage of the complete audio-visual content understanding approach (as given as “*Proof of Concept*”) is more accurate to detect the action/event location in a given video, the summarized video as a video clip is more efficient in terms of computational complexities and space complexities.

### 3.3 Sound Event Detection and Speech Recognition

The preparation part is the most challenging part in this multimodal data processing and underlying learning process. With a raw form, classifying the labels of chosen sound classes is very hard by using deep learning. Because spectrograms of different sound clips from different classes may show a similarity which is very difficult to separate them because they are very similar to each other in manner of feature extraction and process. In order to use the model efficiently, different techniques might have applied to the spectrogram such as attention based technique. In this model, the multiple masking [67] technique has been applied to increase the distribution of the dataset. At first, with the help of the Tensorflow library, data in process that is in an audio shape in the dataset has been transformed into spectrogram shape as an image. These spectrograms have been generated by using log-mel features of the audio. Then, multiple data augmentation methods have been used to increase the data variations in amount and presentation for the dataset, and multiple models have been also generated. To give examples for these data augmentation methods, as seen in Figure 1, applying MFCC type feature extraction method [2], applying Chroma type feature extraction method [3], applying Contrast Limited Adaptive Histogram Equalization (CLAHE) [66] to spectrogram images, applying multiple masking to these images, adding Gaussian noise (background noise) [3] to audio segments, adding shifting method to audio segments, getting log-mel spectrograms using Librosa library [41], applying some operations to these images such as flipping, rotating, sharpening and all combinations of these method have been used. To test these rebuilt and extended datasets, multiple programmatic structures have been generated. If an overfitting problem for model training has occurred, the programmatic structure has been regenerated in order to prevent this overfitting. The best result gained from multiple training processes, audio dataset augmented by background noise and image dataset increased by CLAHE and multiple masking methods. In other words, if it is said in an objective evaluation manner in terms of performance metrics for machine/deep learning processes that have been used for the training process, the highest accuracy and the lowest loss has been gained from this training process.

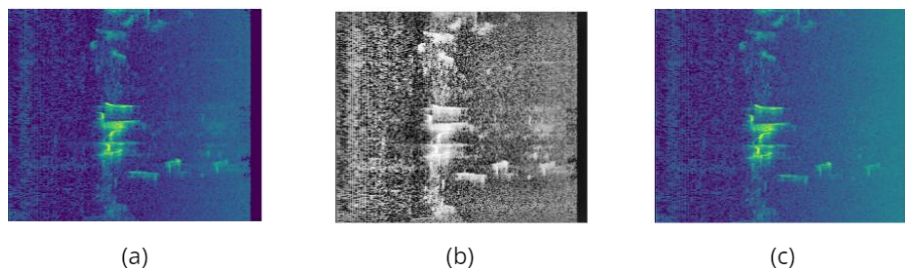


Figure 1 An example of a spectrogram from the Chirping Birds category (a). CLAHE applied (b), Gaussian Noise added (c)

MFCC feature extraction is a method which is a very popular method to be used for feature extraction of an audio file. One of the reasons why MFCC is used so much is that MFCC coefficients are much less affected by changes, audio wave structure [68]. But it is often used in speaker recognition applications. MFCC coefficients are obtained by de-correlating the yield log energies of a channel bank which comprises triangular channels, straightly dispersed on the Mel recurrence scale [69]. There are some libraries that help to apply MFCC type feature extraction to spectrograms, but Librosa library is the most used one.

Chroma feature extraction is a method which is used for obtaining the quality of a pitch class alluding to the color of a musical pitch of a given audio file which has a sampling rate [2, 3]. This pitch can be decomposed into an octave invariant value called chroma, and there is a pitch height value which shows the octave the pitch is already in. Each frame of audio is windowed, by this scope, a short-time

audio window is presented by Chroma that window has the harmonic content such as keys or chords, and etc. Chroma's related feature vector is extricated from the magnitude spectrum with the help of a short time fourier transform (STFT), Constant-Q transform (CQT), Chroma Energy Normalised (CENS), respectively [2, 3, 70].

CLAHE is a different strategy which is used in images. It is used to improve image contrast. To limit the contrast amplification, CLAHE aims to avoid amplification. By considering the transformation of slope function, amplification contrast around the given pixel value is determined. By clipping, the CLAHE limits introduce amplification. It takes a basis which has a predefined value in the histogram prior to cumulative distribution function (CDF) calculation [66]. The main idea of applying it is to improve the contrast of the parts where audio-wave is denoted in the histogram, meanwhile decreasing the contrast of the other parts. With this way, the model can be focused more to the parts where it is wanted the model to focus. The OpenCV [39] library has been used to apply CLAHE to spectrograms. In the manner of sound recognition, a deep learning model is constructed to classify and determine the sound classes that it is based on above mentioned techniques and processes. This model and its multimodal aspect's usage is explained at the following sections of this study as well.

Audio shifting [3] is another method which is used for data augmentation by shifting the audio signal content  $n$  steps ( $n$  denotes time) and saves shifted audio file as a new audio file. Overflowing audio data from the end of this audio file is cut and added to the beginning of the shifted audio file. With this way, limited data is being increased and with not using the same audio file, possible overfitting problem may be prevented.

Flipping, sharpening, and rotating a spectrogram are the techniques which are used to increase the amount of the dataset. Obtained spectrograms are shown as matrices while programming. Each spectrogram denotes the  $m \times n$  matrix by its width and height. If it is said a spectrogram is shown as a  $4 \times 1$  matrix  $[1, 1, 0, 0]$ , flipping this matrix will be resulted as  $[0, 0, 1, 1]$  which means horizontally reversed. Rotating an image is used to rotate an image by a given degree. A matrix that has  $x$  and  $y$  is the image. Multiplying a matrix that has sine and cosine with the image matrix is a rotated matrix that shows the degree of how many degrees the image is rotated [71]. Sharpening is a method to remove blur, enhancing details and dehazing. There are some methods that help to sharpen an image. The used sharpen matrix is given as  $[-1, -1, -1; -1, 10, -1; -1, -1, -1]$  as a two dimensional matrix. Finally, here is Figure 2 that has an example that each sharpening, flipping and rotating are applied to a spectrogram from the chirping birds category. With the help of trigonometric functions, a rotating process is applied where the rotating angle theta is 45 degrees. All of these processes have been applied using OpenCV library functions and all of these processes have been applied after applying CLAHE method to every spectrogram which can be seen from Figure 2 below.

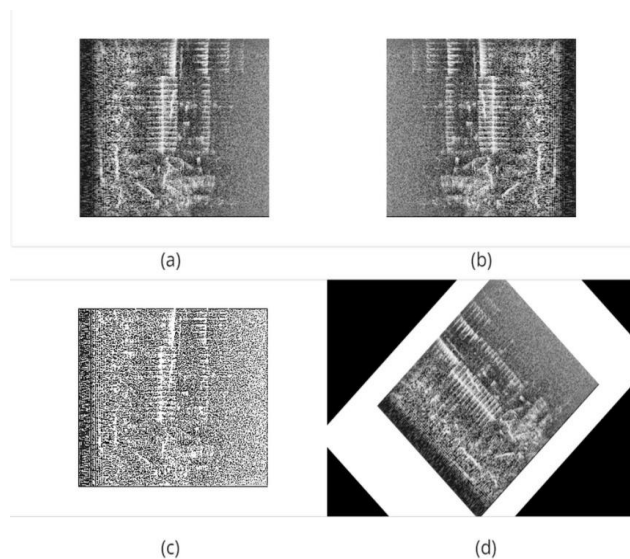


Figure 2 Examples of every method applied to a spectrogram. (a) denotes normal spectrogram, (b) denotes flipped spectrogram, (c) denotes sharpened spectrogram and (d) denotes rotated spectrogram

At first, after the dataset has been gained, background noise [3] has been added to all of the audio files and has been saved as another audio file in order to augment the dataset. This background noise audio clip is a clip that has Gaussian noise in it for one minute long. In order to add this clip to audios, a 5000 millisecond long sample has been taken from the Gaussian noise clip. Then, this clip has been applied to audio segments and has been saved by using Librosa [41] library. With this process, the dataset has increased two times of its original amount. The Gaussian noise formula can be seen as denoted by Eq. 1 as below where  $z$  represents the gray level,  $\mu$  represents the mean gray value and  $\sigma$  represents its standard deviation.

$$\text{GaussianNoise}(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad (1)$$

Another method which has been used to augment data is applying the multiple masking method. It is a method which is used for data augmentation and to get a sort of mixed data. It is split into two steps as applying frequency and time mask by the given parameters (i.e., consecutive time frames  $T$ , consecutive frequency bins  $F$ ) as the first step and mixing them as the second. [67]. The frequency mask is based on a masking method which is applied to the spectrogram which is in a frequency zone. With this method, frequency channels are masked. This mask is chosen from a random uniform distribution between 0 to  $F$  frequency value. However, the time mask is based on a masking method which is applied for masking consecutive time steps. This mask is chosen from a random uniform distribution between the number of frames 0 to  $T$  frames as presented as time steps in milliseconds. According to a parameter that denotes how many times this frequency and time mask will be applied ( $t_0, f_0$ ) these masking processes are applied. After parameters are given, the distribution for training purposes is expanded by drawing the shear plane from random parts. There finishes the first step. Thinking about a spectrogram that has been generated by its frequency/time features, horizontal shear plane denotes time mask and vertical shear plane denotes frequency mask. Given parameters to perform the multiple masking process that have been chosen from different trials can be shown as Table 1 below.

Table 1 Given parameters to perform Multiple Masking

$T$	$F$	$t_0$	$f_0$
24	36	2	2

The second step begins where the first step ends. This step is about mixing spectrograms which frequency and time masks have been applied as seen in Figure 3. The most important part about mixing spectrograms is to decide which categories have to be mixed. In other words, mixing which categories to augment the data helps to build the most efficient sound recognition model. In this part, categories which are shown the most similarity between their spectrograms are chosen to be mixed. For example “Footsteps” and “Chirping Birds” sounds are both a sound which their spectrograms have similarity in the same frequency. The other categories which have chosen to be mixed are: “car horn” - “clapping”, “football” - “racing”, “human speech”- “sea waves” and “rain” - “laughing”. After applying the multiple masking process, the amount of dataset has increased, and the distribution provided.

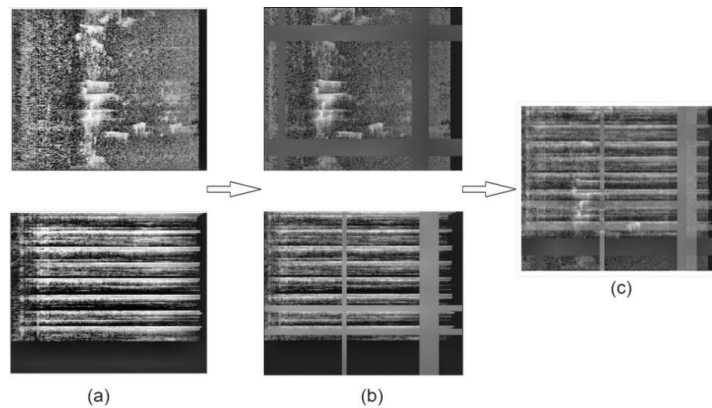


Figure 3 Example of applying multiple masking process in “Chirping birds”- “Footsteps” categories. (a) denotes normal spectrograms. (b) denotes multiple masking applied spectrograms and (c) denotes mixed spectrogram

### 3.4 Multimodal Deep Learning Methodology

The deep learning methodology has some superior properties among other machine learning paradigms [15]. For a learning system, representation learning is an important process that helps to discover the feature patterns in given data [72], and also, it automatically proceeds to discover the representations needed for feature detection, selection or classification from given data. For image based learning, the semantic gap means that there is a difference between primitive (low level) features of an image and semantic meanings that humans recognize from the image. With the help of the representation learning to achieve a successful understanding of content, the semantic gap between low level features and semantically rich high level features would be filled. In the deep learning concept, to fill the semantic gap between low level features such as color, texture, shape etc. and high level semantically rich features, representation learning is used. With the use of transfer learning, representation learning usually can produce a good result of content understanding. To solve a classification problem in solving another similar problem, transfer learning is used as a training style where the original problem’s knowledge and experience are used to solve this similar problem. In this scope, our base model for sound event recognition is constructed with MobileNetV2 [32] which is trained for by using 1000 classes of ImageNet dataset [73]. Its knowledge about learned salient features is transferred to our integrated model with the transfer learning process, and then we apply the fine tuning over this kind of base model as shown in Table 2.

Video summarization is used to generate a short synopsis. By the manner of selecting the video's most informative and important parts, this synopsis summarizes the video content. To form a shorter video, this summarized video is usually composed of a set of representative video frames (i.e., video’s keyframes), video fragments or video segments (i.e., video’s shots) that have been stitched in proper chronological order [54, 55, 59]. In our study, video summarization is built on the dynamic programming that solves a combinatorial optimization problem treated as a knapsack problem [64] for video’s action parts selection as given as frames in the video.

Our audio-visual content understanding methodology starts with obtaining the above mentioned summarized video and sound recognition process applied on this video. This methodology works with the following steps. At the first step as Stage I in Figure 4, the relevant video summarization datasets were used to train a meta deep learning model to detect parts of the video involving action to summarize the video. In the scope of KTS process [62], related video segments (i.e., video shots) needed for video summarization have been chosen based on change points that have been detected. This meta model was built with VGG19 [60] and Bi-LSTM [14-16] parts as told in the above sections of this study. After this kind of summarization training, the original full length video may come as an input when it is used to summarize with the use of shot level scores as average frames scores. After these processes, the summary has been obtained by using *Dynamic Programming Solution* for Knapsack Problem. Since action/event location detection is applied to this full length video to split the parts of this video as given input for Stage III in Figure 4 as told below that involve most important

actions or events in it, and then these multiple parts or just one splitted part get concanarated as a new video clip with selected new frames in a summarized manner. At the second step as Stage II in Figure 4, the audio dataset which is used for training the sound event recognition model, has been illustrated as spectrograms and increased by data augmentation techniques such as CLAHE [66], gaussian noise [3], and multiple masking [67]. With this way, the data has been prepared for training the model. Then, this audio dataset has been given to the MobileNetV2 based integrated deep learning model for training and the appropriate model has been produced to use for predicting sound classes. After this point, our content understanding process also has two complementary steps. At the first step of this process as Stage III in Figure 4, first, the original full length input video can be processed to be summarized as a video clip with newly chosen frames. These frames can also be chosen by producing a summary at Stage I using dynamic programming as well. Therefore, the sound recognition process is applied to this new video clip to recognize the sound classes of the environment. In order to strengthen the result of sound recognition, a content understanding process is defined to compile the speech from the video taken as plain text in the form of a script. After the prediction results, NER [10] techniques are applied to the script of speech that is taken from the new video clip to categorize the words or phrases from the speech if these words or phrases evoke any of the words or phrases that are prepared for each category owned. With this way, the sound recognition result is getting stronger. At the second step of this process as Stage IV in Figure 4, our contribution basis on the sentiment analysis is applied to the speech that is taken from the new video to determine whether the sentiment of the sentence is positive, negative or neutral. In Figure 4, a general presentation of our deep learning based audio-visual content understanding methodology is figured out which is explained as above.

For sound processing, we applied data augmentation and then data preprocessing methods, in order to generate an efficient model based on the training of a sound recognition model. The training data has been split as 70% train, 20% validation and 10% test part. Our integrated sound recognition network structure that has been generated programmatically to train the model can be shown as the structure as given in Table 2. The training process of the integrated sound event recognition model has been split into two subparts. First part is started by deploying a pre-trained model from Keras library MobileNetV2 to the generated model. MobileNetV2 is a convolutional neural network architecture with 53 layers equipped with a deep learning concept. At a glance, it is based on an inverted residual structure where the residual connections are between the bottleneck layers of a given network. Furthermore, it seeks to perform very well on mobile devices with low memory consumption. Its model works with a specified resolution for a given input image which has exactly 3 input channels as (224, 224, 3) [32].

Before starting the training process, the added layers from MobileNetV2 have been frozen. The model has been compiled by using batch size equals 8. In Table 2, the architectural structure of the integrated network model for sound event recognition can be seen.

Table 2 Architectural structure of the integrated sound event recognition network model

<b>MobileNetV2 and Convolutional Structure</b>
MobileNetV2
GlobalAveragePooling2D
512 Dense-ReLU
Dropout 0.5
256 Dense-ReLU
10 Dense-Softmax

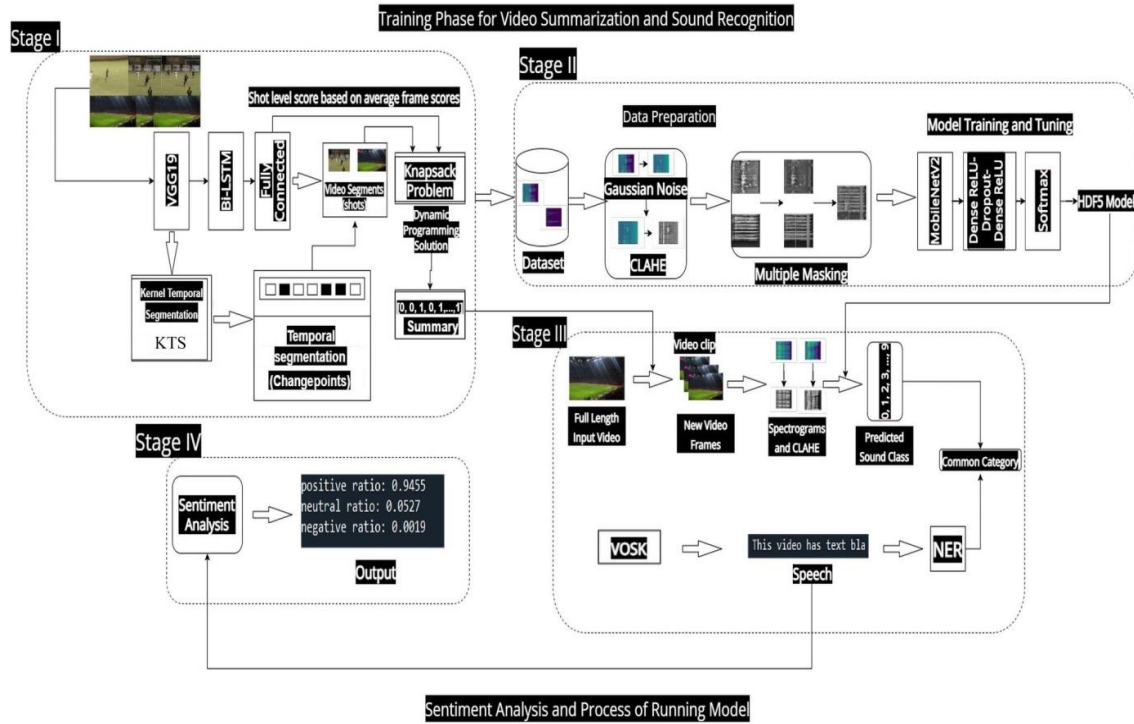


Figure 4 General representation of our deep learning based audio-visual content understanding methodology.

Adaptive Moment Estimation (ADAM) is an optimization method that keeps an exponentially decaying average past gradient to the similar momentum value. In this way, it computes adaptive learning rates for each parameter as well [74]. In our study, the ADAM method is used by us as optimization function and its determined first learning rate is given equals as  $2 * e^{-5}$ . Categorical cross entropy is used as the loss function and the training process lasts 100 epochs. Model's checkpoint record has been enabled so that the optimum part or training model can be taken before the overfitting problem occurs. Therefore, the *GlobalAveragePooling2D* method is applied to use average pooling on the spatial dimensions. A dense layer that has 512 neural nodes and Rectified Linear Unit (ReLU) activation function in it is added to this structure. ReLU function is a function which is commonly used in hidden layers [15, 16]. A dropout process has been applied to the connections between 512 neural nodes layer and 256 neural nodes layer to find the strong connections between each layer and to prevent possible overfitting problems. Another dense layer which has 256 neural nodes and again the ReLU activation function has been added. And the last dense layer which has 10 neural nodes denotes the output layer. This layer is where the classification happens. Each layer denotes categories owned. As having ten categories, there are 10 neural nodes at the output layer. The activation function that has been used at the output layer is the Softmax function which is commonly used in the output layer as the classifier. It turns a vector of  $K$  real values into a vector of  $K$  real values that sum to 1. This  $K$  term usually denotes the number of classes in the multiclass classifier. Its normalization properties ensure that all the output values of the function will sum to 1, The softmax function transforms the input values into values between 0 and 1. The equation of Softmax function can be seen in Eq. 2. In Eq. 2, the  $s_i$  values are the elements of the  $s$  input vector and can take any real value, The term on the bottom side of the Eq. 2 is the normalization term [75].

$$\text{Softmax}(\vec{s})_i = \frac{e^{(-s_i)}}{\sum_{j=1}^K e^{(s_j)}} \quad (2)$$

Beyond the MobileNetV2 as the second part of Table 2 denotes the fine tuning part of network architecture of our work. This time, the frozen MobileNetV2 layers have been unfrozen and the model's fine tuning process has begun. Only different parameter between training and tuning is the learning rate. In the tuning part, learning rate has been chosen as  $1 * e^{-5}$ . After 100 epochs of the tuning process, the model has been saved with h5 file extension. The Hierarchical Data Format version 5



(HDF5) is an open source file format, and also, it is given as "h5" formatted file in our study. It is capable of involving complex, large heterogeneous data [76]. In our experiments, the model has been trained and tuned about 200 epochs and after the processes have been done, the incoming result has not been a good result and the model has started to overfit. Also, a checkpoint has been used to save the model while the model has the minimum validation loss value. This checkpoint has saved the model structure with its weights in the tuning process while it has been in the 48th epochs. It could have been seen from the plots given at the experimental results section, the model also has started to overfit after 48th epochs. In this way, the training and tuning processes have taken 148 epochs in total.

With the help of NER [10] techniques, the result that is incoming from the sound recognition process has been strengthened. In order to use these techniques, the first thing that has been done is to get the data that evokes any of the categories owned. For example, "shoot", "goal" from "football" category, "Lewis Hamilton", "drifting" from "racing" category, "sparrow" from "chirping birds" category, "applause" from "clapping" category can be given as some examples. These data have been collected for both "English" and "Turkish" languages. There are at least 50 words and phrases that could evoke any of the categories owned. Gained text from the video that has been taken by using VOSK [12] library has been processed by some NLP [7] techniques. First, if the language is "English", with the help of SpaCy [37] library, the words from the text have been lemmatized which means all the suffixes from the word have been expelled. For example, the present tense's token "-s" has been expelled, and the past tense's token "-ed" will be expelled. Also, if the word is an irregular verb, the word has been transformed into its base form. All the letters from a word have been transformed into lower case. Characters such as ".", ",", etc. and numbers have been removed. It is a little bit different in "Turkish" language because there is a lack of libraries for Turkish NLP techniques and Turkish language is an agglutinative language. A word could have more than one stem and it could be a bit hard to find which stem created the word. Also, a new algorithm has been created for this case. This algorithm works as follows. First, with the help of Zeyrek [45] library, all the stems for a word have been found and words from the sentence have been split into a list. Among these stems, if the word has more than one stem, the lowest length between word and the stem most likely denotes the correct stem of the word. Otherwise, if the word has one stem, this stem has been selected as the correct stem of the word. After that, the same processes that have been done as the "English" sentence have been applied such as transforming lower case, expelling the characters and numbers. After applying the preparation of the sentence processes, the process of matching words and phrases from the sentence between words and phrases from the previously prepared NER dataset has been started. Matching words have been controlled like this: Because of having a maximum of six words of phrases in the NER dataset, the first word has been selected first and it has been controlled if it is matched word from the dataset. Then, the second word has been added to the first one and these two words of phrase have been controlled and this matching process has been continued like this starting from the first word and by getting six words of phrases. Later that, the second word has been selected and it has been controlled with the post-added five words until the last word. Applying this process to the "English" words is easy. Because words from the dataset and words from the sentence are without suffixes. So, the matching is easy. But it is hard for "Turkish" words. So, another algorithm has been generated for the Turkish ones. This algorithm depends on similarities between the word or the phrase from the sentence and the word or the phrase from the dataset. In order to calculate similarities, four different calculating methods have been used. First method is from the DiffLib [44] library and it is used to find the similarity between two strings by using similarity ratio [77]. It is calculated by using the formula of ratio as given in Eq. 3, where  $M$  denotes matches and  $T$  denotes the total number of elements.

$$\text{Similarity ratio} = 2.0 * \frac{M}{T} \quad (3)$$

The second method which is used to find the similarity score is by the help of Jellyfish [46] library, it is used to find the Jaro [78] distance between the words or the phrases. It depends on words' length and the number of matching characters. It returns 1 when all the characters between two strings are the same and in the same order. But, if all characters are the same but not in the same order or not all the characters are the same, transpositions between words are taken. In order to take transpositions, the

matching words are controlled. The half of the number of the unmatching words in the same index denotes transpositions [78]. The third method is used with the help of NLTK [42] library and it is used to calculate Bilingual Evaluation Understudy (BLEU) Score [79]. The method which is used to calculate BLEU Score is Sentence BLEU Score. To find the BLEU score, the number of candidate words that are in the reference word are divided with the number of total words in the candidate word. N grams term denotes that the words that are the same and match in the same order. Every match changes the N. When the possible proposing high-precision hypothesis translations are too short, to compensate them, a brevity penalty (BP) is given as calculated in Eq. 4 as below [80].

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \quad (4)$$

where the length of the corpus of hypothesis translations is denoted by  $c$  term, and  $r$  is the effective reference corpus length. The BLEU Score is given in Eq. 5 as it calculated as:

$$BLEU \text{ Score} = BP * \exp\left(\sum_{n=1}^N \omega_n \log p_n\right) \quad (5)$$

In Eq. 5, each modified N gram precision  $p_n$  is combined up to length N and can be weighted by specifying a weight  $\omega_n$ ; this formula is used in our study as well. In order to smooth that match, there are some smoothing functions. In this work, a related method for smoothing functions has been used. This function does that instead of scaling the similarity score between the words by using the multiplicative inverse of  $2^m$ , where the  $m$  term is word size, it uses the multiplicative inverse of the natural logarithm value of the length of the translation. And the last method as fourth has been used to find cosine similarity. It uses two vectors. The first one is generated by if the word from the first sentence, phrase etc. exists in the other sentence, the value of that index from the first vector becomes 1 else it becomes 0. The same process is made for the other vector [81]. Then, by the help of Eq. 6, the cosine similarity score is found. A and B denote these two vectors.

$$\text{Cosine similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \quad (6)$$

All of these similarity score metrics return a value between 0 and 1. In order to find the similarity between the word or the phrase from a sentence that has been gained from the video and the word or the phrase from the NER [10] dataset from these collected values from four different metrics, the minimum and the maximum of them have been selected in our study. Therefore, the harmonic mean has been calculated and if the calculated value is higher than the threshold that has been assigned, it is said that there is a similarity between them. This threshold value has been assigned as equals as 0.6. At the end, if the model finds a topic from the NER dataset which is spoken in the video, the model will give it as an output to detect the related category. This output is limited by showing only the top-first two of the most spoken topics.

With the help of sentiment analysis, the sentiment of the sentence has been determined. In order to determine it, previously prepared BERT [82] techniques with related models have been used. These BERT models have been taken from the HuggingFace [83] services. As a platform, HuggingFace helps users to create their own interface in which pre-trained deep learning models can be tested by other end-users. For “English”, a model which is named “Twitter Roberta Based Sentiment” has been used [84]. This model is generated by using 58 million tweets. For “Turkish”, a model named “Bert-base Turkish Sentiment Model” has been used [85]. For training this model, 5331 positive and 5331 negative sentences from different comments of different websites and a dataset which is generated for collecting tweets from twitter have been used [86]. In order to apply this model to sentences, first words from the sentence have been prepared. This preparation process includes having lower case letters, deleting all characters and numbers from words and deleting words which are tagged as stopwords. Stopwords are words that do not make sense in a sentence. Also, it is preferable to discard such words rather than occupy space in the sentence [87]. As an example of these words for “English” language “the”, “a”, “my” and for “Turkish” language “ama”, “her”, “ne”. After these processes, the sentiment analysis process has begun. Sentiment analysis model returns one of three labels based on whether the sentence is “Positive”, “Negative” or “Neutral”. But the result returns as probability. The probability of being “Positive”, “Neutral”, “Negative” can be seen as a result.

The Association Rule Mining (ARM) [11] is used in our study to determine the relationship between categories used in different videos. It is used for the model to comment. In order to apply ARM, the Apriori Algorithm is used [88]. The Apriori algorithm is one of the most used techniques for ARM. The reason for using Apriori, it is a successful algorithm to reveal the connection between categories. In order to apply Apriori, firstly, minimum support number equals 0.01 and minimum confidence number equals 0.2 has been selected. Support value of all categories has been found. If there is a value lower than minimum support value, this category has been disabled. Dual partnerships have been generated from remaining categories. If there is a partnership that has a value lower than minimum support value, this partnership has been disabled. Then, triple partnerships have been generated and so on. The algorithm goes like this to find multiple partnerships between categories from a dataset that has been generated before.

The experimental results of our study are presented in the following sections as well as some of them are also obtained with the use of graphical user interface (GUI) on the basis of open source Gradio interface. Gradio is a web based interface to deploy and test machine learning models with user interactions [47]. For testing purposes, our “*Proof of Concept*” demonstration is introduced with the Gradio web interface as given in the Github repository [89]. The end-users can effortlessly reach and use interactively from the web site of our GUI based application as given in the Github platform. It is called “*Audio-Visual Event Sentiment Analysis*” (AVESA) that works with sample videos and uploaded videos by end-users as well [89].

Our GUI is presented in two sections: the first one (i.e., left hand side at Figure 5) is a video input container to acquire video from a file, and the second one (i.e., right hand side at Figure 5) is able to show the experimental results and the output video as given in Figure 5 as given below. By dragging or clicking to the section for uploading video, the video can be uploaded. Therefore, the language should be selected from the section for selecting language. The language of GUI depends on the user’s computer language. By clicking the “*Yükle*” or “*Upload*” button, the process will begin. By clicking the “*Temizle*” or “*Clear*” button, the video can be deleted from the section for uploading video or after the video is uploaded, the button “*X*” from the top of the video can be clicked to delete the video. After the processes are done, the output will be shown in the second side (i.e., right hand side). All figures related our web GUI are taken from our web GUI application in its original form which is based on the open source Gradio interface Python library [47].

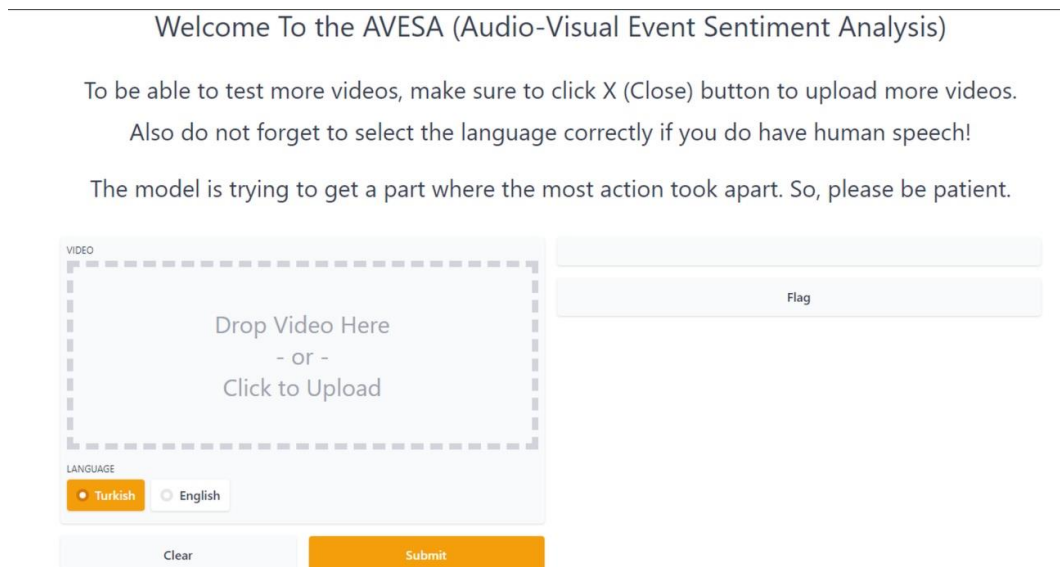


Figure 5: Our graphical user interface in action to recognize the sound event from video [47, 89]

In addition, videos that were chosen by us before can be used by clicking any of the videos below the “*Examples*” or “*Örnekler*” section. There are three different videos chosen by having any of the two languages. By clicking any of them, results for any of them can be seen at the second section. This section can be seen at Figure 6 below.

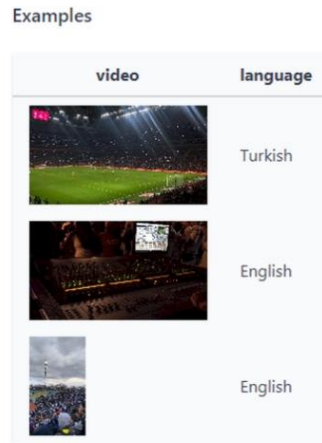


Figure 6 Examples shown by our graphical user interface [47, 89]

By considering the GUI, it seems to be able for an end-user to interact with a specific or randomly chosen attribute of our deep learning based audio-visual content understanding approach. By this way, it can be concluded that a sound event's outcome (i.e., class) and the mostly correct sentiment from this event is determined by our system which is shown in our GUI [89].

#### 4. Experimental Results

The experimental setup of our study is applied to establish our deep learning based audio-visual content understanding concept. In our experiments, we used a mix of sounds from a subset of ESC-50 [48] and other sound samples to detect and recognize the sound event classes. In other words, we applied an infrastructure to handle the above mentioned dataset which explained in Section 3.1. After that, we used some videos to test our video based NLP [7], NER [10] and other speech to text infrastructure of our methodology. These videos were chosen from CPSM dataset [50] and non-copyrighted materials from the Internet as provided on the Pexels website [90]. Since our real world problem, we are addressing is based on the real time speech to text translation and sound event recognition, the complete model design and its work on the knowledge discovery and processing results to a certain event class decision with the help of salient emotional situations come to a conclusion with underlying sentiment analysis.

In our experiments we used some computers equipped with Intel Core i7-8750H CPU that works with 2.20 GHz and has 16 GB RAM memory, NVidia GTX1060 graphic card (GPU) with 6 GB RAM and Intel Core i7-9750H CPU that works with 2.6 GHz and has 8 GB RAM, NVidia GTX1650 GPU with 4 GB RAM. In addition, the experiments were carried out with some experimental setups that were run several times on these machines for training and testing purposes. Our underlying software platform is based on some well-known libraries and frameworks which are supported by the Python programming language such as Keras [31], Tensorflow [30], NLTK [42] and etc.

In the following subsections, we address the objective performance evaluation via metrics for speech recognition, deep learning based classification and appropriate sentiment analysis. In addition, these kinds of evaluation in this section are supported with related experimental results given in detail. With the help of the experimental results, we clarified our concept and proof it as a “*Proof of Concept*” as well.

##### 4.1 Performance Evaluation

In this section, we introduce the objective performance evaluation based on metrics for sound event classification, speech recognition and appropriate sentiment analysis. Accuracy rate is one of the objective criteria commonly used to determine the class discrimination ability of the classifier on the dataset in an experiment. According to the confusion matrix table in the literature, it is evaluated by

true positive ( $TP$ ), false positive ( $FP$ ), true negative ( $TN$ ) and false negative ( $FN$ ) measurements. This value can be shown as both decimal and expanded as a percentage value in studies. It is given as Eq. 7 below [91].

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \quad (7)$$

The Precision metric expresses how many of the values estimated to be positively labeled are actually positively labeled in the experiments. This is shown in the Eq. 8 as below [91].

$$Precision = TP/(TP + FP) \quad (8)$$

The Recall metric denotes how many values the model predicted positively while they should have predicted as positively labeled in the experiments. This is shown in the Eq. 9 as below [91].

$$Recall = TP/(TP + FN) \quad (9)$$

The  $F1$  Score is known as the harmonic mean of precision and recall metrics. In the literature, it also called  $F$  measure,  $F$  metric or  $F$  Score as well. It creates a useful evaluation result in order to consider the extreme situations in the experiment where performance is affected. This is shown as Eq. 10 [91].

$$F1 = 2 * (Precision * Recall)/(Precision + Recall) \quad (10)$$

The Word Error Rate (WER) Score is one of the other common metrics to calculate accuracy between words. In performance evaluation scope, the WER can be found by dividing the number of errors to total words. These error numbers can be said as the addition of the substitution number, the addition number and the deletion number. The *Substitution* (S) occurs by replacing a word with another. As an example of this, transcribing “noose” as “moose”. The *Insertion* (I) occurs by adding a word which was not said before. For example, transcribing “hostess” as “host is”. The *Deletion* (D) occurs by deleting the word from the transcript. By considering this, transcribing “let it burn” is converted to “let burn”. Therefore, the sum of S, I and D terms denotes the total number of errors. Dividing this value to the total number of words will give the WER Score [92].

Above mentioned metrics and measurements are used in our study to prove the consistency and reliability of objective performance evaluation of our deep learning base audio-visual content analysis approach.

#### 4.2 Experimental Results with the Proof of Concept

In this section, the experimental results of our deep learning based approach to understand audio-visual content from videos are presented in an objective evaluation manner with plots and metric values which are well-known in the literature. Our experiments conducted on the above mentioned computer and software equipped with appropriate infrastructure. Every experimental setup was tried at least on 5 different runs to ensure the reliability of the test. Firstly, the training and validation of our sound recognition model and its fine tuning procedures results are given. In this scope, the changes of accuracy value while model tuning and training processes is given as accuracy plot in Figure 7 below. In our training process of sound recognition, loss function is treated as an objective function to analyze and reduce the average error that occurs in learning with given epochs for deep neural networks. The loss function’s plot from the training and tuning process can be seen in Figure 7 below.

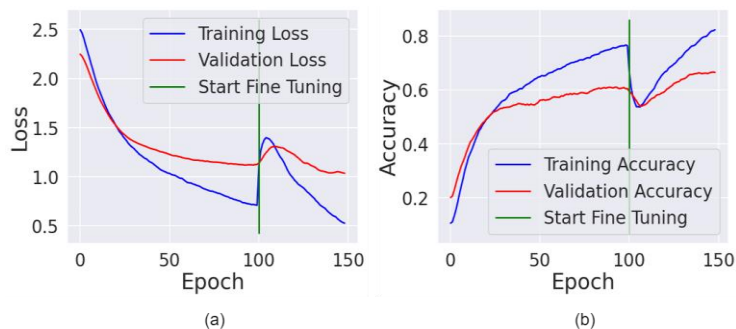


Figure 7: Loss plot (a) and Accuracy plot (b) for sound recognition model training.

It can be clearly seen from Figure 7, when the accuracy is increasing, the loss value at related epochs is decreasing as well. Values which are training loss, training accuracy, validation loss and validation accuracy started in the first epoch as respectively: 2.39, 0.10, 2.24, and 0.20. And it became the 148th epoch as respectively: 0.48, 0.83, 0.99, and 0.67. With the help of the beginning of fine tuning stage, the loss values a little bit increase for a number of epochs, but then it decreased as fast as possible than more before fine tuning point of training process. After the training and validation process, the model obtained an accuracy value as 0.70 and its respective loss values as 1.00 in the manner of fine tuning. Macro Average and Weighted Average values are equal to 0.73 for precision, 0.70 for recall and 0.70 for *F1* Score, where total sample test size equal to 120. Therefore, it can be said that the model has performed quite well with the result of 70% accuracy in terms of percentage ratio.

By considering the objective performance metrics in testing of our model, the performance results of sound recognition and confusion matrix for sound recognition in the experiments in our study can be seen from Figure 8 to Figure 9. Every class has an outcome in training with 12 samples as given in experiments. Total test sample size is defined as 120 samples for ten classes of overall testing.

In this sound event recognition’s performance evaluation, the average precision, average recall and average *F1* Score are obtained as 0.72, 0.70 and 0.69, respectively.

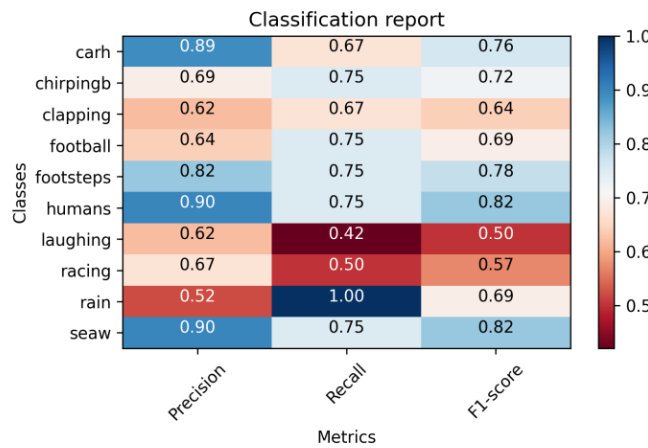


Figure 8 Performance results for sound recognition

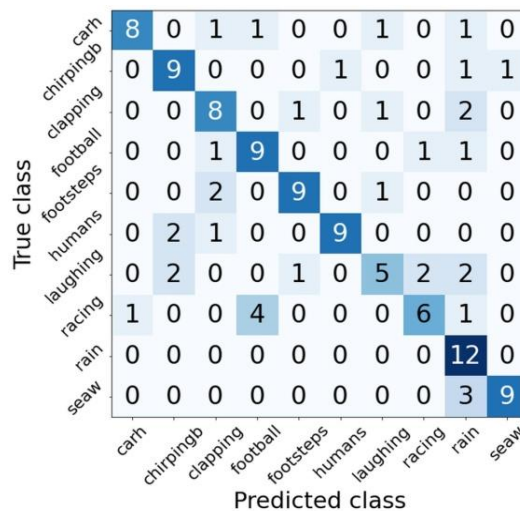


Figure 9 Confusion matrix for sound recognition

In addition, the performance results of getting speech with VOSK [12] language models for both languages in the experiments in our study can be seen in Figure 10. As seen from Figure 10, these tests for “*English*” texts have been done by using five different videos which are randomly chosen from the “*English*” speech dataset [50]. As seen from Figure 10, for “*Turkish*” texts have been done by using

four different videos which are randomly chosen from the “Turkish” speech dataset [52]. According to the sentiment analysis reports given in these figures, the best results in the experiments are obtained with “video-4” for “Turkish” texts, and “video-2” for “English” texts as well. In these tests, we assemble independent ground-truth text for each one of the videos in the test for NLP [7] and NER [10] processes that these texts are used to determine the similarity between predicted results and original human understanding of the spoken content with some metrics. These test results are given in Figure 10 as below. By considering the above mentioned VOSK speech tests for English videos, the average precision, average recall, average *F1* score [91], average cosine similarity score [81], average BLEU score [79] and WER score [92] are obtained as 0.89, 0.95, 0.92, 0.92, 0.90 and 0.21, respectively. In addition, these metrics obtained values for Turkish videos in VOSK speech tests are 0.96, 0.94, 0.95, 0.95, 0.81 and 0.41, respectively.

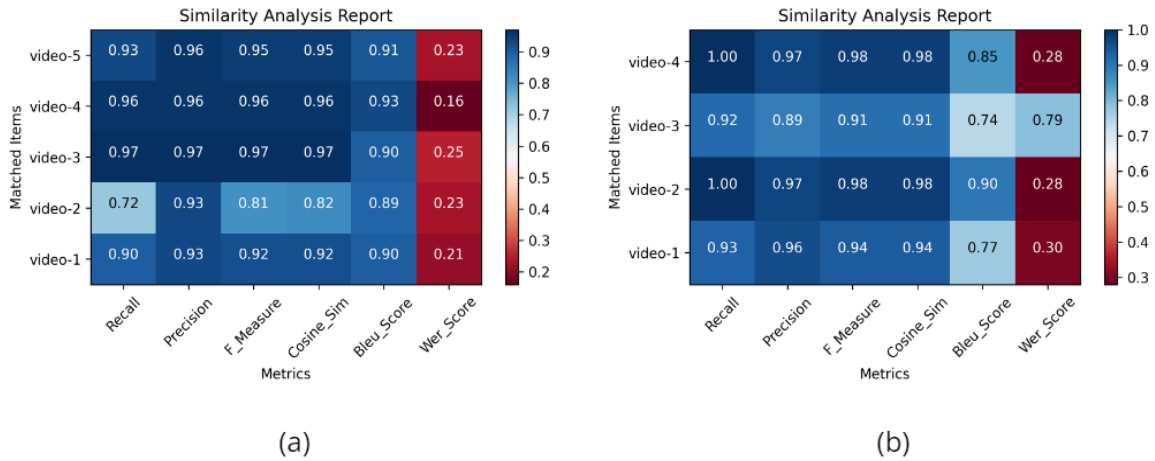


Figure 10 VOSK speech tests applied to (a) the English, and (b) Turkish videos.

At a glance, the cosine similarity score and BLEU score are much higher as they are already expected to show higher performance achievement for similarity analysis between original ground truth transcribed text and predicted text with our model. In addition, WER score is also much lower as it is expected to reflect the low error rate of prediction for accurate sound event class(es) and its correct sentimental conclusion in given videos as well.

The Receiver Operating Characteristics (ROC) Curve is created based on the confusion matrix and the experimental results are given as plot in Figure 11 below. Accordingly, if the curve approaches the upper left corner of the graph, it is understood that the model’s ability to distinguish between classes in classification is quite good. As can be seen from Figure 11, it provides evidence that a very good performance was obtained in the experiments in our study in terms of separation between classes [93].

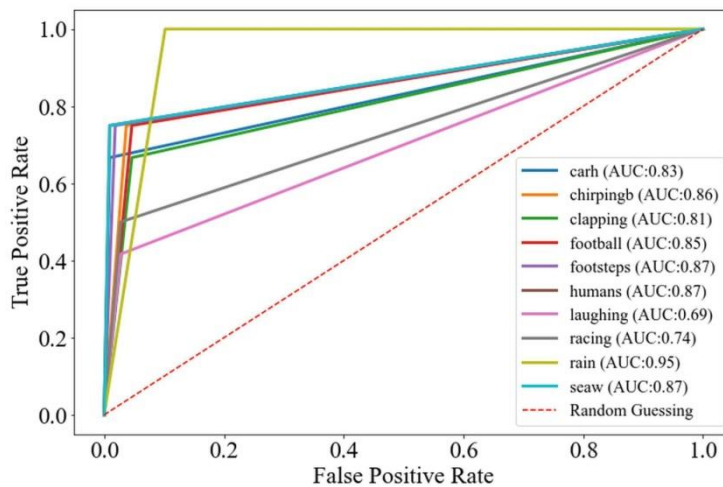


Figure 11 ROC Curve and AUC values for sound recognition process

The Area Under Curve (AUC) value shows the numerical value of the area under the ROC curve, proving how successful this model is in distinguishing between the given sound classes. This value is in the range of 0 to 1, and the closer it is to 1, the higher the performance of the model. These ROC-AUC values can be seen in Figure 11 above. Accordingly, this value proves that our model can make a very successful classification for data of this scale [94]. It is clearly seen from the ROC plot, the “rain” sound class is a more distinguishing class for our model than other recognized sound classes with the ROC-AUC value obtained as 0.95 as well.

In Figure 12, some experimental results taken from an English video are given which involve the prediction results for sound event class in the given video and its sentiment analysis with its ratio. And also in Figure 13, some sound event class prediction is given from a Turkish video without spoken content.

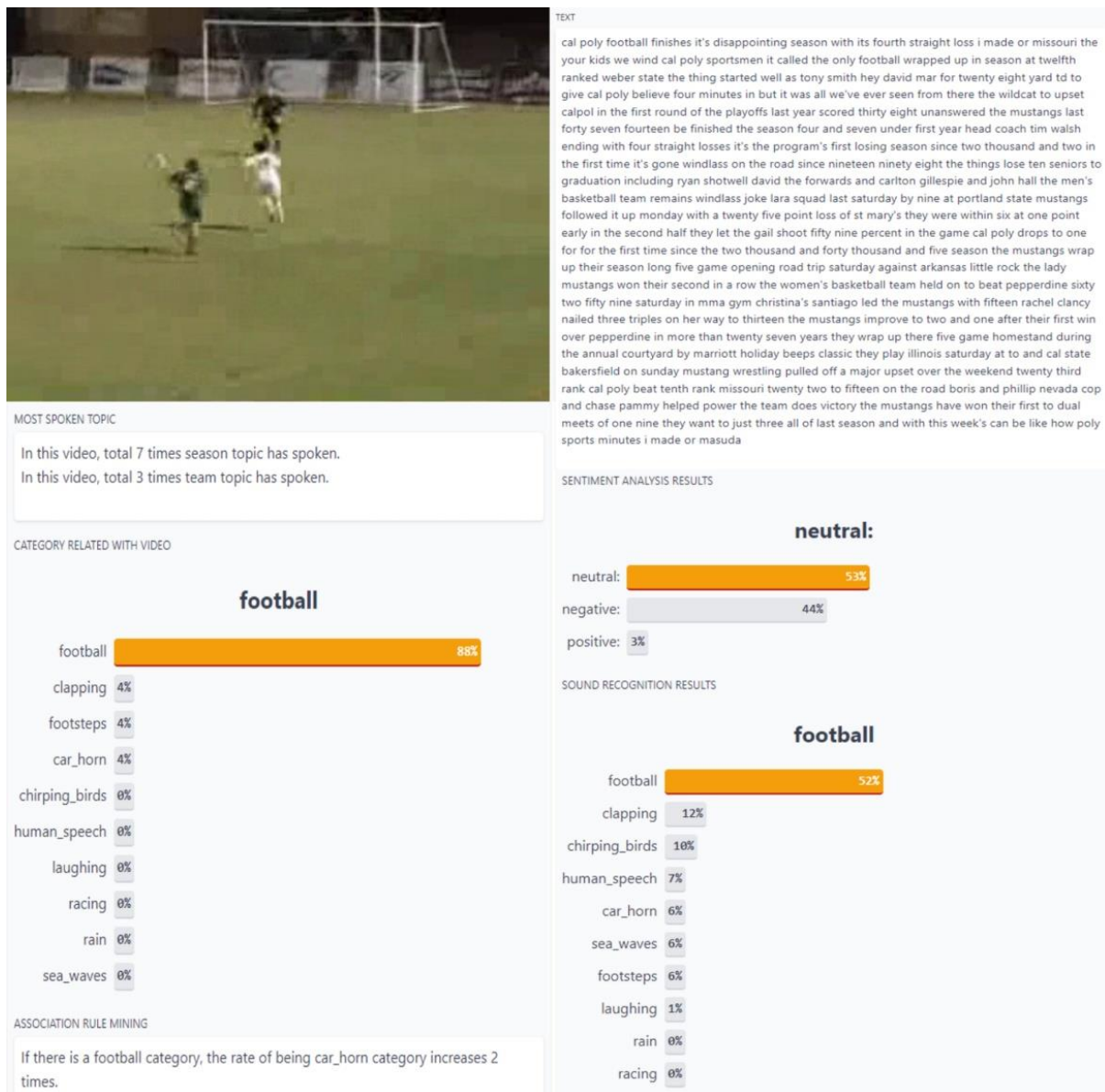


Figure 12 An experimental result on GUI for English video presented with its sentiment analysis, sound event recognition result and the related topic category [47, 89]



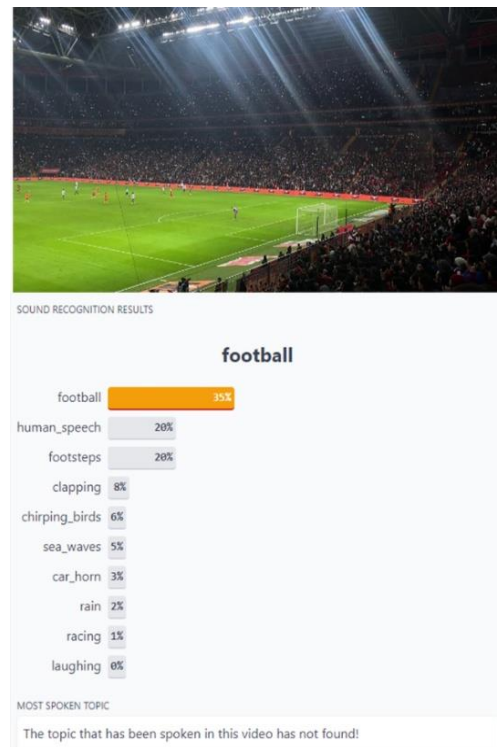


Figure 13 Experimented result for predicted sound event class for Turkish video without spoken content [47, 89]

In this scope, our study achieves consistent and reliable prediction results to prove our main concept about audio-visual content understanding with the help of deep learning.

## 5. Conclusion

In this study, the content understanding methodology for determination of the sentiment from spoken content based on the multimodal information is presented in which it is given in the manner of audio and video processing. This is based on the multilingual audio-visual content extracted using a deep learning approach without any other manual intervention. In addition, the sound scene classification and sound event/action recognition is applied by our approach to reach a refined sentiment analysis from spoken content. Its related sentences are taken from the processed video clip using speech recognition, natural language processing (NLP) [7] and Apriori algorithm [88]. To present our “*Proof of Concept*”, our experiments were conducted to prove the reliability and consistency of our methodology for real world scenarios with the help of deep learning’s high level interpretation of abstract features on raw data. As seen from these experiments’ results, both English and Turkish spoken content with related sentiment were processed and analyzed quite well with high accuracy to reach a conclusion of event/action on a given video clip in the tests. By considering average WER scores in our experiments, lower WER scores indicates its superiority that our audio-visual content understanding approach is more accurate to recognize topic and sentiment, emotional states in videos. In further studies, we plan to add some other deep learning models to our methodology, and thus, we will expand the test set with different scenarios involving other sound classes synthetic or natural sounds in the environment that our approach can be used to concluding the interaction with different event/action pairs for resolving the overlapping issues in complex audio signals in need to cleaning the channels for source separation.

## References

- [1] B. Karakaya, E.B. Boztepe, and B. Karasulu, "Development of a Deep Learning Based Model for Recognizing the Environmental Sounds in Videos," in *The SETSCI Conference Proceedings*

- Book, vol. 5, no. 1, pp. 53-58, 2022.
- [2] B. Karasulu, "Çoklu Ortam Sistemleri İçin Siber Güvenlik Kapsamında Derin Öğrenme Kullanarak Ses Sahne ve Olaylarının Tespiti," *Acta Infologica*, vol. 3, no. 2, pp. 60-82, 2019.
- [3] E. A. Kıvrak, B. Karasulu, C. Sözbir ve A. Türkay, "Ses Özneliklerini Kullanan Ses Duygu Durum Sınıflandırma İçin Derin Öğrenme Tabanlı Bir Yazılımsal Araç," *Veri Bilim Dergisi*, vol. 4, no. 3, pp.14-27, 2021.
- [4] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a Convolutional Neural Network," in *Proceedings of the International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, pp. 1-6, 2018.
- [5] Y. Zhao, X. Jin, and X. Hu, "Recurrent Convolutional Neural Network for Speech Processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5300-5304, 2017.
- [6] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal Deep Learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML11)*, Bellevue, Washington, USA, pp. 689–696, 2011.
- [7] S. Bird, E. Loper, and J. Baldridge, "Multidisciplinary Instruction with the Natural Language Toolkit," in *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, Columbus, Ohio, pp. 62–70, 2008.
- [8] J. Joseph, and J. R. Jeba, "Information Extraction Using Tokenization And Clustering Methods," *International Journal of Recent Technology and Engineering*, vol. 8 no. 4, pp. 3680-3692, 2019.
- [9] H. van Halteren, J. Zavrel, and W. Daelemans, "Improving Accuracy in NLP Through Combination of Machine Learning Systems," *Computational Linguistics*. vol. 27, no. 2, pp. 199–229, 2001.
- [10] A. Roy, "Recent Trends in Named Entity Recognition (NER)," arXiv preprint arXiv:2101.11420 [cs.CL], 2021.
- [11] K. Shaukat, S. Zaheer, and I. Nawaz, "Association Rule Mining: An Application Perspective," *International Journal of Computer Science and Innovation*, vol. 2015, no. 1, pp.29-38, 2015.
- [12] VOSK Offline Speech Recognition Library Website, 2022, [Online]. Available: <https://alphacephei.com/vosk/>. [Accessed: 01-July-2022]
- [13] Ö. Şahinaslan, H. Dalyan ve E. Şahinaslan, "Naive Bayes Sınıflandırıcısı Kullanılarak YouTube Verileri Üzerinden Çok Dilli Duygu Analizi," *Bilişim Teknolojileri Dergisi*, vol. 15, no. 2, pp. 221-229, 2022.
- [14] M.C. Yılmaz ve Z. Orman, "LSTM Derin Öğrenme Yaklaşımı ile Covid-19 Pandemi Sürecinde Twitter Verilerinden Duygu Analizi," *Acta Infologica*, vol. 5, no. 2, pp. 359-372. 2021.
- [15] N. Buduma and N. Lacascio, *Designing Next-Generation Machine Intelligence Algorithms Fundamentals of Deep Learning*, O'Reilly Media UK Ltd., 2017.
- [16] F. Chollet, *Deep Learning with Python*, Manning Publications, 2017.
- [17] Y. Shen, C.-H. Demarty, and N.Q.K. Duong, "Deep Learning for Multimodal-Based Video Interestingness Prediction," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1003-1008, 2017.
- [18] Y.-G. Jiang, Y. Wang, R. Feng, X. Xue, Y. Zheng, and H. Yang, "Understanding and Predicting Interestingness of Videos," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pp. 1113–1119, 2013.
- [19] D. M. Agrawal, H. B. Sailor, M. H. Soni, and H. A. Patil, "Novel TEO-based Gammatone Features for Environmental Sound Classification," in *Proceedings of the 25th European Signal Processing Conference*, pp.1859-1863, 2017.
- [20] Z. Mushtaq and S.-F. Su, "Efficient Classification of Environmental Sounds through Multiple Features Aggregation and Data Enhancement Techniques for Spectrogram Images," *Symmetry*, vol. 12, no. 11:1822, pp. 1-34, 2020.
- [21] DenseNet Documentation, 2022, [Online]. Available: <https://github.com/liuzhuang13/DenseNet>. [Accessed: 01-July-2022].
- [22] A. Khamparia, D. Gupta, N.G. Nguyen, A. Khanna, B. Pandey, and P. Tiwari, "Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network," *IEEE Access*, vol. 7, pp. 7717-7727, 2019.

- [23] K.J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proceedings of the IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Boston, MA, USA pp. 1-6. 2015.
- [24] R. A. Khalil, E. Jones, M. I. Babar, T. Jan, M. Haseeb Z., and T. Alhussain, "Speech Emotion Recognition Using Deep Learning Techniques: A Review," *IEEE Access*, vol. 7 pp. 117327-117345, 2019.
- [25] M. Gygli, H. Grabner, and L. V. Gool, "Video Summarization By Learning Submodular Mixtures Of Objectives," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, pp. 3090-3098, 2015.
- [26] B. A. Plummer, M. Brown, and S. Lazebnik, "Enhancing Video Summarization Via Vision-Language Embedding," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 1052-1060, 2017.
- [27] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Summary Transfer: Exemplar-Based Subset Selection For Video Summarization," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1059-1067, 2016.
- [28] K. Petros, and M. Petros, "SUSiNet: See, Understand and Summarize It," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Long Beach, CA, USA 16-17 June, pp. 809-819, 2019.
- [29] Python Programming Language and Python Modules Documentation, 2022, [Online]. Available: <https://www.python.org/doc/>. [Accessed: 01-July-2022]
- [30] Tensorflow Library Documentation, 2022, [Online]. Available: [https://www.tensorflow.org/api\\_docs](https://www.tensorflow.org/api_docs). [Accessed: 01-July-2022]
- [31] Keras Library Documentation, 2022, [Online]. Available: <https://keras.io/api/>. [Accessed: 01-July-2022]
- [32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, pp. 4510-4520, 2018.
- [33] Python Data Analysis Library (Pandas) Website 2022, [Online]. Available: <https://pandas.pydata.org/>. [Accessed: 01-July-2022].
- [34] Library for Visualization with Python (Matplotlib) Website, 2022, [Online]. Available: <https://matplotlib.org/>. [Accessed: 01-July-2022].
- [35] Python Statistical Data Visualization Library (Seaborn) Website, 2022, [Online]. Available: <https://seaborn.pydata.org/introduction.html>. [Accessed: 01-July-2022].
- [36] Numerical Library for Python (NumPy), 2022, [Online]. Available: <https://numpy.org/>. [Accessed: 01-July-2022]
- [37] SpaCy Natural Language Processing Library for Python, 2022, [Online]. Available: <https://spacy.io/api/doc>. [Accessed: 01-July-2022].
- [38] Manipulate Audio Library (PyDub) Website, 2022, [Online]. Available: <https://pydub.com/>. [Accessed: 01-July-2022].
- [39] OpenCV Library Documentation, 2022, [Online]. Available: <https://docs.opencv.org/4.6.0/>. [Accessed: 01-July-2022].
- [40] Moviepy Library Documentation, 2022, [Online]. Available: <https://zulko.github.io/moviepy/>. [Accessed: 01-July-2022].
- [41] B. McFee, C. Raffel, D. Liang, D. Ellis, M. Mcvicar, E. Battenberg, and O. Nieto, "Librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the Python in Science Conference*, 2015.
- [42] E. Loper and S. Bird, "NLTK: the Natural Language Toolkit," in *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, vol. 1, pp. 63-70, 2002.
- [43] Transformers Library Documentation, 2022, [Online]. Available: <https://huggingface.co/docs/transformers/main/en/index>. [Accessed: 01-July-2022].
- [44] Diffilib module computing deltas for Python, 2022, [Online]. Available: <https://docs.python.org/3/library/diffilib.html>. [Accessed: 01-July-2022].
- [45] Zeyrek: Morphological Analyzer and Lemmatizer GitHub Website, 2022, [Online], Available:

- <https://github.com/obulat/zeyrek>. [Accessed: 01-July-2022].
- [46] Library for approximate and phonetic matching of strings for Python, 2022, [Online]. Available: <https://github.com/jamesturk/jellyfish>. [Accessed: 01-July-2022].
- [47] Gradio Library Documentation, 2022, [Online]. Available: <https://gradio.app/docs/>. [Accessed: 01-July-2022].
- [48] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.
- [49] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, QLD, Australia, pp. 5206 - 5210, 2015.
- [50] T. M. Hospedales, S. Gong, and T. Xiang, "Learning Tags from Unsegmented Videos of Multiple Human Actions," in *Proceedings of the IEEE 11th International Conference on Data Mining*, Vancouver, BC, Canada, pp. 251-259, 2011.
- [51] Youtube. 2022. [Online]. Available: <https://www.youtube.com>. [Accessed: 01-July-2022].
- [52] R. Kolobov et al., "MediaSpeech: Multilanguage ASR Benchmark and Dataset," arXiv preprint arXiv:2103.16193, 2021.
- [53] M. Rochan, L. Ye, and Y. Wang, "Video Summarization Using Fully Convolutional Sequence Networks," in *Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science*, vol. 11216. pp 358–374, 2018.
- [54] S. Jadon and M. Jasim, "Unsupervised video summarization framework using keyframe extraction and video skimming," in *Proceedings of the IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, UP, India, Oct 30-31, pp. 140-145, 2020.
- [55] J. Park, J. Lee, S. Jeon, and K. Sohn, "Video Summarization by Learning Relationships between Action and Scene," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South), 27-28 October, pp. 1545-1552, 2019.
- [56] Z. Li, G. M. Schuster, A. K. Katsaggelos, and B. Gandhi, "Rate-distortion optimal video summarization: a dynamic programming solution," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, QC, Canada, vol. 3, pp. iii-457, 2004.
- [57] S. Lu, M. R. Lyu, and I. King, "Video summarization by spatial-temporal graph optimization," in *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, Vancouver, BC, Canada, pp. II-197, 2004.
- [58] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid, "Category-specific video summarization", in *Proceedings of the European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, 6-12 September, pp. 540–555, 2014.
- [59] M. Otani, Y. Nakashima, E. Rahtu, and J. Heikkila, "Rethinking the Evaluation of Video Summaries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, pp. 7588-7596, 2019.
- [60] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv preprint arXiv:1409.1556v6 [cs.CV], 2015.
- [61] K. Zhou, Y. Qiao and T. Xiang, "Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity-Representativeness Reward," arXiv preprint arXiv:1801.00054, 2018.
- [62] Kernel Temporal Segmentation (KTS). 2022. [Online]. Available: <https://github.com/TatsuyaShirakawa/KTS>. [Accessed: 01-July-2022]
- [63] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "TVSum: Summarizing web videos using titles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 07-12 June, pp. 5179-5187, 2015.
- [64] R. Andonov, V. Poirriez, and S. Rajopadhye, "Unbounded knapsack problem: Dynamic programming revisited," *European Journal of Operational Research*, vol. 123, no. 2, pp. 394-407, 2000.
- [65] M. Gygli, H. Grabner, H. Riemenschneider, and L. van Goo, "Creating Summaries From User

- Videos,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, 6-12 September, pp. 505–520, 2014.
- [66] P. Musa, F. Rafi, and M. Lamsani, “A Review: Contrast-Limited Adaptive Histogram Equalization (CLAHE) Methods to Help the Application of Face Recognition,” in *Proceedings of the Third International Conference on Informatics and Computing (ICIC)*, Palembang, Indonesia, 17-18 October, pp. 1-6, 2018.
- [67] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, “Learning Attentive Representations for Environmental Sound Classification,” *IEEE Access*, vol. 7, pp. 130327 - 130339, 2019.
- [68] Ö. Eskidere ve F. Ertaş, “Mel Frekansı Kepstrum Katsayılarındaki Değişimlerin Konuşmacı Tanımaya Etkisi,” *Uludağ Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, vol. 14, no. 2, pp. 93-110, 2009.
- [69] Md. A. Hossan, S. Memon, and M. A. Gregory, “A Novel Approach for MFCC Feature Extraction,” in *Proceedings of the 4th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Gold Coast, QLD, Australia, 13-15 December, pp. 1-5, 2010.
- [70] N. Jiang, P. Grosche, V. Konz, and M. Müller, "Analyzing chroma feature types for automated chord recognition", in *Proceedings of the 42nd AES International Conference on Semantic Audio*. Ilmenau, Germany, pp. 285-294, 22-24 July, 2011.
- [71] Rotating Images Information Website, 2022, [Online]. Available: <https://datagenetics.com/blog/august32013/index.html>. [Accessed: 01-July-2022].
- [72] Y. Bengio, A. Courville, and Pa. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1798-1828, 2013.
- [73] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, pp. 248-255, 20-25 June, 2009.
- [74] D. P. Kingma, and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proceedings of the 3rd International Conference for Learning Representations*, San Diego, USA, pp. 1-13, 2015.
- [75] S. Albelwi and A. Mahmood, “A framework for designing the architectures of deep convolutional neural networks,” *Entropy*, vol. 19, no. 6:242, 2017.
- [76] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, “An Overview of the HDF5 Technology Suite and its Applications,” in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, Uppsala, Sweden, March 25, pp. 36-47, 2011.
- [77] M. Mednis and M. K. Aurich, “Application of String Similarity Ratio and Edit Distance in Automatic Metabolite Reconciliation Comparing Reconstructions and Models,” *Biosystems and Information Technology*, vol.1, no.1, pp. 14-18, 2012.
- [78] K. Dreßler and A.-C. Ngonga Ngomo, “On the Efficient Execution of Bounded Jaro-Winkler Distances,” *Semantic Web, Issue title: Ontology and linked data matching*, vol. 8, no 2, pp 185–196, 2017.
- [79] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia Pennsylvania, USA, 7 - 12 July, pp. 311–318, 2002.
- [80] C. Callison-Burch, M. Osborne, and P. Koehn, “Re-evaluating the Role of BLEU in Machine Translation Research,” in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, 3-7 April, pp. 249-256, 2006.
- [81] F. Rahutomo, T. Kitasuka, and M. Aritsugi, “Semantic Cosine Similarity,” in *Proceedings of the 7th International Student Conference on Advanced Science and Technology ICAST*, Seoul, South Korea, 2012.
- [82] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Computation and Language (cs.CL)*, arXiv preprint arXiv:1810.04805 [cs.CL], 2018.
- [83] Hugging Face Services Documentation, 2022, [Online]. Available: <https://huggingface.co/docs>. [Accessed: 01-July-2022].
- [84] Roberta Sentiment Model Documentation, 2022, [Online]. Available:

- <https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment>. [Accessed: 01-July-2022].
- [85] BERT-Turkish Sentiment Model Documentation, 2022, [Online]. Available: <https://huggingface.co/savasy/bert-base-turkish-sentiment-cased>. [Accessed: 01-July-2022].
- [86] S. Yildirim, "Comparing Deep Neural Networks to Traditional Models for Sentiment Analysis in Turkish Language," In: B. Agarwal, R. Nayak, N. Mittal, and S. Patnaik, (eds) *Deep Learning-Based Approaches for Sentiment Analysis. Algorithms for Intelligent Systems*. Springer, Singapore, pp. 311-319, 2020.
- [87] S. Sarica and J. Luo, "Stopwords in Technical Language Processing," *Plos One*, vol.16, no.8, pp. 1-13, 2021.
- [88] S. Panjaitan, Sulindawaty, M. Amin, S. Lindawati, R. Watrianthos, H. T. Sihotang, and B. Sinaga, "Implementation of Apriori Algorithm for Analysis of Consumer Purchase Patterns," in *Proceedings of the International Conference on Computer Science and Applied Mathematic, IOP Conf. Series: Journal of Physics: Conf. Series*, vol. 1255, no. 1, pp. 1-8, 2019.
- [89] AVESA GitHub Repository, 2022, [Online]. Available: <https://github.com/berayboztepe/AVESA>. [Accessed: 01-July-2022].
- [90] Pexels Website, 2022, [Online]. Available: <https://www.pexels.com>. [Accessed: 01-July-2022].
- [91] B. Karasulu, "Kısıtlanmış Boltzmann makinesi ve farklı sınıflandırıcılarla oluşturulan sınıflandırma iş hatlarının başarımının değerlendirilmesi", *Bilişim Teknolojileri Dergisi*, vol. 11, no. 3, pp. 223-233, 2018.
- [92] A. Ali and S. Renals, "Word Error Rate Estimation for Speech Recognition: e-WER," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, 15 - 20 July, pp. 20-24, 2018.
- [93] T. Fawcett, "Introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, 2006.
- [94] D. M. W. Powers, "The Problem of Area Under the Curve," in *Proceedings of the IEEE International Conference on Information Science and Technology (ICIST)*, Wuhan, China, 23-25 March, pp. 567-573, 2012.

# Transfer of Analogies in Traditional Programming Languages to Teaching VHDL

 Halit ÖZTEKİN<sup>1</sup>,  Ali Gülbağ<sup>2</sup>

<sup>1</sup>Corresponding Author; Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, TURKEY, halitoztekin@subu.edu.tr; +90 533 547 06 18

<sup>2</sup>Department of Computer Engineering, Sakarya University, Sakarya, TURKEY, agulbag@sakarya.edu.tr; +90 533 344 34 99

Received 6 June 2022; Revised 25 October 2019; Accepted 7 July 2022; Published online 31 August 2022

## Abstract

One of the languages available to describe a digital system in FPGA is the VHDL language. Since programming in hardware requires a different way of thinking than developing software, the students face some difficulties when trying to design in VHDL language with the previous and long experiences kept in mind in the learning of software imperative programming. These are its concurrency, parallel and sequential model. Due to the insufficient understanding of these topics, it is difficult for students to master the VHDL language. Analogies change the conceptual system of existing knowledge by linking the known to the unknown and by changing and strengthening their relationships. This study contributes to overcoming the problems that students encounter in the coding of the above-mentioned topics in VHDL language by using their experiences in traditional programming languages through analogies. Analogies were used in an undergraduate embedded systems course to explain complex concepts such as those related to signals, concurrent/parallel process; and to encourage comprehensive projects in digital circuit design. Computer science and similar degrees include courses focused mainly on high-level programming languages. This has a strong case for shortening the learning curve in teaching hardware description languages of the skills acquired in programming courses. In feedback from students, the discussion and negotiation of analogies seems to minimize confusion and from using inappropriate expressions in using VHDL language.

**Keywords:** VHDL, FPGA, Teaching, Adaptive learning, Programming languages, Analogy, Think Hardware

## 1. Introduction

The traditional approach to knowledge transfer is maintained with ideas from the early 20th century. In the current psychological theories, there are particularly seminal ideas of Thorndike regarding the importance of overlapping characteristics between learning and transfer situations [1, 2]. However, in most psychological research on transfer, especially analogical reasoning theories have been used [3-5].

An analogy is similar in some ways between unlike things, and a comparison is based on this similarity. Analogies connect the known to the unknown and change the conceptual system of existing knowledge by changing and strengthening their relationships [6]. Well-defined metaphors are important for students to establish a connection with what they already know and to offer order to the chaos of unfamiliar concepts [7]. There are not many studies on the use of analogies as an active teaching strategy in engineering education. A study was conducted to determine what kind of analogies students use to explain the basic circuit concepts, and what properties the structural analogies have, which are the most common in students' discussions about circuit concepts [8]. However, there are studies on the use of the concept of analogy in other scientific fields such as medical science [9]. In this study, analogies were used in an undergraduate embedded systems course for electrical and electronics students for two purposes: a) to explain complex concepts such as those related to signals, concurrent/parallel and process; and b) to provide practice for students to increase interest in VHDL and encourage comprehensive projects in digital circuit design using these strategies [10].

VHDL is a description language for digital electronic circuits to accelerate the design process. As stated in its acronym (VHDL- Very High Speed Integrated Circuits Hardware Description Language), it is used to accelerate the design process. Performing this process in high-level programming language such as C++ could be an option. However implementing about the hardware aspects of computer systems can not be an easy task. To this end, there are studies emphasizing the importance of hardware description languages [11]. The fact that VHDL is not

a programming language is often forgotten. Therefore, knowing syntax as in a standard programming language does not mean being able to design digital circuits with it. A program in VHDL language is not executed sequentially as in a standard programming language such C++. Due to the behaviour of a real hardware, VHDL allows concurrent statements and operations triggered by events. It would be very difficult to explain this to someone who creates the program in a classical programming language [12-14]. These issues tend to distract them from the understanding of digital components. The number of studies using hardware description languages is increasing in the literature. This situation reveals the importance of learning these languages. There are studies in the literature about teaching these languages. Detailed information about this is given in section 2. In this study, it is aimed to facilitate hardware thinking by showing the equivalents of the expressions in VHDL in a programming language. The purpose of teaching methodology is trying to help the transfer of the concepts that students have difficulty in understanding to the familiar environment. Thus, they learn the importance of HDL-based digital design by better understanding the complexities of HDLs.

## 2. Related Work

There are various methods developed in the literature to overcome the difficulties of using the HDL languages. The main purpose of the studies is to make digital design easier by reducing the complexity of hardware description languages [15]. In one of these studies, developed the tutorial showing students how to design digital circuit systems first, and then introducing them to the language [16]. In other words, it is the transition to language teaching by placing the idea of "think hardware" in students. In the reference [17], a set of design guidelines for HDL-based design explain how to avoid the common conceptual error of synchronizing HDL code with procedural software and instead think in terms of hardware. Technology is a tool that should be used in the teaching-learning process [18]. VerilogTown that is a video game used to control the world has been developed to better understand Verilog, and for the experience to be less jarring in their learning process [19]. Similarly, a language called Abstract Verilog has been defined in order to reduce the cognitive load for new students [20]. The spiral approach method was incorporated into the digital systems course in undergraduate electrical and computer engineering programs at Ohio Northern University (ONU), as HDLs can be comprehensive and very difficult to understand in such an early course as digital systems [21]. A lightweight and cross-platform open-source environment has been developed that is suitable for use as an introductory tool for students learning HDLs [22]. In some studies in the literature, the problems encountered while teaching HDL language [23] and suggestions to instructors wishing to implement the class are given [24-26]. Literature [27-29] indicates an approach that would enable students to focus more on the design and less on the individual bits instead of HDL in order to perform bit-level operations.

## 3. Materials and Methods

In this section, the syntax of the HDL languages will not be discussed, only the equivalents of some of its components in one of the classical programming languages (preferred C++ in this work) will be shown and used VHDL as hardware description language. The general structure of this language is given in Figure 1.

```

architecture name of entity is
  -- architecture declarations
  -- signals
  -- components
  -- types
  begin
  -- concurrent statements
  -- conditional statements
  process(sensitivity list) begin
  -- sequential codes
  end process;
end name;

```

Figure 1 The general architecture of VHDL language



### 3.1 Entity Component

Entity is the part where the input and output ports of a digital system are defined. In other words, it accepts data from external units through these ports and transmits the result to external units via these ports. Since only one entity can be defined in a VHDL file, a declaration of a function (function prototype or function interface) can be given in C++ programming language. The idea that entity components with different names can be created in the same VHDL file may come to mind here. This problem would be eliminated by considering the entity component as a main function in C++. The example below (Code 1) shows the C++ equivalent of the entity component in VHDL. The number and types of parameters in the function signature in the function prototype and the return type of the function represent the port block of the entity component. If the entity component has more than one output port, then the return type of the function will need to be represented by a struct.

Code 1 The entity declaration in VHDL language

<u>VHDL</u>	<u>C++</u>
1 entity example is	1 double example ( double, int, int );
2 port ( A,B,C : in std_logic;	3
3 D : out std_logic );	3
4 end example;	4

### 3.2 Components

While giving information about the functioning of this component, mentioning the subroutine subject in programming languages will help the student to remain in her/his mind. In particular, calling a sub procedure from another procedure is a good example to illustrate this one (Code 2). Unlike an entity component in a program snippet written in a VHDL language, the components can have more than one. The operation of this component in a high-level programming language is the same as that of the entity component.

Code 2 The component instantiation in VHDL language

<u>VHDL</u>	<u>C++</u>
1 component sub_program	1 .....
2 port ( A,B : in BIT;	2 Y=sub_program(A, B)
3 Y : out BIT );	3 .....
4 end component;	4

### 3.3 Execution Modes

The statements in VHDL can be executed in the following three modes. These modes are namely:

- i) Sequential mode
- ii) Concurrent mode
- iii) Parallel mode

Sequential mode is an environment where commands are executed sequentially from top to bottom as in all programming languages and the user knows this environment very well. The commands in this mode can only be written in the process block in VHDL that are scheduled for execution by events. In order to understand the operation of the process block, there are 2 issues to consider: event and scheduling. The first one is a mechanism called an event (sensitivity list) that activates the execution of sequential commands in the process block. The sequential commands in the process are executed when changes in any signal specified in the sensitivity list. In other words, the process will be executed in an infinite loop. The events such as timer, click etc. used in classical programming languages can be used to explain the functioning of the concept of process. Secondly, the instructions in the process block are executed with

the current values of the signals and their updated values occur after the end of *end process* clause. This situation can be associated with a queue structure that stores the current value at the top of the queue and the updated value at the bottom. The pop command is configured to leave at least one element in the queue. In other words, pop command will not perform data retrieval if there is only one element in the queue. Otherwise, there may be no data in the queue. The following example (Code 3) describes these one. In this example, let's assume that three queue structures that work according to the FIFO rule are created for the variables. Their names are q\_a, q\_b and q\_c respectively. If we accept that there are value1, value2 and value3 values in the queues at the beginning, that is, before the block of process runs, after line 3 in the VHDL code snippet, the value(value2) at the top of the b queue is pushed to the queue named q\_a. After the piece of code in line 4 runs, the value(value1) from the queue named q\_a is pushed to the queue named q\_c. At the end of the process block, the elements below the queues are transferred to the top.

Code 3 The equivalent of process block in C++

<u>VHDL</u>	<u>C++</u>
1 process (e)	1 private void button1_click(object sender, EventArgs
2 begin	2 e) {
3 q_a<=q_b;// (1)	3 q_a.push(q_b.pop());
4 q_c<=q_a;// (2)	4 q_c.push(q_a.pop());
5 end process;	5 return update_queues;}
6	6 void update_queues() {
7	7 q_a.pop();
8	8 q_b.pop();
9	9 q_c.pop();}

The changes in the example above are given step by step below(Figure 2).

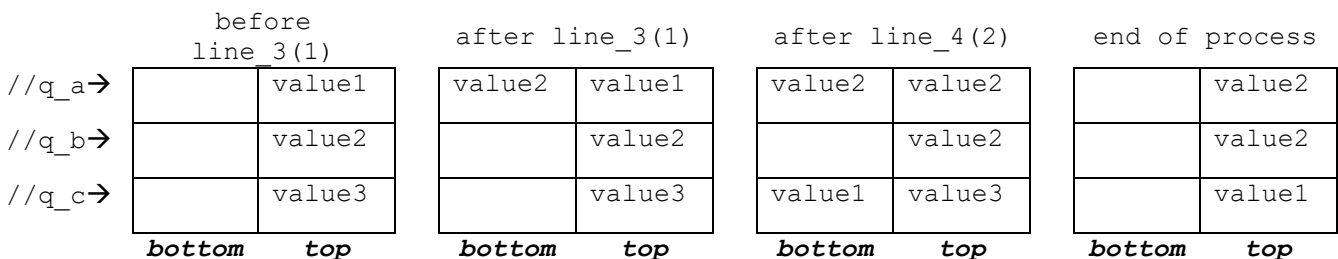


Figure 2 The changes of signals in the process

One of the most important reasons why FPGAs are popular is that their structure is suitable for parallelism. The term that is often confused with the term parallelism is concurrency. Considering that the VHDL language is a hardware description language, it is useful to specify what these two terms correspond to in terms of hardware. If there is cooperation between hardware units, the term to be mentioned is concurrency, if not parallelism. Code 4 explains how two different kinds of execution method were compared.

Code 4 The examples for concurrent and parallel operations in VHDL

<u>Concurrent</u>	<u>Parallel</u>
1 architecture example of	1 architecture example of exp_parallel is
2 exp_concurrency is	2 begin
3 begin	3 c <= a and b;
4 e <= c or d;	4 d <= a or b;
5 c <= a and b;	5 end exp_parallel ;
6 end exp_concurrency;	6

While commands are executed as they are written in classical programming languages, this does not matter in VHDL language. The order of execution is depended only by events on the signals. In addition, the commands are executed repeatedly in every change of the signal. Concurrent or parallel operations in VHDL language are similar to thread structure in traditional programming languages. While there is no need for an extra command in VHDL language to do these operations, software description languages require an extra instruction (Code 5). In other words, each line (block) in VHDL language can be evaluated as a thread. The order of execution is defined only by events occurring on the signals that the assignments are sensitive to. First of all, the order of the command lines with the "<=" operator in the VHDL language is not important. Even changing only one of the variables to the right of the "<=" operator in these lines is sufficient to execute the relevant line. In this example, two separate threads are created for concurrent execution of two lines. However, while these threads are running once, the lines in the VHDL language will run multiple times. To realize this situation in high-level programming language, threads must be written inside an event. If the variable/s to the left of the "<=" operator are not included in the variables to the right of the "<=" operator in other lines, it means that these lines will be executed in parallel.

Code 5 The examples for concurrent and parallel operations in C++

<u>Concurrent</u>	<u>Parallel</u>
1 void task1(int m){	1 #pragma omp parallel
2 for(int i=1;i<=m;i++)	2 {
3 cout << "Hello";}	3 cout<<"Hi!"
4 void task2(int n){	4 }
5 for(int i=1;i<=n;i++)	5 //For dual-core system,twice text "Hi!"
6 cout << "World";}	6 //It may output: HiH!i!,
7 int main(){	7 //printing is //parallel
8 thread t1(task1, 3);	8
9 thread t2(task2, 2);}	9
10 //Output(machine dependent)	10
11 //HelloHelloWorldHelloWorld	11

#### 4. Results

Feedback is an important part of learning effectively and improving students' learning experiences. Student feedback has become a widely used method for evaluating and improving teaching effectiveness, as even small changes in the classroom can make a big difference. Students' opinions are asked to evaluate the effectiveness of this teaching approach through a survey. All of the students enrolled in this course have successfully completed Algorithms and Computer Programming courses in which C ++ is taught. Therefore, they are familiar with the programming language concepts mentioned in the study. The participants enrolled in the Embedded System Course were divided into two groups, one experimental and another one control. For participants in the control group, the instructor used PowerPoint slides and delivered in the traditional manner of the lecture style. On the contrary, in the experimental group participants were engaged in the lecture format plus use of analogy in classical programming languages. At the end of the semester, a questionnaire consisting of 5 questions was made to both groups and the answers given are given in the Table 1.

Table 1 Student Feedback on Embedded Systems Course

Statements	Control Group(%)			Experimental Group(%)		
	Disagree	Neutral	Agree	Disagree	Neutral	Agree
1. I realized that the VHDL language is a hardware description language.	12.5	37.5	50	0	10	90
2. Decreased complexity in understanding features such as Concurrent / Parallel.	37.5	25	37.5	10	10	80
3. Increased in understanding how the process block performs schedule.	62.5	0	37.5	0	30	70
4. I realized the operating characteristics of Signal and variables (in computer programming languages)	0	25	75	0	10	90
5. Increased interest in VHDL language and encouraging more comprehensive projects in digital circuit design	25	50	25	0	30	70

The comparison of midterm and final exam averages of both groups is given in the graphic below (Figure 3). The midterm exam averages of the two groups are very close to each other. This situation can be explained as follows: It is due to the use of the analogy approach presented in this article on the subjects mentioned after the midterm exam. This situation is reflected in the final exam averages of the groups.

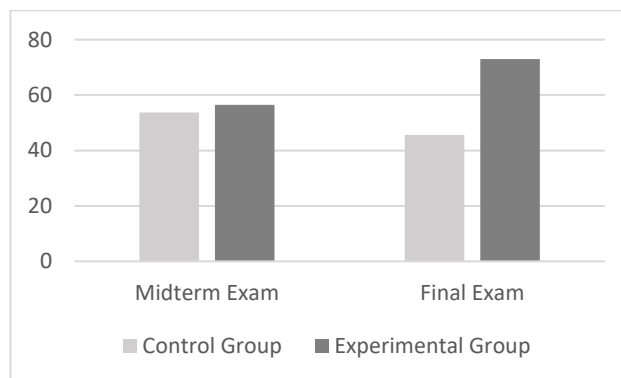


Figure 3 A comparison of student performance on Midterm and Final exams

#### 4. Discussion and Conclusion

In this study, the pits and falls of teaching VHDL language to students having learned the programming languages such as C or C++ are mentioned. There are three main types of problems we encounter in course. The first of these is the confusion created by the concepts of signal and variable. The second is thoughts on how to execute concurrent and parallel processes without an additional instruction. Finally, the details of how the processes in the process block are scheduled.

Transfer of learning is a method of applying the knowledge and skills learned in one situation to a different situation. The transfer is more likely to occur if instructors use transfer strategies that include students' knowledge and skills. Students' knowledge and experience in classical programming languages were used to contribute to the solution of these problems.

In the teaching and learning process of VHDL, the following suggestions can be of great help from the author's experience. At the beginning of the teaching process of each new concept, students should be reminded that the VHDL language is NOT a programming language, and it should be contributed to reduce the effects of the conventional method on minds by providing a clue of what kind of hardware as much as possible. In other words, thinking about hardware will contribute to the teaching process. It should be said that every statement outside of the process block is like calling a sub-function at every signal change, and the calling of operation happens automatically. It should be pointed out that the processes in the process block are executed sequentially as in programming languages as in C++, and only the updating of each signal happens at the end of the process. Therefore, only each signal in the process section has a queue data structure that works according to the FIFO rule. The signals outside process behaves the variables as in the programming languages.

## References

- [1] E. L. Thorndike, "Mental discipline in high school studies," *Journal of Educational Psychology*, vol. 15, pp. 83–98, 1924.
- [2] E. L. Thorndike and R. S. Woodworth, "The influence of improvement in one mental function upon the efficiency of other functions," *Psychological Review*, vol. 8, pp. 247–261, 1901.
- [3] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive Science*, vol. 7, pp. 155–170, 1983.
- [4] J. E. Hummel, and K. J. Holyoak, "Distributed representations of structure: A theory of analogical access and mapping," *Psychological Review*, vol. 104, pp. 427–466, 1997.
- [5] J. E. Hummel, and K. J. Holyoak, "A symbolic-connectionist theory of relational inference and generalization," *Psychological Review*, vol. 110, pp. 220–264, 2003.
- [6] A. Harrison and D. Treagust, "Teaching and Learning with Analogies: friend or foe?," In: *Metaphor and Analogy in Science Education*, pp. 11-24, Netherlands, Springer, 2006.
- [7] U. Teucher, "Metaphor in crisis: The language of suffering," *Pain and Suffering Interdisciplinary Research Network*, University of British Columbia, 2004.
- [8] N. Pitterson, N. Perova-Mello, and R. Streveler, "Engineering students' use of analogies and metaphors: Implications for educators," *International Journal of Engineering Education*, vol. 35, pp. 2-14, 2018.
- [9] R. Kanthan and S. Mills, "Using Metaphors, Analogies and Similes as Aids in Teaching Pathology to Medical Students," *The Journal of the International Association of Medical Science Educators*, vol. 16, no. 1, pp. 19-26, 2006.
- [10] Ž. Nakutis, and M. Saunoris, "Challenges of Embedded Systems Teaching in Electronic Engineering Studies," *Elektronika Ir Elektrotechnika*, vol. 102, no. 6, pp. 83-86, 2010.
- [11] R. Obeidat, and H. Alzoubi, "Why Are Hardware Description Languages Important for Hardware Design Courses?," *International Journal of Information and Communication Technology Education (IJICTE)*, vol. 17, no. 2, pp. 1-16, 2021.
- [12] R. J. Duckworth, "Embedded system design with FPGA using HDL (lessons learned and pitfalls to be avoided)," *2005 IEEE International Conference on Microelectronic Systems Education (MSE'05)*, pp. 35-36, 2005.
- [13] M.D.L.Á. Cifredo-Chacón, Á. Quirós-Olozábal, and J.M. Guerrero-Rodríguez, "Computer architecture and FPGAs: A learning-by-doing methodology for digital-native students," *Comput Appl Eng Educ*, vol. 23, pp. 464-470, 2015.
- [14] C. M. Kellett, "A Project-Based Learning Approach to Programmable Logic Design and Computer Architecture," *IEEE Transactions on Education*, vol. 55, no. 3, pp. 378-383, 2012.
- [15] S. A. Shinde and R. K. Kamat, "FPGA based Improved Hardware Implementation of Booth Wallace Multiplier using Handel C," *Elektronika Ir Elektrotechnika*, vol.109, no. 3, pp. 71-74, 2011.
- [16] F. Vahid and R. Lysecky, *VHDL for Digital Design*, John Wiley & Sons, 2007.
- [17] J. A. Nestor, "HDL coding guidelines for student projects," *2011 IEEE International Conference on Microelectronic Systems Education*, pp. 86-89, 2011.
- [18] M. Meral, C. Akuner, and I. Temiz, "Competencies of Teachers' use of Technology in Learning and Teaching Processes," *Elektronika Ir Elektrotechnika*, vol. 18, no. 10, pp. 93-97, 2012.

- [19] P. Jamieson, "VerilogTown - Improving Students Learning Hardware Description Language Design - Verilog - with a Video Game," *124th American Society for Engineering Education Annual Conference and Exposition*, vol. 29, 2017.
- [20] C. Ebeling and B. French, "Abstract Verilog: A Hardware Description Language for Novice Students," *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pp. 105-106, 2007.
- [21] S. Vemuru et al. "A spiral learning approach to hardware description languages," *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pp. 2759-2762, 2013.
- [22] A. Kumar, R. C. Panicker, and A. Kassim, "Enhancing VHDL learning through a light-weight integrated environment for development and automated checking," *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pp. 570-575, 2013.
- [23] V. Bonato, M. M. Fernandes, J. M. P. Cardoso, and Eduardo Marques, "Practical Education Fostered by Research Projects in an Embedded Systems Course," *International Journal of Reconfigurable Computing*, vol. 2014, pp. 1-12, 2014.
- [24] S. Edwards, "Experiences teaching an FPGA-based embedded systems class," *ACM Sigbed Review*, vol 2, no. 4, pp. 56-62, 2005.
- [25] G. Wang, "Lessons And Experiences Of Teaching Vhdl," *Proceedings of the 2007 American Society for Engineering Education Annual Conference & Exposition*, pp. 12.1015.1-12.1015-10, 2007.
- [26] G. Wang, "Bridging the gap between textbook and real applications: A teaching methodology in digital electronics education," *Comput. Appl. Eng. Educ.*, vol. 19, pp. 268-279, 2011.
- [27] W. Balid and M. Abdulwahed, "A novel FPGA educational paradigm using the next generation programming languages case of an embedded FPGA system course," *2013 IEEE Global Engineering Education Conference (EDUCON)*, pp. 23-31, 2013.
- [28] A. Kumar, R. C. Panicker, and A. Kassim, "Enhancing VHDL learning through a light-weight integrated environment for development and automated checking," *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pp. 570-575, 2013.
- [29] A. Kumar, S. Fernando, and R. C. Panicker, "Project-Based Learning in Embedded Systems Education Using an FPGA Platform," *IEEE Transactions on Education*, vol. 56, no. 4, pp. 407-415, 2013.

# An Implementation of Traffic Signs and Road Objects Detection Using Faster R-CNN

 Emin Güney<sup>1</sup>,  Cüneyt Bayılmış<sup>2</sup>

<sup>1</sup>Corresponding Author; Department of Computer Engineering,  
Sakarya University of Applied Sciences, Sakarya, Turkey; eminguney@subu.edu.tr

<sup>2</sup>Department of Computer Engineering, Sakarya University, Sakarya, Turkey; cbayilmis@sakarya.edu.tr

Received 14 February 2022; Revised 1 March 2022; Accepted 22 July 2022; Published online 31 August 2022

## Abstract

Traffic signs and road objects detection is a significant issue for driver safety. It has become popular with the development of autonomous vehicles and driver-assistant systems. This study presents a real-time system that detects traffic signs and road objects in the driving environment with a camera. Faster R-CNN architecture was used as a detection method in this study. This architecture is a well-known two-stage approach for object detection. Dataset was created by collecting various images for training and testing of the model. The dataset consists of 1880 images containing traffic signs and objects collected from Turkey with the German Traffic Sign Recognition Benchmark (GTSRB) dataset. These images were combined and splitted into the training and testing sets with a ratio of 80/20. The model's training was carried out in the computer test environment for about 8.5 hours and approximately 10000 iterations. The experimental results achieve a total loss rate of 0.220 and the best accuracy of 88.99% for the real-time performance. Therefore, the proposed system can be easily used for robust traffic signs and objects detection.

**Keywords:** traffic sign detection and recognition (TSDR), faster R-CNN, object detection, deep learning.

## 1. Introduction

Traffic sign and road objects detection and recognition (TSDR) play a fundamental role in keeping vehicle-driver safe in traffic and improving the driving experience. With the correct detection of traffic signs, autonomous vehicles can be developed, which have recently been an essential study field. In addition, accidents can be reduced by ensuring safe driving. Therefore, TSDR systems have been a challenging problem for many years and have become an indispensable task for driver assistance systems in addition to autonomous vehicles. Developing solution systems for this critical task is crucial for safe driving in challenging road and traffic conditions. The effective operation of recognition will also help reduce traffic accident risks [1]. Li et al. developed a system that uses color segmentation and shape matching based Pyramid Histogram Directed Gradient (PHOG) to detect traffic signs. The results were obtained by analyzing the studies tested on the original dataset in various weather conditions [2].

On the other hand, Yin et al. discussed a quick and robust system that recognizes traffic signs to increase driving safety [3]. The consisting of three stages system work using Hough and SIFT transformations and Artificial Neural Network (ANN) structure. The experimental results stated that the study was superior in training and recognition speed in traffic sign recognition. Qian et al. designed a deep Convolutional Neural Networks (CNN) system to detect traffic signs. They used colour space thresholding to identify the candidate region in the input image, while multitasking CNN was used for detection. The system has been tested on different traffic signs and texts, reaching an accuracy level of nearly 90% [4]. Changzhen et al. proposed a traffic sign detection algorithm based on Deep Convolutional Neural Network (D-CNN) using the Region Proposal Network (RPN). A Chinese traffic sign dataset was created by collecting the seven main traffic sign categories and their subclasses. Afterward, the trained network was tested on 33 videos and the real-time detection rate was 99% [5]. On the other hand, Zhang et al. used the enhanced YOLO v2 model for detecting and recognizing traffic signs in real-time [7]. Xu et al. have proposed a method using the Adaptive Color Threshold and Shape Symmetry based on Cumulative Histogram Distribution Function. In a comprehensive experimental

study on the GTSRB [6] dataset, the detection accuracy was increased up to 94%. They reached detection accuracy and time efficiency in complex traffic environments [13].

Table 1 Literature studies for traffic signs detection and recognizing task

Article	Application environment	Traffic Sign Application	Category	Technology	Datasets
Li et al. 2015 [2]	Video Series	Detection	Color segmentation and Shape matching based	Pyramid histogram of directed gradients	Private to work dataset
Yin et al. 2015 [3]	Video Series	Recognition	Hough and SIFT transform, ANN	Feature based fixed binary pattern rotation	GTSRB [6] and STS
Qian et al. 2015 [4]	Video Series	Detection	Based on deep convolutional neural networks	Edge detection and Connected component analysis	GTSRB [6], MNIST [8], CASIA [9]
Changzhen et al. 2016 [5]	Video Series	Detection	Based on deep convolutional neural networks	Region Proposal Network (RPN)	Belgium TS [10] and GTSRB
Zhang et al. 2017 [7]	Real-Time	Detection and Recognition	Based on enhanced YOLOv2	Technique with grid cells in one step	GTSRB [6] and CTSD [7]
Xu et al. 2019 [13]	Video Series	Detection	Based on Adaptive Color Threshold and Shape Symmetry	Cumulative Histogram Distribution and Shape symmetry detection	GTSRB [6]
Öztürk et al. 2020 [14]	Video Series	Recognition	Based on SSD and Faster R-CNN models	Transfer learning and grid cells	COCO [11] and GRAZ [12] datasets
Zhou et al. 2021 [16]	Video Series	Recognition	Region-Based Attention Network	Parallel Attention Network (PFANet)	ITSDB and ITSDB datasets
<b>This study (2022)</b>	Real-Time	Detection and Recognition	Based on Faster R-CNN	Detection method with Grid cells in two stages	GTSRB [6] and Private to work dataset

Öztürk et al. tested the SSD and Faster RCNN models on the image and video of traffic signs. They have reached high speed with SSD and high accuracy with Faster RCNN [14]. Han et al. achieved approximately 13% improvement in mAP by proposing an Online Hard Examples Mining (OHEM) based system for real-time small traffic sign detection with the revised Faster-RCNN [15]. Zhou et al. proposed the two-block PFANet architecture, which can extract more valuable features, to detect traffic signs even in harsh weather conditions. They have produced an innovative study, significantly increasing the accuracy rate [16]. Shao et al. presented a TSD using VGG16 and Faster R-CNN. The experimental results significantly accelerated the detection accuracy by exceeding 98% for various traffic signs [17]. Dai et al. developed the ResNet-50 architecture and presented a multi-tasking Faster R-CNN detector that simultaneously performs distance estimation and pedestrian detection [18].

The literature studies are detailed given in Table 1. This study uses the Faster R-CNN architecture to detect and recognize traffic signs and road objects. Training and testing processes were carried out in the test computer environment, and the results were obtained. Afterthat a real-time working system was obtained. The study performs the TSDR task using the Faster R-CNN architecture to recognize 23 different traffic signs and other objects. In order to increase the diversity of the dataset, in addition to the GTSRB, a job-specific data set was also collected for the study where the signs were intense in Turkey. This enriched study can assist researchers in analyzing TSDR tasks.



The remaining paper is structured as follows: Section 2 describes the used datasets and object detection model in detail. Section 3 elaborates on the comparative analysis of numerical results obtained from the experimental evaluation of the Faster R-CNN object detection algorithm for the TSDR task. Finally, Section 4 is the conclusion and discussion of the realized study.

## 2. Materials And Methods

This section presents the development stages of the created model for detecting non-vehicle traffic signs. Deep learning applications require high computing power and processing speed. To ensure learning, the number of hidden layers must be increased, and the parameters must be adjusted by constantly updating the weights. The flow chart of the implemented TSDR system is shown in Figure 1.

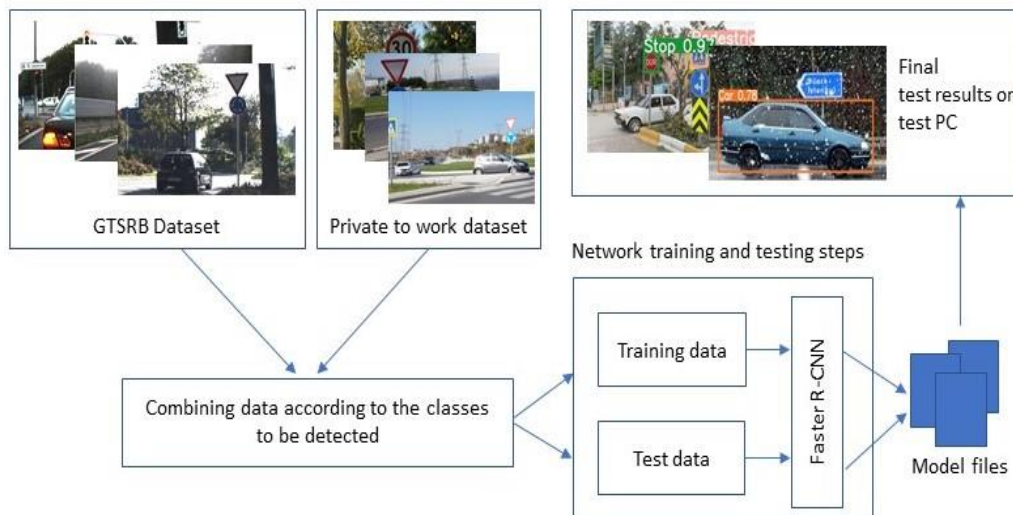


Figure 1 The implemented TSDR system architecture based on Faster R-CNN

### 2.1 Datasets

The dataset used in this article includes the German Traffic Sign Recognition Benchmark [6] (GTSRB) dataset, which is frequently used for TSDR studies in the literature. In addition to the original data, the dataset was expanded according to the classes to be detected with the data collected under various light and weather conditions. Figure 2 shows the images taken from the dataset used in the study.



Figure 2 Different images from our dataset

The dataset consists of 1880 images containing different traffic signs, vehicles and pedestrians. The data is primarily divided into various classes for detection and recognition. Afterwards, labelling processes were performed on these image data. The online software tool makesense.ai [19] was used while labelling all of the images in the datasets. In this program, the images were examined sequentially and the existing objects were selected with bounding boxes. A sample labeled image is given in Figure 3.

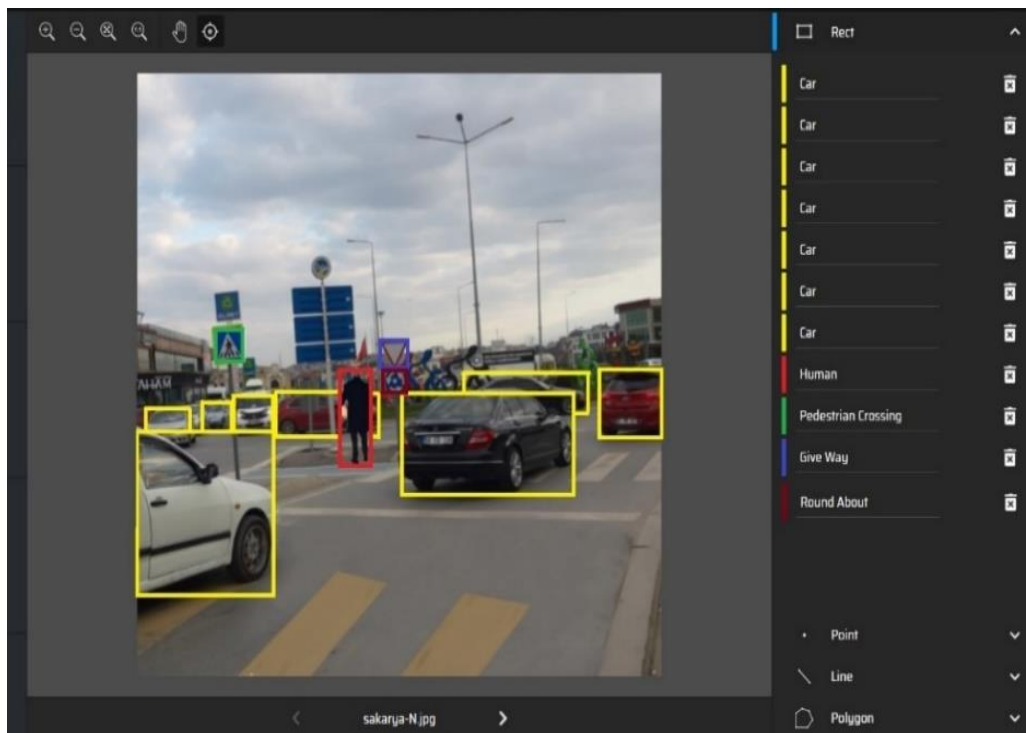


Figure 3 A sample image from the labeling process

Makesense.ai is a free-to-use online tool for labeling images. It makes the dataset preparation process easy and fast. Since the study performance is directly related to these images to be trained, it is critical to choose the corrected objects in this process. However, images in datasets contain multiple objects. Additionally, class names and distributions used in this study are shown in Table 2. After all these data processes, the images are made ready to train the model to be created and test the results by measuring.

Table 2 Brief description of our dataset. It contains information about the used classes and their amounts

Sign name	No. of Data	Sign name	No. of data
Speed Limit 20	40	Slippery Road	50
Speed Limit 30	114	Road Narrows	53
Speed Limit 50	71	Construction	50
Speed Limit 100	50	School Crossing	34
No Overtaking	50	Go Right	52
Priority Road	77	Go Left	42
Give Way	126	Go Straight	30
Stop	50	Roundabout	58
No Entry	50	Pedestrian Crossing	91
Danger	50	Human	267
Bend	51	Bicycle	106
Uneven Road	52	Car	220
Priority at Next Intersection	46		

## 2.2 Object Detection Models

Object detection is finding and classifying objects in the images. There are many approaches for object detection in terms of accuracy and speed. These approaches for the detection can be divided into two groups, and Figure 4 shows this two-stage and one-stage detection. Two-stage sensors such as Mask R-CNN [20] or Faster R-CNN use a RPN network to generate regions of interest (RoI) in the first step. It sends region proposals for object classification and bounding box regression in a pipeline. Although these kinds of models have high accuracy, they generally run slower than single-stage object detection models [21]. On the other hand, single-stage object detectors, in which an input image is predicted with a single pass through the neural network, have a higher prediction speed. Single-stage models such as YOLO [22] and SSD [23] achieve lower accuracy rates.

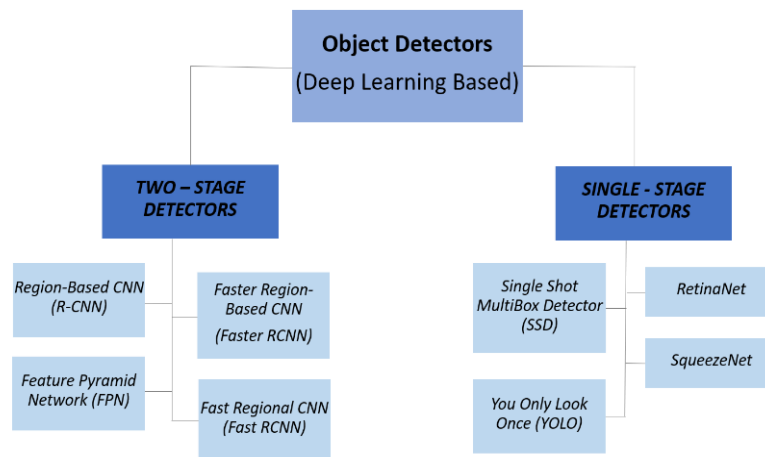


Figure 4 Classification of object detection approaches

## 2.3. Faster R-CNN

Faster R-CNN [24] was presented by Girshick et al. in 2015. This object detection model uses Convolutional Neural Networks (CNN) to classify the image in two stages. This region-based model takes a more state-of-the-art approach to accelerate region proposals similar to R-CNN [25] and Fast R-CNN [21]. Previous models used the Selective Search Algorithm [26] to find region proposals. However, this algorithm is a time-consuming process that negatively affects network performance. For this reason, the Faster R-CNN algorithm has been developed, which improves the performance of the network by eliminating Selective Search Algorithm and allows it to learn region recommendations. A schematic image from the RCNN, Fast RCNN, and Faster R-CNN models is given in Figure 5. Figure 5 shows the working principle of the Faster R-CNN approach, similar to Fast R-CNN. Region suggestions estimate offset values to classify the image within the proposed region.

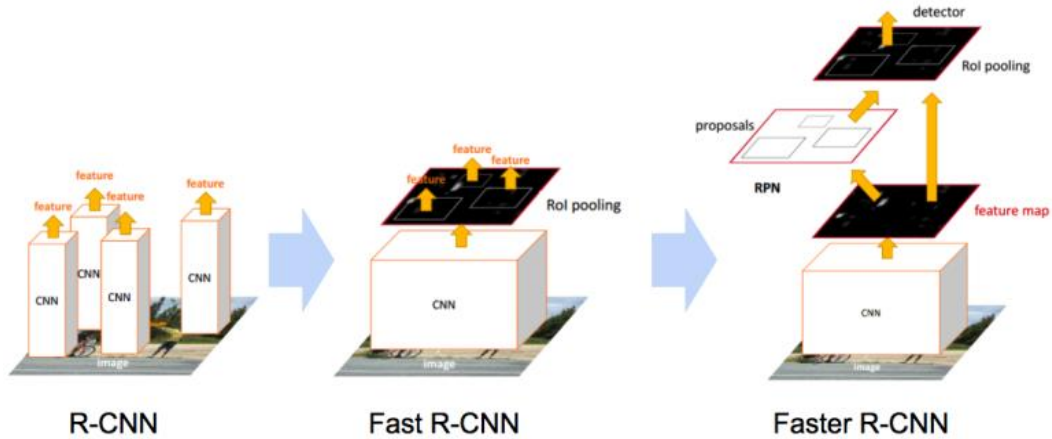


Figure 5 An illustration of R-CNN, Fast R-CNN and Faster R-CNN models [27].

### 3. Experimental Results

The experimental results was performed on the test computer given in Table 3. The dataset, consisting of 1880 images, is divided 80% to 20% for training and validation processes. Test procedures were carried out in a computer environment for 8.5 hours and approximately 10000 iterations. The training continued until the threshold value reached the appropriate value for the detection accuracy. The results show that the study with Faster R-CNN is suitable for the real-time detection of traffic signs and objects.

Table 3 Characteristics of the test environment used for the model performance

Name	Properties
Operation System (OS)	Windows 10
Central Processing Unit (CPU)	Intel Core i7 6700 HQ
Graphics Processing Unit (GPU)	Nvidia Geforce GTX 950 M 2 GB
Memory (RAM)	8 GB DD4
Harddisk (HDD)	1 TB HDD

The various loss graphics data are recorded in Table 4. As can be easily seen from Table 4, classification, localization, and total loss, etc., both training and validation rates appear to decrease consistently on a large scale. As the number of iterations increases, the loss rate decrease clearly shows the model's performance. Also, this decrement means that the model measurements are correct. In this study, the model's performance was measured according to the loss rates from loss graphics. For example, a loss rate of 0.144 for classification ( $L_{class.}$ ) and a total loss rate ( $L_{total}$ ) of 0.220 were recorded in 10000 iterations, as given in Table 4.

Table 4 Loss rates for classification and other parameters according to number of iterations

		Number of iterations									
		1k	2k	3k	4k	5k	6k	7k	8k	9k	10k
Loss rates	$L_{class.}$	0.605	0.376	0.337	0.253	0.447	0.257	0.329	0.245	0.228	<b>0.144</b>
	$L_{local.}$	0.760	0.642	0.202	0.259	0.264	0.191	0.208	0.152	0.349	<b>0.090</b>
	$L_{object.}$	0.028	0.050	0.009	0.008	0.027	0.015	0.010	0.040	0.017	<b>0.003</b>
	$L_{clone}$	1.346	1.037	0.562	0.463	0.651	0.706	0.496	0.518	0.507	<b>0.231</b>
	$L_{total}$	1.325	1.028	0.555	0.456	0.645	0.712	0.504	0.317	0.495	<b>0.220</b>

The comparison of the achievements obtained from the study with the literature is shown in Table 5. The first of the compared studies was presented by Qian et al. [4], a deep Convolutional Neural Networks (CNN) system to detect traffic signs. The other is the SSD and Faster RCNN based system presented by Öztürk et al. [14]. When we look at the other studies in the literature, compared to the

accuracy values obtained for the TSDR task performed with datasets with three classes and 10 class labels, a success value of 88.99% was calculated at 1500 iterations from test images with 25 classes in the presented study.

Table 5 The detection accuracies for traffic sign detection and recognition task obtained by other studies

Study	Qian et al. [4]	Öztürk et al. [14]	Our study
Accuracy	95%	85.1%	<b>88.99%</b>
Used dataset	GTSRB, MNIST, CASIA	COCO and GRAZ	GTSRB and Private dataset
Class no.	3 objects	10 objects	<b>25 objects</b>

The dataset was analyzed by combining the data with the study-specific dataset added to reproduce the GTSRB dataset for the test process. Also, the data that was previously reserved at the rate of 20% was used in this process. The dataset used in this study was created concerning the classes to be specified. Some of the results regarding the detection process from the test results are shown in Figure 6. While some of the test results from the dataset were estimated correctly, some estimates were inconsistent as the images of the traffic signs were similar and the data diversity was insufficient.

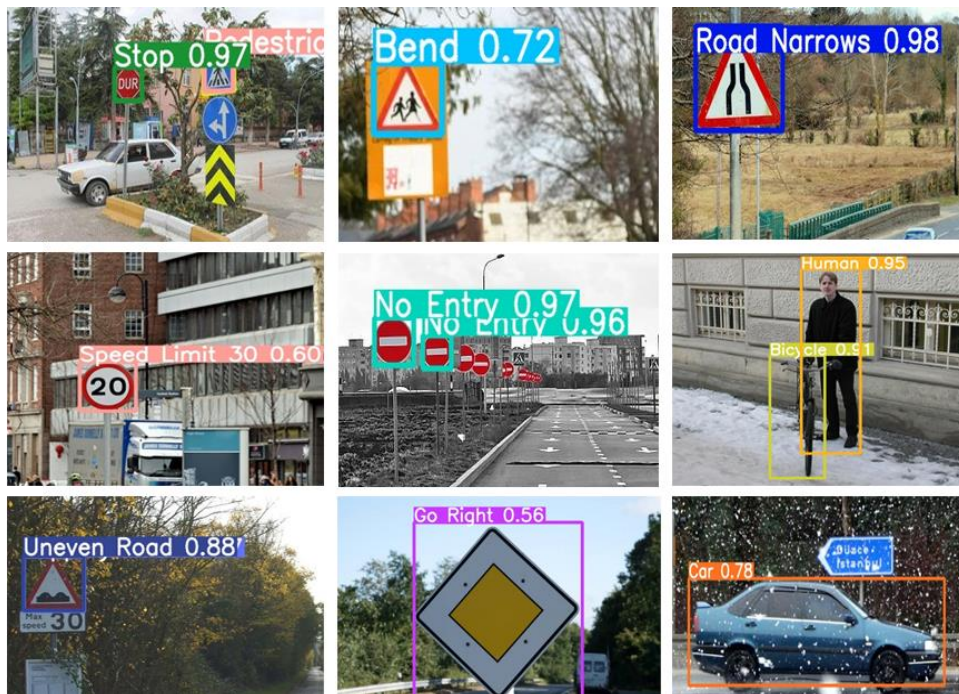


Figure 6 Test images from the developed model

#### 4. Conclusion and Discussion

Traffic sign detection and recognition systems have been a challenging problem for many years. Several methods and technologies have been developed for its solution in the past. This paper proposes a two-stage deep learning method to detect and recognize traffic signs and specific objects. For this purpose, Faster R-CNN, one of the two-stage detection methods, was used in the study. This method is frequently used in the literature for real-time object detection.

We detected 22 different traffic signs, people, bicycles and vehicles. Model was created by training the system with the dataset specifically created for working in the computer environment and the GTSRB dataset. Then, this model was tested with different images. The performance analysis of the proposed method was performed and the results were obtained. The training continued until the threshold value reached the appropriate value for the detection accuracy. Also, testing results achieve the best accuracy

of 88.99% at 1500 iterations for the real-time TSDR performance. The results show that Faster R-CNN can be efficiently used in the real-time detection of traffic signs and objects.

In future studies, it is planned to develop some TSDR methods for enhance the system performance. In addition, the dataset used can be expanded to increase accuracy and performance. Also, this article focuses on traffic signs and object detection. However, considering the applications in traffic, lane violations can be added to this system. In addition, the system can be tested on an embedded device and contribute to mobility.

### Acknowledgments

We want to thank our hardworking teammates Neslihan ÇAKIRBAŞ, Havva Selin ÇAKMAK, Ali Göktuğ YALÇIN and Dilara KOCA. They helped us prepare the datasets and the development of the system. This study was carried out with Sakarya University Scientific Research Projects Coordination Unit (Project No: 2021-7-24-20).

### References

- [1] A. Ruta, Y. Li, and X. Liu, "Real-time traffic sign recognition from video by class-spec. discriminative features," *Pattern Recognition*, vol. 43, no. 1, pp. 416–430, 2010.
- [2] H. Li, F. Sun, L. Liu, and L. Wang, "Neurocomputing A novel traffic sign detection method via color segmentation and robust shape matching," *Neurocomputing*, vol. 169, pp. 77–88, 2015.
- [3] S. Yin, P. Ouyang, L. Liu, Y. Guo, and S. Wei, "Fast Traffic Sign Recognition with a Rotation Invariant Binary Pattern Based Feature," pp. 2161–2180, 2015.
- [4] R. Qian, B. Zhang, Z. Wang, and F. Coenen, "Robust Chinese Traffic Sign Detection and Recognition with Deep Convolutional Neural Network," pp. 791–796, 2015.
- [5] X. Changzhen, W. Cong, M. Weixin, and S. Yanmei, "A Traffic Sign Detection Algorithm Based on Deep Convolutional Neural Network," pp. 6–9, 2016.
- [6] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.
- [7] J. Zhang, M. Huang, X. Jin, and X. Li, "A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2," pp. 1–13, 2017.
- [8] Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," vol. 86, no. 11, 1998.
- [9] C. Liu, F. Yin, D. Wang, and Q. Wang, "Chinese Handwriting Recognition Contest," 2010.
- [10] M. Mathias, R. Timofte, R. Benenson, and L. van Gool, "Traffic sign recognition - How far are we from the solution?," *Proceedings of the Int. Joint Conference on Neural Networks*, 2013.
- [11] T.-Y. Lin et al., "LNCS 8693 - Microsoft COCO: Common Objects in Context," 2014.
- [12] "INRIA Annotations for Graz-02 (IG02)." <https://lear.inrialpes.fr/people/marszalek/data/ig02/> (accessed Nov. 20, 2021).
- [13] X. Xu, J. Jin, S. Zhang, L. Zhang, S. Pu, and Z. Chen, "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry," *Future Generation Computer Systems*, vol. 94, pp. 381–391, 2019.
- [14] G. Ozturk, R. Koker, O. Eldogan, and D. Karayel, "Recognition of Vehicles, Pedestrians and Traffic Signs Using Convolutional Neural Networks," Oct. 2020.
- [15] C. Han, G. Gao, and Y. Zhang, "Real-time small traffic sign detection with revised faster-RCNN," *Multimedia Tools and Applications*, vol. 78, no. 10, pp. 13263–13278, May 2019.
- [16] K. Zhou, Y. Zhan, and D. Fu, "Learning region-based attention network for traffic sign recognition," *Sensors (Switzerland)*, vol. 21, no. 3, pp. 1–21, 2021.
- [17] F. Shao, X. Wang, F. Meng, J. Zhu, D. Wang, and J. Dai, "Improved faster R-CNN traffic sign detection based on a second region of interest and highly possible regions proposal network," *Sensors (Switzerland)*, vol. 19, no. 10, May 2019.

- [18] X. Dai et al., “Multi-task faster R-CNN for nighttime pedestrian detection and distance estimation,” *Infrared Physics and Technology*, vol. 115, Jun. 202.
- [19] “Make Sense,” *Victorian Studies*, vol. 48, no. 3. pp. 395–438, 2006.
- [20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN.”
- [21] R. Girshick, “Fast R-CNN,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1440–1448, 2015.
- [22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object sdetection,” *IEEE Computer Vision and Pattern Recognition*, pp. 779–788, 2016.
- [23] W. Liu et al., “SSD: Single shot multibox detector,” *Lecture Notes in Computer Science*, vol. 9905 LNCS, pp. 21–37, 2016.
- [24] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”.
- [25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” pp. 580–587, 2014.
- [26] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for obj recog,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [27] “Flagly.” <https://www.flagly.org/project/projects/4/sections/42/> (Accessed Dec. 15, 2021).

# A Relational Database Design for The Compounds Cytotoxically Active on Breast Cancer Cells

 Zeynep OKTAY<sup>1</sup>,  Çiğdem SELÇUKCAN EROL<sup>2,3</sup>,  Nazlı ARDA<sup>4,5</sup>

<sup>1</sup>Corresponding Author; Institute of Graduate Studies in Science, Istanbul University, Turkey;  
zeynepoktay2012@ogr.iu.edu.tr

<sup>2</sup>Department of Botany in Department of Biology, Faculty of Science, Istanbul University, Turkey

<sup>3</sup>Department of Informatics, Istanbul University, Turkey;  
cigdems@istanbul.edu.tr

<sup>4</sup>Department of Molecular Biology and Genetics, Faculty of Science, Istanbul University, Turkey

<sup>5</sup>Center for Research and Practice in Biotechnology and Genetic Engineering, Istanbul University, Turkey;  
narda@istanbul.edu.tr

Received 2 August 2022; Accepted 5 August 2022; Published online 31 August 2022

## Abstract

Breast cancer is one of the most important global health problems affecting both developed and developing countries. The identification of anticancer compounds, effective on breast cancer cells, is of key importance in chemoprevention investigations and drug development studies. In the literature, there are numerous compounds that have been analyzed for their cytotoxic effects on breast cancer cells, but there is no database where the researchers who want to design a new study on breast cancer can find these compounds all at once. This paper presents a relational database that stores the data of natural and synthetic compounds cytotoxically active on breast cancer cells. The database contains 381 cytotoxicity results and data of 159 compounds, compiled from selected 80 studies. When all this data in our database was queried, it was found out that quercetin, which is a dietary flavonoid, is the most analyzed compound, and MCF-7 cell line is the most used breast cancer cell line.

**Keywords:** relational database, relational model, breast cancer, cytotoxic compounds, cytotoxicity analyses

## 1. Introduction

Female breast cancer is the most diagnosed cancer worldwide [1]. In 2020, female breast cancer with 2.26 million new cases had an 11.7% rate in other cancer types [2]. Chemotherapeutic drugs used in treatment also damage normal (healthy) cells and cause serious side effects [3]. In addition, when drug resistance to chemotherapy occurs [4, 5], the treatment becomes ineffective.

In this regard, studies are progressing to find new compounds effective in inhibiting the growth of breast cancer cells. In the scientific literature, there is a wide range of cell culture studies (*in vitro* studies) analyzing the cytotoxic effects and anticancer activities of natural and synthetic compounds on breast cancer cell lines. Especially, plant secondary metabolites, mainly flavonoids, are at the center of these studies [6, 7, 8]. Plant extracts [9], synthetic derivatives of natural compounds [10], and compounds derived from different organisms, such as fungal taxol isolated from *Pestalotiopsis pauciseta* VM1 [11] and propolis as a honeybee product [12], are the other examples of focus of these cell culture studies. The identification of compounds with cytotoxic effects is crucial to both define compounds with therapeutic potential and to provide the basis for advanced studies.

As in numerous fields, in life sciences, data has critical importance and an effect on directing the decision-making process. Databases are computerized systems used for storing, accessing, retrieving, and managing data. The relational database management system (RDBMS) is the most used database technology today. RDBMS is based on Edgar F. Codd's "relational data model" [13]. SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS [14]. The main principle in RDBMS is to store the data in the form of relations. In the RDBMS, data is stored in tables where each record is held in a row and each attribute of the data is held in a column [15, 16]. The



advantages of relational databases are the ease of adding, inserting, updating, deleting data, and practicability in data retrieval, querying, and reporting [17].

In the present study, we designed a relational database compiling natural and synthetic compounds with cytotoxic effects on breast cancer cells. We aimed this database to be the first example of a digital library of literature that presents the whole data related to those compounds in the context of compound name, publication to which it belongs, breast cancer cell line, the origin organism of the compound, the class of the compound, dose, treatment time, and cytotoxicity level for the researchers who study on breast cancer.

## 2. Material and Method

### 2.1. Literature Search and Study Selection

Scholar Google and Science Direct databases were searched for cell culture studies assessing the effects of natural and synthetic compounds on breast cancer cell lines. Having cytotoxicity analysis results and % inhibition data were the criteria for selecting studies that would be included in our database. In selecting studies, the date range was not taken into account.

### 2.2. The Scope of Literature Review and General Characteristics of The Selected Studies

When we reviewed the literature by using the expressions of “breast cancer and natural compounds”, “breast cancer and synthetic compounds”, “breast cancer and cytotoxicity”, “breast cancer and *in vitro* assays” as keywords, we came across two main group of studies with different specifications. We found that while the vast majority of studies focused solely on the cytotoxic effects of compounds (such as quercetin, lycopene, curcumin, apigenin, etc.), the other group of studies focused on the cytotoxic effects of herbal extracts. For instance, in [18] Takeshima et al. investigated the effect of lycopene, which is a plant pigment called carotenoid mostly found in tomatoes, in breast cancer cells, and in [19] Duo et al. investigated quercetin’s effect. On the other hand, in [20] Alsabah et al. studied the effects of *Xanthium strumarium*, which is an annual herb, against breast cancer cell lines.

In our literature review, we also saw multi-component studies that assess more than one compound’s cytotoxic effect on more than one breast cancer cell line. For instance, in [21] Ligresti et al. investigated the antitumor activity of five different plant cannabinoids (cannabidiol, cannabigerol, cannabichromene, cannabidiol acid and THC acid) on two different breast cancer cell lines (MCF-7 and MDA-MB-231 cell lines). In [22] the authors investigated the cytotoxic effects of six different compounds (beta-carotene, lycopene, all-trans-, 9-cis- and 13-cis-retinoic acid and all-trans-retinol) on three breast cancer cell lines (MCF-7, MDA-MB-231, Hs578T cell lines).

As it is known, Scholar Google and similar databases are the most popular databases used for literature review. While reviewing the literature by using these databases, various keywords and keyword combinations related to the required study are used. Most of the time, tens of Scholar Google pages need to be scanned one by one, and each study in those pages needs to be looked at individually. When it is taken into account that there are hundreds of studies on breast cancer, this process is quite time-consuming for the researcher and may cause the studies needed by the researcher to escape the attention. Based on these disadvantages of Scholar Google and similar databases, we decided to design a relational database that brings together the literature centered on cytotoxically active compounds on breast cancer cells.

### 2.3. Creating Tables

To design the database, Microsoft Access (2007-2016 file format) was used. In the first step, five tables belonging to the database were created. These tables were named Publications, Cell Lines, Compounds, Treatments, and Cytotoxicity, respectively.

In the publications table, five columns were created, and these columns were named as publication ID, authors, publication year, publication name, and journal name, respectively. Publication ID was described as the primary key in this table. 80 publication records were entered into publications table. The first 21 rows of this table are shown in Figure 1.

PublicationID	Authors	PublicationYear	PublicationName	JournalName
1	Alosi, J. A., McDoi	2010	Pterostilbene inhibits breast cancer in vitro through mi	Journal of Surgical Research
2	Atmaca, H., Bozku	2016	Effects of Galium aparine extract on the cell viability, c	Journal of Ethnopharmacology
3	Bielawski, K., Czar	2013	Cytotoxicity and induction of apoptosis of human brea	Environmental Toxicology and Pharma
4	Chang, H. C., Chei	2011	Capsaicin may induce breast cancer cell death through	Human and Experimental Toxicology
5	Choi, E. J., & Kim,	2009	Apigenin induces apoptosis through a mitochondri	J.Clin.Biochem.Nutr.
6	Chou, C. C., Yang,	2010	Quercetin-mediated cell cycle arrest and apoptosis inv	Archives of Pharmacal Research
7	Choudhury, B., Ke	2018	Garcinia morella fruit, a promising source of antioxid	Biomedicine & Pharmacotherapy
8	Damianaki, A., Ba	2000	Potent inhibitory action of red wine polyphenols on hu	Journal of Cellular Biochemistry
9	Duo, J., Ying, G. G	2012	Quercetin inhibits human breast cancer cell proliferat	Molecular Medicine Reports
10	Elamin, M. H., Da	2013	Olive oil oleuropein has anti-breast cancer properties v	Food and Chemical Toxicology
11	Fan, P., Fan, S., W	2013	Genistein decreases the breast cancer stem-like cell po	Stem Cell Research & Therapy
12	Graidist, P., Martl	2015	Cytotoxic activity of Piper cubeba extract in breast can	Nutrients
13	Gloria, N. F., Soar	2014	Lycopene and beta-carotene induce cell-cycle arrest an	Anticancer Research
14	Han, J., Talorete,	2009	Anti-proliferative and apoptotic effects of oleuropein a	Cytotechnology
15	Hu, H., Ahn, N. S.,	2002	Ganoderma lucidum extract induces cell cycle arrest ar	Int. J. Cancer
16	Hui, C., Bin, Y., Xie	2010	Anticancer activities of an anthocyanin-rich extract fro	Nutrition and Cancer
17	Jada, S. R., Matth	2008	Benzylidene derivatives of andrographolide inhibit gro	British Journal of Pharmacology
18	Khanavi, M., Nab	2010	Cytotoxic activity of some marine brown algae against	Biological Research
19	Khare, N., & Chan	2019	Stevioside mediated chemosensitization studies and cy	Saudi Journal of Biological Science
20	Chryssanthi, D. G	2007	Inhibition of breast cancer cell proliferation by style	Anticancer Research
21	Lee, C. J., Wilson,	2010	Hesperidin suppressed proliferations of both human b	Phytotherapy Research
22	Li Y., Zhang T. Ki	2010	Sulforanhanane a dietary component of Broccoli/Brocco	Clinical Cancer Research

Figure 1 Publications table

Into the cell lines table, breast cancer cell lines (such as MCF-7, MDA-MB-231, etc. are defined as “cell lines” in the related literature) used in the selected studies were entered. This table’s three columns were named as cell line ID, publication ID and cell line name. Cell line ID was described as the primary key in this table and unique for every individual breast cancer cell line used in every individual study. In this way, the confusion that may be caused by using the same breast cancer cell lines in more than one study was prevented. Totally, 141 records of breast cancer cell lines used in the selected studies were entered into this table. The first 21 rows of this table are shown in Figure 2.

CellLineID	PublicationID	CellLine_Name
1	1	MCF-7
2	1	MDA-MB-231
3	2	MCF-7
4	2	MDA-MB-231
5	3	MCF-7
6	3	MDA-MB-231
7	4	MCF-7
8	4	BT-20
9	5	MDA-MB-453
10	6	MCF-7
11	7	MCF-7
12	7	MDA-MB-231
13	7	SKBR3
14	8	MCF-7
15	8	MDA-MB-231
16	8	T47D
17	9	MCF-7
18	10	MCF-7
19	10	MDA-MB-231
20	11	MCF-7
21	12	MCF-7
22	12	MDA-MB-231

Figure 2 Cell lines table

The compounds table was created to include the details of compounds used in the selected studies. Compound ID, publication ID, compound name, compound class, origin, solvent/extract type were the columns created in this table. If there was not any data about the class, origin, solvent, or extraction type of the compound in the relevant study, the part of the relevant column was intentionally left blank. Totally, 159 records of compounds were entered into this table. The first 21 rows of the compounds table are shown in Figure 3.

CompoundID	PublicationID	CompoundName	CompoundClass	Origin	Solvent (or Extract Type)
1	1	Pterostilbene	analogue of resveratrol		DMSO
2	2	Galium aparine extract		Galium aparine	MeOH extract
3	3	Pt2(3-ethylpyridine)4(bereni)2 (Pt10)	dinuclear platinum(II) compound		
4	3	Pt2(3-butylpyridine)4(bereni)2 (Pt11)	dinuclear platinum(II) compound		
5	4	Capsaicin			
6	5	Apigenin	flavone subclass of flavonoids		DMSO
7	6	Quercetin	flavonoid		
8	7	Garcinia morella extract		Garcinia morella	GFCH (chloroform fraction)
9	8	Resveratrol	Red wine polyphenols		absolute ethanol
10	8	Quercetin	Red wine polyphenols		absolute ethanol
11	8	Catechin	Red wine polyphenols		absolute ethanol
12	8	Epicatechin	Red wine polyphenols		absolute ethanol
13	9	Quercetin	flavonoid		DMSO
14	10	Oleuropein	phenolic compound	olive leaf	
15	11	Genistein	isoflavone		DMSO
16	12	Piper cubeba extract		Piper cubeba	Methanolic and dichloromethane
17	13	Lycopene	carotenoid		
18	13	Beta-carotene	carotenoid		
19	14	Oleuropein	phenolic compound	olive oil	
20	14	Hydroxytyrosol	phenolic compound	olive oil	
21	15	Ganoderma lucidum extract		oriental fungus	alcohol extract
22	16	Anthocyanin-Rich Extract from Black Rice(AFBR)		Oruza sativa L. indica	

Figure 3 Compounds table

The treatments table's primary key, treatment ID, is specific to breast cancer cell line-compound-treatment time combination. For instance, as seen in Figure 4, in the record of the publication which has Publication ID 1 [23], we see 6 different Treatment IDs belonging to it. The reason for this is that Pterostilbene is administered to MCF-7 and MDA-MB-231 breast cancer cell lines for three different administration periods and each administration period is represented by a unique Treatment ID (Treatment ID 1, 2, 3 for 24, 48, 72 h on MCF-7 and Treatment ID 4, 5, 6 for 24, 48, 72 h on MDA-MB-231 consecutively).

TreatmentID	PublicationID	CellLineID	CellLine_Name	CompoundID	CompoundName	Treatment_Time(h)
1	1	1	MCF-7	1	Pterostilbene	24
2	1	1	MCF-7	1	Pterostilbene	48
3	1	1	MCF-7	1	Pterostilbene	72
4	1	2	MDA-MB-231	1	Pterostilbene	24
5	1	2	MDA-MB-231	1	Pterostilbene	48
6	1	2	MDA-MB-231	1	Pterostilbene	72
7	2	3	MCF-7	2	Galium aparine extract	72
8	2	4	MDA-MB-231	2	Galium aparine extract	72
9	3	5	MCF-7	3	Pt2(3-ethylpyridine)4(bereni)2 (Pt10)	24
10	3	5	MCF-7	4	Pt2(3-butylpyridine)4(bereni)2 (Pt11)	24
11	3	6	MDA-MB-231	3	Pt2(3-ethylpyridine)4(bereni)2 (Pt10)	24
12	3	6	MDA-MB-231	4	Pt2(3-butylpyridine)4(bereni)2 (Pt11)	24
13	4	7	MCF-7	5	Capsaicin	72
14	4	8	BT-20	5	Capsaicin	72
15	5	9	MDA-MB-453	6	Apigenin	24
16	5	9	MDA-MB-453	6	Apigenin	72
17	6	10	MCF-7	7	Quercetin	48
18	7	11	MCF-7	8	Garcinia morella extract	24
19	7	11	MCF-7	8	Garcinia morella extract	48
20	7	11	MCF-7	8	Garcinia morella extract	72
21	7	12	MDA-MB-231	8	Garcinia morella extract	24
22	7	12	MDA-MB-231	8	Garcinia morella extract	48

Figure 4 Treatments table

The effects of compound administrations on breast cancer cell viabilities were entered into the cytotoxicity table as cytotoxicity data in the context of dose of compound, % inhibition, and % survival rate. In Figure 5, the cytotoxicity table and its columns are shown. With the purpose of creating columns

(fields) as simple and in atomic structure as possible, we placed unit of dose and standard deviations in separate columns. The cytotoxicity table has 381 records (rows) of cytotoxicity data.

ID	TreatmentID	Dose	+/-SD_of_Dose	Unit_of_Dose	%Inhibition	+/-SD_of_%Inhibition	%SurvivalRate	+/-SD_of_%SurvivalRate
1	1	59.42	7.89	micromolar	50			
2	2	40.51	10.72	micromolar	50			
3	3	26.42	10.84	micromolar	50			
4	4	56.37	17.56	micromolar	50			
5	5	29.60	4.77	micromolar	50			
6	6	20.21	2.88	micromolar	50			
7	7	503		ug/ml	50			
8	8	486		ug/ml	50			
9	9	45	2	micromolar	50			
10	10	20	2	micromolar	50			
11	11	30	2	micromolar	50			
12	12	11	2	micromolar	50			
13	13	200		micromolar			58.50	2.5
14	14	200		micromolar			36.24	2.3
15	15	59.44		micromolar	50			
16	16	35.15		micromolar	50			
17	17	92.4		micromolar	50			
18	18	5.18	0.38	ug/ml	50			
19	19	4.84	0.22	ug/ml	50			
20	20	3.98	0.15	micromolar	50			
21	21	6.25	0.62	ug/ml	50			
22	22	5.40	0.36	ug/ml	50			

Figure 5 Cytotoxicity table

The data extraction from the publications and the data input were carried out manually.

## 2.4. Creating Relationships Between Tables

The relationships between tables were created according to relationship defining instructions [24, 25]. The created relationships are shown in Figure 6.

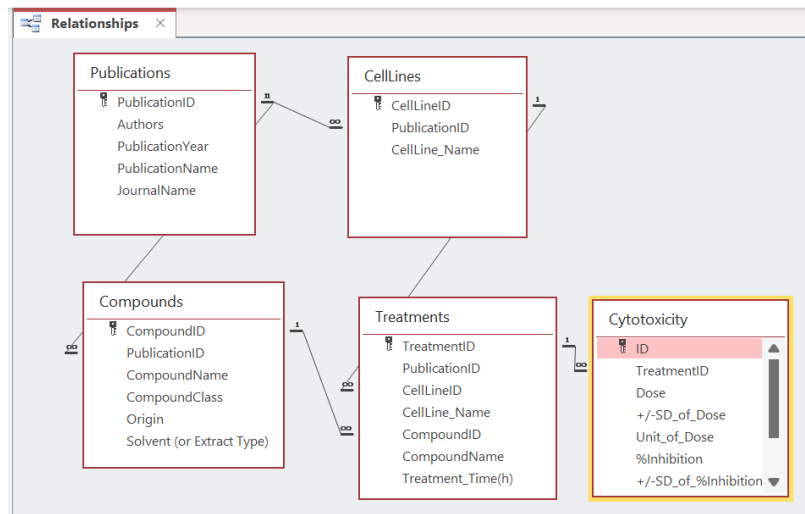


Figure 6 Relationships between tables

## 3. Results

When we queried with SQL, which breast cancer cell line was the most used one in 80 studies in our database, it was seen that MCF-7 was the most researched cell line with 64 studies. MDA-MB-231 cell line is following that with 34 studies. The SQL query for publications using MCF-7 cell line is shown in Figure 7 and first 25 of 64 results are seen in Figure 8.

```

MCF-7 and Publication Query
SELECT Publications.PublicationID, Publications.PublicationYear, Publications.PublicationName, CellLines.CellLine_Name
FROM Publications INNER JOIN CellLines ON Publications.[PublicationID] = CellLines.[PublicationID]
WHERE CellLine_Name='MCF-7';

```

Figure 7 SQL query for publications using MCF-7 cell line

PublicationID	PublicationYear	PublicationName	CellLine_Name
1	2010	Pterostilbene inhibits breast cancer in vitro through mi	MCF-7
2	2016	Effects of Galium aparine extract on the cell viability, c	MCF-7
3	2013	Cytotoxicity and induction of apoptosis of human brea	MCF-7
4	2011	Capsaicin may induce breast cancer cell death through	MCF-7
6	2010	Quercetin-mediated cell cycle arrest and apoptosis inv	MCF-7
7	2018	Garcinia morella fruit, a promising source of antioxidant	MCF-7
8	2000	Potent inhibitory action of red wine polyphenols on hu	MCF-7
9	2012	Quercetin inhibits human breast cancer cell proliferati	MCF-7
10	2013	Olive oil oleuropein has anti-breast cancer properties v	MCF-7
11	2013	Genistein decreases the breast cancer stem-like cell po	MCF-7
12	2015	Cytotoxic activity of Piper cubeba extract in breast canc	MCF-7
13	2014	Lycopene and beta-carotene induce cell-cycle arrest an	MCF-7
14	2009	Anti-proliferative and apoptotic effects of oleuropein a	MCF-7
15	2002	Ganoderma lucidum extract induces cell cycle arrest ar	MCF-7
16	2010	Anticancer activities of an anthocyanin-rich extract fro	MCF-7
17	2008	Benzylidene derivatives of andrographolide inhibit gro	MCF-7
20	2007	Inhibition of breast cancer cell proliferation by style co	MCF-7
22	2010	Sulforaphane, a dietary component of Broccoli/Brocco	MCF-7
23	2017	Diosmin-induced senescence, apoptosis and autophag	MCF-7
24	2006	Antitumor activity of plant cannabinoids with emphasi	MCF-7
25	2004	In vitro anti-proliferative activities of ellagic acid	MCF-7
26	2015	Antitumor effects of crocin on human breast cancer ce	MCF-7

Figure 8 Results of query for publications using MCF-7 cell line

Similar SQL queries were also performed for the compounds. Quercetin was found to be the most preferred compound among the studies in our database. The SQL query for publications using quercetin is shown in Figure 9 and the results are presented in Figure 10.

```

Quercetin and Publication Query
SELECT Publications.PublicationID, Publications.PublicationYear, Publications.PublicationName, Compounds.CompoundName
FROM Publications INNER JOIN Compounds ON Publications.[PublicationID] = Compounds.[PublicationID]
WHERE CompoundName='Quercetin';

```

Figure 9 SQL query for publications using quercetin

PublicationID	PublicationYear	PublicationName	CompoundName
5	2010	Quercetin-mediated cell cycle arrest and apoptosis inv	Quercetin
8	2000	Potent inhibitory action of red wine polyphenols on hu	Quercetin
9	2012	Quercetin inhibits human breast cancer cell proliferati	Quercetin
40	2015	Apigenin inhibits growth of breast cancer cells: The rol	Quercetin
46	2015	Cytotoxic effect of Turkish propolis on liver, colon, bre	Quercetin

Figure 10 Results of query for publications using quercetin

#### 4. Discussion

In the sense of user experience, this relational database has remarkable differences from the conventional literature search databases. Firstly, this database enables users to access compounds effective in breast cancer cells and the cytotoxic effect levels of these compounds together. For instance, a researcher who wants to review the cytotoxic effects of quercetin in breast cancer cells in conventional databases should perform a search with the keywords such as breast cancer, quercetin, cytotoxicity, and also look at every publication on the search pages one by one to see the cytotoxic analysis results.

On the other hand, a researcher using our relational database can access cytotoxic compounds, breast cancer cell lines in which these compounds are effective and cytotoxicity test results (in the context of numeric data) at the same time with SQL query commands quickly and easily. Continuing from the example of quercetin mentioned above, the researcher can see in which cell lines quercetin is administered, during which treatment time and in which dosage intervals it causes a cytotoxic effect.

In addition to these, with the help of this relational database, the most and the least investigated cell lines and compounds can be queried with simple SQL commands and the popular study focuses can be specified.

## 5. Conclusion and Future Works

Cytotoxic analyses are critical in drug development. Especially in the investigation of new cancer therapeutics, it is essential to identify compounds with cytotoxic effects on cancer cells. Studies analyzing the cytotoxic effects of the compounds with anti-cancer properties on different cancer cells constitute a very rich source of data. However, there is no database where researchers can collectively access the results of these studies in the context of cell type-compound-dose and % inhibition. This database we have designed for the compounds possessing cytotoxic effects on breast cancer cells is the first one in this sense.

The data in this database has the potential to be updated and enriched continuously by the inclusion of new publications in the literature. It is planned to make the database accessible to the end-user in the internet environment. In this way, this database will be a reference source for researchers, who want to design a study on breast cancer, where they can see cytotoxic compounds and the studies made with these compounds collectively, and also save time in the literature search step.

With the experience gained in designing a relational database while creating this database prototype, similar databases can be designed for the investigations on other cancer types in the future.

## Acknowledgments

The authors wish to thank Assoc. Prof. Dr. Emre AKADAL from Department of Management Information Systems in Istanbul University Faculty of Economics, for his precious contributions.

## References

- [1] WHO, "Newsroom Cancer," 2022. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/cancer>. [Accessed: 24-June-2022].
- [2] H. Sung, J. Ferlay, R. L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, and F. Bray, "Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries," *CA: A Cancer Journal for Clinicians*, vol. 71, no. 3, pp. 209-249, 2021.
- [3] S. Łukasiewicz, M. Czezelewski, A. Forma, J. Baj 3, R. Sitarz, and A. Stanisławek, "Breast Cancer—Epidemiology, Risk Factors, Classification, Prognostic Markers, and Current Treatment Strategies— An Updated Review," *Cancers*, vol. 13, no. 17, pp. 1-30, 2021 266–275, 2021.
- [4] F. Leonessa and R. Clarke, "ATP binding cassette transporters and drug resistance in breast cancer," *Endocrine-Related Cancer*, vol. 10, pp. 43–73, 2003.
- [5] T. Saeki, T. Tsuruo, W. Sato, and K. Nishikawsa, "Drug resistance in chemotherapy for breast cancer," *Cancer Chemotherapy and Pharmacology*, vol. 56, pp. 84-89, 2005.
- [6] N. L. Vukovic, A. D. Obradovic, M. D. Vukic, D. Jovanovic, and P. M. Djurdjevic, "Cytotoxic, proapoptotic and antioxidative potential of flavonoids isolated from propolis against colon (HCT-116) and breast (MDA-MB-231) cancer cell lines," *Food Research International*, vol. 106, pp. 71-80, 2018.
- [7] A. Damianaki, et al., "Potent Inhibitory Action of Red Wine Polyphenols on Human Breast Cancer Cells," *Journal of Cellular Biochemistry*, vol. 78, pp. 429-441, 2000.

- [8] H. Nakagawa, Y. Kiyozuka, Y. Uemura, H. Senzaki, N. Shikata, K. Hioki, and A. Tsubura, "Resveratrol inhibits human breast cancer cell growth and may mitigate the effect of linoleic acid, a potent breast cancer cell stimulator," *J. Cancer Res. Clin. Oncol.*, vol. 127, pp. 258-264, 2001.
- [9] E. Safarzadeh, et al., "The Cytotoxic and Apoptotic Effects of Scrophularia Atropatana Extracts on Human Breast Cancer Cells," *Advanced Pharmaceutical Bulletin*, vol. 7, no.3, pp. 381-389, 2017.
- [10] E. Pierpaoli, A. G. Arcamone, F. Buzzetti, P. Lombardi, C. Salvatore, and M. Provinciali, "Antitumor effect of novel berberine derivatives in breast cancer cells," *BioFactors*, vol. 39, no.6, pp. 672-679, 2013.
- [11] R. Vennila, S. Kamalraj, and J. Muthumary, "In vitro Studies on anticancer activity of fungal taxol against human breast cancer cell line MCF-7 cells," *Asian Pacific Journal of Tropical Biomedicine*, vol. 2, no. 2, pp. 1159-1161, 2012.
- [12] I. Turan, S. Demir, S. Misir, K. Kilinc, A. Mentese, Y. Aliyazicioglu, and O. Deger, "Cytotoxic Effect of Turkish Propolis on Liver, Colon, Breast, Cervix and Prostate Cancer Cell Lines," *Tropical Journal of Pharmaceutical Research*, vol. 14, no. 5, pp. 777-782, 2015.
- [13] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377-387, 1970.
- [14] Y. Bassil, "A Comparative Study on the Performance of the Top DBMS Systems," *Journal of Computer Science & Research*, vol. 1, no. 1, pp. 20-31, 2012.
- [15] S. Sumathi and S. Esakkirajan, *Fundamentals of Relational Database Management Systems*. Berlin, Heidelberg: Springer-Berlin Heidelberg, 2007.
- [16] K. Ito, "Relational Database," 2021. [Online]. Available: <https://www.czc.hokudai.ac.jp/bioinform/pdf/2021-08-30-RDB-KI.pdf>. [Accessed: 02-June-2022].
- [17] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, "A Survey and Comparison of Relational and Non-Relational Database," *International Journal of Engineering Research & Technology (IJERT)*, vol.1, no. 6, pp. 1-5, 2012.
- [18] M. Takeshima, M. Ono, T. Higuchi, C. Chen, T. Hara, and S. Nakano, "Anti-proliferative and apoptosis-inducing activity of lycopene against three subtypes of human breast cancer cell lines", *Cancer Science*, vol. 105, no. 3, pp. 252-257, 2014.
- [19] J. Duo, G. G. Ying, G. W. Wang, and L. Zhang, "Quercetin inhibits human breast cancer cell proliferation and induces apoptosis via Bcl-2 and Bax regulation", *Molecular Medicine Reports*, vol. 5, pp. 1453-1456, 2012.
- [20] A. S. Alsabah, A. H. Abd, A. M. Al-Shammari, "Cytotoxicity of Xanthium Strumarium against Breast Cancer Cell Lines", *Journal of Global Pharma Technology*, vol. 10, no. 3, pp. 767-776, 2018.
- [21] A. Ligresti, et al., "Antitumor Activity of Plant Cannabinoids with Emphasis on the Effect of Cannabidiol on Human Breast Carcinoma", *The Journal of Pharmacology and Experimental Therapeutics*, vol. 318, no. 3, pp. 1375-1387, 2006.
- [22] P. Prakash, R. M. Russell, and N. I. Krinsky, "In Vitro Inhibition of Proliferation of Estrogen Dependent and Estrogen-Independent Human Breast Cancer Cells Treated with Carotenoids or Retinoids", *The Journal of Nutrition*, vol. 131, no. 5, pp. 1574-1580, 2001.
- [23] J. A. Alosi, D. E. McDonald, J. S. Schneider, A. R. Privette, and D. W. McFadden, "Pterostilbene Inhibits Breast Cancer In Vitro Through Mitochondrial Depolarization and Induction of Caspase-Dependent Apoptosis," *Journal of Surgical Research*, vol. 161, pp. 195-201, 2010.
- [24] Microsoft, "How to define relationships between tables in an Access database," 2022. [Online]. Available:<https://docs.microsoft.com/en-us/office/troubleshoot/access/define-table-relationships>. [Accessed: 30-May-22].
- [25] E. Akadal, *Veritabanı Tasarlama Atölyesi*. İstanbul, Türkmen Kitabevi, 2021.

# The Effect of Ambient Temperature On Device Classification Based On Radio Frequency Fingerprint Recognition

 Ozkan Yilmaz<sup>1</sup>,  Mehmet Akif Yazici<sup>2</sup>

<sup>1</sup>Aselsan, Communications and Information Technologies Business Sector, Ankara, Türkiye;  
ozkanyilmaz@aselsan.com.tr

<sup>2</sup>Corresponding Author; Istanbul Technical University, Informatics Institute, Information and Communications Research Group, Istanbul, Türkiye; yazicima@itu.edu.tr; Tel: +90 212 285 71 94

Received 30 Jun 2022; Revised 06 August 2022; Accepted 06 August 2022; Published online 31 August 2022

## Abstract

Physical layer authentication is an important technique for cybersecurity, especially in military scenarios. Device classification using radio frequency fingerprinting, which is based on recognizing device-unique characteristics of the transient waveform observed at the beginning of a transmission from a radio device, is a promising method in this context. In this study, the effect of the ambient temperature on the performance of radio device classification based on RF fingerprinting is investigated. The waveforms of the transient regions of the transmissions are recorded as images, and ResNet50 and InceptionV3 networks for image classification are used to determine the radio devices. The radio devices used in the study belong to the same brand, model, and production date, making the problem more difficult than classifying radio devices of different brands or models. Our results show that high levels of accuracy can be attained using convolutional neural network models such as ResNet50 and InceptionV3 when the test data and the training data are collected at the same temperature, whereas performance suffers when the test data and the training data belong to different temperature values. We provide the performance figures of a blended training model that uses training data taken at various temperature values. A comparison of the two networks is also provided.

**Keywords:** cybersecurity, device classification, radio frequency fingerprint, double sliding window, image classification, resnet50, inceptionV3

## 1. Introduction

Radio device recognition and classification is important from a cybersecurity point of view in many applications, such as law enforcement and military use cases. Radio frequency (RF) fingerprinting is one of the techniques that can be used in device classification. When a radio transmitter first turns on or starts broadcasting to the air, the signal emitted from the transmitter exhibits a transient behavior. The transient region duration may be in the order of microseconds, depending on the hardware of the transmitter. It has been shown that the transient behavior region contains unique features of the radio transmitter [1]. These unique features are due to the unique characteristics of hardware components such as analog converters, filters, power amplifiers, and frequency mixers used during the manufacturing of the transmitter layer, and various defects in the soldering process during the assembly of these components on the boards. In addition, the aging of the radio transmitter may cause the transient region to differ in devices with the same brand, model, and production date, even if they are products of a high quality manufacturing process. The signal characteristics in the transient region are different for each radio transmitter, which is called the RF fingerprint.

Encryption algorithms are mainly used to identify a wireless device that has been authorized by the system. In an encrypted communication system, a two-way communication is required to generate a session key [2]. However, the security algorithm will be compromised during access to the key, thus making it difficult to distinguish a legitimate key. Such problems encountered in encrypted communication systems can be effectively solved by using physical layer security [3]. At this point, it would be correct to explain the physical layer security. Physical layer security is the practice of identifying wireless devices by extracting unique features embedded in the electromagnetic waves



emitted from transmitters [4]. Physical layer security based on the recognition of these unique features is known as Radio Frequency (RF) fingerprinting [5]. Radio frequency fingerprinting has been applied to various communication technologies and standards, including cognitive radio networks [6], Universal Mobile Telecommunications System (UMTS) [7], Wi-Fi [8], push-to-talk transmitters [9], bluetooth [10], and Radio-Frequency Identification (RFID) [11].

RF fingerprint extraction has been done by high-end receiver devices in many studies. Rehman et al. [12] showed that there is practically no need for high-level receiver equipment to obtain RF fingerprinting, but low-level receiver equipment can also be used. In addition, they tested the performance of the RF fingerprinting systems they developed against impersonation attacks. Tekbas et al. [13] classified different models and brands of radios with RF fingerprints and examined the effects of ambient temperature, battery voltage, and ambient noise on the classification success during classification. High-end receiver equipment is used for RF fingerprinting. Riyaz et al. [14] examined the use of convolutional neural networks for device classification with RF fingerprinting. Rehman et al. [15] counts the effects of the ages of the devices, the ambient temperature, and the mobility of the devices during the fingerprinting process on the fingerprints as further issues to be explored. Wang et al. [16] discussed the low performance of device classification by RF fingerprinting, and proposed deep complex residual networks as a new method to overcome this problem. The deep complex residual network has been integrated into the RF fingerprint extraction and the device classification model, and it has been found that the accuracy rates in device classification have increased. Suski et al. [17] investigated the use of RF fingerprinting for the security of commercial devices broadcasting in the IEEE 802.11a standard. In many studies, the ambient temperature is ignored as a parameter of the device classification with RF fingerprinting.

In this study, narrowband radios with the same brand, model, and production date were used to investigate the effect of the ambient temperature on RF fingerprinting. We also demonstrate that device classification, identification, and similar procedures can be performed at low cost using low-level receiver hardware for RF fingerprinting. RF fingerprints of the radios were obtained at ambient temperatures of  $-5$ ,  $10$ ,  $25$ , and  $40$  °C. The obtained RF fingerprints were classified at different temperatures with the convolutional neural network models ResNet50 and InceptionV3, which are branches of deep learning. Contrary to the studies in the literature, the images of the waveforms of the transient regions of the radios were used to train the ResNet50 and InceptionV3 networks. In other words, RF fingerprint extraction is combined with image processing. The results show that the ambient temperature significantly affects the performance of device classification based on RF fingerprints.

## 2. Radio Frequency Fingerprint

Physical layer authentication is one of the key technologies used to secure wireless communications. RF fingerprints [18], which are the results of the electrical properties of the components on the device hardware, contain features that are difficult to clone. Classification of devices with RF fingerprinting consists of the steps shown in Figure 1.

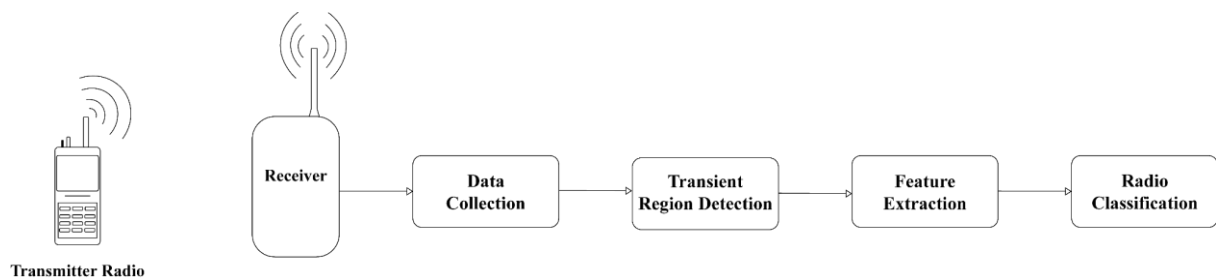


Figure 1 Device classification model with radio frequency fingerprint.

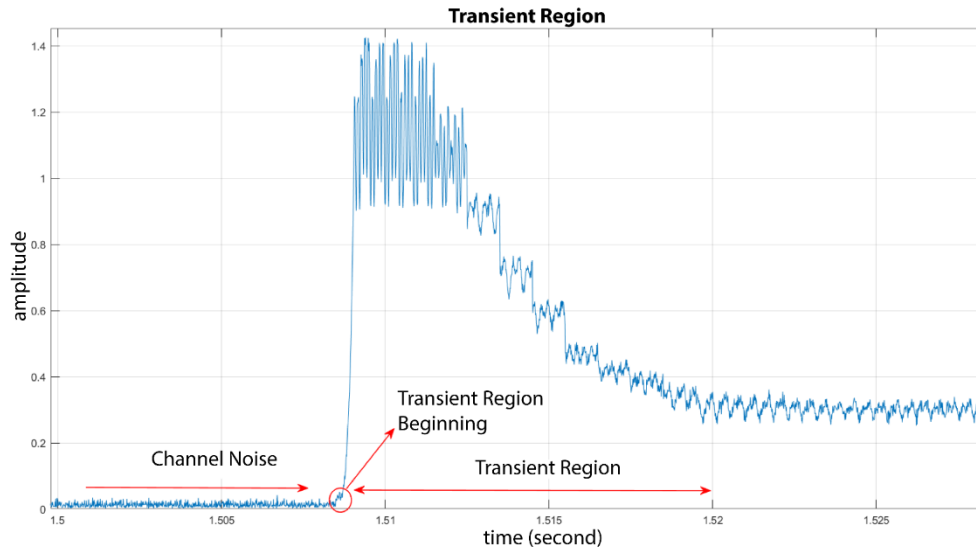


Figure 2 A sample transient region obtained during the study.

For this study, radios with the same brand, model, and production date operating in the UHF band are used as transmitter radios. In order to avoid the possible effects of aging on RF fingerprinting, care has been taken to ensure that the production dates are the same. A low-level commercial product, Adalm Pluto software defined radio (SDR) was used as the receiving device. Data collection was carried out with the help of the Adalm Pluto plugin of the Matlab software. Transient region detection, feature extraction, and device classification processes were also performed on Matlab.

## 2.1 Transient Region Detection

Detection of the transient region and fingerprinting from this region is the most important step in device identification with RF fingerprinting. An incorrectly detected transient region may adversely affect the fingerprinting step, and the classification process may therefore be inaccurate. For transient region detection, the features of the signal in the time domain are extracted. Bayesian step change, threshold detection, and double sliding window are the most widely used methods for transient region detection [19]. All three methods basically use the differences in the amplitudes of the signals in the noise region and transient region. The noisy region is the period where the channel is not carrying any data signals. During the transition from this region to the transient region, a sudden change occurs in the signal. In

For signals with gradual transitions, Bayesian step change and threshold detectors may be delayed to detect the beginning of the transient region. For this reason, their performance may suffer. Hence, the double sliding window method is preferred for the transient region detection.

## 2.2 The Double Sliding Window Method

The double sliding window method is a kind of rising edge detection algorithm that detects the energy increase in the incoming signal [20]. For this, the incoming signal is convolved with a two-window filter of a certain length. The signal energy covered under the right window is divided by the signal energy covered under the left window. At the starting point of the signal, the energy of the right window significantly exceeds the left window and an upward peak is formed in the running ratio. Likewise, when the double sliding window exits the signal, while the left window is still in the signal region, the right window switches to the noise section. Thus, the ratio of the energy of the signal corresponding to the right window to that of the left window peaks downward [19]. In Figure 3, the graph of the energy generated when the double sliding window moves over the signal while it passes into the transient region of the signal is demonstrated.

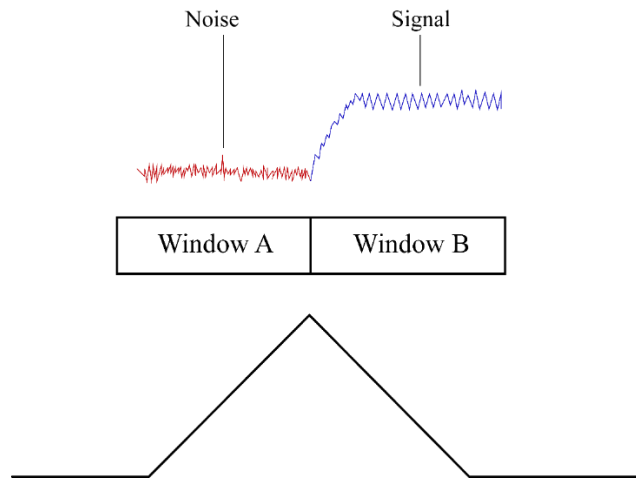


Figure 2 The ratio of the energy of the signal in window A to the energy of the signal in window B in the double sliding window method. The peak marks the beginning of the transient region.

Figure 4 shows the detection of the beginning of the transient region with the double sliding window method on a sample signal obtained in the study. After the beginning of the transient region is determined, its end must also be determined. In this study, the end of the transient region was chosen based on the observations, going forward a certain time from the starting point on the time axis.

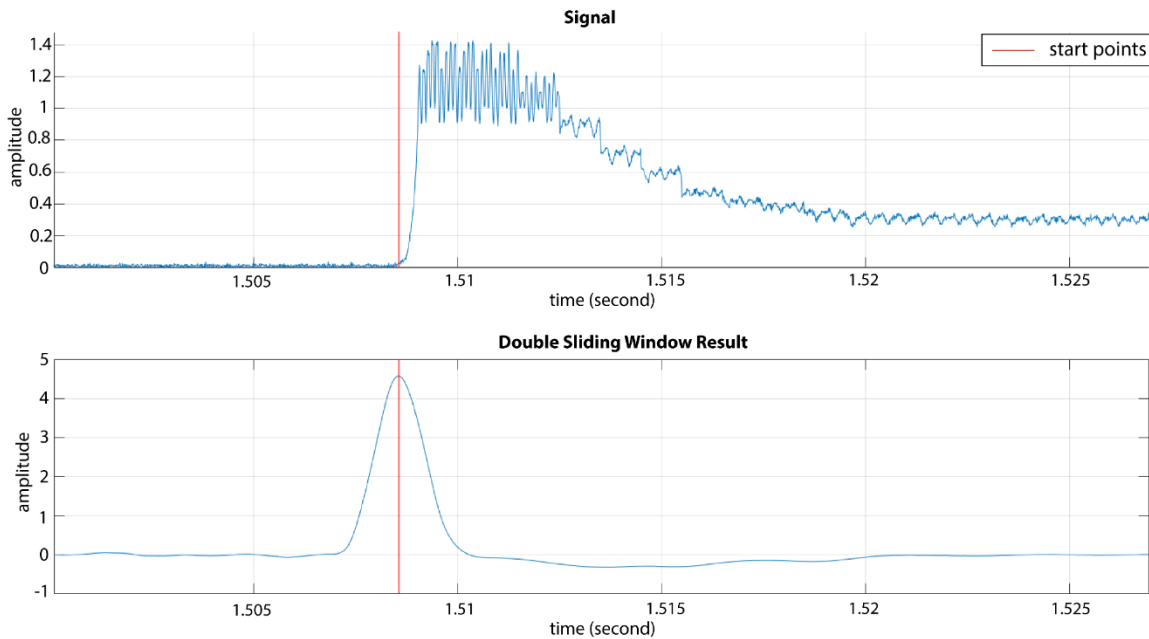


Figure 4 Detection of the transient region by the double sliding window method. The red bar marks the peak of the energy ratios in the bottom figure, and the beginning of the transient region in the top figure.

### 3. Methodology

The effect of ambient temperature on device classification with RF fingerprinting was investigated in a laboratory environment. For transient region detection, which is the most important step for RF fingerprinting, the double sliding window method is used. After the beginning of the transient region was determined, the end of the transient region was determined empirically, based on the experimental study. Unlike other studies, instead of using the I/Q data of the signals in the transient region, images of

the transient region in png format were used. 288.6 Gb data was collected in baseband (.bb) format with the help of Adalm Pluto SDR for 5 different radios of the same brand, model, and year of manufacturing at 4 temperature values. ResNet50 and inceptionV3 convolutional neural network models, two of the most successful models for image classification, in Matlab library were used for classification. A Weissstechnik brand temperature cabinet, which is an advanced ambient temperature test cabinet, was used to adjust the temperature of the environment. This cabinet constantly monitors the temperature inside, and ensures that the surface temperature of the radio device and the temperature of the inside of the cabinet are equal. After the devices were placed inside the cabinet, they were kept for 1 hour until the desired ambient temperature was achieved. The radio device under test is programmed to automatically transmit for 2 seconds via an option cable extended outside through the heat-proof slots at the entrance of the cabinet, followed by a 3-second silence, periodically.

In addition, an external thermocouple is mounted on the temperature pulse surface of the radio inside the cabinet. Thus, the temperature on the device was also monitored outside the cabinet's own thermometer. It was observed during the tests that the temperature on the surface of the devices increased up to a further 2 degrees beyond the temperature of the test cabinet. Adalm Pluto SDR is positioned as far away from the temperature test cabinet as possible so that the radio under test is not affected by the signals that may be reflected from its body as well as the antenna, and to avoid noise. The power outputs are programmed to be 1 Watt so that the radios do not overheat while transmitting. Furthermore, in order not to be affected by ambient noise, the air interface was scanned with an Aeroflex IFR device, and a frequency value (415.125 MHz) with minimal noise was selected. During the test process, batteries with high capacity were used to avoid possible current fluctuations that may occur near the end of the battery charge. The test setup is shown in Figure 5. The ResNet50 and inceptionV3 models were run on an NVIDIA GEFORCE 940MX graphics card.

#### 4. Numerical Results

5 radios of the same brand, model, and production year were used for the study. The radios were given identification labels as A, B, D, E, and X. Each of the radios were placed in the test cabinet at temperatures of  $-5$ ,  $10$ ,  $25$ , and  $40$  °C\*. Each radio made 2200 transmissions at each temperature value. Therefore, a total of  $5 \times 4 \times 2200 = 44000$  png images were obtained as the data set. For each radio and temperature pair, 2000 images out of the total 2200 images were used for training the ResNet50 model, and the remaining 200 images were used for testing the device classification. The total number of images taken from 5 radios at each temperature for the training of ResNet is  $5 \times 2000 = 10000$ .

Table 1 shows the ResNet50 performance of the classification by RF fingerprints from the baseband signals collected from these temperature values radios. We trained the model using the data from the four different temperature values, and tested for the four different temperature values. As can be seen in Table 1, the performance of the classification depends on the temperature.

---

\* Extreme increases were detected in the surface temperatures of the radios under test in the temperature cabinet, measured by thermocouple at temperatures of  $55$  °C, and hence, we decided not to collect data for  $55$  °C and above.

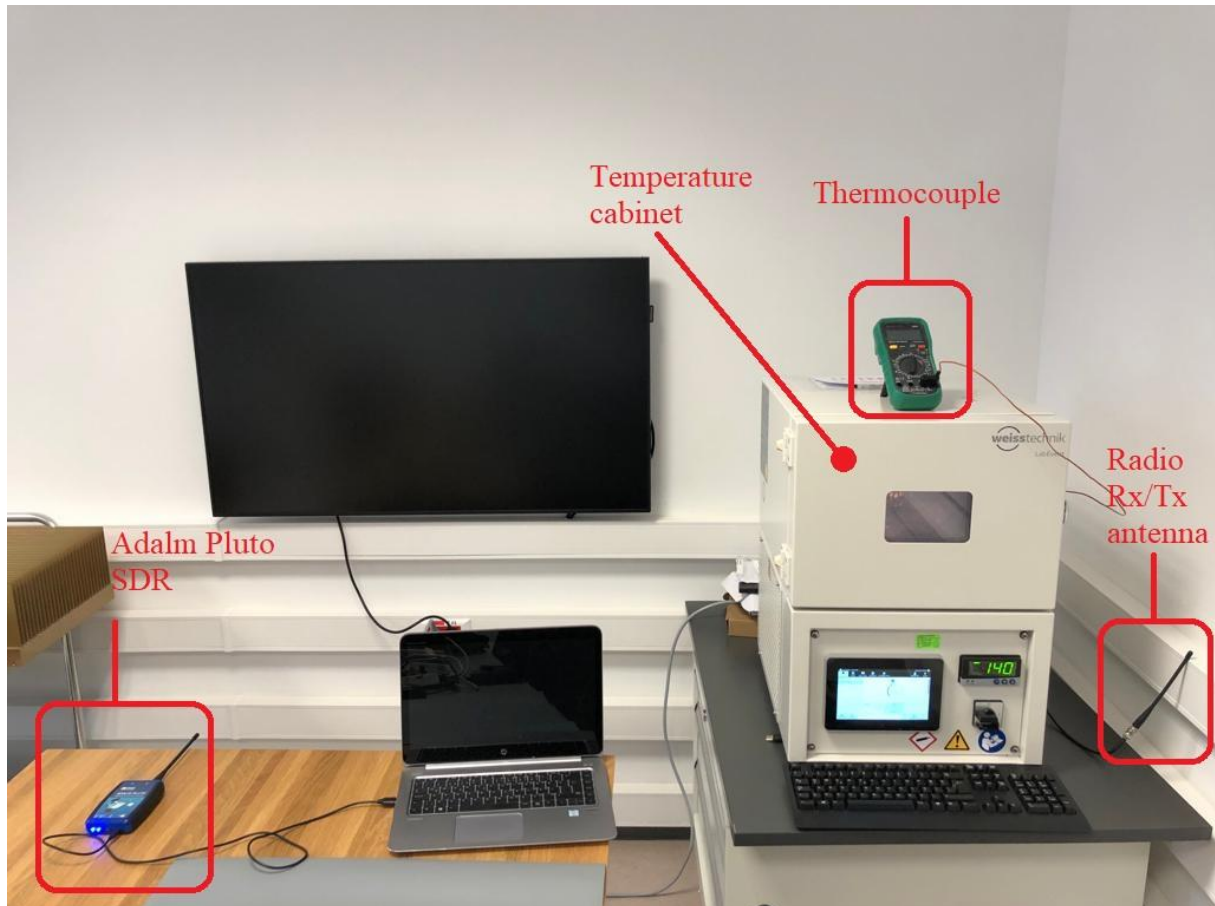


Figure 3 Data collection setup.

Table 1 ResNet50 classification performance of temperature dependent radios.

Training Temperature	Test Temperature			
	-5°C	10°C	25°C	40°C
-5°C	88.35%	42.04%	39.08%	43.04%
10°C	36.79%	91.02%	41.65%	21.47%
25°C	20.28%	51.73%	96.39%	32.94%
40°C	39.91%	19.9%	21.96%	94.7%

The best classification performance, 96.39% accuracy, was obtained with training and test data both collected at 25 °C. On the other hand, the classification performance among the cases where the training and the test data belong to the same temperature value was the lowest at -5 °C with an accuracy of 88.35%. While the performance is relatively high when training and test data come from the same temperature experiments, testing data from different temperatures than the training data was obtained seriously reduces the performance. The lowest performance was observed when the training data was taken at 40 °C and the test data at 10 °C.

Table 2 shows the InceptionV3 performance of the classification by RF fingerprints under the same scenario. As can be seen from Table 2, the best classification performance, 96.47% accuracy, was obtained with training and test data both collected at 40 °C, closely followed by the case where the training and the test data both collected at 25 °C. In the case of using test and training data at the same temperature, the lowest performance was obtained at -5 °C with 86.42%. Similar to ResNet50, the classification performance obtained with the training and the test data selected at the same temperature in InceptionV3 is higher than the classification performance made with the test and training data at different temperatures. The lowest performance was observed when the training data was taken at 40 °C and the test data at 25 °C.

Table 2 InceptionV3 classification performance of temperature dependent radios.

Training Temperature	Test Temperature			
	-5°C	10°C	25°C	40°C
-5°C	86.42%	44.90%	41.75%	44.22%
10°C	35.57%	89.29%	43.95%	20.59%
25°C	22.36%	52.24%	95.67%	21.82%
40°C	39.81%	24.90%	20.31%	96.47%

As can be clearly understood from Tables 1 and 2, the ambient temperature is an important factor in device classification with RF fingerprinting and cannot be ignored. Another inference that can be made here is that the temperature of the devices significantly contributes to the RF fingerprints. Increases and decreases in ambient temperature cause the components in the hardware layer of the device to exhibit different characteristics, thus leading to somewhat different RF fingerprints for the same device at different temperatures.

#### 4.1 Device Classification with Blended Training Model

In an effort to reduce the effect of ambient temperature on the performance, especially for the scenarios where the test case temperature differs from the temperature the training data was collected, the training data of the ResNet50 and InceptionV3 networks were selected for each radio in an equal amount from each temperature. Test data was given to these trained networks in an equal amount of data from each radio at every temperature, and the classification process was repeated. Classification performances are given in Tables 3 and 4 for ResNet50 and InceptionV3, respectively. As can be seen from the ResNet50 classification performance in Table 3, while the performances of the model for the test data collected at 10, 25, and 40 °C are decent, the performance at -5 °C turns out to be significantly lower.

As can be seen from the InceptionV3 classification performance in Table 4, the lowest performance was obtained at the test temperature of -5 °C, as in ResNet50. On the other hand, at test temperatures of -5, 10, and 25 °C, ResNet50 showed better performance than InceptionV3. At 40 °C, Inceptionv3 has a slightly better performance than ResNet50.

Table 3 ResNet50 classification performance of radios with training data including all temperatures.

Training Temperature	Test Temperature			
	-5°C	10°C	25°C	40°C
-5°, 10°, 25°, 40°C	52.08%			
-5°, 10°, 25°, 40°C		84.80%		
-5°, 10°, 25°, 40°C			88.35%	
-5°, 10°, 25°, 40°C				85.98%

Table 4 InceptionV3 classification performance of radios with training data including all temperatures.

Training Temperature	Test Temperature			
	-5°C	10°C	25°C	40°C
-5°, 10°, 25°, 40°C	44.62%			
-5°, 10°, 25°, 40°C		69.08%		
-5°, 10°, 25°, 40°C			71.96%	
-5°, 10°, 25°, 40°C				87.25%

#### 4.2 Normalized Confusion Matrices

Figure 6 shows the classification performances and errors in percentage of the 5 radios trained at -5 °C and classified with test data at -5 °C for ResNet50 and InceptionV3 models. The horizontal axis shows the estimated classes of the radios on the model outputs, and the vertical axis shows the correct classes. The diagonal elements of the matrix indicate the percentage of a radio being classified correctly, and the other elements indicate the percentages of misclassification. Similarly, Figures 7 – 9 show the confusion matrices for training and test at 10, 25, and 40 °C, respectively. Furthermore, Figures 10 – 13 show the confusion matrices for the blended training model for tests with -5, 10, 25, and 40 °C. One observation from the confusion matrices is that in some settings, one or two radio devices are difficult to correctly classify, whereas the other devices are classified with high accuracy. One such example is radio device X under the setting where ResNet50 is used on the training and the test data collected at -5 °C, seen in Figure 6.

The results demonstrate that while device classification based on RF fingerprinting using transient region images is viable, a difference in the temperature values that the training data was taken and the classification is executed significantly affects the accuracy. When the blended training data is used, the accuracy is improved compared to the settings where the temperature values for the training and the test data are different. In this case, ResNet50 performs better than InceptionV3. When the test is performed at -5 °C, both methods suffer significantly. Therefore, when operating at lower temperatures, better classifiers should be designed.

## 5 Conclusion and Future Work

In this study, we investigated the effect of the ambient temperature on the performance of radio device classification based on RF fingerprinting. The radio devices used in the study belong to the same brand, model, and production date, making the problem more difficult than classifying radio devices of different brands or models. In our study, we performed the classification of devices by processing the images of radio frequency fingerprints. In terms of this approach, we used a different method from other studies. Our results show that high levels of accuracy can be attained using convolutional neural network models such as ResNet50 and InceptionV3 when the test data and the training data are collected at the same temperature, whereas performance suffers when the test data and the training data belong to different temperature values. Hence, we conclude that the ambient temperature cannot be ignored in future studies in the field of RF fingerprinting. We have also shown that device classification via RF fingerprinting can be done using SDRs with lower cost, easier to use and open source applications, instead of using dedicated devices such as high-cost complex circuit receivers and oscilloscopes. On the other hand, no big difference in performance was observed in the classification made with ResNet50 and InceptionV3. It has also been observed that ResNet50 can classify more successfully at temperatures of  $-5$ ,  $10$ ,  $25$  °C in blended training model. In the same model, at  $40$  °C, InceptionV3 had slightly better accuracy. For this reason, a mixed CNN model can be used for different temperatures as well as a blended training model.

Future studies will focus on improving the classification performance under different temperature values. Possible approaches to be investigated include training multiple models for different temperatures and making a classification decision based on the majority of these models.

During the collection of RF fingerprint data in this study, the transmitting and receiving devices were kept stationary. Collecting RF fingerprint data in scenarios where the receiver and transmitter are mobile will be an important study to see the effects of channel distortions. The aging of RF fingerprinted devices will change fingerprints. Therefore, the effect of device aging requires further research. The effects of the output power of the transmitter devices on the RF fingerprint should also be investigated as an open issue.

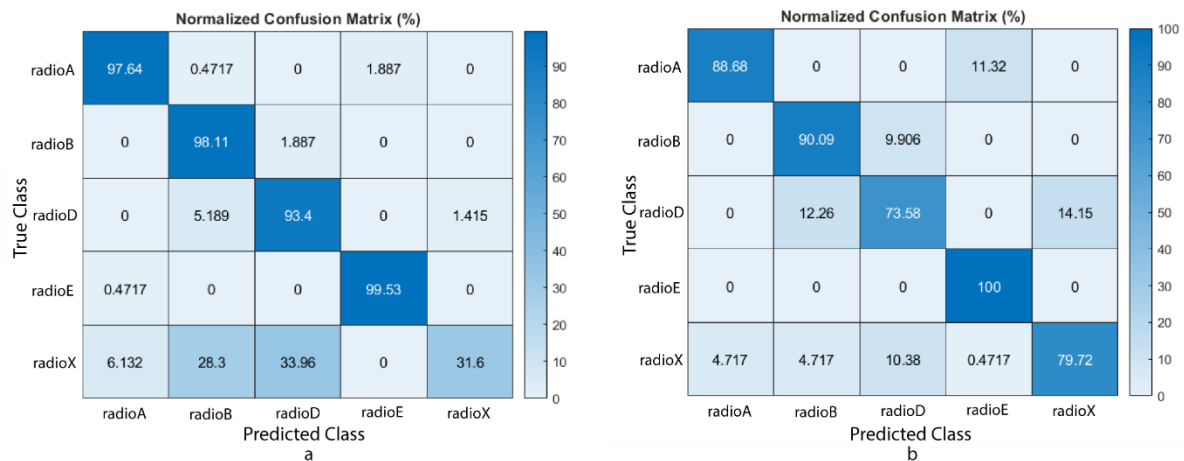


Figure 4 Normalized confusion matrices for: (a)  $-5$  °C training and test data for ResNet50, (b)  $-5$  °C training and test data for InceptionV3



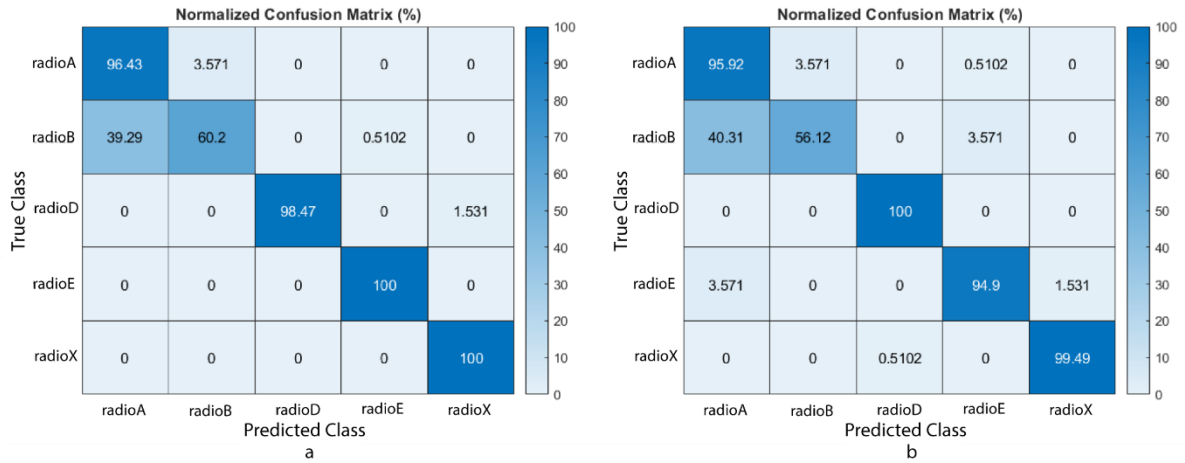


Figure 7 Normalized confusion matrices for: (a) 10 °C training and test data for ResNet50, (b) 10 °C training and test data for InceptionV3.

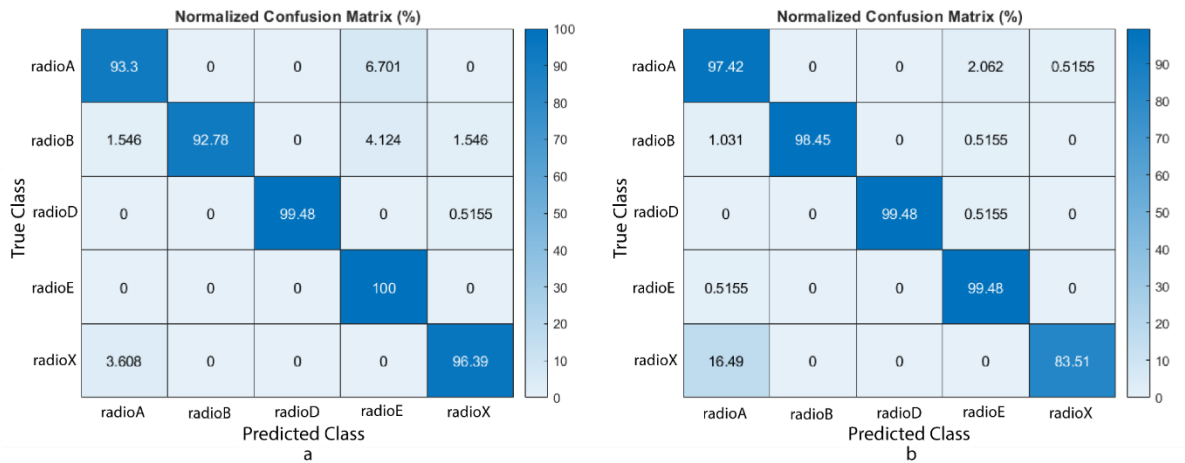


Figure 8 Normalized confusion matrices for: (a) 25 °C training and test data for ResNet50, (b) 25 °C training and test data for InceptionV3.

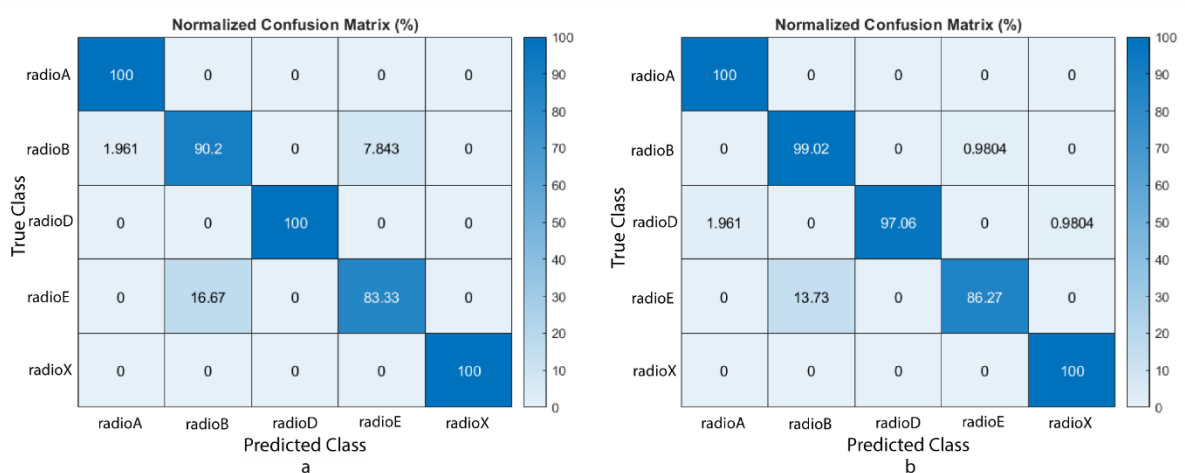


Figure 9 Normalized confusion matrices for: (a) 40 °C training and test data for ResNet50, (b) 40 °C training and test data for InceptionV3.

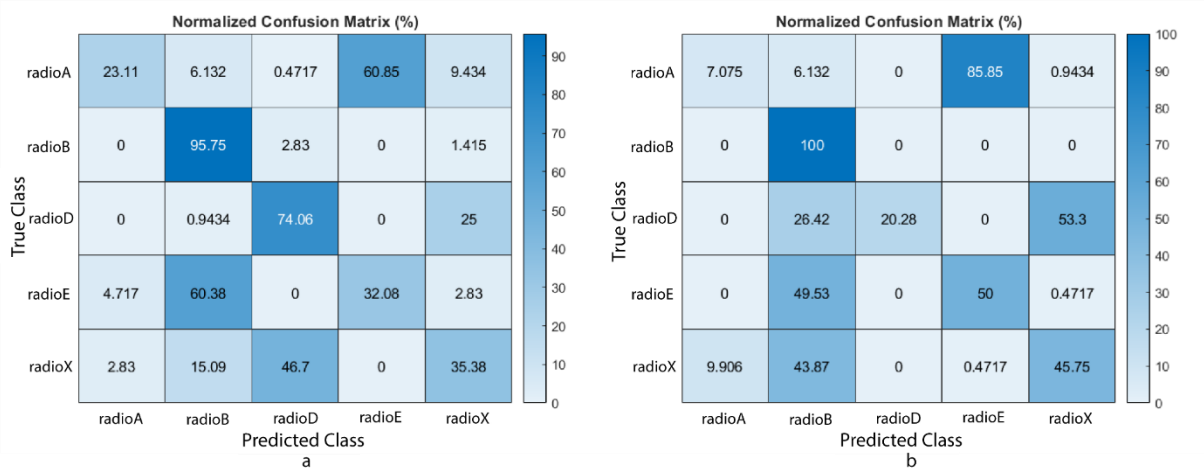


Figure 10 Normalized confusion matrices for  $-5, 10, 25,$  and  $40\text{ }^{\circ}\text{C}$  blended training data and (a)  $-5\text{ }^{\circ}\text{C}$  test data for ResNet50, (b)  $-5\text{ }^{\circ}\text{C}$  test data for InceptionV3.

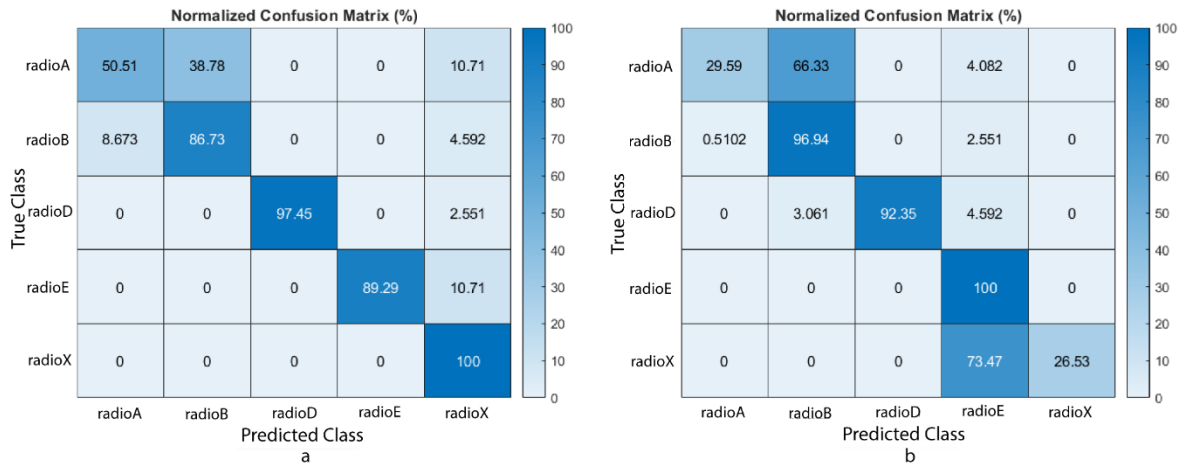


Figure 11 Normalized confusion matrices for  $-5, 10, 25,$  and  $40\text{ }^{\circ}\text{C}$  blended training data and (a)  $10\text{ }^{\circ}\text{C}$  test data for ResNet50, (b)  $10\text{ }^{\circ}\text{C}$  test data for InceptionV3.

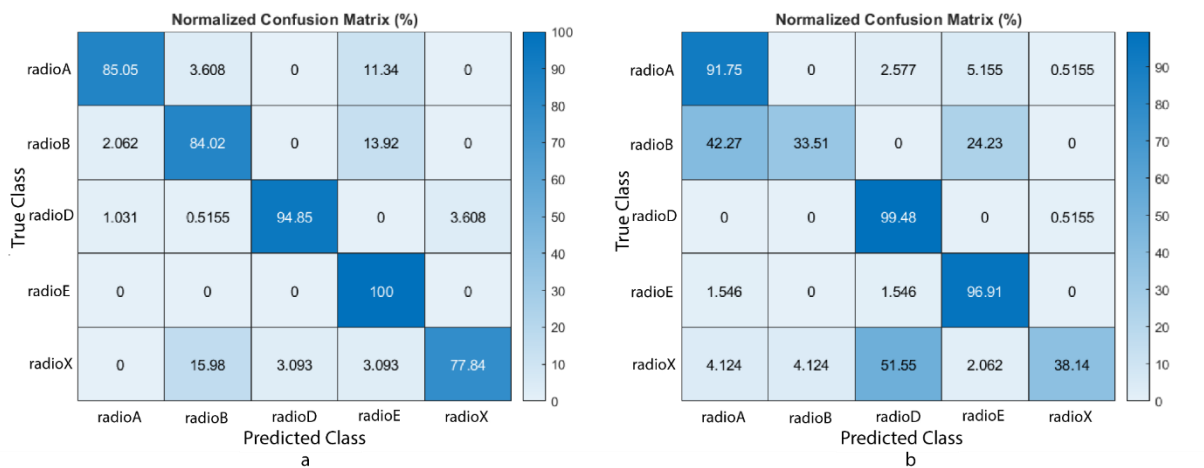


Figure 12 Normalized confusion matrices for  $-5, 10, 25,$  and  $40\text{ }^{\circ}\text{C}$  blended training data and (a)  $25\text{ }^{\circ}\text{C}$  test data for ResNet50, (b)  $25\text{ }^{\circ}\text{C}$  test data for InceptionV3.

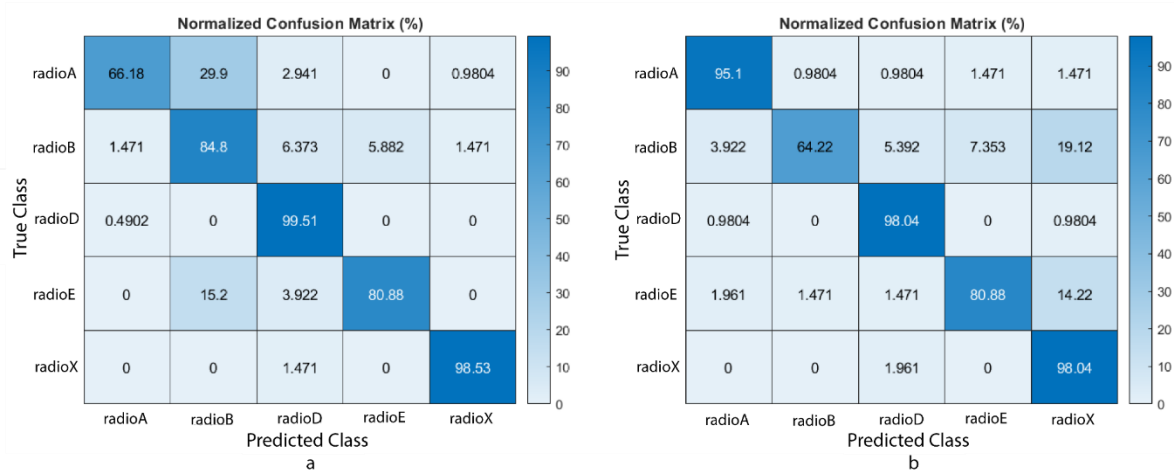


Figure 13 Normalized confusion matrices for  $-5$ ,  $10$ ,  $25$ , and  $40$  °C blended training data and (a)  $40$  °C test data for ResNet50, (b)  $40$  °C test data for InceptionV3.

## References

- [1] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27-33, 2007.
- [2] D. R. Reising, M. A. Temple and M. J. Mendenhall, "Improving intra-cellular security using air monitoring with RF fingerprints," in *2010 IEEE Wireless Communication and Networking Conference*, Sydney, NSW, Australia, 2010.
- [3] A. C. Polak, S. Dolatshahi and D. L. Goeckel, "Identifying wireless users via transmitter imperfections," *IEEE Journal on selected areas in communications*, vol. 29, no. 7, pp. 1469-1479, 2011.
- [4] S. Mathur, A. Reznik, C. Ye, R. Mukherjee, A. Rahman, Y. Shah, W. Trappe and N. Mandayam, "Exploiting the physical layer for enhanced security [security and privacy in emerging wireless networks]," *IEEE Wireless Communications*, vol. 17, no. 5, pp. 63-70, 2010.
- [5] B. Danev, H. Luecken, S. Capkun and K. El Defrawy, "Attacks on physical-layer identification," in *Proceedings of the third ACM conference on Wireless network security*, Hoboken, New Jersey, USA, 2010.
- [6] K. Merchant, S. Revay, G. Stantchev and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160-167, 2018.
- [7] I. O. Kennedy and A. M. Kuzminskiy, "RF fingerprint detection in a wireless multipath channel," in *7th International Symposium on Wireless Communication Systems*, York, UK, 2010.
- [8] O. Ureten and N. Serinken, "Bayesian detection of Wi-Fi transmitter RF fingerprints," *Electronics Letters*, vol. 41, no. 6, pp. 373-374, 2005.
- [9] J. Toonstra and W. Kinsner, "A radio transmitter fingerprinting system ODO-1," in *Proceedings of 1996 Canadian Conference on Electrical and Computer Engineering*, Calgary, AB, Canada, 1996.
- [10] M. Woelfle, M. Temple, M. Mullins and M. Mendenhall, "Detecting, identifying and locating bluetooth devices using RF fingerprints," in *2009 Military Communications Conference (MILCOM 2009)*, Boston, MA, USA, 2009.
- [11] D. Zanetti, B. Danev and S. Capkun, "Physical-layer identification of UHF RFID tags," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, Chicago, IL, USA, 2010.

- [12] S. U. Rehman, K. W. Sowerby and C. Coghill, "Analysis of impersonation attacks on systems using RF fingerprinting and low-end receivers," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 591-601, 2014.
- [13] O. Tekbas, N. Serinken and O. Ureten, "An experimental performance evaluation of a novel radio-transmitter identification system under diverse environmental conditions," *Canadian Journal of Electrical and Computer Engineering*, vol. 29, no. 3, pp. 203-209, 2004.
- [14] S. Riyaz, K. Sankhe, S. Ioannidis and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146-152, 2018.
- [15] S. U. Rehman, K. W. Sowerby, S. Alam and I. Ardekani, "Radio frequency fingerprinting and its challenges," in *2014 IEEE Conference on Communications and Network Security*, San Francisco, CA, USA, 2014.
- [16] S. Wang, H. Jiang, X. Fang, Y. Ying, J. Li and B. Zhang, "Radio frequency fingerprint identification based on deep complex residual network," *IEEE Access*, vol. 8, pp. 204417-204424, 2020.
- [17] W. C. Suski II, M. A. Temple, M. J. Mendenhall and R. F. Mills, "Radio frequency fingerprinting commercial communication devices to enhance electronic security," *International Journal of Electronic Security and Digital Forensics*, vol. 1, no. 3, pp. 301-322, 2008.
- [18] N. Soltanieh, Y. Norouzi, Y. Yang and N. C. Karmakar, "A review of radio frequency fingerprinting techniques," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, pp. 222-233, 2020.
- [19] D. Shaw and W. Kinsner, "Multifractal modelling of radio transmitter transients for classification," in *IEEE WESCANEX 97 Communications, Power and Computing*, Winnipeg, MB, Canada, 1997.
- [20] J. Terry and J. Heiskala, *OFDM wireless LANs: A theoretical and practical guide*, Indianapolis, Indiana, USA: Sams publishing, 2002.
- [21] J. Li, D. Bi, Y. Ying, K. Wei and B. Zhang, "An improved algorithm for extracting subtle features of radiation source individual signals," *Electronics*, vol. 8, no. 2, p. 246, 2019.

# A Software Defined Networking-based Routing Algorithm for Flying Ad Hoc Networks

 Berat Erdemkılıç<sup>1</sup>,  Mehmet Akif Yazıcı<sup>2</sup>

<sup>1</sup>Istanbul Technical University, Electronics and Communication Engineering Department;  
berdemkilig@itu.edu.tr;

<sup>2</sup> Corresponding Author; Istanbul Technical University, Informatics Institute, Information and Communications Research Group, Istanbul, Türkiye; yazicima@itu.edu.tr; Tel: +90 212 285 71 94

Received 24 July 2022; Accepted 7 August 2022; Published online 31 August 2022

## Abstract

Flying Ad-Hoc Networks (FANET) are wireless mobile ad-hoc networks composed of unmanned aerial vehicles (UAV) as communicating nodes. As with any computer network, routing is an essential problem that has to be solved efficiently for high performance. FANETs present unique challenges with respect to routing, due to their structures. FANET systems have high dynamicity as the nodes move at very high speeds and UAVs can behave in accordance with various mobility models. The nodes usually have line of sight between them, but FANET systems frequently operate on large topologies with low node density. Hence, the structure of the topology changes rapidly, and the frequency of link disconnections between UAVs increases. Traditional topology-based and position-based routing algorithms do not work well in the face of this problem. In this study, we propose a novel SDN-based Routing Protocol which comprises both proactive and reactive components in order to improve the performance. Software Defined Networking technology is used as the network management architecture. To investigate the performance of the proposed protocol against legacy MANET routing protocols, a comparison study was conducted in terms of throughput, end-to-end delay, and control packet overhead. Simulation results show that the proposed SDN-based Routing Protocol performs better than the selected legacy protocols.

**Keywords:** unmanned aerial vehicle, flying ad-hoc network, software defined networking, openflow protocol, routing algorithm

## 1. Introduction

Ad hoc networks are networks without dedicated infrastructure, often formed temporarily for certain purposes. Such networks are usually wireless, and the hosts may be mobile, in which case the network is named a mobile ad hoc network (MANET). MANETs consisting of vehicles are called vehicular ad hoc networks (VANET), which are studied as a special case due to their certain properties and use cases. Similarly, flying ad hoc networks (FANET) are special types of VANET/MANETs that are made up of unmanned aerial vehicles (UAV). FANETs are used mainly in search and rescue operations, traffic and surveillance applications, military and law enforcement, and natural disaster scenarios. A typical FANET consists of UAVs moving with very high speeds (compared to earth-bound vehicles), in a variety of mobility patterns leading to highly dynamic network topologies, which is usually detrimental to routing performance. On the other hand, FANET nodes usually communicate via line-of-sight transmissions as they do not see many obstacles in the air matching their speeds [1].

The major difficulty regarding routing in MANETs is the dynamicity of the network topology. Routes become obsolete much more frequently in MANETs compared to wireline or infrastructure networks. Therefore, there exist specialized routing algorithms for MANETs [2]. In FANETs, this is an even more profound problem as nodes move much faster and use larger areas, which lead to highly dynamic but sparse networks, making routing more difficult. Depending on the use case [3], UAV nodes in a FANET might move in predetermined paths, that can be modeled using the semi-random circular movement model [4] or the paparazzi model [5]; or they might move randomly, which are usually represented by stochastic mobility models such as the random waypoint model [6].

In general, routing protocols for MANETs can be topology-based or position-based. Topology-based routing protocols can be:

- (i) static, where all routes to destinations are predefined and do not change,
- (ii) proactive, where routes to all possible destinations are discovered before any need for transmission and refreshed periodically,
- (iii) reactive, where routes are only discovered for a destination when a transmission is to be made to that particular destination, and
- (iv) hierarchical, where nodes use a proactive protocol for a subset (usually a neighbourhood) of the topology and use a reactive protocol for the rest.

On the other hand, position-based routing protocols generally rely on position services such as the global positioning service (GPS).

In this study, we propose a novel routing algorithm for FANETs. The proposed algorithm is based on the software defined networking (SDN) paradigm and exploits central SDN controller(s). We provide simulation-based comparisons with existing and widely known algorithms from the literature to demonstrate the performance of the proposed algorithm. In the sequel, we give an overview of existing routing algorithms for MANETs and FANETs in the second section. We describe the proposed algorithm in the third section, whereas we present simulation results for the performance evaluation in the fourth section. Finally, we conclude in the fifth section.

## 2. An Overview of Routing Algorithms for MANETs and FANETs

Routing is one of the essential problems in MANETs, VANETs, and FANETs. As FANETs are special cases of MANETs, many FANET routing protocols are based on MANET routing protocols. A comprehensive survey of routing protocols for UAV networks is given in [7]. Static/deterministic protocols are useful if the flight path and the formation of the FANET is known beforehand and no change is expected. However, overwhelmingly, FANETs require dynamic routing protocols in most use cases. FANET routing protocols can be roughly classified into topology-based and position-based protocols, although there are stochastic routing protocols based on estimation [8] and node movement [9], and cluster-based approaches such as mobile infrastructure based VANET routing [10], modularity-based dynamic clustering [11], and the hybrid self-organized clustering scheme [12]. Based on the use cases, there are routing protocols supporting multicast [13] [14], which we do not consider in this study.

Among the topology-based routing protocols, proactive algorithms such as the destination-sequenced distance-vector routing (DSDV) [15], the optimized link state routing protocol (OLSR) [16], the better approach to mobile ad hoc networking (BATMAN) protocol [17]; reactive algorithms such as the ad-hoc on-demand distance vector routing (AODV) [18] and dynamic source routing (DSR) [19]; and hierarchical protocols such as the zone routing protocol (ZRP) [20] and the temporally-ordered routing algorithm (TORA) [21] can be mentioned. Proactive routing protocols keep the topology information and the routing tables at the nodes fresh by updating them periodically, whereas the reactive routing protocols discover routes when necessary. Hierarchical protocols define clusters or neighbourhoods inside which the protocol act proactive, maintaining routes, whereas to the outside of the cluster, the protocol behaves as a reactive protocol. The advantage of the proactive routing protocols is low latency but they suffer from high overhead and reduced throughput. On the other hand, reactive routing protocols avoid high control overheads at the cost of considerably larger latencies. Hierarchical protocols are a hybrid of both approaches, trying to combine the best aspects of both worlds. The effectiveness of such hybrid protocols highly depend on their definition of the clusters. These algorithms are better suited for systems such as sensor networks where communication between nodes in close proximity to each other is much more frequent than the communication between nodes that are further apart.

Position-based routing algorithms use geographical location information obtained through positioning methods such as GPS, which provides position information every second within 10 to 15 m accuracy. This can be made more accurate via assisted GPS (AGPS) or differential GPS (DGPS), and more speedy using an inertial measurement unit (IMU) [22].

Location-aware routing for delay-tolerant networks (LAROD) [23] is a position-based routing algorithm that is equipped with a location service that keeps a local database of node locations, which is updated using broadcast gossip and routing overhearing. LAROD uses partial knowledge of geographic position to decrease latency and overhead. The greedy perimeter stateless routing (GPSR) algorithm [24] makes greedy forwarding decisions using only information about a router's immediate neighbors based on their geographical positions. The decisions are greedy in the sense that at every forwarding, the packet is sent to a node that is closer to the destination. When that is not possible, GPSR routes the packet around the perimeter, and goes back to its greedy behaviour when possible. GPSR performs well in terms of routing speed and scale. An algorithm based on GPSR was proposed in [25], which works using adaptive (as opposed to periodic) beacons to reduce overhead and position prediction. The Adaptive Density-based Routing Protocol (ADRP) is described in [26]. ADRP adjusts the transmission probability of nodes, giving preference to those with fewer neighbours to increase the probability that their packets get through. Ad-hoc Routing Protocol for Aeronautical Mobile Ad hoc Networks (ARPAM) [13] is a protocol rooted in AODV that also employs geographical location information as well as proactive functions in specific circumstances. The position aware, secure, and efficient mesh routing (PASER) [27] is a position-based routing algorithm that puts special emphasis on security.

A typical FANET is considerably more dynamic than a typical MANET, both in node speeds and coverage area. FANETs may also have less node density, giving rise to packet loss issues. Therefore, legacy MANET routing protocols such as AODV, DSR, OLSR etc. may not always perform well in a FANET setting. Protocols designed specifically for FANETs are expected to perform better, but this also depends on the scenario. On the other hand, comparing specially designed protocols for FANETs is not straight-forward, as most of these protocols are not readily-available in major wireless network simulators.

### 3. The SDN-based Routing Algorithm for FANETs

The proposed algorithm is a position-based routing protocol that runs at the SDN controller. It has both reactive and proactive components, thus mitigating the disadvantages of both kinds of protocols. The SDN controller keeps track of the positions of each node, and makes routing decisions accordingly. For the communication between the SDN controller and the nodes, OpenFlow protocol [28] is used. Accordingly, the OpenFlow packet types listed in Table 1 are adapted and used for the purposes of the proposed protocol.

Table 1 Adapted OpenFlow packets for the implementation of the proposed algorithm

Packet Type	Packet Purpose
Hello	Packet sent from the nodes to the controller during initialization
Configuration	Configuration updates for the nodes
Echo	Request/reply packet pair to maintain connectivity of the nodes to the controller
Barrier	Ensuring message order and dependencies
Error	Notifying the controller of any problems at the nodes
Role-Request	Assigning roles to a node by the controller, particularly when there are multiple controllers a node can connect to
Packet-in	Transferring the control of a packet to the controller, when there is a miss in the rule table of the node, or the rule requires forwarding the packet to the controller
Packet-out	Forwarding packets that are received via Packet-in messages, along with the associated rules

The controller executes the following services:

- *Data layer unit configuration service*: This service consists of initializing the nodes in the FANET, as well as updating the timer period value at the nodes.
- *Topology mapping service*: Nodes in the FANET share their location information periodically every second with the SDN controller, and the controller builds an internal map of the topology of the FANET on which the routing decisions are made.

- *Table control service*: The routing tables are built based on the location data obtained from the nodes. Then, any modifications are communicated to the relevant nodes. Instead of the entire tables, only the changes are sent in order to minimize the protocol overhead.
- *Proactive routing service*: Routes are found and refreshed periodically based on the topology map built by the topology mapping service. Dijkstra's shortest path algorithm is used to find the minimum hop route.
- *Reactive routing service*: In case of asynchronous route request arrival due to an absence of a valid route to a destination in the routing table, this service is used along with the topology mapping service to provide up-to-date routing information to the requester node. Similar to the proactive routing service, the reactive routing service uses Dijkstra's shortest path algorithm.
- *Timer management service*: The timer for the proactive routing service is initiated with a period of 1 second. Afterwards, the dynamicity of the network is determined based on the location data received from the nodes. As a result, the period of the timer is either decreased or increased. Then, the updated period value is broadcast to the nodes via the data layer unit configuration service.

A general flow diagram summarizing the behaviour of the proposed algorithm is given in Figure 1. After the initial configuration of the nodes, the topology mapping service builds a map at the controller, who builds the routing tables both periodically in a proactive manner, and reactively when the routing table of a node does not have an entry for an incoming packet with a certain destination. The period of the timer is adaptively controlled by the timer management service. The routing tables produced at every proactive cycle are compared to the previous table, as indicated as "Service 4" under the topology mapping service in Figure 1. If the table remains the same three consecutive times, it is concluded that the dynamicity of the FANET does not require the current frequency of refreshing, and hence the period of the timer is reduced by 200 ms. Similarly, if the table changes three consecutive times, it is concluded that the dynamicity of the FANET requires more frequent route updates, and thus the period of the timer is increased by 200 ms.



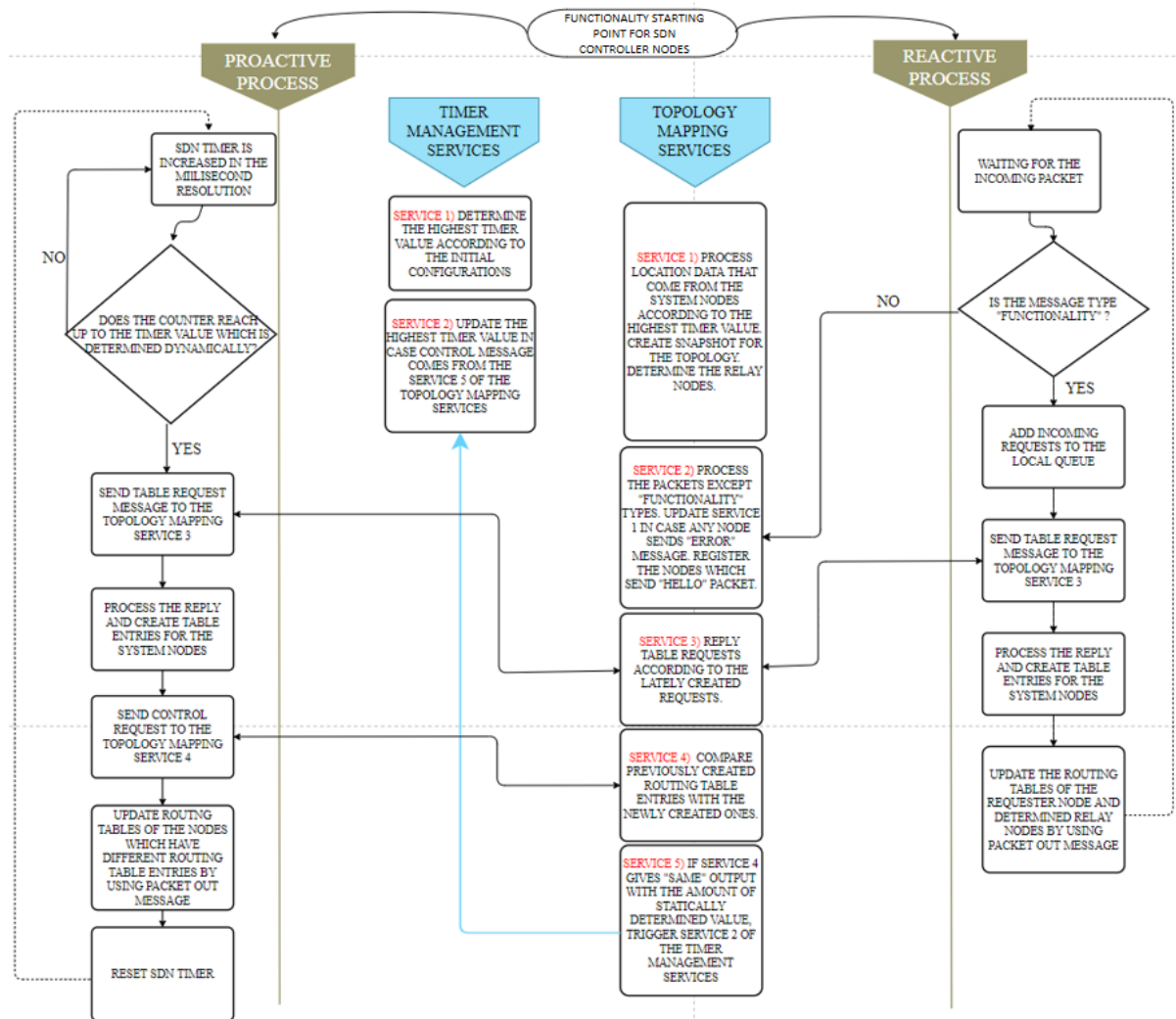


Figure 1 The flow diagram of the proposed routing algorithm.

#### 4. Performance Comparison

The performance of the proposed algorithm is compared with three well-known routing algorithms for MANETs, namely AODV, a reactive topology-based protocol, DSDV, a proactive topology-based protocol, and GPSR, a position-based protocol, with respect to the throughput achieved, end-to-end latency, and control messaging overhead. As stated in section 2, more specialized protocols for FANETs are mostly unavailable in wireless network simulators such as OMNeT++ [29]. Therefore, comparison of the proposed algorithm with such protocols is left as future work. We provide a comparison with three representative protocols from the main families of routing protocols for MANETs to indicate a general performance level. For this purpose, a simulation study has been conducted on OMNeT++. AODV, DSDV, and GPSR have already been implemented on OMNeT++, whereas we implemented the proposed algorithm by defining and/or modifying the necessary packet types and writing the functionality for the services as defined in section 3.

A scenario with a FANET consisting of 8 user nodes and 2 SDN controllers is simulated on a geographical area of 3.2 km by 2.2 km. 4 of the nodes are assigned as two source-destination pairs. 1500-byte UDP packets are generated at each source according to independent Poisson processes with a rate of 1 packet per second. All nodes move according to the random waypoint model with 200 ms pause time between direction changes. A snapshot of a sample topology under the scenario is presented in Figure 2. In four different experiments conducted for each of the protocols, the nodes are moved with constant speeds of 5, 10, 15, and 20 m/s. The simulation parameters are summarized in Table 2. The

four protocols are compared with respect to the throughput obtained, end-to-end latency, and the total protocol overhead due to the control messaging.



Figure 2 The snapshot of a sample topology under the simulation scenario.

Table 2 The summary of the simulation parametres and the environment.

Parametre/Property	Value
Topology model	Unit Disk Radio Medium
Detection range (m)	Uniformly distributed in [995, 1005]
Communication range (m)	1000
Interference range (m)	1000
Link layer protocol	IEEE 802.11ah
Network layer protocol	IPv4
Transport layer protocol	UDP
Application layer packet size (B)	1500
Packet generation model	Poisson arrival process (mean rate: 1 packet/s)
Node mobility model	Random waypoint model (200 ms pause time)
Number of nodes	8 terminal nodes and 2 SDN controller units
Node speeds (m/s)	5, 10, 15, 20
Simulation iterations (per speed value)	10
Total simulation time (hours)	5
Topology area (m <sup>2</sup> )	3200 × 2200

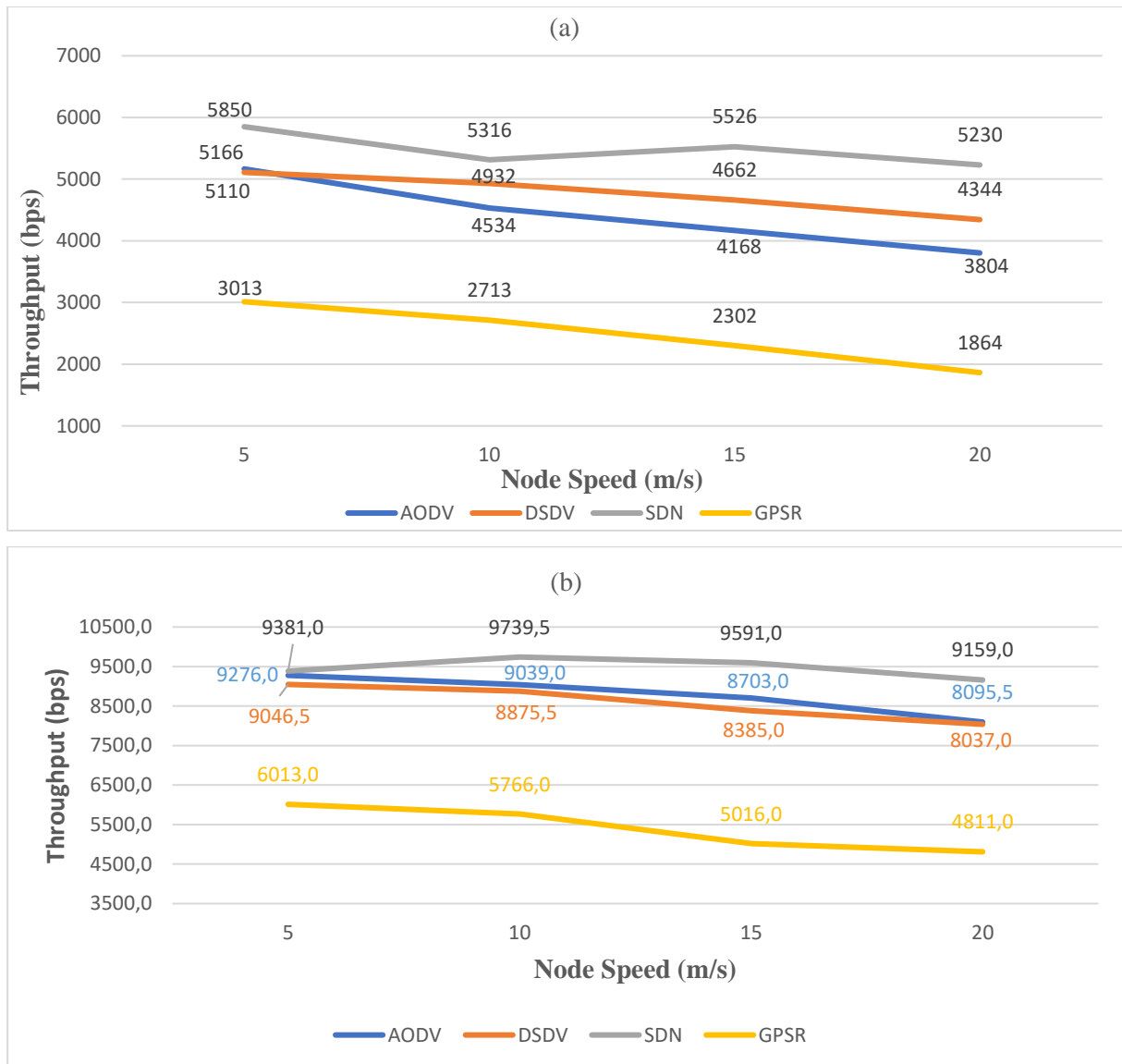


Figure 3 Throughputs achieved by the protocols for (a) Source1 - Destination1 pair, (b) Source2 – Destination2 pair, under the described scenario

The throughputs achieved by the protocols for each node speed value is given in Figure 3. For both source-destination pairs, it is observed that the proposed protocol performs the best, while AODV and DSDV are not far behind. Another important observation is that the throughput obtained with the proposed protocol is more or less maintained, whereas the other protocols, especially GPSR has a decreasing trend with increasing node speeds.

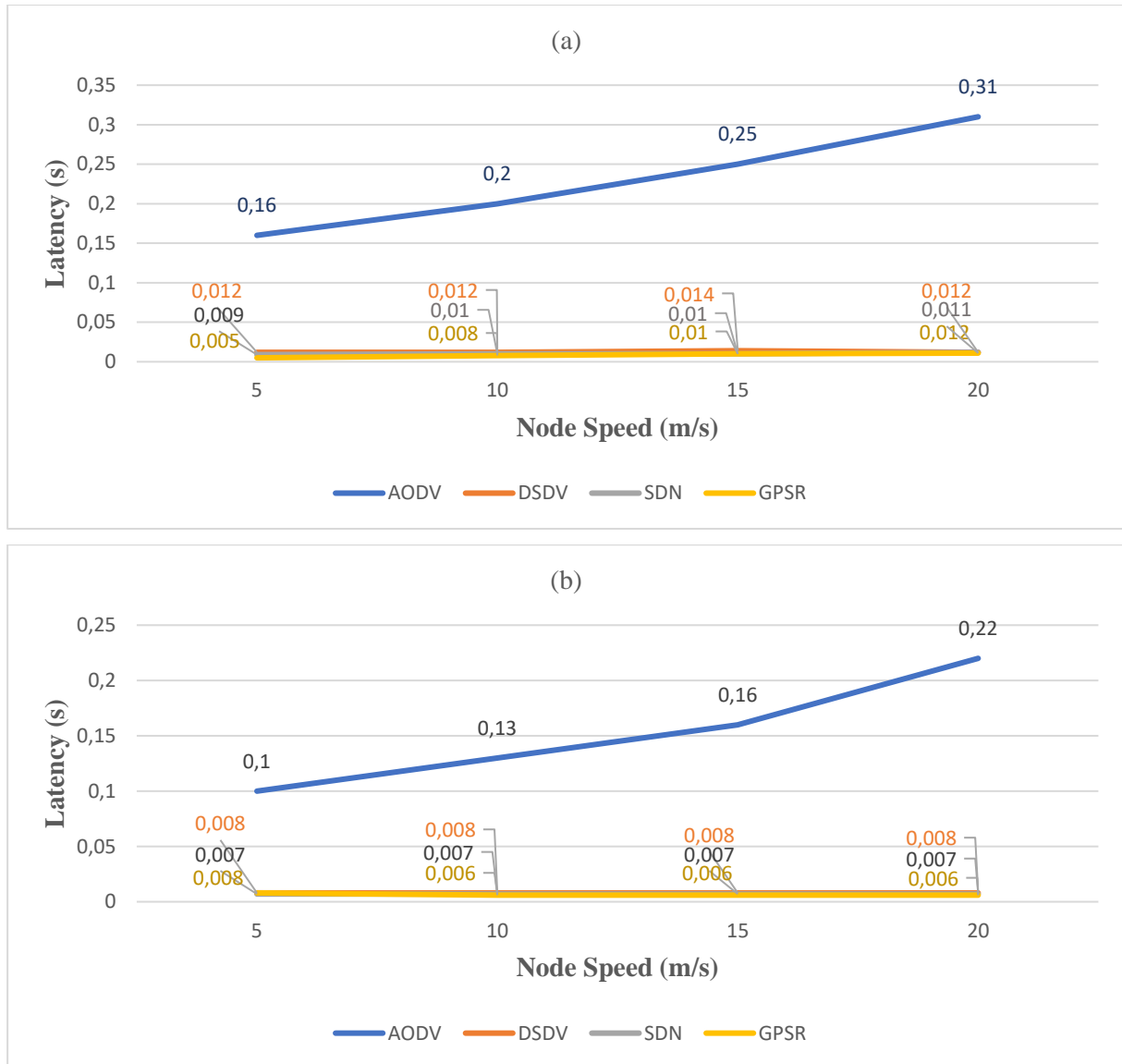


Figure 4 Average end-to-end latency values experienced by the packets for each of the protocols for (a) Source1 – Destination1 pair, (b) Source2 – Destination2 pair, under the described scenario.

The average end-to-end latency values experienced by the packets for each of the protocols is given in Figure 4. The proposed algorithm achieves latency values no more than 11 ms and this seems to be insensitive to the topology dynamicity (in terms of node speeds). The proactive component of the proposed algorithm makes sure that the routes are fresh. DSDV and GPSR behave similarly, whereas AODV, being a pure reactive protocol, leads to delays that are an order of magnitude larger and that increase with increasing node speeds. This demonstrates that any protocol used for especially highly dynamic FANETs must have a proactive component in order to keep the latency under a certain value.

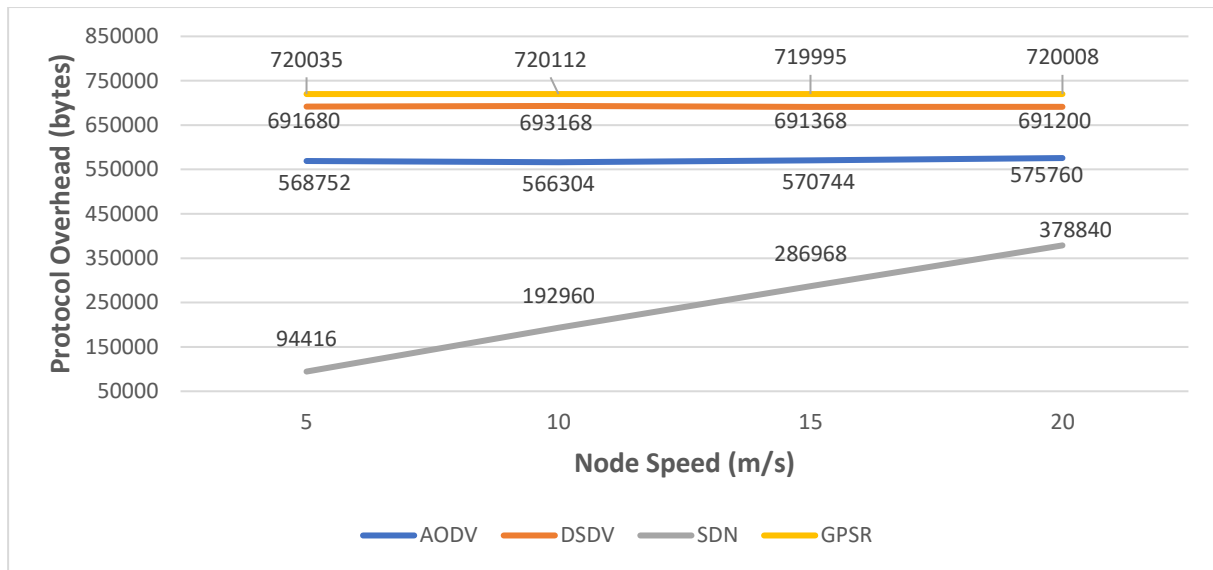


Figure 5 The total protocol overhead in bytes for each of the protocols.

The total protocol overhead for each of the protocols due to control messaging is presented in Figure 5. The proposed algorithm outperforms the others, especially when the dynamism of the network is relatively low. As the proposed algorithm takes precautions against unnecessary overhead, such as the table control service checking for changes in the routing tables before broadcasting the fresh routes to the nodes, it achieves the best performance in this regard among the investigated algorithms. Although the amount of protocol overhead for the proposed algorithm is increasing with the network dynamism as opposed to the other three protocols, it is still well below the others even at the speed of 20 m/s, which is very close to the optimal speed of a drone in a delivery scenario described in [30]. It requires further research to determine at what speed the proposed algorithm ceases to be the best in terms of protocol overhead. However, taking the Figures 3 and 4 into account, it could be expected that at such speeds, the proposed algorithm would offer better throughput and/or latency performance.

## 5. Conclusion and Future Work

In this paper, we proposed an SDN-based position-aware routing protocol for FANETs. The algorithm has proactive and reactive components, and merges the low latency advantage of proactive routing protocols with the low control overhead advantage of reactive routing protocols. Being SDN-based, the algorithm runs on the SDN controllers, which keep track of the geographical topology via periodic position updates and informs the nodes of any changes in the routes. The period of the position updates is dynamically adapted in order to keep up with highly dynamic scenarios, while keeping control overhead low for less dynamic systems.

The simulation results show that the proposed algorithm outperforms AODV, DSDV, and GPSR in terms of throughput, latency, and control overhead. Although the control overhead for the proposed algorithm is below that of the others for all of the investigated scenarios, it increases with the node speeds, owing to the fact that routes become obsolete more frequently in such scenarios. This means that there exists a speed value above which the control overhead will exceed that of AODV, DSDV, and GPSR. However, at such high speeds, these three protocols will suffer in terms of throughput and latency. As future work, scenarios with higher speeds can be investigated to determine how these protocols perform. A composite metric that is a function of the three figures investigated, namely the throughput, the latency, and the control overhead, can be defined to measure the overall performance. Another direction that can be further investigated is scenarios with higher node density.

Finally, a comparison with more position-based routing protocols that have been specifically designed for FANETs should be executed to demonstrate the effectiveness of the proposed algorithm. However,

this requires a considerable amount of effort as these protocols are not available as predefined functionality in simulation tools such as OMNeT++.

## References

- [1] İ. Bekmezci, O. K. Sahingoz and Ş. Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254-1270, May 2013.
- [2] S. Corson and J. Macker, *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, RFC 2501, DOI 10.17487/RFC2501, <https://www.rfc-editor.org/info/rfc2501>, January 1999.
- [3] A. Bujari, C. E. Palazzi and D. Ronzani, "FANET application scenarios and mobility models," in *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, Niagara Falls, NY, USA, 2017.
- [4] W. Wang, X. Guan, B. Wang and Y. Wang, "A novel mobility model based on semi-random circular movement in mobile ad hoc networks," *Information Sciences*, vol. 180, no. 3, pp. 399-413, February 2010.
- [5] O. Bouachir, A. Abrassart, F. Garcia and N. Larrieu, "A mobility model for UAV ad hoc network," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, Orlando, FL, USA, 2014.
- [6] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, Boston, MA, USA, Kluwer Academic Publishers, 1996, pp. 153-181.
- [7] M. Y. Arafat and S. Moh, "Routing protocols for unmanned aerial vehicle networks: A survey," *IEEE Access*, vol. 7, pp. 99694-99720, 2019.
- [8] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 1, p. 24–37, 2006.
- [9] Q. Li and D. Rus, "Communication in disconnected ad hoc networks using message relay," *J. Parallel Distrib. Comput.*, vol. 63, no. 1, p. 75–86, 2003.
- [10] J. Luo, X. Gu, T. Zhao and W. Yan, "A mobile infrastructure based VANET routing protocol in the urban environment," in *Int. Conf. Commun. Mobile Comput.*, Shenzhen, China, 2010.
- [11] J. Yu, R. Zhang, Y. Gao and L.-L. Yang, "Modularity-based dynamic clustering for energy efficient UAVs-aided communications," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, p. 728–731, 2018.
- [12] F. Aftab, A. Khan and Z. Zhang, "Hybrid self-organized clustering scheme for drone based cognitive Internet of Things," *IEEE Access*, vol. 7, p. 56217–56227, 2019.
- [13] M. Iordanakis, D. Yannis, K. Karras, G. Bogdos, G. Dilintas, M. Amirfeiz, G. Colangelo and S. Baiotti, "Ad-hoc routing protocol for aeronautical mobile ad-hoc networks," in *Fifth International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Patras, Greece, 2006.
- [14] R. Shirani, M. St-Hilaire, T. Kunz, Y. Zhou, J. Li and L. Lamont, "Combined Reactive-Geographic routing for Unmanned Aeronautical Ad-hoc Networks," in *8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Limassol, Cyprus, 2012.
- [15] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, p. 234–244, 1994.
- [16] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *IEEE International Multi Topic Conference (IEEE INMIC 2001), Technology for the 21st Century.*, Lahore, Pakistan, 2001.

- [17] D. Johnson, N. Ntlatlapa and C. Aichele, "A simple pragmatic approach to mesh routing using BATMAN," in *2nd IFIP Int. Symp. Wireless Commun. Inf. Technol. Developing Countries*, 2008.
- [18] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, USA, 1999.
- [19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*, Boston, MA, USA, Springer, 1996, pp. 153-181.
- [20] Z. J. Haas and M. R. Pearlman, "The performance of a new routing protocol for the reconfigurable wireless networks," in *IEEE International Conference on Communications. Conference Record. Affiliated with SUPERC0MM'98*, Atlanta, GA, USA, 1998.
- [21] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *IEEE INFOCOM'97 The Conference on Computer Communications*, Kobe, Japan, 1997.
- [22] G. Mao, S. Drake and B. D. O. Anderson, "Design of an extended kalman filter for uav localization," in *2007 Information, Decision and Control*, Adelaide, SA, Australia, 2007.
- [23] E. Kuiper and S. Nadjm-Tehrani, "Geographical Routing With Location Service in Intermittently Connected MANETs," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 592 - 604, 2011.
- [24] B. N. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, Boston, MA, USA, 2000.
- [25] X. Li and J. Huang, "ABPP: An Adaptive Beacon Scheme for Geographic Routing in FANET," in *18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Taipei, Taiwan, 2017.
- [26] X. Zheng, Q. Qi, Q. Wang and Y. Li, "An adaptive density-based routing protocol for flying Ad Hoc networks," in *AIP Conference Proceedings 1890*, 2017.
- [27] M. Sbeiti, N. Goddemeier, D. Behnke and C. Wietfeld, "PASER: Secure and Efficient Routing Approach for Airborne Mesh Networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 3, pp. 1950 - 1964, 2016.
- [28] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69–74, April 2008.
- [29] "OMNeT++ Discrete Event Simulator," [Online]. Available: <https://omnetpp.org/>. [Accessed July 2022].
- [30] O. Dukkanci, B. Y. Kara and T. Bektaş, "Minimizing energy and cost in range-limited drone deliveries with speed optimization," *Transportation Research Part C: Emerging Technologies*, vol. 125, 2021.