TOWARDS CONTEXT-AWARE COMMUNICATIONS: USER IDENTITY MATCH IN RADIO AND VIDEO DOMAINS USING MACHINE LEARNING

Vinicius Mesquita de Pinho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Marcello L. R. de Campos

Rio de Janeiro
Fevereiro de 2021

TOWARDS CONTEXT-AWARE COMMUNICATIONS: USER IDENTITY MATCH IN RADIO AND VIDEO DOMAINS USING MACHINE LEARNING

Vinicius Mesquita de Pinho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Marcello L. R. de Campos

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2021

Mesquita de Pinho, Vinicius

Towards Context-Aware Communications: User Identity Match in Radio and Video Domains Using Machine Learning/Vinicius Mesquita de Pinho. – Rio de Janeiro: UFRJ/COPPE, 2021.

XXI, 81 p.: il.; 29, 7cm.

Orientador: Marcello L. R. de Campos

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2021.

Referências Bibliográficas: p. 45 – 48.

1. Wireless Communications. 2. Machine Learning. 3. Computer Vision. I. L. R. de Campos, Marcello. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Para Valdecy, Kátia e Amanda.
Sempre.

Agradecimentos

Agradeço à minha família, meus pais, Kátia e Valdecy, e minha irmã, Amanda. Só tenho a agradecer pelo suporte infinito e amor incondicional. Obrigado por sempre estarem ao meu lado. Essa conquista é mais de vocês do que minha. Uma nota especial à ajuda da Amanda em algumas das figuras deste trabalho, muito obrigado.

Ao meu orientador, Marcello, que é um dos responsáveis por estar concluindo este trabalho. Obrigado por sempre acreditar em mim, por me conceder tantas oportunidades. É difícil escrever um agradecimento à altura do que você já fez por mim ao longo desses anos. Os caminhos que decidi seguir em minha vida têm grande influência sua. Obrigado por todo o aprendizado e por ter a paciência de me orientar por tanto tempo.

Ter amigos é essencial para terminar um mestrado. Mesmo com a distância desses últimos tempos, tenho a obrigação de escrever os nomes de algumas pessoas muito especiais. Agradeço muito à Rebeca, Domenica, Ingrid, Pedro, Renata, Yuri, Felipe, Wesley, Rafael, Gabriel, Gabi, Padilla, Igor, Cinelli, Matheus, Roberto, Carol, Lucas, Jéssica, Bia, Vettore, Ainoã, Thalyson, baby Dom e Anna Lara.

Agradeço ao Laboratório de Sinais, Multimídia e Telecomunicações, o SMT, por ter sido uma segunda casa durante todos os meus anos por lá. Agradeço aos professores e toda a equipe. Em especial, agradeço a Dona Edinalva, por sempre nos fazer rir com sua energia e seu bom humor.

Agradeço à todas as pessoas do Nokia Bell Labs, por terem me recebido tão bem em minha passagem durante este mestrado.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de mestrado concedida.

Muito obrigado ao Programa de Engenharia Elétrica da COPPE, aos colegas discentes e aos docentes que contribuíram para minha formação.

Agradeço aos professores Eduardo e Charles, e ao Luis, que compõem a banca examinadora, por aceitarem o convite para avaliar este trabalho e pelas contribuições essenciais para melhoria do mesmo.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

TOWARDS CONTEXT-AWARE COMMUNICATIONS: USER IDENTITY MATCH IN RADIO AND VIDEO DOMAINS USING MACHINE LEARNING

Vinicius Mesquita de Pinho

Fevereiro/2021

Orientador: Marcello L. R. de Campos

Programa: Engenharia Elétrica

Os sistemas de comunicação de quinta geração (5G) estão sendo desenvolvidos e aprimorados para serem peças-chave na promoção de uma infraestrutura essencial para as novas demandas de um mundo cada vez mais conectado. Ferramentas de aprendizado profundo e visão computacional podem ser usadas para fornecer informações ao sistema de comunicação, aumentando seu conhecimento à respeito da dinâmica do ambiente e seus usuários. Informações extraídas por ferramentas de visão computacional podem fornecer dados sobre posição de usuários, direções de movimento e velocidades, que podem ser fornecidos à rede 5G. Porém, a rede 5G necessita de um mecanismo para casar informações advindas do sistema visual, formado por câmeras e a tecnologia de visão computacional, e o sistema de rádio. Atualmente, este sistema de casamento de informações não está presente na literatura. Portanto, este trabalho propõe um *framework* que realiza o procedimento de correspodência entre informações oriundas de um sistema visual e de rádio, usando um classificador baseado em aprendizado de máquina. Essa etapa de casamento de informações é essencial para incrementar o conhecimento espacial dos sistemas de comunicação. Este trabalho detalha o *framework* proposto, tanto sua fase de treinamento quanto a de experimentos práticos aplicados a uma configuração de testes. Os experimentos de teste foram realizados com informações coletadas em quatro ambientes com características diferentes. Este trabalho compara o uso de dois classificadores no *framework* proposto, um baseado em redes neurais profundas e o *random forest*. Os resultados dos experimentos mostram que o *framework* com o classificador baseado em redes neurais é capaz de atingir bons resultados, com acurácia de mais de 99% em todas as situações. As próximas gerações para além do 5G lançarão mão de soluções que exploram o uso de informações de diferentes domínios, e a nossa solução é um componente essencial que permite tais usos, como por exemplo em antecipação de *handover* assistido por câmeras e seleção de feixes em MIMO massivo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TOWARDS CONTEXT-AWARE COMMUNICATIONS: USER IDENTITY MATCH IN RADIO AND VIDEO DOMAINS USING MACHINE LEARNING

Vinicius Mesquita de Pinho

February/2021

Advisor: Marcello L. R. de Campos

Department: Electrical Engineering

5G is designed to be an essential enabler and a leading infrastructure provider in the communication technology industry by supporting the demand for the growing data traffic and a variety of services with distinct requirements. Deep learning learning and computer vision tools can increase the network's environmental awareness with information from visual data. Information extracted via computer vision tools such as user position, movement direction, and speed can be promptly available for the network. However, the network must have a mechanism to match a user's identity in both visual and radio systems. This mechanism is absent in the present literature. Therefore, we propose a framework to match the information from both visual and radio domains. The user-identity match is an essential step to practical applications of computer vision tools in communications. We detail the proposed framework training and deployment phases for a presented setup. We carried out practical experiments using data collected in different types of environments. The work compares the use of Deep Neural Network and Random Forest classifiers and shows that the former performed better across all experiments, achieving classification accuracy greater than 99%. The next generations beyond 5G will explore solutions that embrace the use of information from different domains, and our solution is a building block to enable such use, for example in video-assisted proactive handover and beam selection on massive MIMO.

Sumário

Lista de Figuras

Lista de Tabelas

List of Abbreviations

AI artificial intelligence.

AP access point.

BBOX bounding box.

CIR channel impulse response.

CV computer vision.

DNN deep neural network.

GPU graphical processing unit.

MIMO multiple-input-multiple-output.

MLP multilayer perceptron.

OFDM orthogonal frequency-division multiplexing.

RAN radio access networks.

RFC random forest classifier.

UE user equipment.

USRP universal software radio peripheral.

Capítulo 1

Introduction

1.1 Motivation

5G systems and artificial intelligence (AI) have been highlighted as fields of innovation, emblematic for the transition to a smarter society. Researchers envision 5G, and beyond 5G, offering a plethora of services and capabilities, addressing a wide range of use cases, including enhanced mobile broadband, ultra-reliable low-latency communications, and massive machine-type traffic [1, 2].

Due to the advancements in AI techniques, especially deep learning, and the availability of extensive data, there has been an overwhelming interest in using AI for the improvement of wireless networks [3, 4]. Combining deep learning and computer vision (CV) techniques have seen great success in diverse fields, such as security [5] and healthcare [6], where they deliver state-of-the-art results in multiple tasks. Applying computer vision with deep learning in wireless communications has seen recent growing interest. Computer vision brings powerful tools to improve current communications systems. Visual information enriches the environmental awareness of networks and can enable context-aware communications to a level that has not yet been explored [7]. These capabilities are envisioned to be used in the next generations, beyond 5G.

Computer vision and deep learning have direct applications in the physical layer. We can exemplify an application with the following case. When using multiple-input-multiple-output (MIMO) beamforming communication systems, beams' direction and power can be scheduled using the knowledge of users' locations and blocking cases readily available from the visual information. The immediate availability of data reduces overhead in communication, minimizing power consumption, and interference. Moreover, CV tools can give motion information about a user at the edge of the coverage area. This data can be used to predict and estimate whether or when a terminal goes out or comes into its serving area. The network can then allocate channel resources for the handover process to improve the system resources' utilization efficiency.

1.2 Related Work

Machine learning techniques have been used to solve various problems in communications systems. In [8–12], some interesting use-cases of machine learning in the field of wireless communication and networking are surveyed: MAC layer protocols designed with reinforcement learning, deep neural networks for MIMO detection, user equipment (UE) positioning with neural networks, and others. In [12], the authors address the problem of designing signaling protocols for the MAC layer using reinforcement learning. The results show a promising future for nonhuman-made protocols; they are faster and cheaper to construct when compared to the ones standardized by humans during several long meetings. Machine learning has been applied to MIMO detection; examples are the works with deep neural networks in [13] and [14]. UE positioning with neural networks as in [15] and [16] can achieve mean positioning errors of less than 2 m, essential for user localization in communication networks. Furthermore, machine learning-based solutions for communications can work with more than just radio signals to extend its capabilities. The use of computer vision-based on deep neural networks brings another source of useful tools.

Recently, the scientific community started exploring the possibility of bringing intelligence from CV systems to radio networks. In [17], the authors presented a framework for generating datasets with visual and radio information to facilitate research towards vision-aided wireless communication. The framework uses a game engine and a wireless propagation tool to recreate outdoor urban scenarios. This framework has been used for addressing beam-tracking and link-blockage problems.

The beam-tracking problem has been tackled in [7] and [18], using visual information from a dataset generated with the framework from [17]. The authors from [18] combined images and beam indices from the scene generated by the framework from [17] to fine-tune a pre-trained deep learning model to predict future beam indices. However, the oversimplified scenario with only one user hinders the analysis if the method would scale to more complex scenarios.

The link-blockage problem was addressed in [18] and [19]. The former tackles the problem from a reactive point of view, i.e., the system classifies the present link status as blocked or not. The latter focuses on a proactive treatment of the problem, using recurrent neural networks to predict future blockage. Both works show promising results, but with only a single-moving user in the presence of stationary blockages.

The works in [17], [18], and [19] can be further extended with more realistic scenarios. It is necessary to increase the number of possible users in the scenes and allow non-stationary blockages. With a more dynamic scenario, the need to match the transmitting user in both video feed and radio transmission emerges. This issue is not addressed in [17], [18], or [19].

1.3 Contributions

In a practical scenario, visual data is acquired separately from radio information. However, it is only possible to take advantage of the ready-to-use visual information if the network can match the user identity from both visual and radio domains. Otherwise, the network does not have the means to use the information extracted from the visual data. The information from visual data can be useful for the network, as in the following examples. For improving handover on the cell edge by providing means of estimating a user's trajectories and speed, or reducing the radio control channel usage by contributing to user location instead of relying solely on radio information. To the best of our knowledge, a mechanism to match visual and radio data from the same user has not yet been described in the literature. The usual approach to deal with this problem is to consider only one user at a time in the scenario, or consider the information match that is already provided for the network. Both do not happen in a realistic situation.

We close this gap by proposing a novel framework that enables the user information match from a visual source with its radio counterpart. We model the problem as a classification task using the user position in the video feed and its channel impulse response (CIR). We use a machine learning technique to solve the task of classifying the transmitting user. A solution such as ours is a necessary step to allow the development of more complex scenarios involving visual information, leading towards context-aware communications.

One example of application of the proposed solution is the initial access in massive MIMO, where many beams need to be swept out to find the most suitable one. This example is depicted in the scenario presented Figure 1.1. The proposed framework can be used to match the information from video and radio, then a mechanism to reduce the search space for the best beam can be applied. The cameras in this scenario capture the users (in this case, vehicles), and the users' positions jointly with the radio information are fed to a trained classifier. The output from the classifier can point to a preselected set beams that are suited for the user in that situation.

The proposed framework is flexible; it is possible to incorporate as many users as necessary without critically increasing the computational complexity since the features used in the classification task are one-dimensional. Furthermore, we used

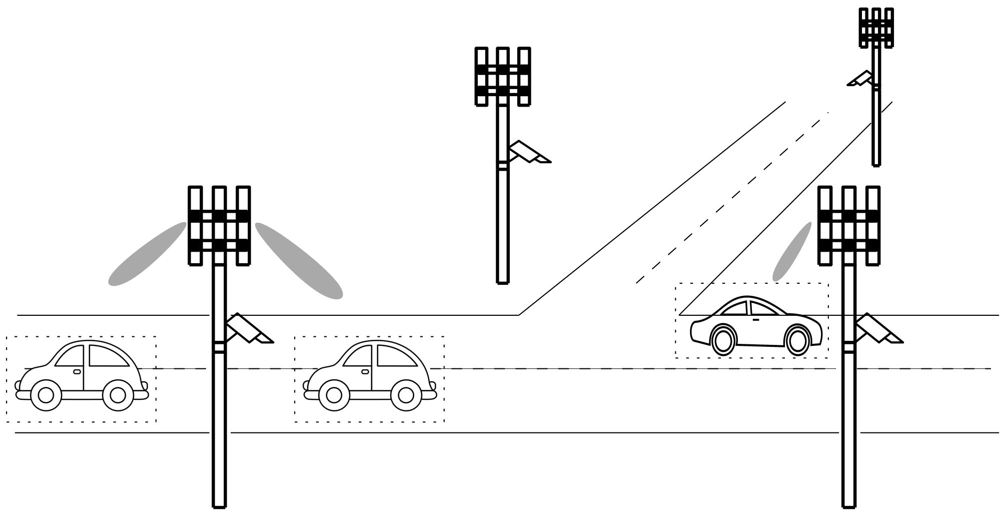


Figura 1.1: Example of scenario where the solution proposed in this work can be applied.

an experimental setup to showcase the proposed framework. We carried out experiments using real data collected in four environments with different characteristics, from indoor spaces to an outdoor area. The high classification accuracy metrics in the experiments demonstrated the potential applicability of the proposed framework.

The industrial private networks can take great advantage of using the proposed framework. The industries' private networks require a customized design due to the strict requirements of ultra-reliable and low latency users and machine-type communications. There are numerous opportunities to explore in this environment, as flexibility increases. The operator owns both the radio access networks (RAN) and the UE, they would have also the permission from every person in the space; therefore, privacy becomes less of an issue. We have access to additional information to the RAN, data otherwise not available, for example, the video feed of the covered area. Hence, the network can extract useful information about the users, readily available on visual data, reducing the communication system's latency.

1.4 Published Works

The following publications and patent were results of this work:

- Part of this work was first published in the 2020 IFIP Networking Conference in the abstract paper "User identification by matching radio 'vision' and computer vision through means of machine learning" [20].

- This work was published in the article entitled "Vision-Aided Radio: User Identity Match in Radio and Video Domains Using Machine Learning" in the IEEE Access special section on Beyond 5G Communications [21].

- The patent "METHOD FOR IDENTIFYING AND TRACKING THE UE DEVICE THROUGH MACHINE LEARNING AND COMPUTER VISION" was filled by Nokia under the application number 20205785 on the Finnish Patent and Registration Office in August of 2020.

1.5 Work Structure

The remainder of this work is organized as follows.

Chapter 2 describes the proposed framework and the testbed used throughout the thesis. We start with the description of the testbed Section 2.1 as it allows a more comprehensive and applied description of the framework. Section 2.2 describes the framework and methods for matching a UE in a video feed to UE identity in a radio transmission using machine learning and computer vision. The framework is described with a direct application on the testbed. The experiments and results obtained in the testbed are detailed in Chapter 3. We extend beyond the testbed in Chapter 4, discuss more complex scenarios where the proposed framework can be used, and present simulations results from the scenarios presented. Finally, conclusions are drawn in Chapter 5, and future work paths are considered.

Capítulo 2

Framework and Testbed Description

This chapter describes a testbed that allows us to illustrate the principle of the proposed identity-matching procedure, its feasibility, and how the experiments can be reproduced. We favor open software and communication entities (looking forward to Open-RAN [22, 23]), yet the concept can be extended to 5G devices for commercial use.

2.1 Experimental Setup Description

The setup for the testbed is illustrated in Figure 2.1. The setup components are a graphical processing unit (GPU)-enabled laptop, a camera, an access point (AP), and two identical, visually indistinguishable UEs.

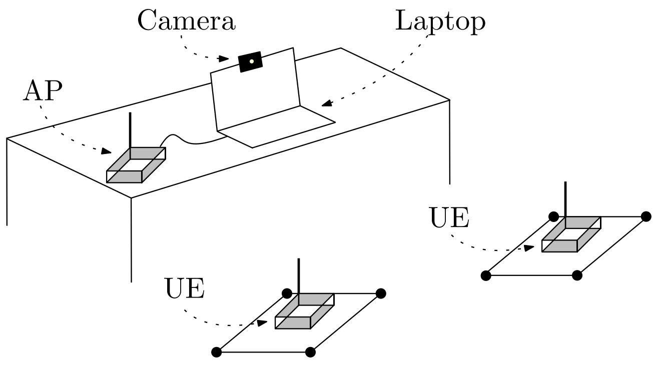


Figura 2.1: Setup for the testbed containing one camera, a laptop, and three USRPs.

The AP and user devices are implemented using universal software radio peripherals (USRPs) model Ettus B210. We implement a simplified uplink transmission using GNU Radio [24] based on the IEEE 802.11a orthogonal frequency-division multiplexing (OFDM) standard [25]. The active user USRP sends a pilot-based frame to the AP. The frame uses a 52-subcarrier OFDM operating at 1 GHz. All the subcarriers are used to transmit pilots. The frame is modulated with a binary phase-shift keying modulation. The USRP playing the AP part is connected to the laptop, where the received signal is processed with GNU Radio.

The video stream acquisition is made with a Logitech C922 Pro Stream HD webcam connected to the laptop.

An equivalent 5G setup would have the following correspondence with our experimental setup. The AP is the 5G base station, gNB, and the two UEs are the 5G User Devices (e.g., robots in industrial networks). The camera can be collocated with the gNB, or the RAN can be connected through a communication interface to the camera. The processing done in the GPU computer can be executed at the gNB site or other entity of the RAN (e.g., the RAN-Location Management Function).

2.2 Framework

We model the user-matching task as a classification problem and use a machine learning approach to solve it. The steps of the framework are visually illustrated in Figure 2.2 and summarized as follows.

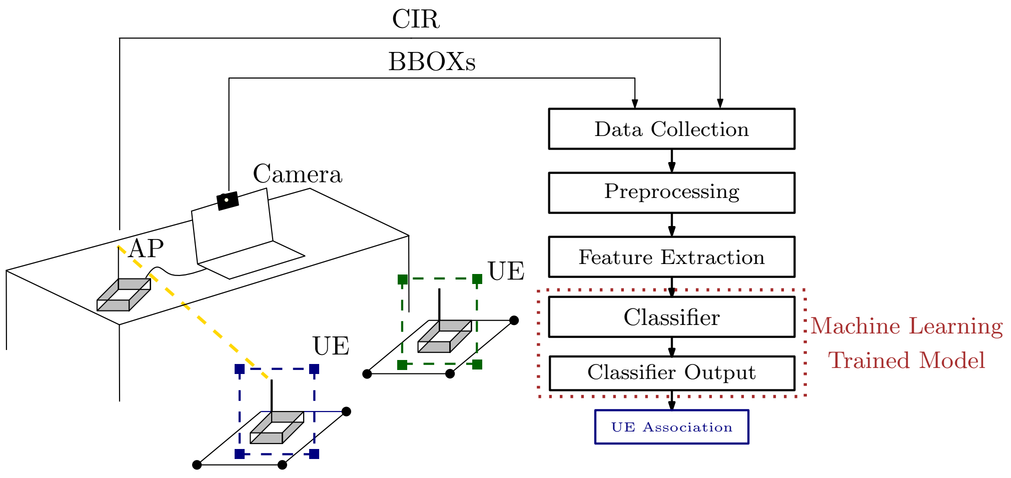


Figura 2.2: Illustration of framework steps linked with the experimental setup.

- Data collection: acquisition of data from the video system and the radio system;
- Preprocessing: merging data from both sources and purge of spurious samples;
- Feature extraction: extraction of relevant features from preprocessed data;
- Training the ML model. In the following we will detail the option using Random Forest and Neural network classifiers:
  - Classifier: classification of input features;

  - Classifier Output: labeling outcomes with corresponding levels of confidence;

- UE Association: association of classifier output with corresponding user information.

2.3 Data Collection

The first essential step for collecting video data is the recognition of radio devices in the video feed. Recognizing an object in a video feed is a well-known computer-vision task. We apply an existing ready-to-use framework to detect radio devices, in our case, USRPs, in the video feed. We use and adapt an object-detection tool available in the Detectron2 framework [26]. The Detectron2 is trained to recognize the devices by fine-tuning a mask region-based convolutional neural network pre-trained on the COCO dataset [27]. Figure 2.3 shows three examples of manually annotated images containing USRPs with surrounding bounding boxes (BBOXs) to fine-tune the model. The reader is referred to [26] for a complete description of the Detectron2 framework and means for fine-tuning to custom data. The tool's output is an array with the BBOXs, which indicates the radio devices' positions in the video feed. In addition, Detectron2 provides levels of confidence of the detection of the objects.

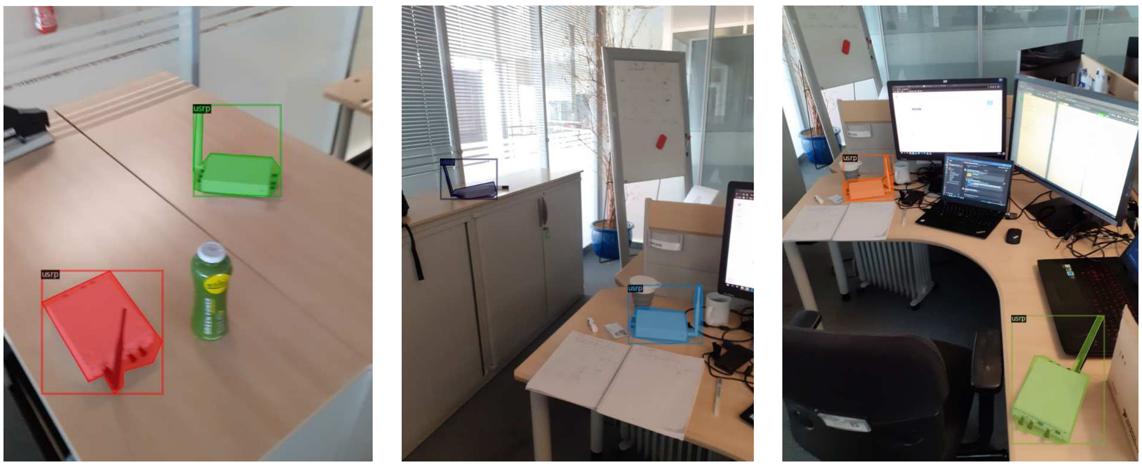


Figura 2.3: Examples of manually annotated images with bounding boxes around USRPs, used for fine-tuning the model pre-trained on the COCO dataset.

The data we collect from the video feed are the arrays with the BBOXs, as illustrated in Figure 2.4, indicating the position of the devices in the scene, along with their levels of confidence of the detection.

The space analyzed by the camera is limited to the area where Detectron2 outputs have levels of confidence of 99% or higher, and the devices can move freely within the area. We filter the outputs with lower levels of confidence. The high accuracy is imposed to avoid spurious measurements in the testbed. We make sure that our dataset only contains visual information of the desired objects (i.e., USRPs), avoiding errors made by the vision system, e.g., classifying a box as an USRP.

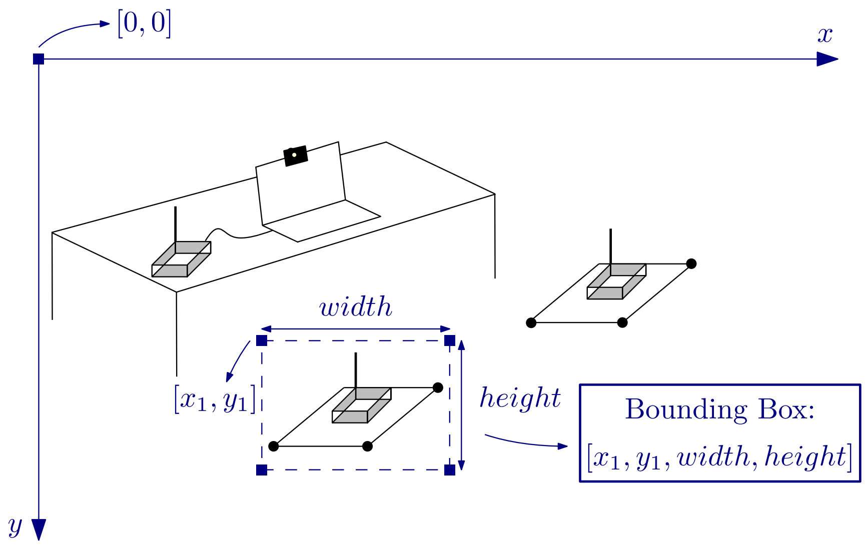


Figura 2.4: The bounding box is composed of a point $[x_1, y_1]$ related to the origin in the superior-left corner of the image and the object's height and width, in pixels.

The data collected from the radio system are the CIRs. The CIR-collecting process is illustrated in Figure 2.5. The CIR is computed in GNU Radio with pilot-based frames from the link between the transmitting device and the AP. The set of CIRs computed during transmission is stored.

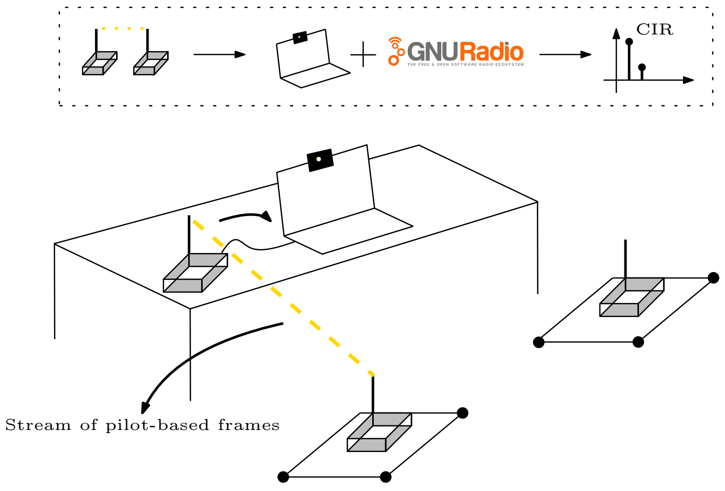


Figura 2.5: The process of computing the channel impulse responses comprises one device transmitting to the access point, processing by the software-defined radio part and computer. The GNU Radio processes the stream of pilot-based frames and computes the CIRs; then, the CIRs are stored.

2.4 Preprocessing

During data collection, the information from the vision and radio systems are acquired concurrently. Each source of data saves the collected measurements with a unique timestamp. We create a unified representation using both vision and radio sources by matching their timestamps. Figure 2.6 illustrates the unification process. The frequency of data acquisition from both domains is different; the radio system is much faster, which results in more CIRs being saved than BBOXs. We match the data from both domains with identical timestamps, which can be viewed as a sampling process to pick a CIR representing an ensemble of CIRs.

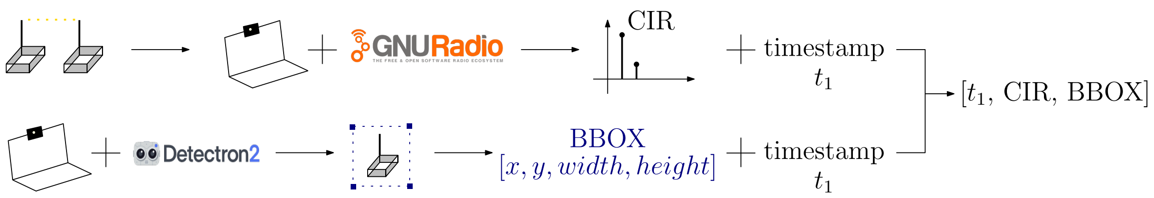


Figura 2.6: The unified representation of data from vision and radio sources is made by matching their unique timestamps, which the BBOXs and CIRs receive when they are stored on the computer.

With the measurements unified, the collected measurements are preprocessed. The CIR records with a maximum magnitude below a threshold δ are discarded. We carry out the removal procedure because a small number of CIRs are wrongly estimated by the GNU Radio due to synchronization issues in transmitted frames. After this data-cleaning step, the remaining signals are fed to the feature extractor.

The BBOXs names are coded into numeric labels for the training phase. The vision system outputs a vector with a BBOX for each of the two devices present in the scene. When there are two devices in the scene, one gets a BBOX named "BBOX 1" and the other the "BBOX 2". Given that there are only two devices in our testbed, the following situations will be treated: when device "BBOX 1" is transmitting and the one named "BBOX 2" is not, the training label generated is $X = 1$. The training label $X = 2$ is generated when the device named "BBOX 2" is transmitting, and "BBOX 1" is not. One example of the situation labeled as $X = 2$ is illustrated in Figure 2.7. When no device is transmitting, the label generated is $X = 0$, also called "NO TX". Hence our system is going to be trained to classify three different situations, designed with the label $X \in \mathcal{X} = \{0, 1, 2\}$.

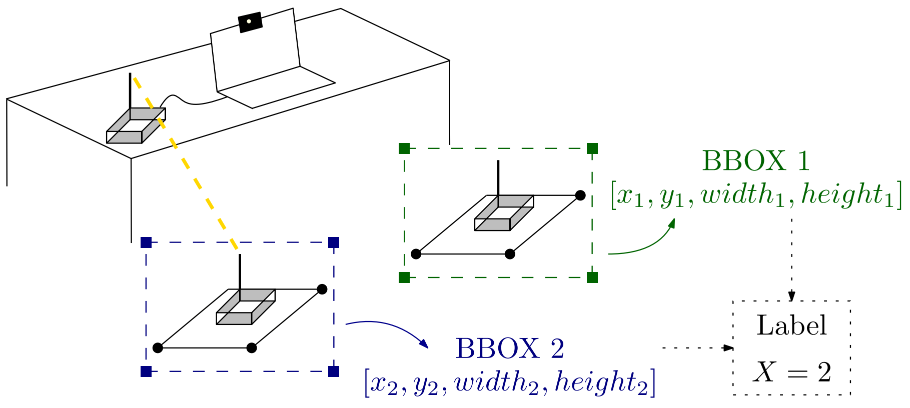

BBOX 1
$[x_1, y_1, width_1, height_1]$
BBOX 2
$[x_2, y_2, width_2, height_2]$
Label
$X = 2$

Figura 2.7: Example on how the labeling process is done. The device with the "BBOX 2" is transmitting in this example as illustrated by the dashed yellow line; therefore, the measurements collected will receive the label $X = 2$.

In this work, we do not consider the case of two users transmitting simultaneously due to equipment limitations. However, the extension to more complex cases are discussed in Chapter 4. Furthermore, for the practical experiments we carried out, the devices were moved throughout the setup area, and the system periodically reassessed the labels of the devices.

2.5 Feature Extraction

We identified the following features of the CIR, defined in (2.1), as being relevant for our problem: the CIR magnitude, phase, and the value and sample index of the CIR magnitude peak in the radio frame.

$$h(n) = \sum_{k=0}^{N-1} a_k \mathrm{e}^{\mathrm{j}\theta_k} \delta(n - \tau_k), \tag{2.1}$$

where k is integer, N is the number of multipath components, a_k, τ_k, and θ_k are the random amplitude, propagation delay, and phase of the kth multipath component, respectively. δ is the Dirac delta function.

From the vision system, we are using the array with the BBOXs. Figure 2.8 shows the feature extraction steps in the framework used for training the model.

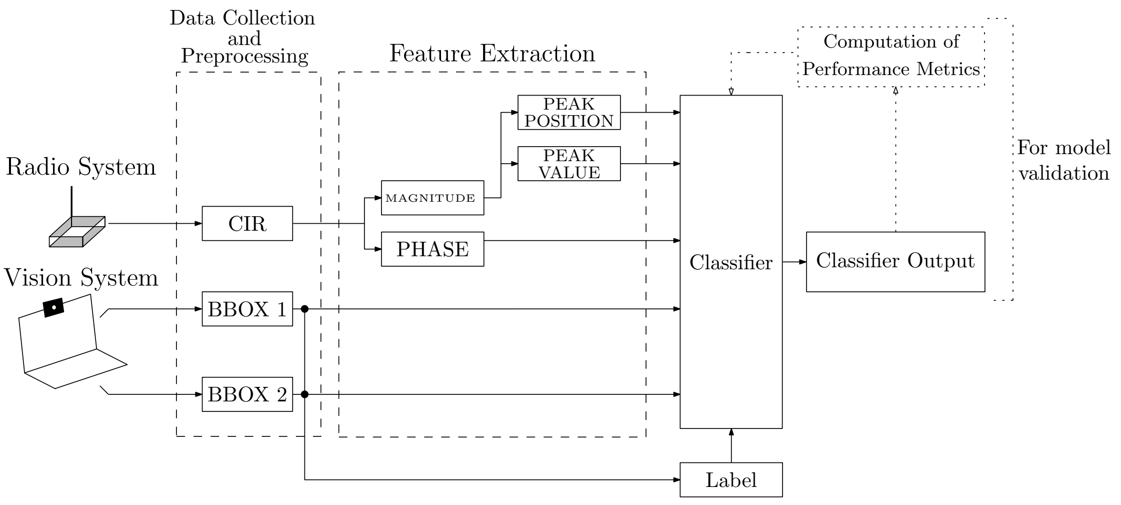


Figura 2.8: Details of the framework used for training and validation phases.

2.6 Random Forest Classifier

Figure 2.9 shows the input for the classifier. The labels are used for supervised model training. Afterward, the trained model can be used in the deployment phase, as illustrated by the framework in Figure 2.10, with only the features to classify new data. In this work, we train the models with random forest classifiers (RFCs) and deep neural networks (DNNs). The proposed framework is agnostic to the classifier used. We used RFCs and DNNs because both techniques are robust and give good classification results.

Features				Label
CIR	CIR-related	BBOX 1	BBOX 2	X

Figura 2.9: Input instance for the classifier, with the radio and video domain features and annotated with a label used for training and validation. One CIR and its related features (peak position and value), because we consider the case where only one USRP is transmitting in the scene.

The RFC is an ensemble learning algorithm for classification that uses decision trees [28], [29]. The RFC constructs a large number of decision trees at training time and outputs the class that is the mode of the output from the individual trees.

We train the model by combining an exhaustive grid search over RFC parameter values. The search space is confined to 20–50 for the number of trees with a maximum depth between 30 and 80. The training uses a 10-fold cross-validation procedure. The training dataset is split into 10 smaller sets. The model is trained using 9 of the folds and validated on the remaining part of the data. To evaluate the performance of the trained model, in each iteration we compute two different metrics: the logarithmic loss and the F_1-score. We choose the best model given the performance metrics. A summary of the parameters used for training the RFCs are displayed in Table 2.1.

Tabela 2.1: Summary of parameters used for training the Random Forest Classifiers.

Parameter	Value
Number of trees	20–50
Max tree depth	30–80
Cross-validation	10-fold
Metrics during training	logarithmic loss and F_1-score

The best-trained model for a given dataset is used for testing, where we compute the confusion matrix, precision, recall, F_1-score, and classification accuracy.

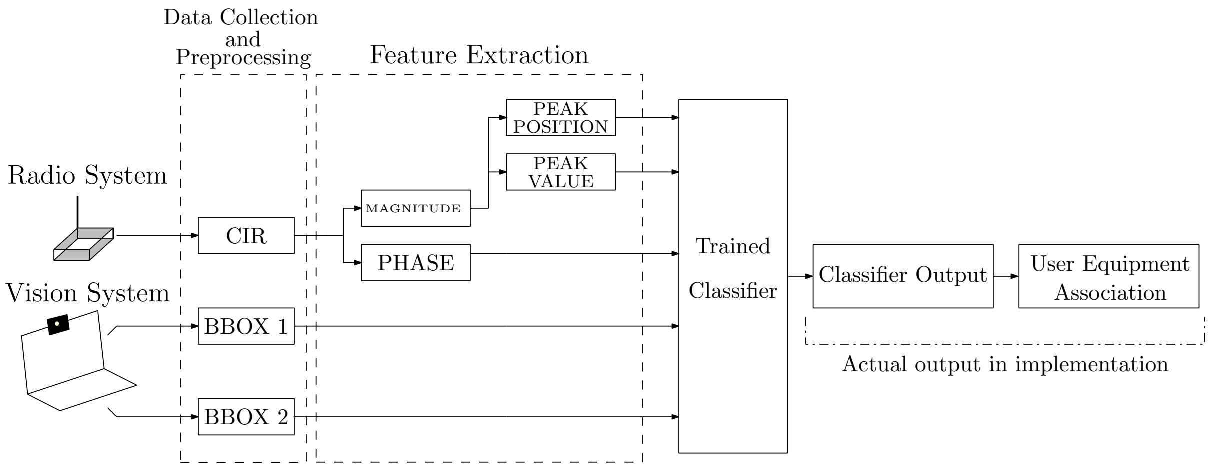


Figura 2.10: Details of the framework used for deployment.

2.7 Classifier Output and UE Association

The classifier outputs are the predicted label number indicating which user is transmitting in the scene and the level of confidence of the output. During the training and validation procedures, the classifier output is used to compute the performance metrics, as illustrated in Figure 2.8, using dotted lines.

For deployment, the framework we use is shown in Figure 2.10. The classifier output is used to make the association with the device. When two possible users are in the scene, if the predicted label is $X = 1$, the device associated with the "BBOX 1" is the one transmitting in the scene, analogously for when the label is $X = 2$. When the predicted labels are $X = 0$, no user is transmitting to the AP in the scene. In the scenario with only one user, the possible outcomes are: the predicted label is $X = 1$ when the user is transmitting, or $X = 0$ when no one is transmitting. With this step done, we have matched the information from both radio and video systems. In summary, the vision system detects two devices and can tell which one is transmitting, successfully matching visual and radio information.

2.8 Deep Neural Network Classifier

The neural network that we use is a feedforward neural network or multilayer perceptron (MLP). An MLP is composed of one input layer (pass-through), one or more hidden layers (with an activation function), and one final layer called the output layer. This structure is illustrated in Figure 2.11. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. When a neural network has two or more hidden layers, it is called a DNN.

An MLP is often used for classification, as in this case of this work. When the output classes are exclusive (e.g., classes 0 through 9 for digit image classification), the output layer is modified by replacing the individual activation functions with a shared softmax function, as depicted in Figure 2.11. The output of each neuron corresponds to the estimated probability of the corresponding class.

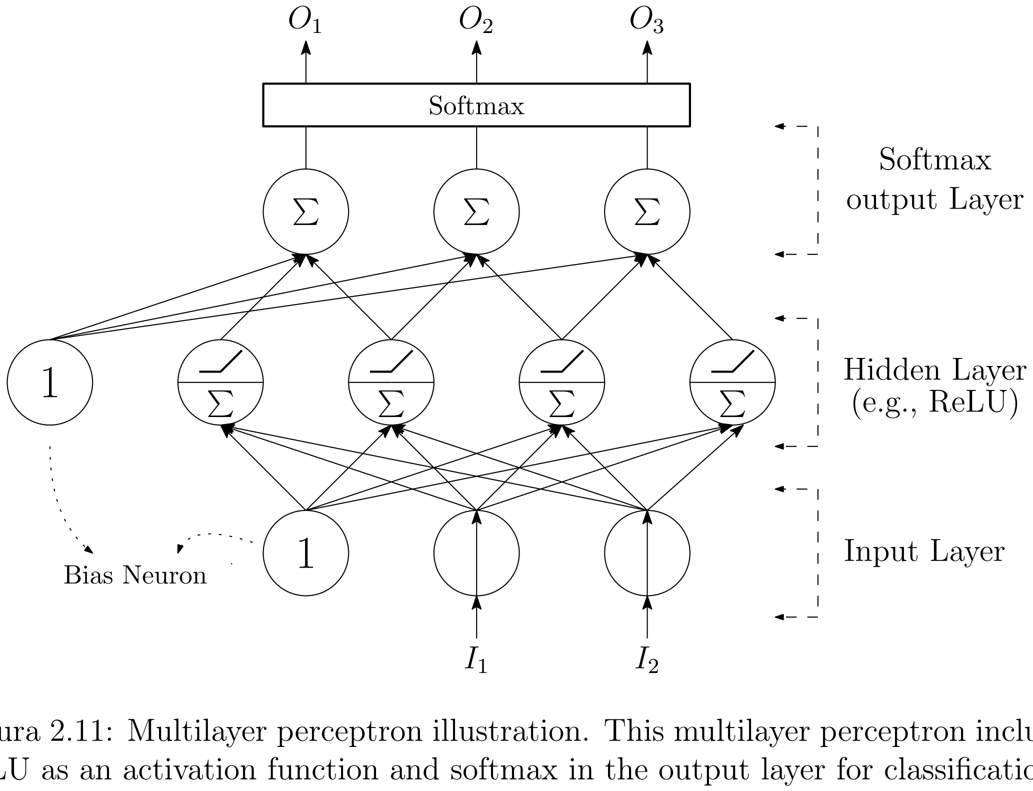


Figura 2.11: Multilayer perceptron illustration. This multilayer perceptron includes ReLU as an activation function and softmax in the output layer for classification.

The architecture we use in this work is detailed in Table 2.2. The DNN consists of an input layer, followed by three hidden layers and an output layer. We use three hidden layers, each with ReLu as an activation function, followed by a dropout layer with a rate of 0.5 to hinder overfitting. The output layer uses softmax as an activation function.

The ReLU function can be defined as

$$\text{ReLU}(z) = \max(0, z).$$

The function ReLU(z) is continuous but not differentiable at $z = 0$, where the slope changes abruptly, making the Gradient Descent bounce around [30]. However, in practice, it works very well and has the advantage of being fast to compute. The ReLU(z) function is depicted in Figure 2.12.

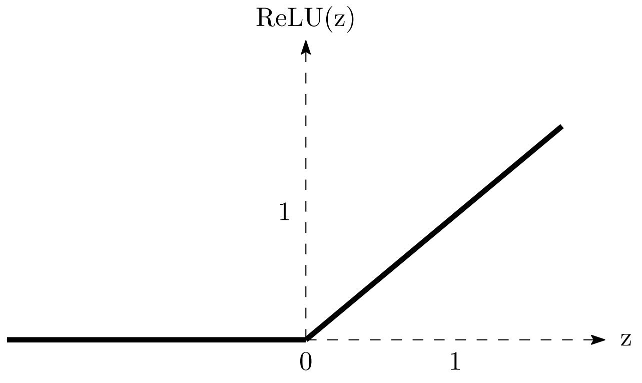


Figura 2.12: The ReLU function.

As aforementioned, we use a dropout after the first two hidden layers. The dropout is a regularization technique for DNNs; it was proposed by [31] and further detailed in [32]. The dropout algorithm is as follows. At every training step, every neuron has a probability p of being (temporarily) dropped out. The act of "dropping out" the neuron means the neural network will ignore it during this training step, but it may be active during the next step [33]. The parameter p is called *dropout rate*; the values used in this work are detailed in Table 2.2.

During training, the labels are encoded using one-hot encoding to transform categorical data into a non-ordinal numerical representation. This encoding is done to prevent the neural network of assuming that two nearby values are more similar than two distant values. To fix this issue, we create one binary attribute per category in the one-hot encoding. For further details on the typical implementation of neural networks, the reader is referred to [33] and [34].

Tabela 2.2: Neural Network Architecture, with Specified Parameters for Each Layer and Number of Trainable Parameters.

Layer	Layer Type	Parameters	# Parameters
Layer 1	Dense + ReLu	Units: 256	15616
Layer 2	Dropout	Dropout Rate: 0.5	0
Layer 3	Dense + ReLu	Units: 128	32896
Layer 3	Dropout	Dropout Rate: 0.5	0
Layer 4	Dense + ReLu	Units: 64	8256
Layer 5	Dense + Softmax	Units: 3	195

2.9 Conclusions

In this chapter, we presented the proposed framework. A setup for a testbed was described, which allowed us to illustrate the identity-matching procedure. Each step of the framework was detailed, from data collection to training and deployment procedures. Finally, two options for classification methods were presented, Random Forest and Neural Networks. However, the proposed framework is agnostic to the classifier used.

Capítulo 3

Experiments and Results

In this chapter, we present practical experiments and discuss their results. We introduce the setups in which the experiments were carried out, the metrics used to evaluate the training and validation performances, the validation results, and a discussion about the results. We conclude the chapter with examples of deployments using the proposed framework.

We carried out experiments to evaluate the performance of the proposed framework in matching the correct users to their identities. Four experimental configurations with different dynamics were used. Hence each set of measurements has distinct characteristics. The different characteristics allowed us to test the capacity of our method to operate in different environments.

3.1 Experimental Setups

Setup 1, illustrated in Figure 3.1, was located in an indoor environment—an 18 m^2 furnished room and only one person inside, to avoid fluctuations in the CIR measurements. For the measurement campaign, the equipment was put in place, as described in Chapter 2, Section 2.1. We defined an area of 2 m^2 in front of the camera, where the user devices could move freely. The object-detection tool could survey the whole space and detect the devices with high accuracy to avoid spurious measurements. We collected data for training and validation separately. The video and radio information was stored in the laptop's hard drive. For the measurements in this setup, there were 233,960 instances collected. They were divided into 176,874 for training and 57,086 for validation. The number of instances acquired during the measurement campaign is detailed in Table 3.1.

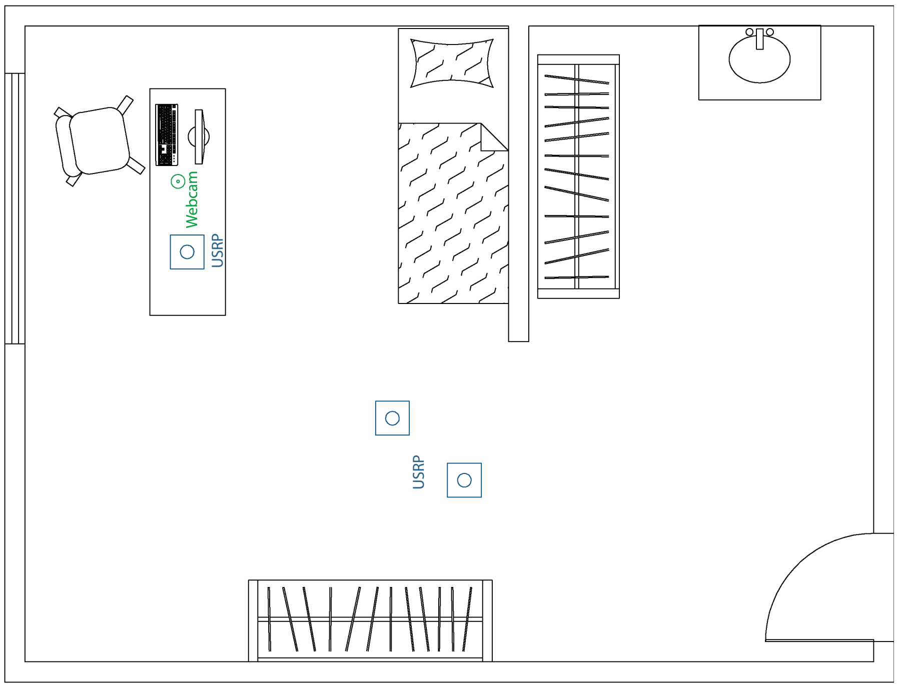


Figura 3.1: Illustration of Setup 1 location. This figure is a not-to-scale simplified depiction of the place, to illustrate to the reader where the experiments were conducted.

Tabela 3.1: Number of Instances in the Training and Validation Datasets per Experiment.

	Number of Instances		
Setup	Training	Validation	Total
Setup 1	176,874	57,086	233,960
Setup 2	242,975	154,098	397,073
Setup 3	380,527	105,187	485,714
Setup 4	38,145	16,013	54,158

Setup 2 was arranged in the corridor of office space, as depicted in Figure 3.2. The environment has a different geometry than the other setup places. There are more reflections of the transmitted signal, which affects the CIR measurements. The setup place also tests the vision system's ability to recognize the USRPs in a different environment. The measurement campaign followed the same procedures as in Setup 1. In this case, a total of 397,073 instances were collected.

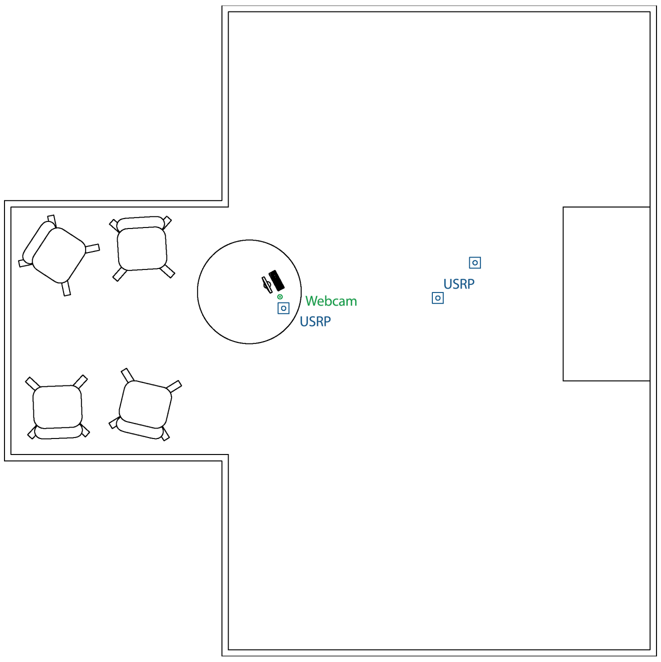


Figura 3.2: Illustration of Setup 2 location. This figure is a not-to-scale simplified depiction of the place, to illustrate to the reader where the experiments were conducted.

Setup 3 was placed inside a laboratory with electronic equipment, illustrated in Figure 3.3. We followed the same steps for the measurement campaigns as the previous setups. The level of noise in the measurements was higher than in the previous experimental configurations. For this reason, the measurement campaign collected more data in this setup. Table 3.1 shows we acquired two times more instances in Setup 3 when compared to Setup 1.

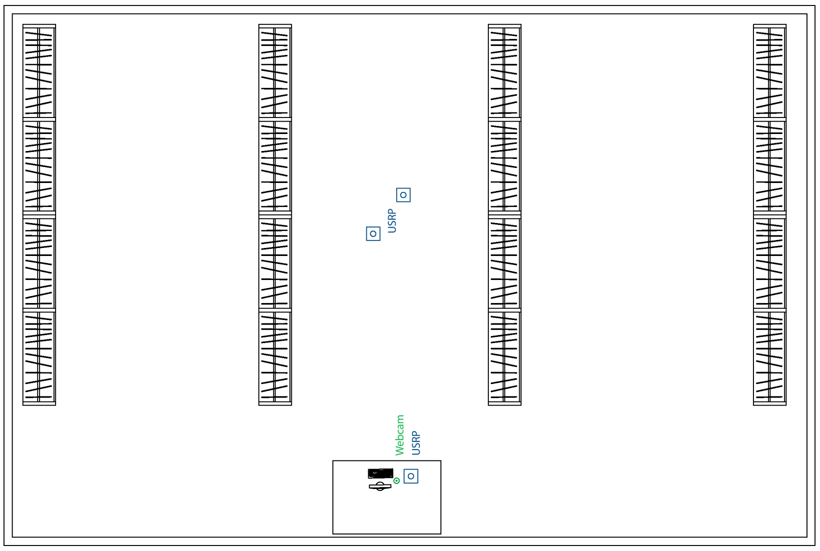


Figura 3.3: Illustration of Setup 3 premises. This figure is a not-to-scale simplified depiction of the place, to illustrate to the reader where the experiments were conducted.

Data collected for Setup 4 test our solution in an outdoor scenario. Setup 4, as shown in Figure 3.4, was situated outside the building. The outdoors measurements affected the CIRs estimations, bringing different characteristics to the datasets acquired in this place. We followed the same steps for the measurement campaign as in the previous setups. For Setup 4, a total of 54,158 instances were collected. They were divided into 38,145 for training and 16,013 for validation.

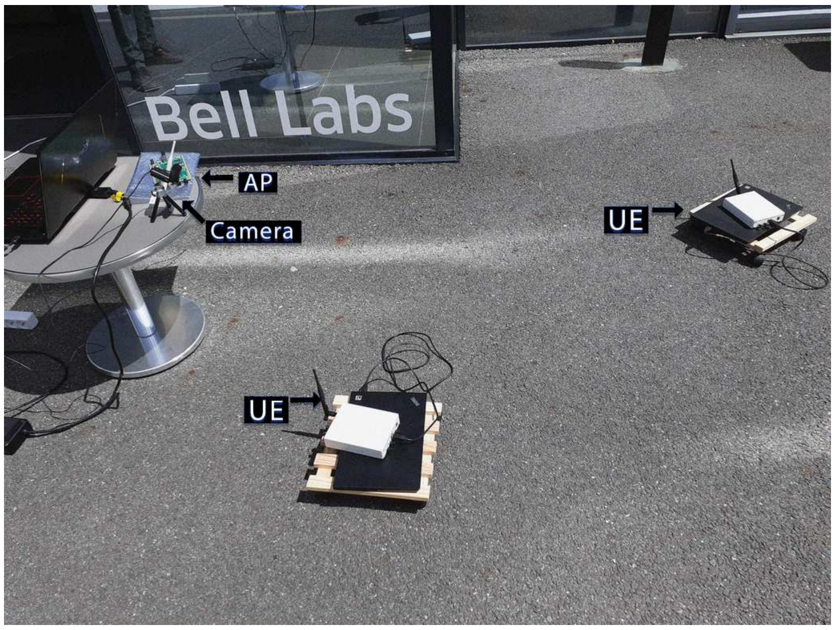


Figura 3.4: Setup for experiment 4 carried out in an outdoor area.

For each setup measurements, we preprocessed the training and validation data and extracted the engineered features, as detailed in Chapter 2. We carried out experiments using RFCs and DNN classifiers. The DNN classifiers were trained during 10 epochs. The learning rate used was 0.001. The architecture is presented in Table 2.2. The layers were initialized using the Glorot uniform initializer and no bias. Training for all the experiments was carried out in a Dell G3 3590 Laptop, with an Intel i7-9750H, 8 GB of RAM, and an NVIDIA GTX 1660 Ti Max-Q 6 GB. The training time is an average of running the same procedure 10 times.

3.2 Performance Metrics

We evaluated the trained models' performance in the classification task on the validation dataset. We plotted the confusion matrix. For easier comprehension, the labels defined in Chapter 2, Section 2.4, are called "NO TX", "BBOX 1" and "BBOX 2" for $X = 0$, $X = 1$ and $X = 2$, respectively. Furthermore, we compute the accuracy, average precision, recall, and F_1-score [35]. Accuracy is the percentage of the predicted outputs that exactly matched the corresponding set of true labels. Moreover, precision is computed as $tp/(tp + fp)$, where tp is the number of true positives and fp the number of false positives. The precision discloses the ability of the classifier not to label as positive a sample that is negative. Recall tells us the ability of the classifier to find all the positive samples. The recall score is computed as $tp/(tp+fn)$, where fn is the number of false negatives. Furthermore, F_1-score is the harmonic mean of precision and recall. It can be computed as $tp/(tp+0.5[fp+fn])$. The highest possible value of the F_1-score is 1, indicating perfect precision and recall, and the lowest possible value is 0 if either the precision or the recall is zero. In this work, the F_1-score is obtained using the weighted-averaging approach, i.e., we considered the class frequency for each individual label when computing the F_1-score, because we have an unbalanced training dataset with fewer instances with labels $X = 0$.

3.3 Results and Discussion

The first experiment was the one with Setup 1 using a random forest classifier. Training time took 12.21 minutes. The validation results are the following. The accuracy was 94.09%, precision 0.96, recall 0.96, and F_1-score 0.96. The confusion matrix is displayed in Figure 3.6. From the confusion matrix, we can see that 11.7% of the instances from "BBOX 1" were mistakenly classified as "BOX 2". The classifier assigns a wrong label to the validation dataset instance. This misclassification happens because the model cannot differentiate the two users due to the devices' close positions in the video feed, which results in similar CIRs. This type of problem can be exemplified in Figure 3.5, where we have CIR 1 and CIR 2 taken directly from two measurement instances that the classifier for Setup 1 wrongly classified. As the input for the classifier are only the BBOX and CIR, we have these occurrences which are hard for the algorithm to distinguish when very CIRs are associated with close BBOX. This type of problem occurs throughout the results in this work, leading to lower accuracy. A solution can can achieved using other sensors that would permit differentiate between two users in this situation.

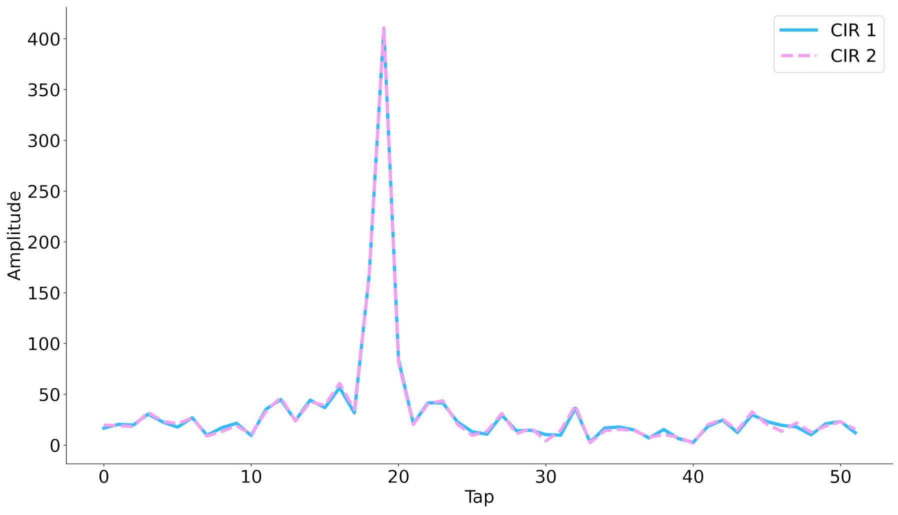


Figura 3.5: Example of two CIRs, CIR 1 and 2, where .

Moreover, all the dataset instances with no device transmitting, labeled as "NO TX", were correctly classified. When no user is transmitting, the dataset instances have null values in their fields, making it easy for the classifier to label them correctly.

The Setup 1 with neural network classifier took 03.50 minutes to train. Figure 3.7 displays the confusion matrix. The metrics computed show 99.91% accuracy, the precision of 0.99, recall of 0.99, and F_1-score of 0.99. Therefore DNN was not as prone to classification errors as RFC.

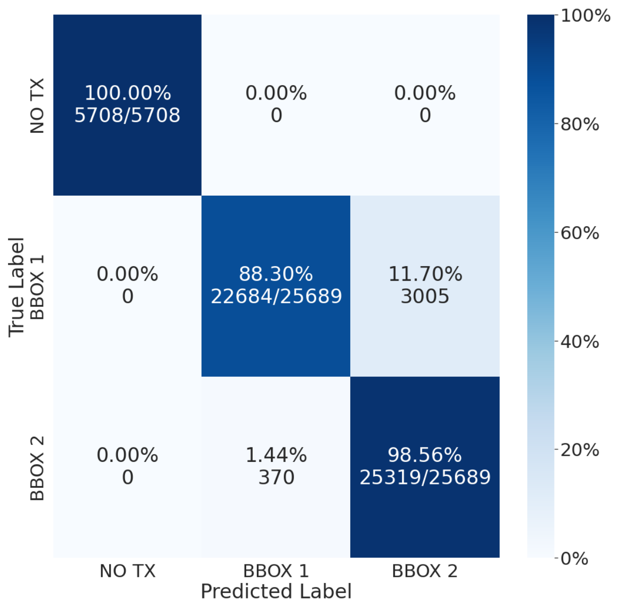


Figura 3.6: Confusion Matrix for Setup 1 data trained with Random Forest Classifier.

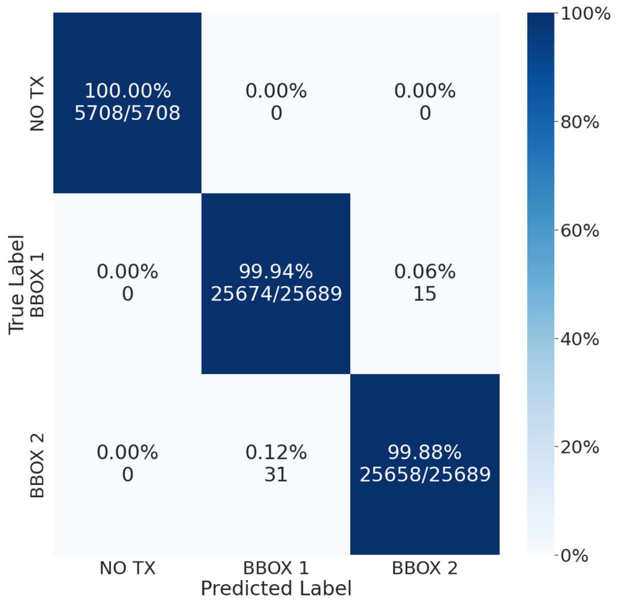


Figura 3.7: Confusion Matrix for Setup 1 data trained with Neural Network Classifier.

The experiment with Setup 2 using the RFC took 14.30 minutes to train. The results were: accuracy 99.77%, precision 0.99, recall 0.99, and F_1-score 0.99. An equivalent analysis can be seen in the confusion matrix in Figure 3.8. The confusion matrix shows that approximately 0.04% (29 cases) of the instances from "BBOX 1" were misclassified as "BBOX 2". For the instances labeled was "BBOX 2", only 0.48% of the measurements instances, the system incorrectly classified them as "BBOX 1".

For Setup 2 with the neural network classifier, the training time was 04.86 minutes. The performance metrics were: accuracy 99.98%, precision 0.99, recall 0.99, and F_1-score 0.99. Figure 3.9 shows the confusion matrix. In this case, only 19 instances were incorrectly classified, which is negligible, result similar to one of the forest.

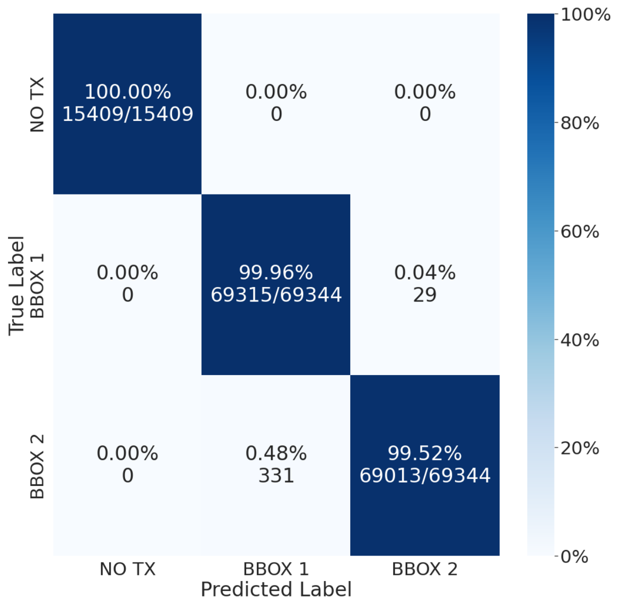


Figura 3.8: Confusion Matrix for Setup 2 data trained with Random Forest Classifier.

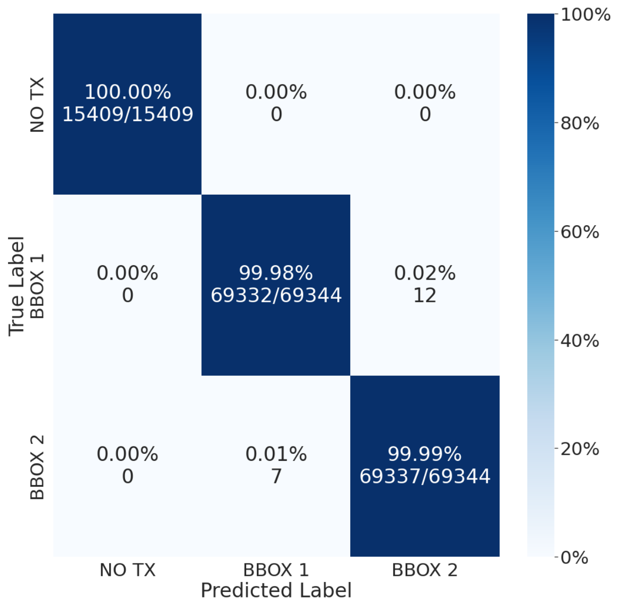


Figura 3.9: Confusion Matrix for Setup 2 data trained with Neural Network Classifier.

For the experiment on Setup 3, using the RFC training time was 16.89 minutes. The training duration was longer than to the other experiments because the training dataset was the largest, as shown in Table 3.1. For the validation dataset, the metrics are the following: accuracy 78.35%, precision 0.84, recall 0.84, and F_1-score 0.84. The accuracy score is lower than the previous ones. However, the confusion matrix in Figure 3.10 shows that the system continues to perform well, depending on the application requirements. It gets 100% correct outputs when no device is transmitting in the scene. The instances with “BOX 2” were correctly classified with an accuracy of 82%.

In the experiment in Setup 3 using a neural network, the training was 06.15 minutes long. The confusion matrix for validation is displayed in Figure 3.11. The neural network classifier was able to handle the measurements in this setup better than the random forest due to the network's architecture capacity of generalization. The accuracy of this experiment was 99.76%. Precision, recall, and F_1-score were all 0.98. The high score values show the robustness of the neural network with the architecture presented in Table 2.2.

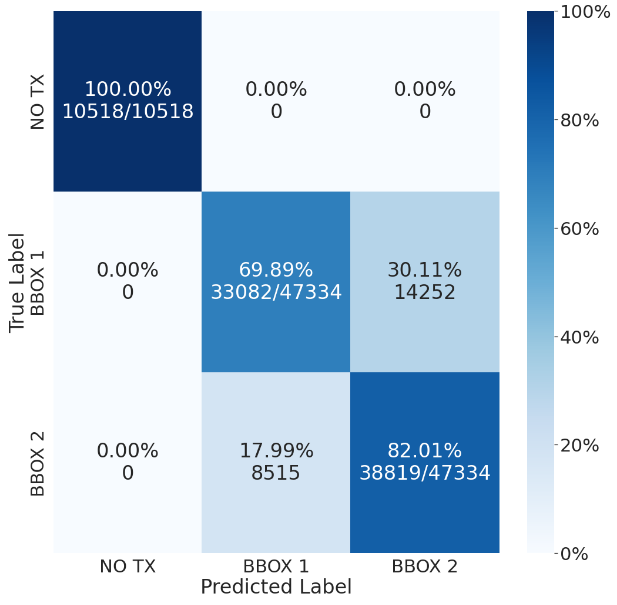


Figura 3.10: Confusion Matrix for Setup 3 data trained with Random Forest Classifier.

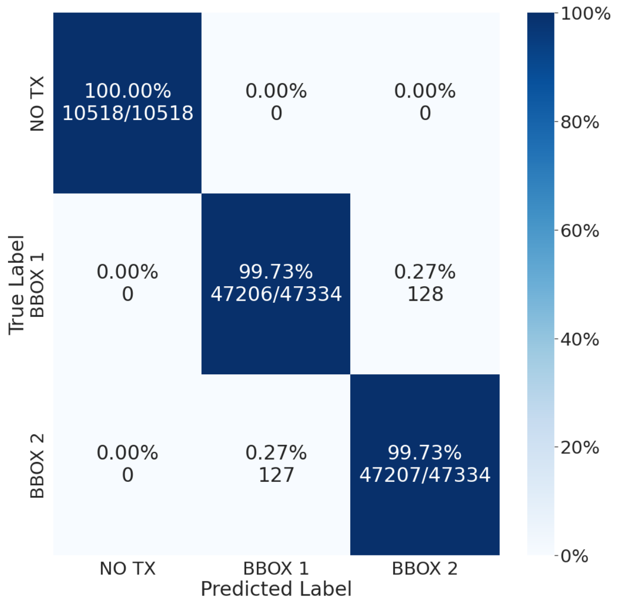


Figura 3.11: Confusion Matrix for Setup 3 data trained with Neural Network Classifier.

Moreover, an experiment using Setup 4 was carried out using RFC. The training time of 06.10 minutes. The measurement campaign for Setup 4 was shorter, leading to smaller training and validation datasets. However, the system achieved excellent results, as the metrics show. The accuracy was 99.66%. Precision was 0.99, the same results for recall and F_1-score. The confusion matrix is shown in Figure 3.12.

The experiment with Setup 4 measurements using a neural network classifier had a training time of 02.01 minutes. The confusion matrix is in Figure 3.13. Accuracy 99.99%, precision, recall, and F_1-score were 0.99.

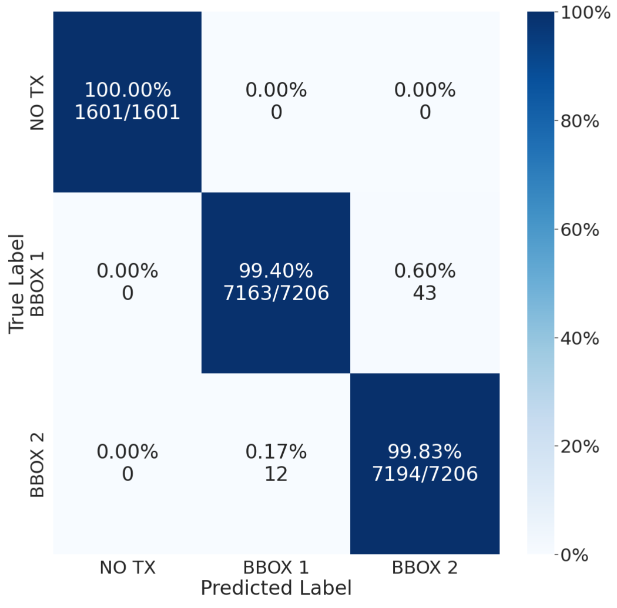


Figura 3.12: Confusion Matrix for Setup 4 data trained with Random Forest Classifier.

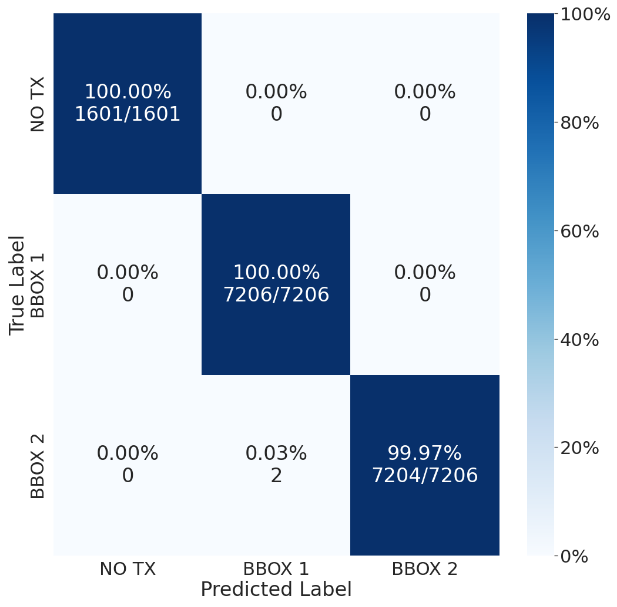


Figura 3.13: Confusion Matrix for Setup 4 data trained with Neural Network Classifier.

The training time for the experiments using neural network classifiers was, on average, three times lower than the ones with random forest classifiers. The longer training duration occurs because the random forest included an exhaustive grid search for parameters and cross-validation during training. The neural network classifiers were training during ten epochs, and no hyper-parameter search was used. Hence the shorter training time for the classifiers using neural networks.

The performance metrics show that experiments with the random forest classifiers had F_1-scores equal to or higher than 0.84. The precision and recall also achieve similar values. These results still give us an accurate and robust classifier; it correctly classifies the instances, even in situations that are difficult to distinguish the transmitting device. For example, when both devices are in positions where the CIRs are similar due to the transmission characteristics, as exemplified in Figure 3.5.

The lower F_1-score of 0.84 is from the experiment using Setup 3 and RFC. The reason for lower performance metrics, when compared to the other experiments, can be found in the search space used for hyperparameter tuning. The numbers of trees and tree depth presented in Chapter 2, Section 2.6, did not contain the hyper-parameter values needed for this experiment to succeed. A better solution can be found with a greater number of trees in the ensemble. With 227 trees and a tree depth of 65, the F_1-scores is 0.97. However, a larger number of trees in the ensemble increases the time necessary for the model to make a classification in the deployment phase. In this work, we maintained the same search space in all the experiments to make the comparisons fair.

In a practical case, the search space for the RFC can be changed until the best solution is found. The training duration is in the order of minutes. Hence it is feasible to train multiple times for the same set of measurements. After the training phase, during the deployment phase, the model gives an output in a negligible amount of time. In this sense, the cost of retraining the dataset is not high, even for the random forest classifiers.

The experiments with the neural network classifiers achieved F_1-score of 0.99 in every setup. Only a minor part of the dataset instances were incorrectly classified. With a small architecture of the neural networks, as displayed in Table 2.2, the models can train fast and still excel in the classification task, as shown by the performance metrics gathered in Table 3.2.

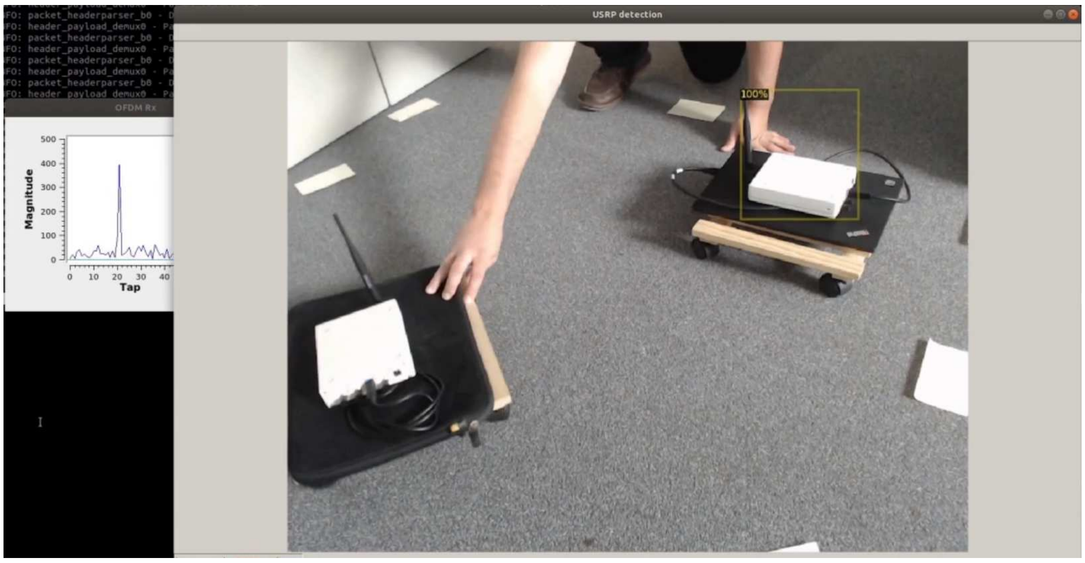
Tabela 3.2: Performance Validation Results of the Experiments.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	94.09	0.96	0.96	0.96	12.21
Setup 1 - NN	99.91	0.99	0.99	0.99	03.50
Setup 2 - RFC	99.77	0.99	0.99	0.99	14.30
Setup 2 - NN	99.98	0.99	0.99	0.99	04.86
Setup 3 - RFC	78.35	0.84	0.84	0.84	16.89
Setup 3 - NN	99.76	0.98	0.98	0.98	06.15
Setup 4 - RFC	99.66	0.99	0.99	0.99	06.10
Setup 4 - NN	99.99	0.99	0.99	0.99	02.01

Overall, the high accuracy and F_1-score in the experiments show the capability of the proposed framework to perform well across different environments. Using the testbed described in Chapter 2, Section 2.1, we tested the proposed framework using datasets with different sizes, collected in different types of places. The results confirm that our solution is capable of correctly match the user identity in a video feed with its corresponding radio signal.

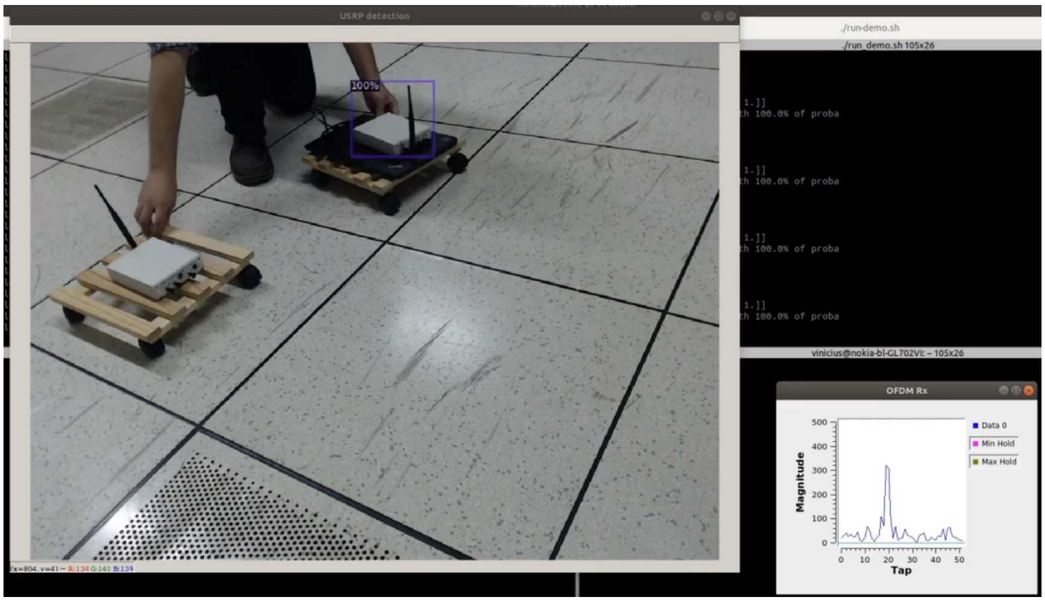
3.4 Prototype Testbed Deployment

In order to test the proposed framework, we deployed our trained classifiers on the previously described setups. Figure 3.14 and Figure 3.15 show examples of the prototype deployment on Setups 2 and 3, respectively. The images display the proposed framework for deployment from Figure 2.10 in action. The vision system captures the BBOXs, and the radio system computes the CIR (shown in the examples), both inputs for the trained classifiers. The "User Equipment Association" from Figure 2.10 becomes visual in the form of a BBOX around the transmitting device.



```
FO: packet_headerparser_b0 - D
FO: header_payload_demux0 - Pa
FO: packet_headerparser_b0 - D
FO: header_payload_demux0 - Pa
FO: packet_headerparser_b0 - D
FO: header_payload_demux0 - Pa
FO: packet_headerparser_b0 - D
FO: header_payload_demux0 - Pa
```

Figura 3.14: First example of testing the proposed framework on Setup 2. On the left-hand side of the image, there is a window with the CIR. On the right-hand side, two devices are in the scene. A yellow BBOX indicates which device is transmitting to the AP. In the superior-left of the BBOX, the classification level of confidence is displayed; in this case, 100%.



```
./run_demo.sh 105x26

 1.]]
th 100.0% of proba

 1.]]
th 100.0% of proba

 1.]]
th 100.0% of proba

 1.]]
th 100.0% of proba

vinicius@nokia-bl-GL702VI: ~ 105x26
```

Figura 3.15: Second example of testing the proposed framework on Setup 3. On the inferior right-hand side of the image, there is a window with the CIR. On the left-hand side, two devices are in the scene. A purple BBOX indicates which device is transmitting to the AP. In the superior-left of the BBOX, the classification level of confidence is 100%.

The prototype deployments were successful. Our solution was capable of correctly identifying the device transmitting to the AP; hence, the prototype deployments were consistent with the validation results. The system was capable of running each part of the framework without any problems. The result was a real-time flow, where the system was capable of tracking the devices across the covered area.

3.5 Conclusions

In this chapter, we presented practical experiments and discussed their results. The validation results showed very high classification accuracy values for each of the different setups. Section 3.4 showed two examples of prototype deployments using the trained models. The prototype deployments were consistent with the validation results, showing that our solution was capable of correctly matching the visual and radio data.

Capítulo 4

Beyond the Testbed

In this chapter, we discuss more complex scenarios using our proposed framework. The experimental testbed described in Chapter 2, Section 2.1, can then be further extended. It is possible to use our proposed framework to include more devices in the scene. Although not strictly necessary, it is possible to use more cameras to capture different angles of the environment. The framework is flexible to adapt and work in more realistic scenarios. This chapter presents three scenarios that cover most of the possible practical situations. We present simulation results for each scenario using extended versions of the datasets from Chapter 3.

4.1 Scenario 1

One possible example is the following. We have two different cameras, C_1 and C_2, to detect four possible transmitting-devices simultaneously. We assume two of the devices are transmitting at the same time, one connected to AP_1 and the other one to AP_2. This scenario is illustrated in Figure 4.1.

In order to use the proposed framework in this scenario, we divide it into two separate systems. One system, S_1, is composed of the camera C_1 and access point AP_1. The other system, S_2, is composed of the camera C_2 and AP_2. Both S_1 and S_2 individually resemble the testbed presented in Chapter 2, Section 2.1. The difference lies in the classifier input. The S_1 and S_2 can have the input for their classifiers as the one depicted in Figure 4.2. S_1 and S_2 classifiers would be trained to output X, where $X \in \mathcal{X} = \{0, 1, 2, 3, 4\}$. In this case, each system operates independently, i.e., S_1 and S_2 can be trained and deployed as independent systems.

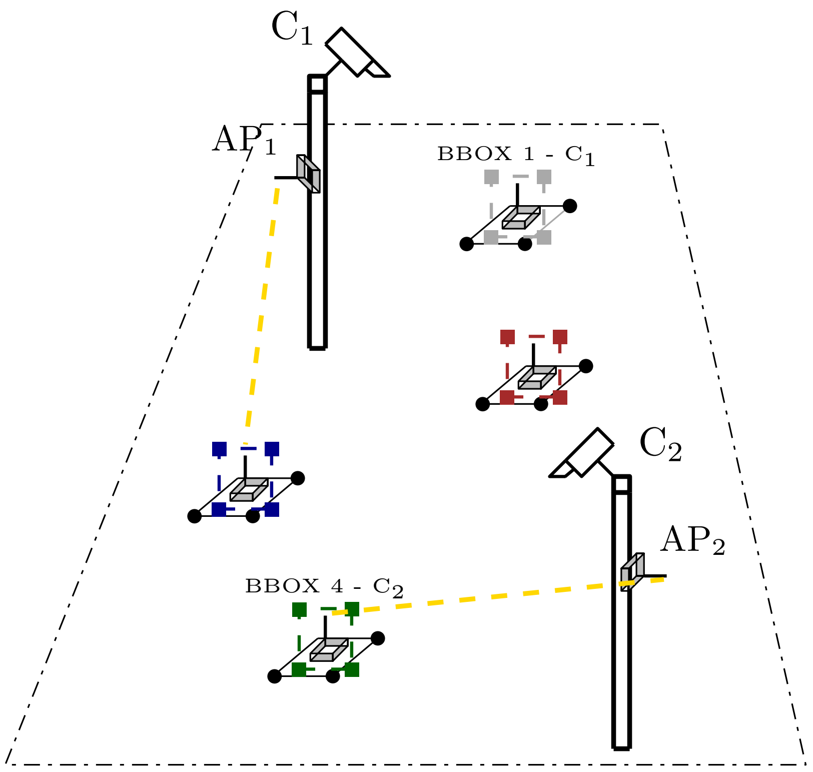


Figura 4.1: First example of a more complex scenario. There are two cameras, C$_1$ and C$_2$, two access points, AP$_1$ and AP$_2$, and four devices in this scene. We indicate the transmitting-device connection to the access point with the yellow-dashed line. The BBOXs related to each camera are shown; we omit some BBOXs' names for a more straightforward illustration, but they can be easily inferred.

Features						Label
CIR	CIR-related	BBOX 1	BBOX 2	BBOX 3	BBOX 4	*X*

Figura 4.2: Example of input for the classifier when there are four different devices in the scene and information from one camera.

There is also the possibility of S$_1$ and S$_2$ sharing information between them. For example, they can exchange visual data. In this case, the input for their classifiers would be as illustrated in Figure 4.3. The number of features increases because S$_1$ and S$_2$ are sharing the BBOXs they captured. However, each BBOX is an array with four floating-point numbers, which does not significantly increase classifier's computational complexity. Furthermore, the case of S$_1$ and S$_2$ sharing CIR is analogous to sharing BBOXs.

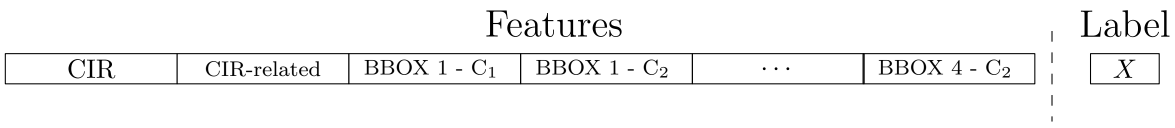


Figura 4.3: Example of input for the classifier when there are four different devices in the scene and information from two cameras.

Furthermore, two situations can occur in any scenario. First, if the classifier is trained with a maximum of four devices, but during one period, only three or fewer devices are present in the scene. All vacant BBOXs are filled with a null-valued BBOX. Second situation: the classifier is the same as in the first case, but five (or more) devices are present in the scene. As the classifier is not trained for this case, the result is not reliable. The transmitting device might not have its BBOX in the classifier input. In this case, the system has to train the classifier again to adjust to the scene's maximum number of simultaneous devices.

4.2 Scenario 2

Another example of a more complex scenario using the proposed framework is illustrated in Figure 4.4. In this scenario, we have a camera, C_1, an access point, AP_1, and four devices. We assume that three devices are transmitting to AP_1.

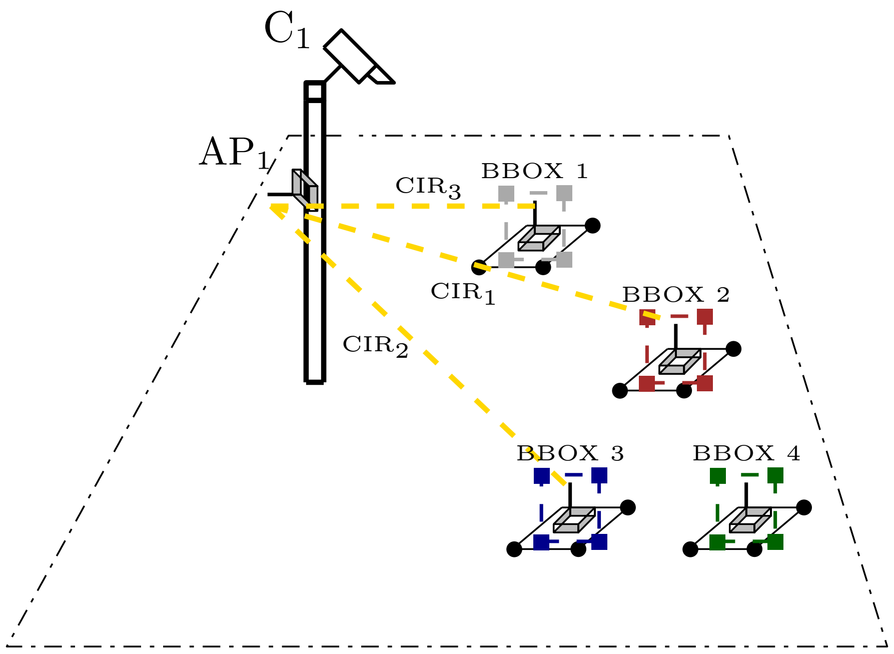


Figura 4.4: The second example of more complex scenarios. There is one camera, C_1, one access point, AP_1, and four devices in this scene. We indicate the transmitting-device connection to the access point with the yellow-dashed line; there are three of them. The BBOXs names are also shown.

To use the proposed framework in this scenario, we can train our system using the same input to the classifier, as presented in Figure 4.2. The training is carried out with one device transmitting throughout the area covered by camera C_1 to capture the different BBOXs and CIRs.

For the deployment, we have to use three instances of trained classifiers simultaneously. The CIRs for each transmitting device goes to a different classifier instance, as illustrated in Figure 4.5.

Features

CIR_1	CIR_1-related	BBOX 1	BBOX 2	BBOX 3	BBOX 4

CIR_2	CIR_2-related	BBOX 1	BBOX 2	BBOX 3	BBOX 4

CIR_3	CIR_3-related	BBOX 1	BBOX 2	BBOX 3	BBOX 4

Figura 4.5: Example of input for three classifier instances when there are four different devices in the scene and information from one camera.

For the scene illustrated in Figure 4.4, the results expected from the classifiers are the following. From top to bottom: the device with CIR_1 is the one with the BBOX 2; hence expected result is $X = 2$. The classifier with CIR_2 yields $X = 3$ and the one with the CIR_3, $X = 1$.

The need to use three classifier instances is not a problem. The notebook responsible for computing the results for this work (see Chapter 3, Section 3.1) can run one instance of the classifier in a negligible amount of time. This speed implicates in a real-time experience. The system is capable of tracking the devices as they move throughout the scene. In this sense, running three classifier instances in a dedicated machine would not be a problem in terms of computational complexity.

4.3 Scenario 3

In this final case proposed a scenario that mixes Scenario 1 from Section 4.1 and Scenario 2 from Section 4.2. In the scenario depicted in Figure 4.6, we have two cameras, C_1 and C_2, two access points, AP_1 and AP_2, and four devices. We are assuming that two devices are connected to each access point in this scenario.

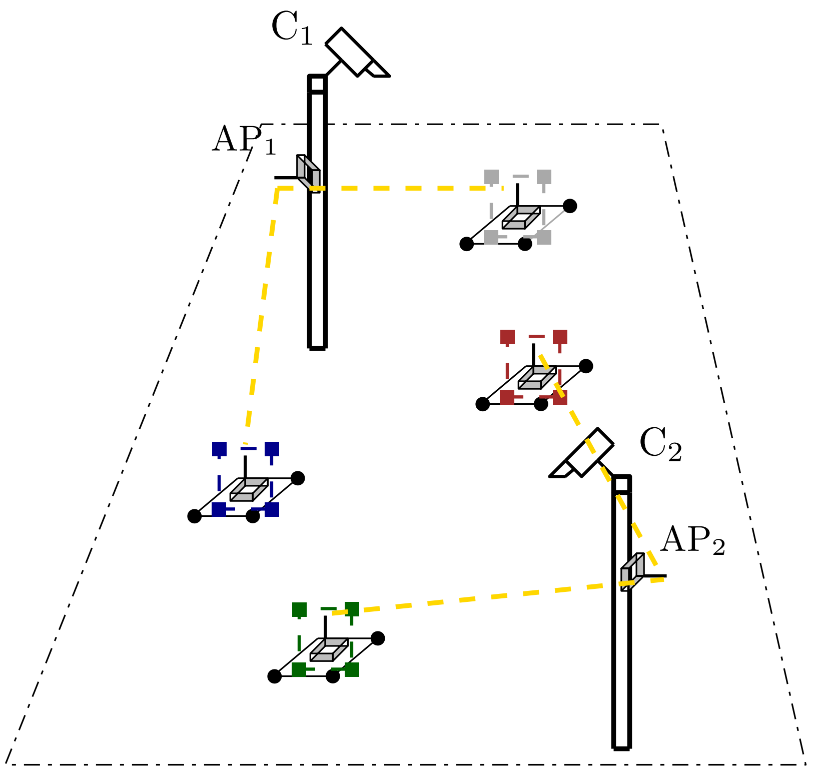


Figura 4.6: The third example of more complex scenarios. In this scene, there are two cameras, C$_1$ and C$_2$, two access points, AP$_1$ and AP$_2$, and four devices. We indicate the transmitting-device connection to the access point with the yellow-dashed line; there are four of them. We omit the BBOXs names for a clearer illustration and because their definition is not necessary to this example.

Using the same division as in Scenario 1 from Section 4.1, we have a system S$_1$, composed by the camera C$_1$ and access point AP$_1$ and S$_2$, composed by C$_2$ and AP$_2$. Let us assume that we are using the model where we share BBOXs information between S$_1$ and S$_2$. In this case, we can train our systems using the same classifier input as the one depicted in Figure 4.3.

For the deployment, we have to use the same strategy devised in Section 4.2. Both systems, S$_1$ and S$_2$, have two connected users; hence, we have to use two classifier instances to match their visual and radio information. The input for each classifier instance is illustrated in Figure 4.7.

Features

CIR$_1$	CIR$_1$-related	BBOX 1 - C$_1$	BBOX 1 - C$_2$	⋯	BBOX 4 - C$_2$

CIR$_2$	CIR$_2$-related	BBOX 1 - C$_1$	BBOX 1 - C$_2$	⋯	BBOX 4 - C$_2$

Figura 4.7: Example of input for two classifier instances when there are four different devices in the scene and information from two cameras.

4.4 Simulation Results

In order to simulate the scenarios presented in this chapter, the datasets from the practical experiments presented in Chapter 3 are extended. In this section, we detail the procedures for modifying the datasets for each scenario, to ensure that the extended versions represent the setups. For each scenario, we present the simulation results using the extended version of the datasets. Confusion matrices for all the simulations in this chapter are presented in the Appendix A, we omit them here for conciseness.

4.4.1 Scenario 1

To emulate the situation described in Figure 4.1, we can simulate the systems S_1 and S_2 separately when they do not share visual information between them. For simulating S_1 and S_2 we used the classifier input as illustrated in Figure 4.2. To simulate the situation where they share visual information, a system $S_{1,2}$ was considered with the classifier input exemplified in Figure 4.3.

S_1 and S_2 individually

To simulate the scenario from Figure 4.1, we needed to extend datasets presented in Chapter 3. For this scenario, we carried out the following procedure.

1. Select one measurement instance of the original dataset.
2. From a different measurement instance from the same dataset, randomly select two BBOXs.
3. Check if the selected BBOXs do not overlap with the already existing BBOXs. If they overlap, repeat '2.'. Proceed, otherwise.
4. Save the modified entry into the extended dataset. Go to '1.' and repeat all the steps until you have selected all the instances from the original dataset.

We assumed that S_1 with the camera C_1 had the same initial coordinate $[0, 0]$ from Figure 2.4 matching the original dataset. For S_2, we moved the initial coordinate to $[200, 200]$, hence C_2 was be positioned below and to right in relation to C_1, similarly as in Scenario 1. Finally, we created extended versions of our original datasets to simulate Scenario 1 and trained and validate this new data. We followed exactly the same steps for training and validation as described in Chapters 2 and 3.

Table 4.1 and 4.2 show the results considering only S_1 and S_2, respectively.

Tabela 4.1: Performance Validation Results of the Simulations for S_1 in Scenario 1.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	93.93	0.94	0.94	0.94	12.27
Setup 1 - NN	99.90	0.99	0.99	0.99	03.53
Setup 2 - RFC	99.76	0.99	0.99	0.99	14.38
Setup 2 - NN	99.96	0.99	0.99	0.99	04.70
Setup 3 - RFC	78.19	0.79	0.78	0.78	16.87
Setup 3 - NN	99.73	0.99	0.99	0.99	06.16
Setup 4 - RFC	99.05	0.99	0.99	0.99	06.23
Setup 4 - NN	99.80	0.99	0.99	0.99	02.12

Tabela 4.2: Performance Validation Results of the Simulations for S_2 in Scenario 1.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	93.91	0.94	0.94	0.94	12.27
Setup 1 - NN	99.89	0.99	0.99	0.99	03.54
Setup 2 - RFC	99.77	0.99	0.99	0.99	14.36
Setup 2 - NN	99.95	0.99	0.99	0.99	04.82
Setup 3 - RFC	78.49	0.79	0.79	0.79	15.91
Setup 3 - NN	99.74	0.99	0.99	0.99	06.09
Setup 4 - RFC	99.05	0.99	0.99	0.99	06.23
Setup 4 - NN	99.91	0.99	0.99	0.99	02.42

The results are very similar to the practical simulations presented in Table 3.2. It is important to note that even adding two more possible users, the training time did not increase, due to the type of features used.

S_1 and S_2 sharing BBOXs

We also considered the case where a unique classifier is used, from Figure 4.3, and we called the resulting system $S_{1,2}$. Table 4.3 shows the results from the simulations.

Tabela 4.3: Performance Validation Results of the Simulations for $S_{1,2}$ in Scenario 1.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	93.93	0.94	0.94	0.94	12.39
Setup 1 - NN	99.93	0.99	0.99	0.99	03.58
Setup 2 - RFC	99.81	0.99	0.99	0.99	14.45
Setup 2 - NN	99.95	0.99	0.99	0.99	04.75
Setup 3 - RFC	79.37	0.79	0.79	0.79	16.37
Setup 3 - NN	99.76	0.99	0.99	0.99	06.19
Setup 4 - RFC	99.24	0.99	0.99	0.99	06.46
Setup 4 - NN	99.88	0.99	0.99	0.99	02.38

The shared information here did not increase the performance of our classifiers. The BBOXs added were only a modified version of the already existing data, this did not help the system because there was new information. However, this idea of exchanging data can be useful if the systems collected different information.

4.4.2 Scenario 2

To simulate the second scenario, presented in Figure 4.4, the process was similar to the Scenario 1. The process was the following.

1. Select one measurement instance of the original dataset.
2. From two different measurement instances from the same dataset, randomly select two CIRs and their respective BBOXs.
3. Check if the selected BBOXs do not overlap with the already existing BBOXs. If they overlap, repeat '2.'. Proceed, otherwise.
4. Create and save the modified entry into the three different extended datasets. Go to '1.' and repeat all the steps until you have selected all the instance from the original dataset.

We considered three different classifier instances for this simulation, one for each CIR, so the results are presented for CIR_1, CIR_2, and CIR_3, separately. One caveat is that we did not consider the effects of acquiring three CIRs at the same time, since we extended the already existing datasets.

Table 4.4, 4.5, and 4.6 show the results considering only CIR$_1$, CIR$_2$, and CIR$_3$, respectively.

Tabela 4.4: Performance Validation Results of the Simulations for CIR$_1$ in Scenario 2.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	94.01	0.94	0.94	0.94	12.33
Setup 1 - NN	99.89	0.99	0.99	0.99	03.38
Setup 2 - RFC	99.72	0.99	0.99	0.99	13.89
Setup 2 - NN	99.95	0.99	0.99	0.99	03.99
Setup 3 - RFC	78.15	0.79	0.78	0.78	15.67
Setup 3 - NN	99.76	0.99	0.99	0.99	05.81
Setup 4 - RFC	99.59	0.99	0.99	0.99	05.33
Setup 4 - NN	99.78	0.99	0.99	0.99	02.01

Tabela 4.5: Performance Validation Results of the Simulations for CIR$_2$ in Scenario 2.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	94.00	0.94	0.94	0.94	12.56
Setup 1 - NN	99.91	0.99	0.99	0.99	02.91
Setup 2 - RFC	99.75	0.99	0.99	0.99	12.70
Setup 2 - NN	99.96	0.99	0.99	0.99	04.24
Setup 3 - RFC	78.51	0.80	0.78	0.78	16.12
Setup 3 - NN	99.75	0.99	0.99	0.99	06.10
Setup 4 - RFC	99.70	0.99	0.99	0.99	06.46
Setup 4 - NN	99.85	0.99	0.99	0.99	02.03

Tabela 4.6: Performance Validation Results of the Simulations for CIR$_3$ in Scenario 2.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	94.22	0.94	0.94	0.94	12.97
Setup 1 - NN	99.91	0.99	0.99	0.99	03.11
Setup 2 - RFC	99.75	0.99	0.99	0.99	14.02
Setup 2 - NN	99.96	0.99	0.99	0.99	04.26
Setup 3 - RFC	77.99	0.78	0.78	0.78	16.11
Setup 3 - NN	99.74	0.99	0.99	0.99	06.16
Setup 4 - RFC	99.62	0.99	0.99	0.99	06.30
Setup 4 - NN	99.83	0.99	0.99	0.99	02.03

4.4.3 Scenario 3

The simulations for the Scenario 3 were carried out as illustrated in Figure 4.7. We carried out considering only CIR_1 and CIR_2, the transmitters connected to S_1, formed by AP_1 and C_1. However, we also used the BBOXs captured by $C2$.

Tables 4.7 and 4.8 show the results considering only CIR_1, CIR_2, and CIR_3, respectively.

Tabela 4.7: Performance Validation Results of the Simulations for CIR_1 from S_1 in Scenario 3.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	94.05	0.94	0.94	0.94	12.71
Setup 1 - NN	99.91	0.99	0.99	0.99	03.88
Setup 2 - RFC	99.75	0.99	0.99	0.99	14.07
Setup 2 - NN	99.95	0.99	0.99	0.99	04.25
Setup 3 - RFC	78.48	0.79	0.78	0.78	16.19
Setup 3 - NN	99.72	0.99	0.99	0.99	06.14
Setup 4 - RFC	99.08	0.99	0.99	0.99	06.01
Setup 4 - NN	99.83	0.99	0.99	0.99	02.44

Tabela 4.8: Performance Validation Results of the Simulations for CIR_2 from S_1 in Scenario 3.

	Validation Results				Training Time
Setup - Classifier	Accuracy (%)	Precision	Recall	F_1-score	Duration (min)
Setup 1 - RFC	93.82	0.94	0.94	0.94	12.26
Setup 1 - NN	99.89	0.99	0.99	0.99	03.51
Setup 2 - RFC	99.77	0.99	0.99	0.99	14.32
Setup 2 - NN	99.96	0.99	0.99	0.99	04.20
Setup 3 - RFC	78.38	0.80	0.78	0.78	16.48
Setup 3 - NN	99.74	0.99	0.99	0.99	06.14
Setup 4 - RFC	99.05	0.99	0.99	0.99	06.28
Setup 4 - NN	99.84	0.99	0.99	0.99	02.23

4.5 Conclusions

In this chapter, we presented more complex scenarios using our proposed framework. Although the testbed presented in Chapter 2, Section 2.1, looks limited, we showed that it is possible to easily extend it to more complex scenarios and use the proposed framework. We presented three scenarios that covered most of the possible situations and showed how to use our framework to tackle the identity-matching problem.

Using extended versions of the datasets presented in Chapter 3, we were able to simulate the proposed scenarios. One caveat is that the simulations are limited because our extended datasets do not consider the iteration between users. The results for accuracy, precision, recall, and F_1-score were very similar to the experiments using the original datasets. However, we could conclude that it is possible to use our framework in these more complex cases, even with more users. Because the features are an array of numbers, even with more cameras and devices, the number of training/deployment data does not increase significantly. We have a scalable framework suitable for more complex scenarios.

Capítulo 5

Conclusions & Future Work

5.1 Conclusions

In this dissertation, we described the procedures for integrating a computer vision system with a radio access network through artificial intelligence. This work showcased the identification of the radio transmitter between devices existing in a video feed.

In Chapter 2, we presented the proposed framework for user-identity matching. The setup for a testbed was described, which allowed us to illustrate the identity-matching procedure. Each step of the framework was detailed, from data collection to training and deployment procedures. Our methodology used the bounding boxes of the detected devices in the video feed and matched it with the channel impulse response of the transmitter using a trained classifier.

To test our proposed method, in Chapter 3, we presented practical experiments. We carried out experiments in four location setups with different spatial dynamics. Each setup generated a distinct dataset to test the framework. For each setup, we trained the model using both random forest and neural network classifiers. We presented the validation results for the training of each experiment in terms of accuracy, precision, recall, F_1-score, and confusion matrix. Overall, the validation results showed high score values across the experiments. We also presented two cases of deployment of the testbed. The deployment's success was consistent with the validation results, which showed that our method was capable of accomplishing the identity-matching objective in real-time.

We went beyond the testbed in Chapter 4. The testbed presented in Chapter 2 was limited, with only two users, but we showed in Chapter 4 that the proposed framework is capable of dealing with complex scenarios. Three scenarios were presented, which covered most of the possible situations. Simulations were carried out using extended versions of the datasets of Chapter 3 for each of the scenario. We showed how to use our framework to tackle the identity-matching problem in each case. We also discussed the increase of computational complexity in the scenarios. We concluded that due to the features used in the classifier, more complex situations could be handled by a dedicated computer.

In summary, we showed that by modeling the identity-matching problem as a classification task and using machine learning techniques, we correctly identified the true transmitter in the scene in several different scenarios presented. The proposed framework was shown to be very robust and reliable yet flexible. We showed that it is possible to extend the testbed used here for a proof-of-concept and experiment with more realistic scenarios. We envision our work being applied to private networks of near-future communication systems, beyond 5G, with the proper development of machine learning-dedicated processing units in the network nodes.

In conclusion, the proposed identity-matching methodology presented in this work is an essential first step in the direction of a more context-aware communication system. Our methodology is a building block for integrating different domains. This integration is going to be ubiquitous in the following generations. For this reason, our solution can be used in other projects working with joint technologies.

5.2 Future Work

The proposed framework uses a classifier that only outputs one result at a time. Although our solution seems to work only in simple cases, we showed that it is possible to use it in more complex scenarios. However, it is possible to investigate solutions with multiple outputs at a time. For example, to use deep convolutional networks to output the bounding boxes from the scene's transmitting devices. This method is similar to already commonplace deep-learning-based computer vision applications. The difference is that the network would also incorporate information from the channel impulse response from the devices to make the identity match.

With the identity-matching procedure done, it is possible to supply more information about the scene to the network (e.g., device's speed and direction, warning about possible blockage). This possibility opens paths to explore applications of our methodology. We believe it is possible to expand our communications systems' knowledge by integrating useful data from different domains.

Chapter 4 discussed ideas of sharing information about detected users among different nodes of the network. For future work, it is possible to explore these different configurations. Study how different nodes would share information about detected devices; the benefits and downsides of each configuration. Another case is to explore whether it is advantageous to centralize the classification part or carry it out in the network's different nodes. There is also the need to establish protocols on how the nodes will communicate with each other. Finally, it is necessary to standardize the procedures. In summary, there is a necessity to test, deploy, and further formalize the technologies necessary to achieve the scenarios presented in Chapter 4.

We envision our work being used not only in indoor scenarios. We did experiments with outdoor measurements, showing that there is room to advance in this direction. There are particular problems with using visual information on outdoor spaces (e.g., privacy issues) that need to be addressed. However, there are a plethora of applications possible, from vehicular communications, on vehicle-to-vehicle to vehicle-to-everything. These communication systems can take advantage of existing infrastructure present on modern cars, such as cameras and proximity sensors, to incorporate more knowledge into the communication systems.

We hope that our work will be further explored and expanded to be extensively employed in future generations beyond 5G. Based on the ideas of ever-faster, more reliable connections, and broader bandwidth available, we will see more knowledge about the environment gathered into the network. Future communications systems will not rely solely on radio signals, but the aggregation of different domains is going to be ubiquitous.

Referências Bibliográficas

[1] GUPTA, A., JHA, R. K. "A Survey of 5G Network: Architecture and Emerging Technologies", *IEEE Access*, v. 3, pp. 1206–1232, Jul 2015.

[2] AGIWAL, M., ROY, A., SAXENA, N. "Next Generation 5G Wireless Networks: A Comprehensive Survey", *IEEE Communications Surveys Tutorials*, v. 18, n. 3, pp. 1617–1655, Feb 2016.

[3] COTE, D. "Using machine learning in communication networks [Invited]", *IEEE/OSA Journal of Optical Communications and Networking*, v. 10, n. 10, pp. D100–D109, Oct 2018.

[4] SHARMA, S. K., WANG, X. "Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions", *IEEE Communications Surveys Tutorials*, v. 22, n. 1, pp. 426–471, May 2020.

[5] MATEI, A., GLAVAN, A., TALAVERA, E. "Deep Learning for Scene Recognition from Visual Data: A Survey". 2020. Disponível em: <`https://arxiv.org/abs/2007.01806`>.

[6] LIU, Z., CHEN, L., ET AL., L. T. "Deep Learning Based Brain Tumor Segmentation: A Survey". 2020. Disponível em: <`https://arxiv.org/abs/2007.09479`>.

[7] TIAN, Y., PAN, G., ALOUINI, M.-S. "Applying Deep-Learning-Based Computer Vision to Wireless Communications: Methodologies, Opportunities, and Challenges". 2020. Disponível em: <`arxiv.org/abs/2006.05782`>.

[8] O'SHEA, T., HOYDIS, J. "An Introduction to Deep Learning for the Physical Layer", *IEEE Transactions on Cognitive Communications and Networking*, v. 3, n. 4, pp. 563–575, Dec 2017.

[9] SIMEONE, O. "A Very Brief Introduction to Machine Learning With Applications to Communication Systems", *IEEE Transactions on Cognitive Communications and Networking*, v. 4, n. 4, pp. 648–664, Nov 2018.

[10] ZHANG, C., PATRAS, P., HADDADI, H. “Deep Learning in Mobile and Wireless Networking: A Survey”, *IEEE Communications Surveys Tutorials*, v. 21, n. 3, pp. 2224–2287, Mar 2019.

[11] ZHANG, C., UENG, Y., STUDER, C., et al. “Artificial Intelligence for 5G and Beyond 5G: Implementations, Algorithms, and Optimizations”, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, v. 10, n. 2, pp. 149–163, Jun 2020.

[12] VALCARCE, A., HOYDIS, J. “Towards Joint Learning of Optimal Signaling and Wireless Channel Access”. 2020. Disponível em: <`https://arxiv.org/abs/2007.09948`>.

[13] GOUTAY, M., AIT AOUDIA, F., HOYDIS, J. “Deep HyperNetwork-Based MIMO Detection”. In: *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, Atlanta, May 2020.

[14] KHANI, M., ALIZADEH, M., HOYDIS, J., et al. “Adaptive Neural Signal Detection for Massive MIMO”. 2019. Disponível em: <`https://arxiv.org/abs/1906.04610`>.

[15] CAMPOS, R. S., LOVISOLO, L., DE CAMPOS, M. L. R. “Wi-Fi multi-floor indoor positioning considering architectural aspects and controlled computational complexity”, *Expert Systems with Applications*, v. 41, n. 14, pp. 6211–6223, Oct 2014.

[16] BUTT, M. M., RAO, A., YOON, D. “RF Fingerprinting and Deep Learning Assisted UE Positioning in 5G”. In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–7, Antwerp, May 2020.

[17] ALRABEIAH, M., HREDZAK, A., LIU, Z., et al. “ViWi: A Deep Learning Dataset Framework for Vision-Aided Wireless Communications”. In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, Antwerp, May 2020.

[18] ALRABEIAH, M., HREDZAK, A., ALKHATEEB, A. “Millimeter Wave Base Stations with Cameras: Vision-Aided Beam and Blockage Prediction”. In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, Antwerp, May 2020.

[19] CHARAN, G., ALRABEIAH, M., ALKHATEEB, A. “Vision-Aided Dynamic Blockage Prediction for 6G Wireless Communication Networks”. 2020. Disponível em: <`https://arxiv.org/abs/2006.09902`>.

[20] DE PINHO, V. M., POPESCU, D. “User identification by matching radio ”vision” and computer vision through means of machine learning”. In: *2020 IFIP Networking Conference (Networking)*, pp. 671–672, Paris, June 2020.

[21] DE PINHO, V. M., DE CAMPOS, M., GARCIA, L., et al. “Vision-Aided Radio: User Identity Match in Radio and Video Domains Using Machine Learning”, *IEEE Access*, v. 8, pp. 1–11, Nov 2020.

[22] SINGH, S. K., SINGH, R., KUMBHANI, B. “The Evolution of Radio Access Network Towards Open-RAN: Challenges and Opportunities”. In: *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, Seoul, Apr 2020.

[23] NIKNAM, S., ROY, A., DHILLON, H. S., et al. “Intelligent O-RAN for Beyond 5G and 6G Wireless Networks”. 2020. Disponível em: <`https://arxiv.org/abs/2005.08374`>.

[24] FOND., G. R. “GNU Radio”. 2020. Disponível em: <`gnuradio.org`>.

[25] HEISKALA, J., TERRY, J. *OFDM Wireless LANs: A Theoretical and Practical Guide*. 1 ed. USA, Sams, 2001.

[26] WU, Y., ET AL. “Detectron2”. 2019. Disponível em: <`github.com/facebookresearch/detectron2`>.

[27] LIN, T.-Y., MAIRE, M., BELONGIE, S., et al. “Microsoft COCO: Common Objects in Context”. 2015. Disponível em: <`https://arxiv.org/abs/1405.0312`>.

[28] DENISKO, D., HOFFMAN, M. M. “Classification and interaction in random forests”, *Proceedings of the National Academy of Sciences*, v. 115, n. 8, pp. 1690–1692, Feb 2018.

[29] BREIMAN, L. “Random Forests”, *Mach. Learn.*, v. 45, n. 1, pp. 5–32, Oct 2001.

[30] AGARAP, A. F. “Deep Learning using Rectified Linear Units (ReLU)”. 2019. Disponível em: <`https://arxiv.org/abs/1803.08375`>.

[31] HINTON, G. E., SRIVASTAVA, N., KRIZHEVSKY, A., et al. “Improving neural networks by preventing co-adaptation of feature detectors”. 2012. Disponível em: <`https://arxiv.org/abs/1207.0580`>.

[32] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, v. 15, n. 56, pp. 1929–1658, Jan 2014.

[33] AUELIEN, G. *Hands-on machine learning with Sckit-learn, Keras and Tensorflow: concepts, tools and techniques to build intelligent system.* 1 ed. Newton, OReilly, 2019.

[34] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning.* 1 ed. Cambridge, MIT Press, 2016.

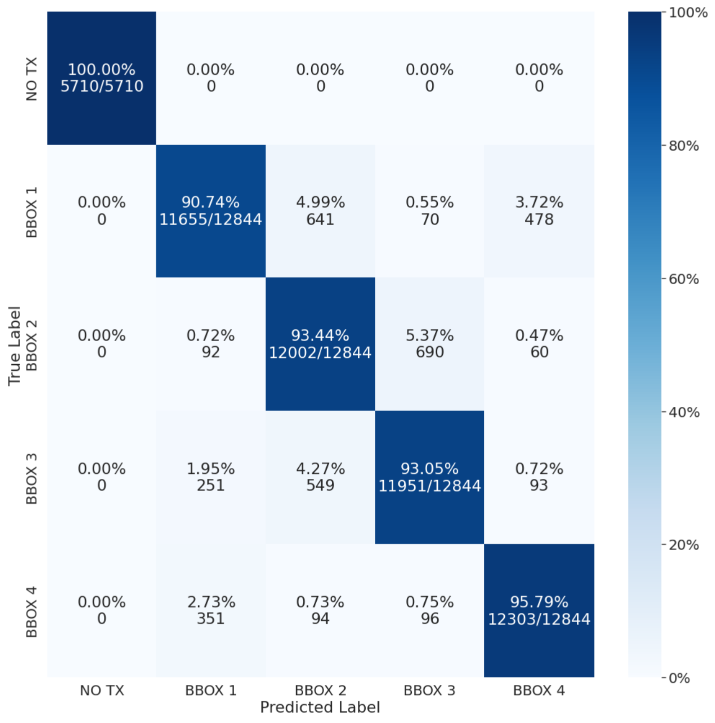
[35] THARWAT, A. "Classification assessment methods", *Applied Computing and Informatics*, Aug 2018.

Apêndice A

Confusion Matrices from "Beyond the Testbed" Simulations

A.1 Scenario 1

A.1.1 S_1 for Setup 1



True \ Predicted	NO TX	BBOX 1	BBOX 2	BBOX 3	BBOX 4
NO TX	100.00% 5710/5710	0.00% 0	0.00% 0	0.00% 0	0.00% 0
BBOX 1	0.00% 0	90.74% 11655/12844	4.99% 641	0.55% 70	3.72% 478
BBOX 2	0.00% 0	0.72% 92	93.44% 12002/12844	5.37% 690	0.47% 60
BBOX 3	0.00% 0	1.95% 251	4.27% 549	93.05% 11951/12844	0.72% 93
BBOX 4	0.00% 0	2.73% 351	0.73% 94	0.75% 96	95.79% 12303/12844

Figura A.1: Confusion Matrix for S_1 in Scenario 1 for Setup 1 trained with Random Forest Classifier.

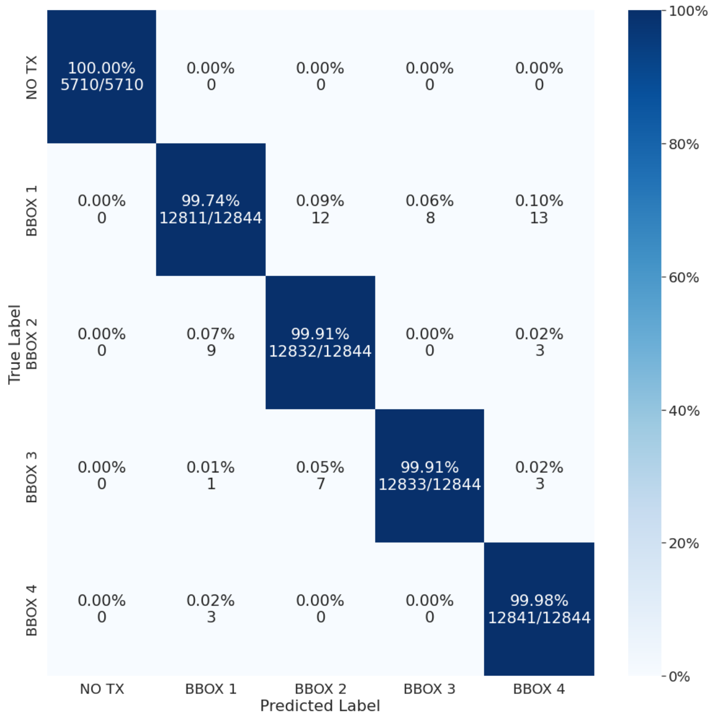


Figura A.2: Confusion Matrix for S_1 in Scenario 1 for Setup 1 trained with Neural Network Classifier.

A.1.2 S_1 for Setup 2

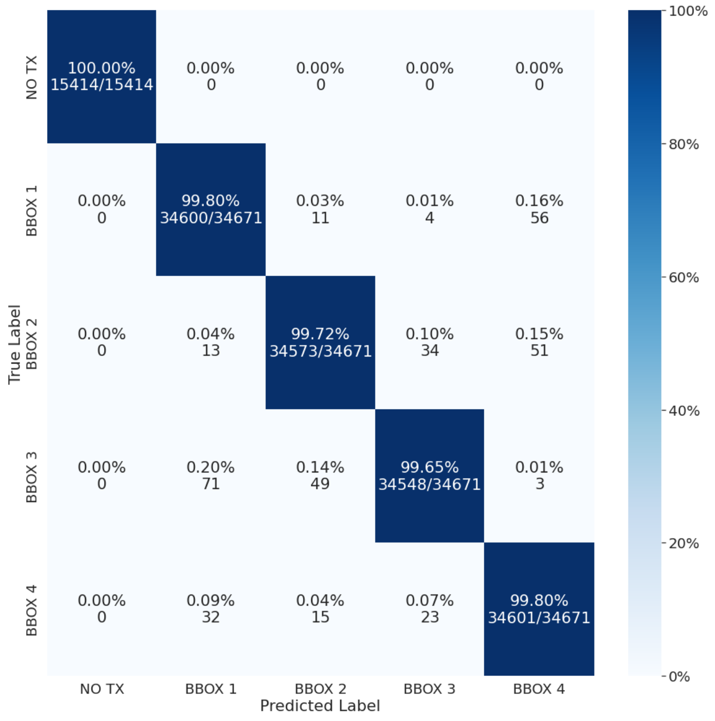


Figura A.3: Confusion Matrix for S_1 in Scenario 1 for Setup 2 trained with Random Forest Classifier.

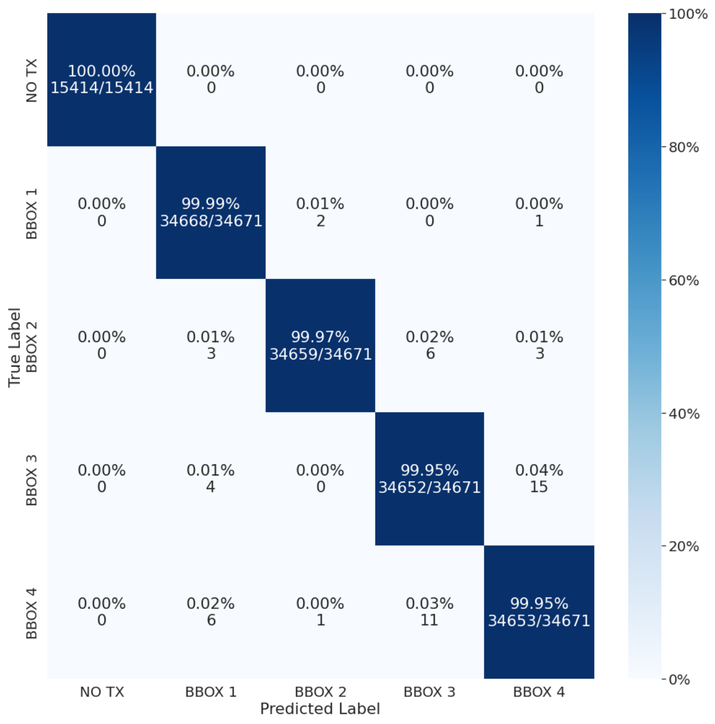


Figura A.4: Confusion Matrix for S_1 in Scenario 1 for Setup 2 trained with Neural Network Classifier.

A.1.3 S_1 for Setup 3

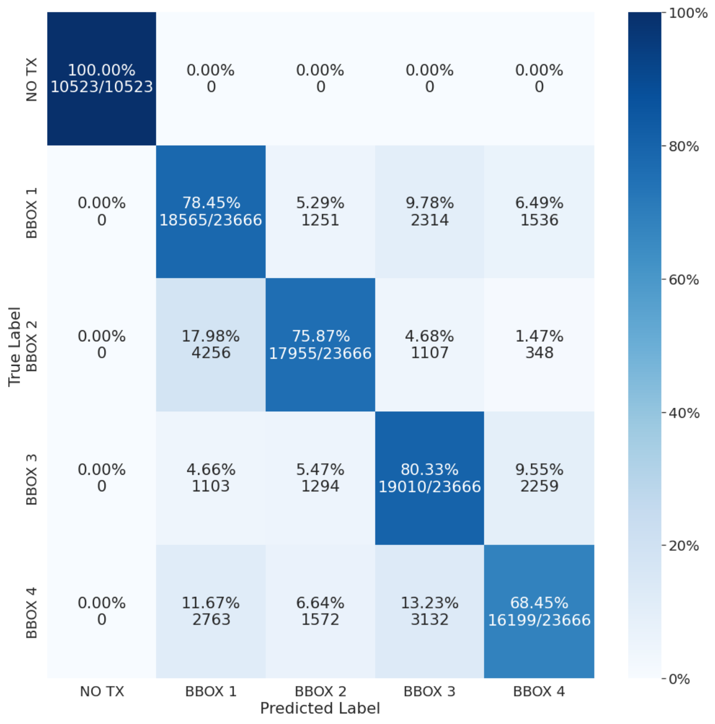


Figura A.5: Confusion Matrix for S_1 in Scenario 1 for Setup 3 trained with Random Forest Classifier.

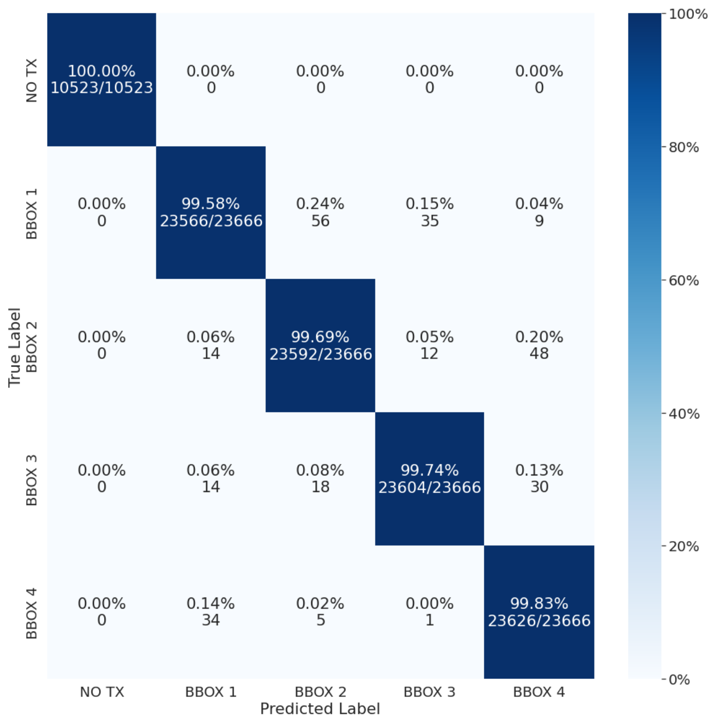


Figura A.6: Confusion Matrix for S_1 in Scenario 1 for Setup 3 trained with Neural Network Classifier.

A.1.4 S_1 for Setup 4

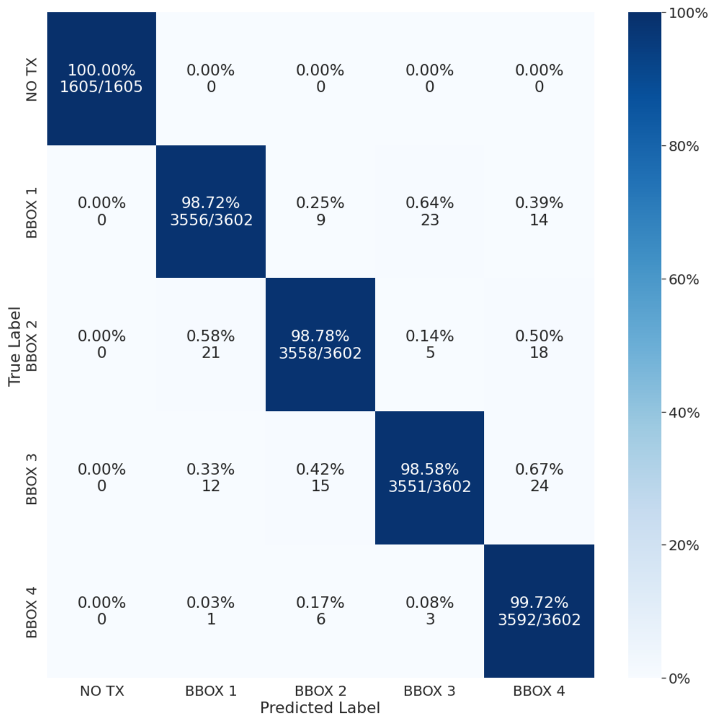


Figura A.7: Confusion Matrix for S_1 in Scenario 1 for Setup 4 trained with Random Forest Classifier.

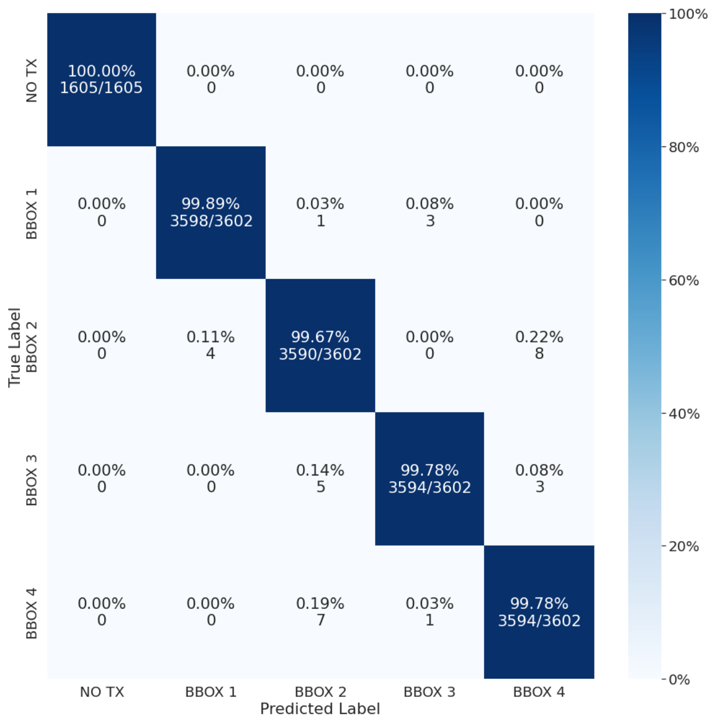


Figura A.8: Confusion Matrix for S_1 in Scenario 1 for Setup 4 trained with Neural Network Classifier.

A.1.5 S_2 for Setup 1

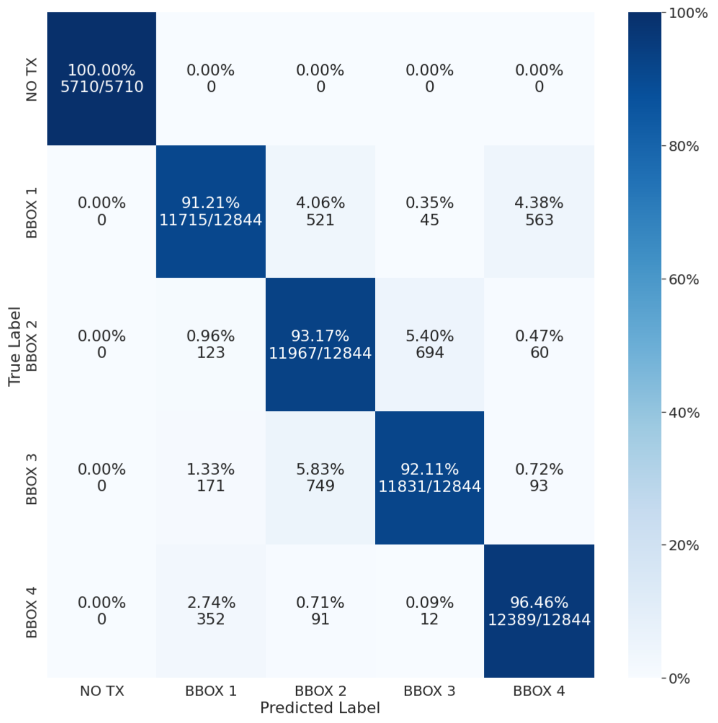


Figura A.9: Confusion Matrix for S_2 in Scenario 1 for Setup 1 trained with Random Forest Classifier.

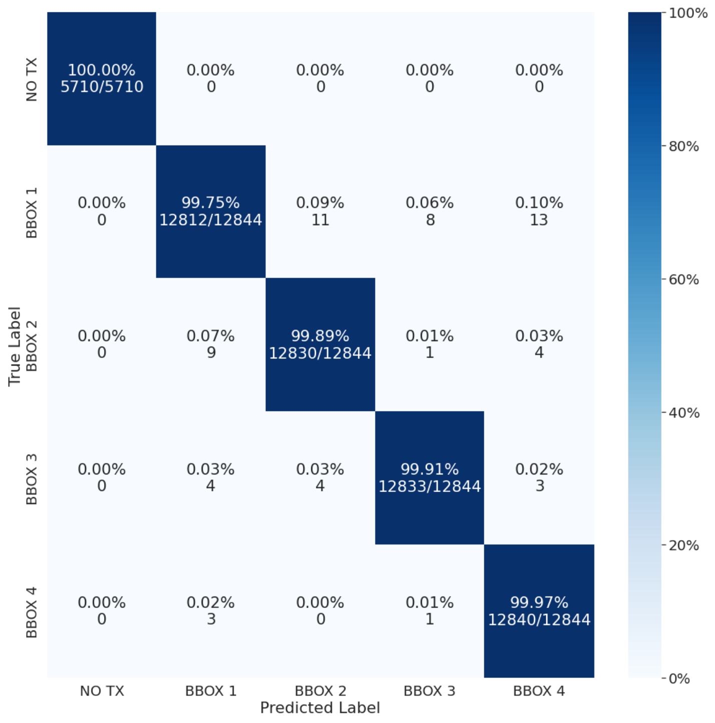


Figura A.10: Confusion Matrix for S_2 in Scenario 1 for Setup 1 trained with Neural Network Classifier.

A.1.6 S_2 for Setup 2

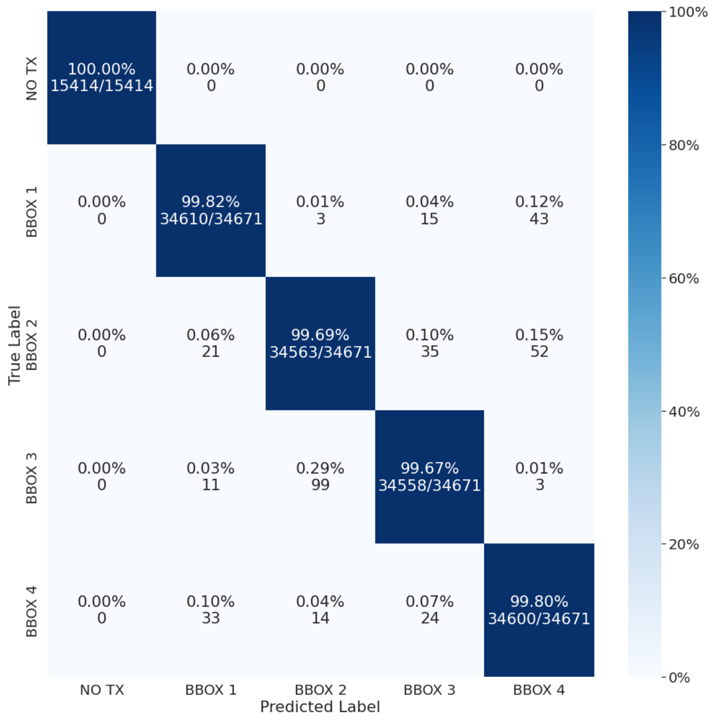
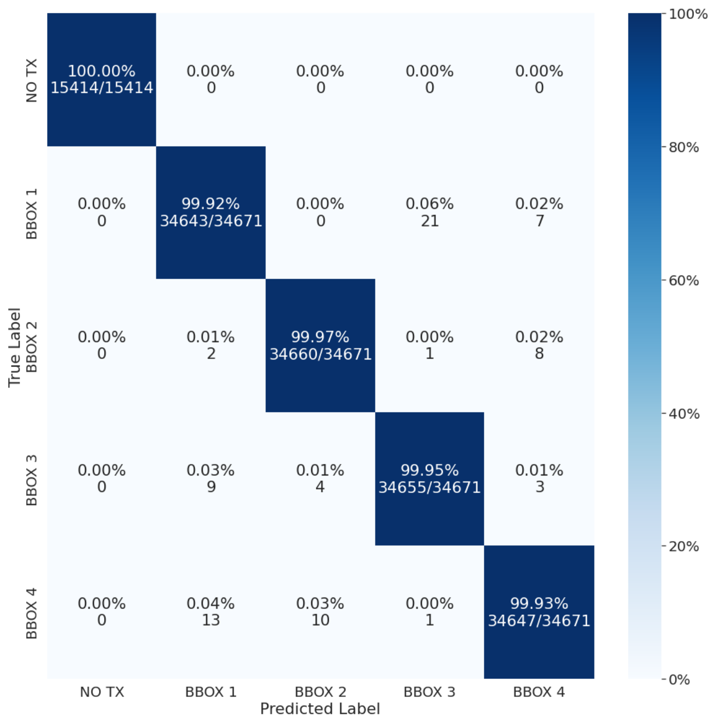


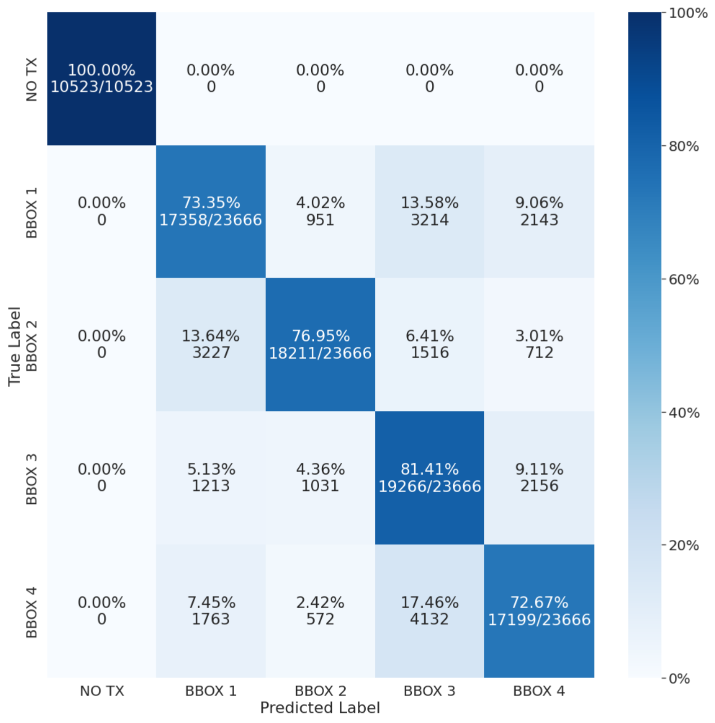
Figura A.11: Confusion Matrix for S_2 in Scenario 1 for Setup 2 trained with Random Forest Classifier.



	NO TX	BBOX 1	BBOX 2	BBOX 3	BBOX 4
NO TX	100.00% 15414/15414	0.00% 0	0.00% 0	0.00% 0	0.00% 0
BBOX 1	0.00% 0	99.92% 34643/34671	0.00% 0	0.06% 21	0.02% 7
BBOX 2	0.00% 0	0.01% 2	99.97% 34660/34671	0.00% 1	0.02% 8
BBOX 3	0.00% 0	0.03% 9	0.01% 4	99.95% 34655/34671	0.01% 3
BBOX 4	0.00% 0	0.04% 13	0.03% 10	0.00% 1	99.93% 34647/34671

Figura A.12: Confusion Matrix for S_2 in Scenario 1 for Setup 2 trained with Neural Network Classifier.

A.1.7 S_2 for Setup 3



	NO TX	BBOX 1	BBOX 2	BBOX 3	BBOX 4
NO TX	100.00% 10523/10523	0.00% 0	0.00% 0	0.00% 0	0.00% 0
BBOX 1	0.00% 0	73.35% 17358/23666	4.02% 951	13.58% 3214	9.06% 2143
BBOX 2	0.00% 0	13.64% 3227	76.95% 18211/23666	6.41% 1516	3.01% 712
BBOX 3	0.00% 0	5.13% 1213	4.36% 1031	81.41% 19266/23666	9.11% 2156
BBOX 4	0.00% 0	7.45% 1763	2.42% 572	17.46% 4132	72.67% 17199/23666

Figura A.13: Confusion Matrix for S_2 in Scenario 1 for Setup 3 trained with Random Forest Classifier.

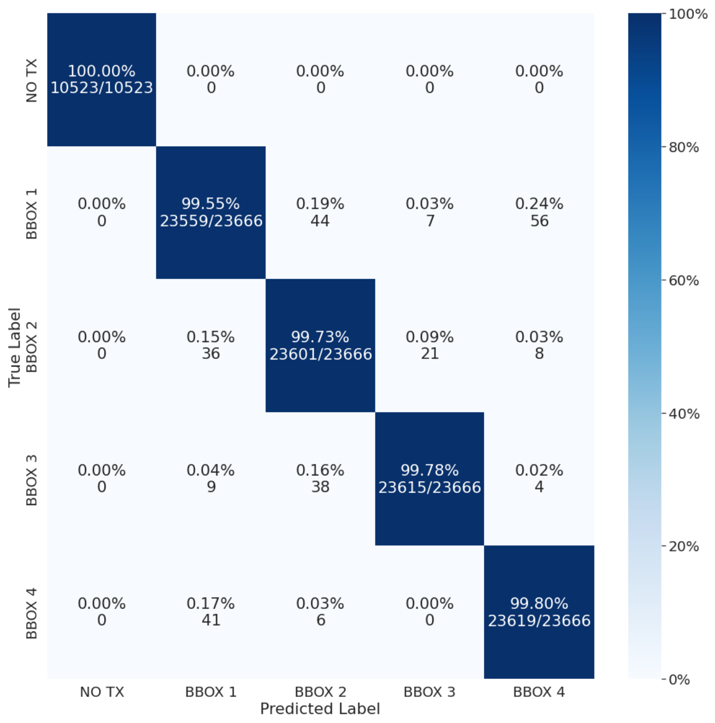


Figura A.14: Confusion Matrix for S_2 in Scenario 1 for Setup 3 trained with Neural Network Classifier.

A.1.8 S_2 for Setup 4

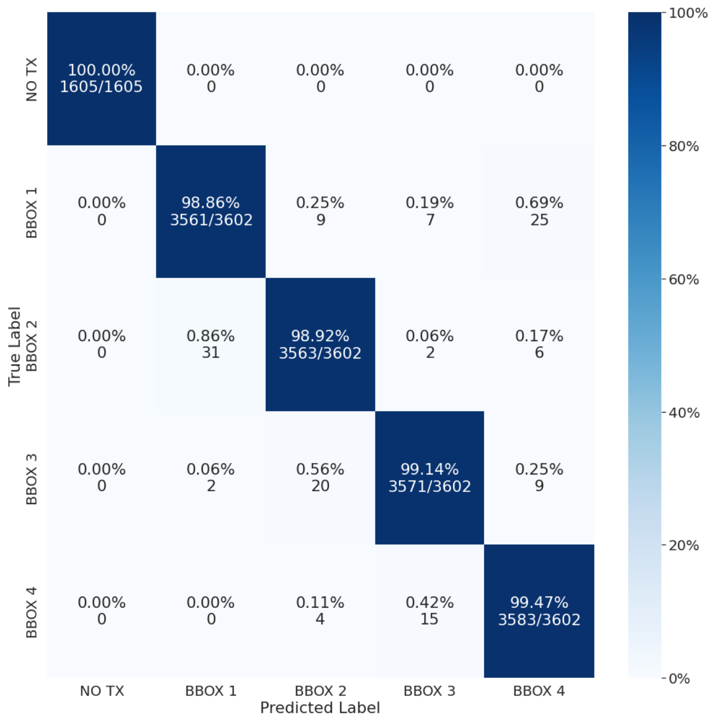


Figura A.15: Confusion Matrix for S_2 in Scenario 1 for Setup 4 trained with Random Forest Classifier.

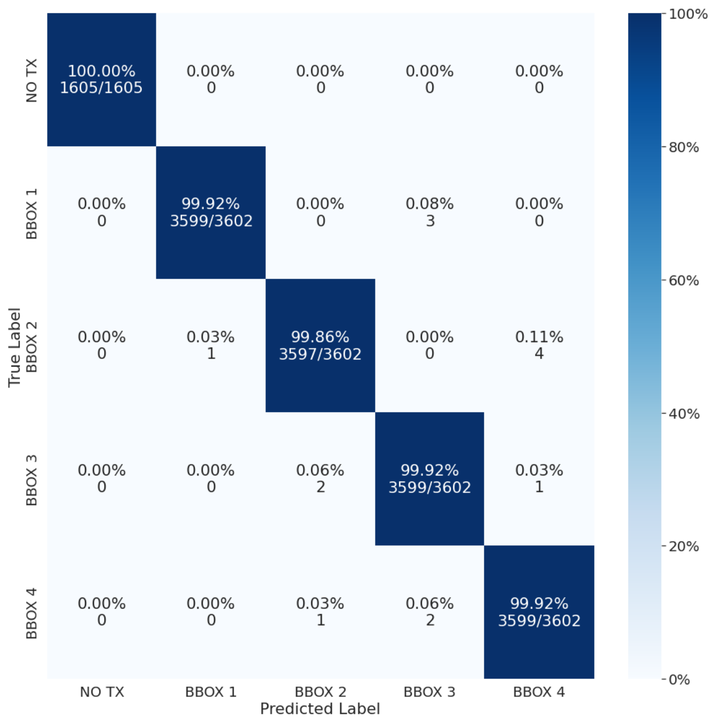


Figura A.16: Confusion Matrix for S_2 in Scenario 1 for Setup 4 trained with Neural Network Classifier.

A.1.9 $S_{1,2}$ for Setup 1

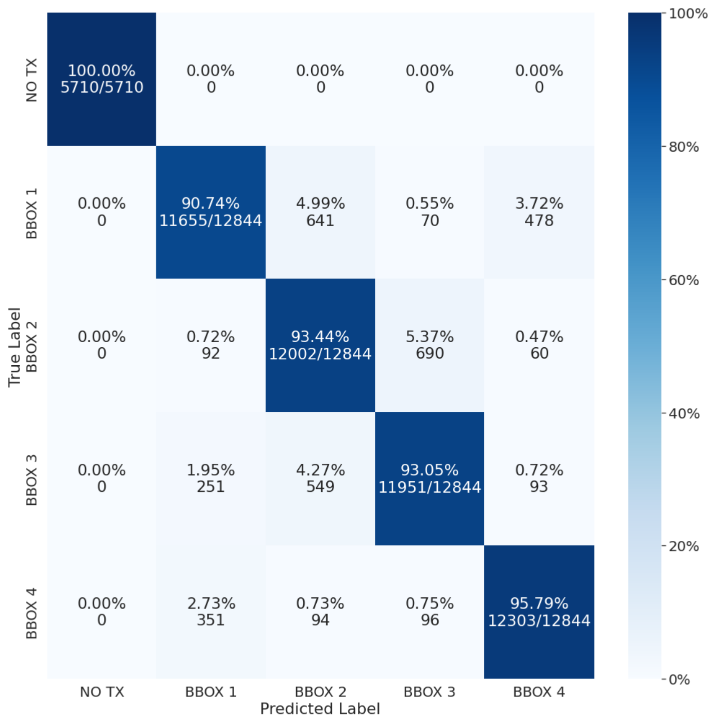


Figura A.17: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 1 trained with Random Forest Classifier.

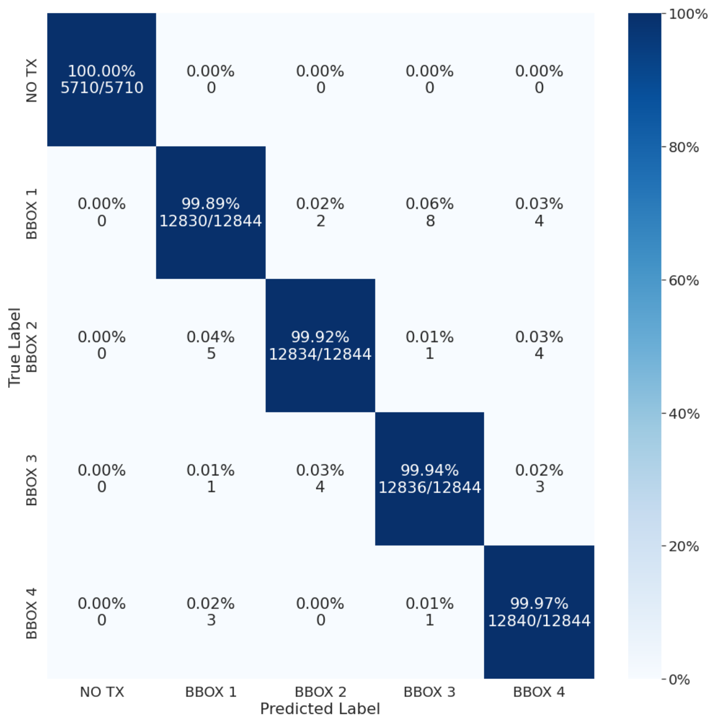


Figura A.18: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 1 trained with Neural Network Classifier.

A.1.10 $S_{1,2}$ for Setup 2

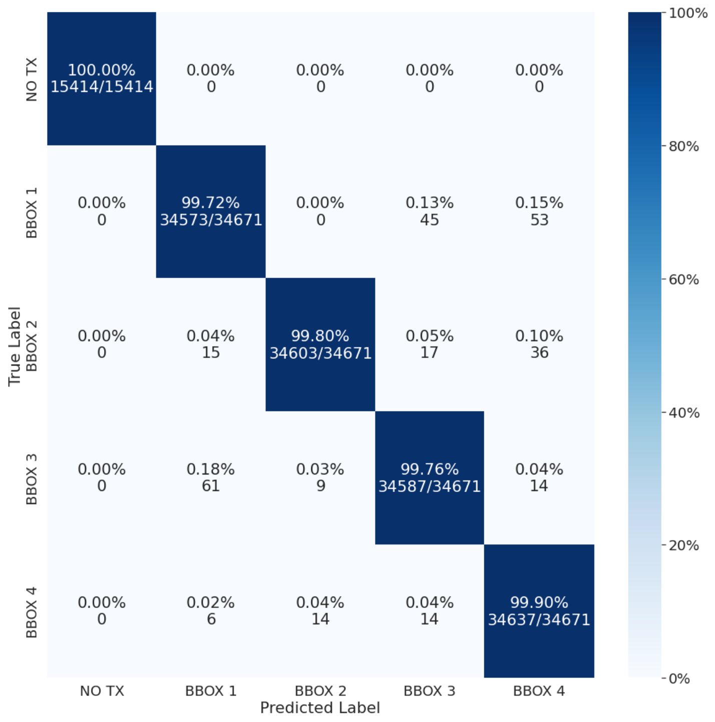


Figura A.19: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 2 trained with Random Forest Classifier.

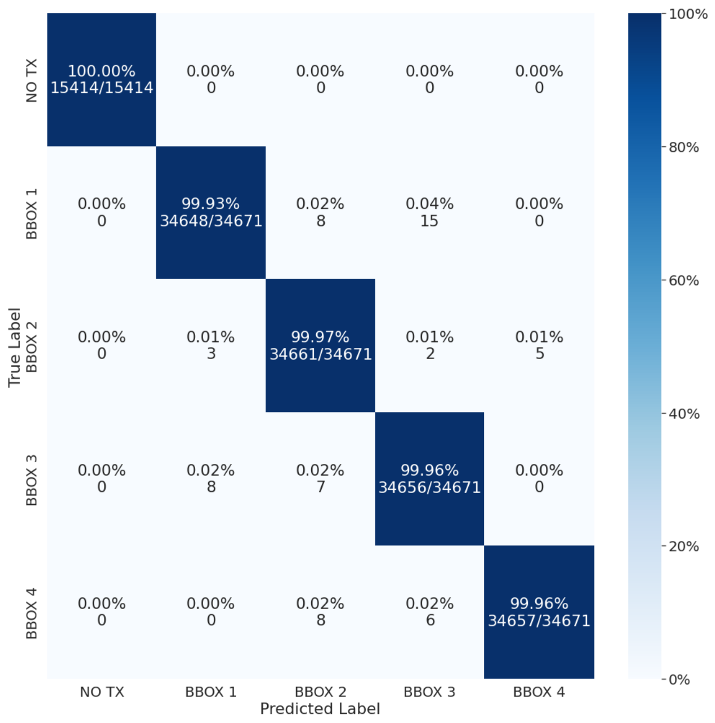


Figura A.20: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 2 trained with Neural Network Classifier.

A.1.11 $S_{1,2}$ for Setup 3

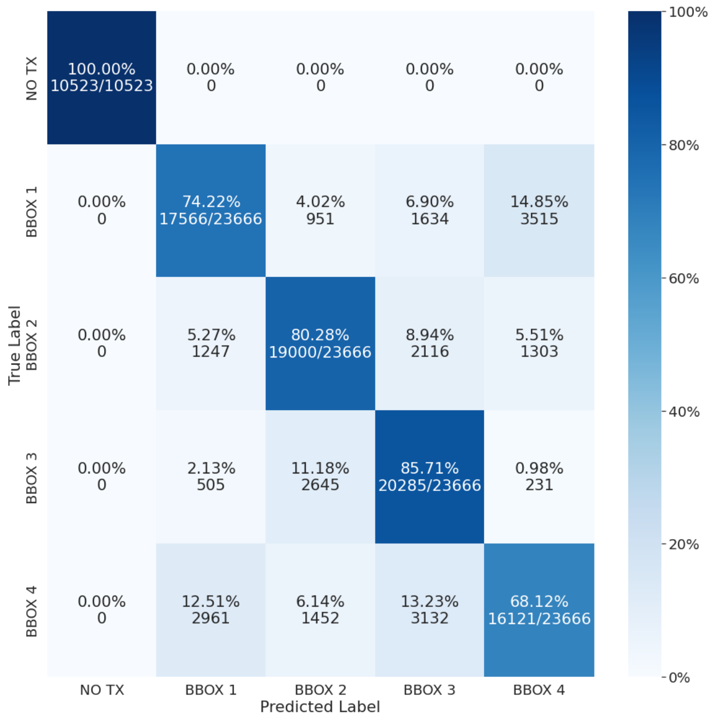


Figura A.21: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 3 trained with Random Forest Classifier.

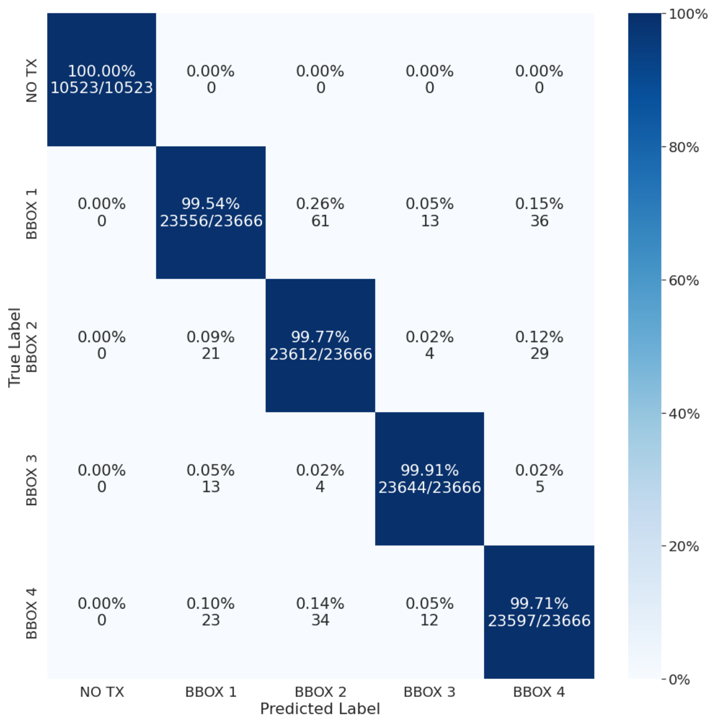


Figura A.22: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 3 trained with Neural Network Classifier.

A.1.12 $S_{1,2}$ for Setup 4

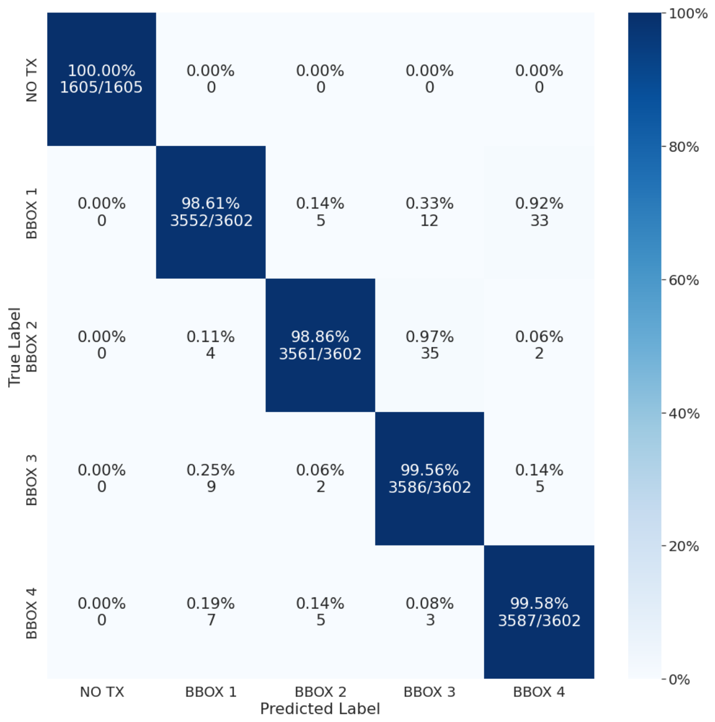


Figura A.23: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 4 trained with Random Forest Classifier.

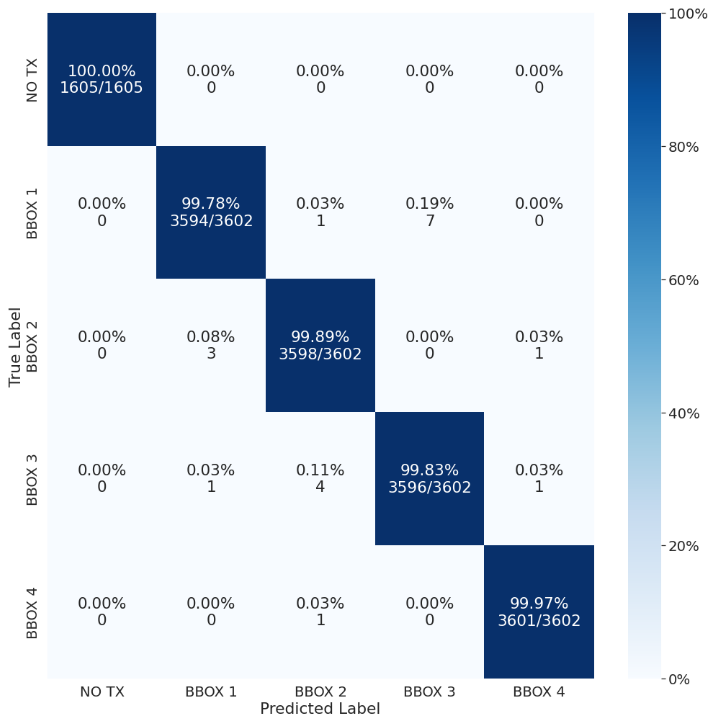


Figura A.24: Confusion Matrix for $S_{1,2}$ in Scenario 1 for Setup 4 trained with Neural Network Classifier.

A.2 Scenario 2

A.2.1 CIR$_1$ for Setup 1

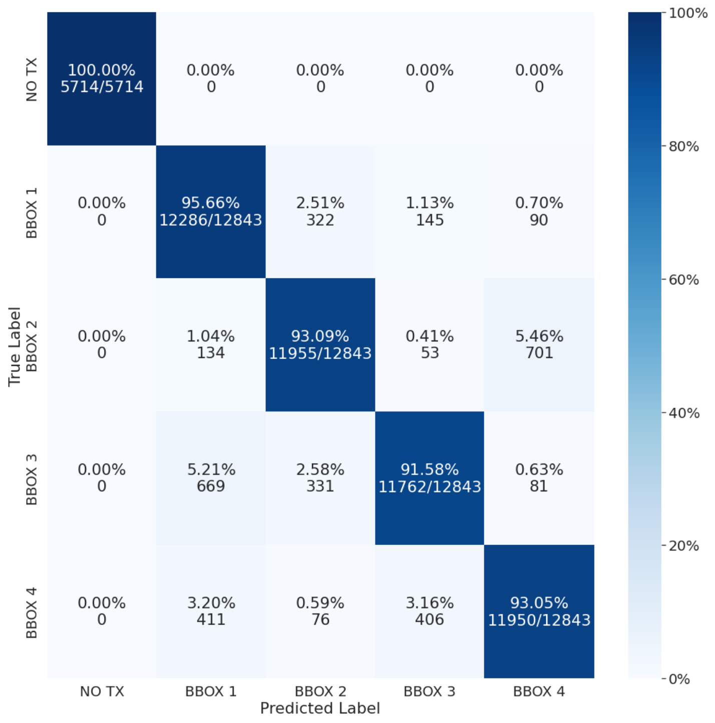
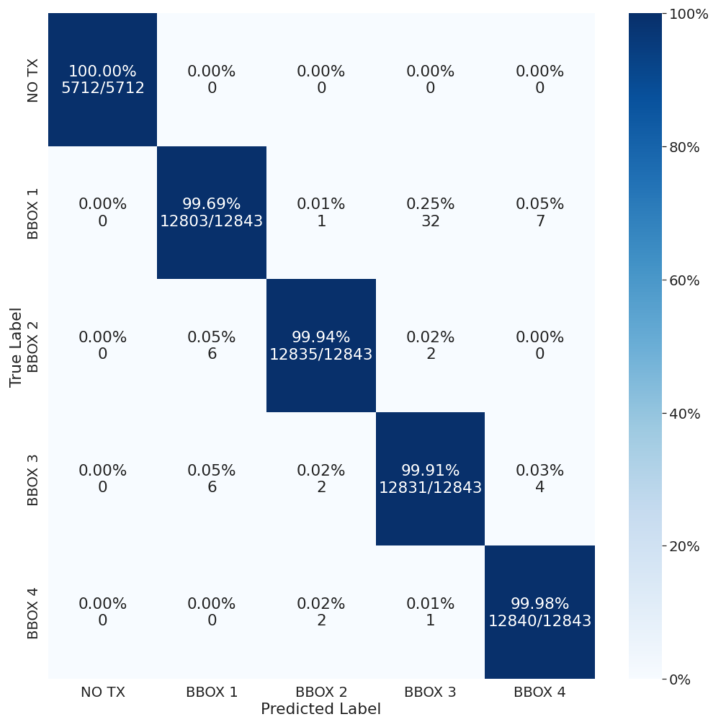


Figura A.25: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 1 trained with Random Forest Classifier.



S_1

Figura A.26: Confusion Matrix for CIR_1 in Scenario 2 for Setup 1 trained with Neural Network Classifier.

A.2.2 CIR_1 for Setup 2

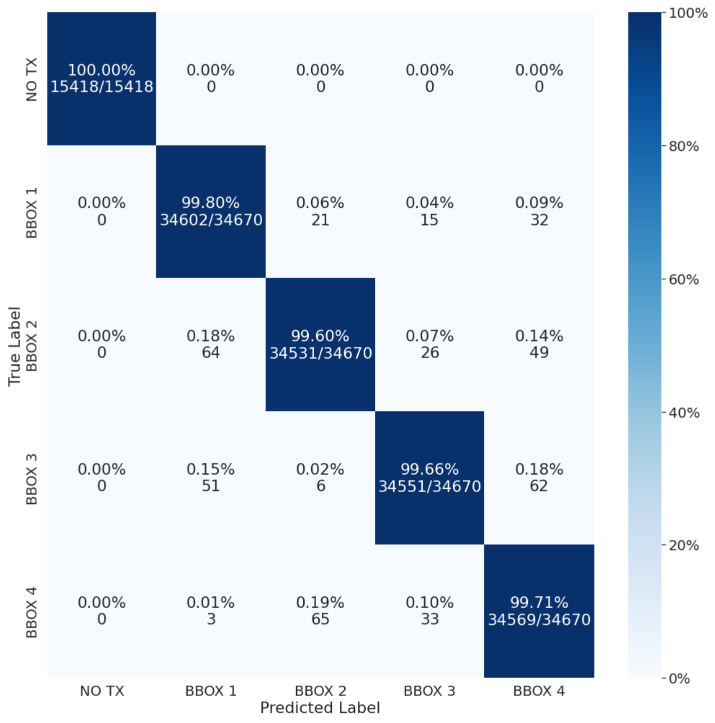


Figura A.27: Confusion Matrix for CIR_1 in Scenario 2 for Setup 2 trained with Random Forest Classifier.

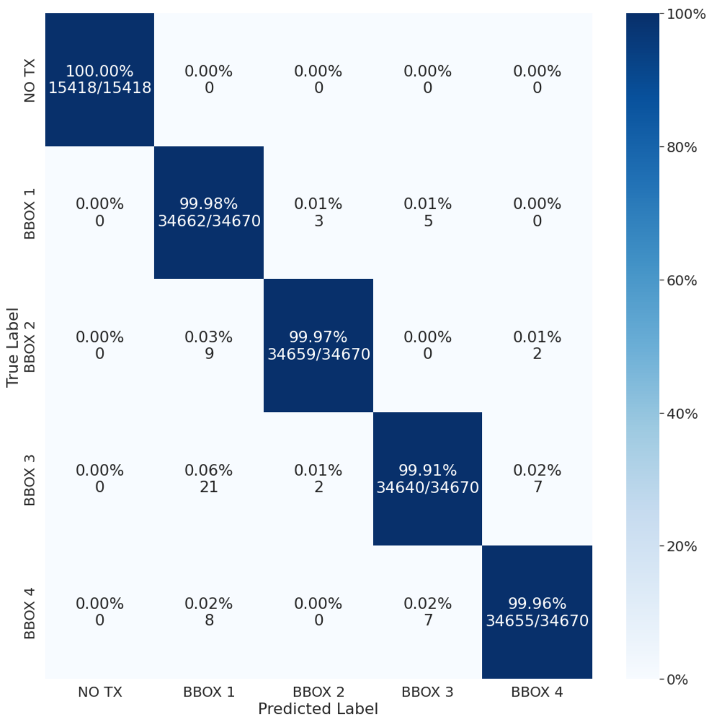


Figura A.28: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 2 trained with Neural Network Classifier.

A.2.3 CIR$_1$ for Setup 3

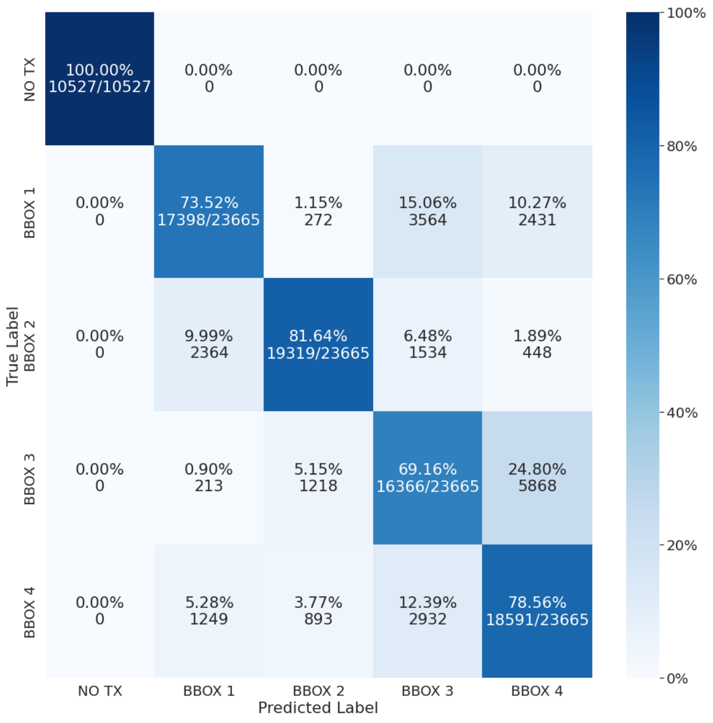


Figura A.29: Confusion Matrix for CIR$_1$ in Scenario 1 for Setup 3 trained with Random Forest Classifier.

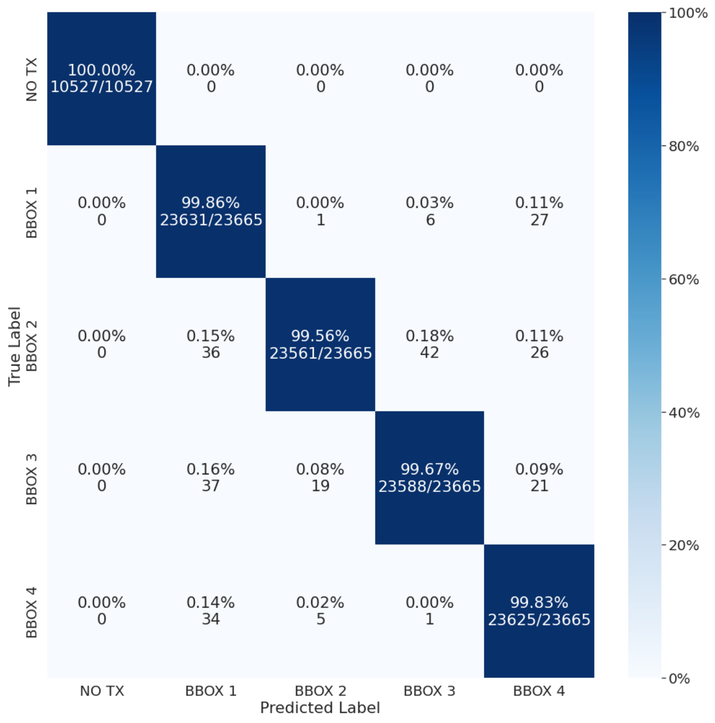


Figura A.30: Confusion Matrix for CIR_1 in Scenario 2 for Setup 3 trained with Neural Network Classifier.

A.2.4 CIR_1 for Setup 4

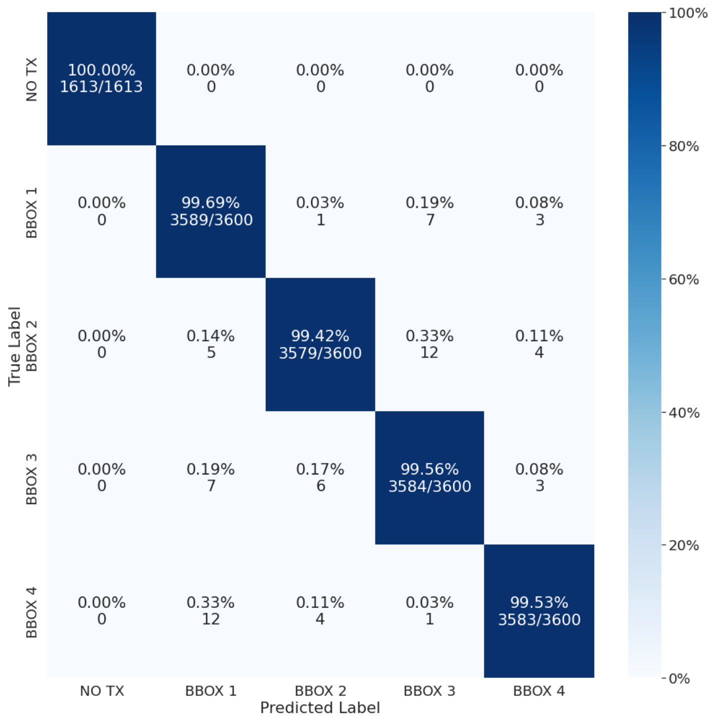


Figura A.31: Confusion Matrix for CIR_1 in Scenario 2 for Setup 4 trained with Random Forest Classifier.

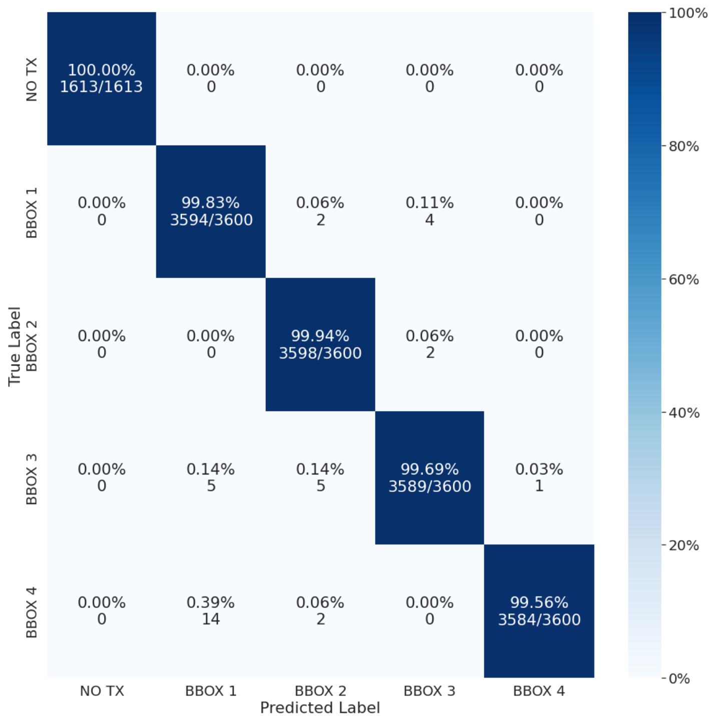


Figura A.32: Confusion Matrix for CIR_1 in Scenario 2 for Setup 4 trained with Neural Network Classifier.

A.2.5 CIR_2 for Setup 1

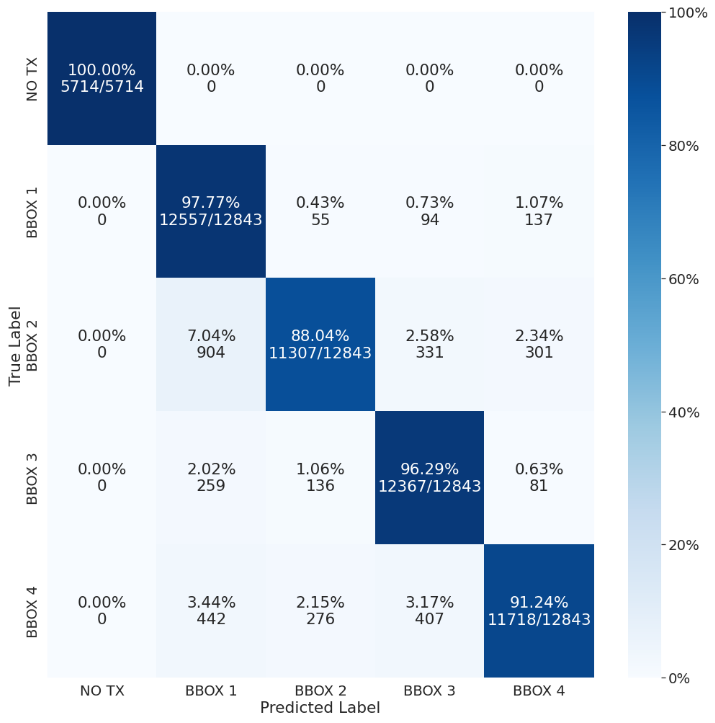
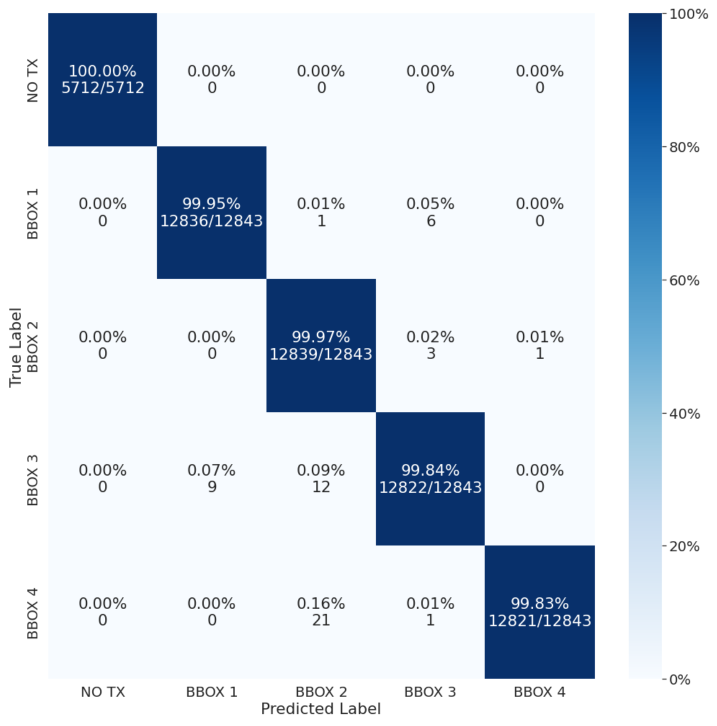


Figura A.33: Confusion Matrix for CIR_2 in Scenario 2 for Setup 1 trained with Random Forest Classifier.



S_1

Figura A.34: Confusion Matrix for CIR_2 in Scenario 2 for Setup 1 trained with Neural Network Classifier.

A.2.6 CIR_2 for Setup 2

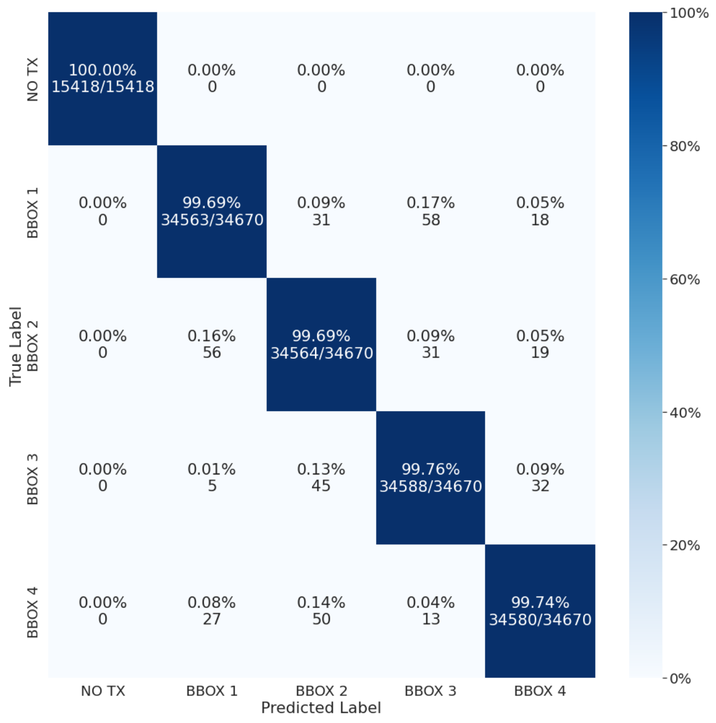


Figura A.35: Confusion Matrix for CIR_2 in Scenario 2 for Setup 2 trained with Random Forest Classifier.

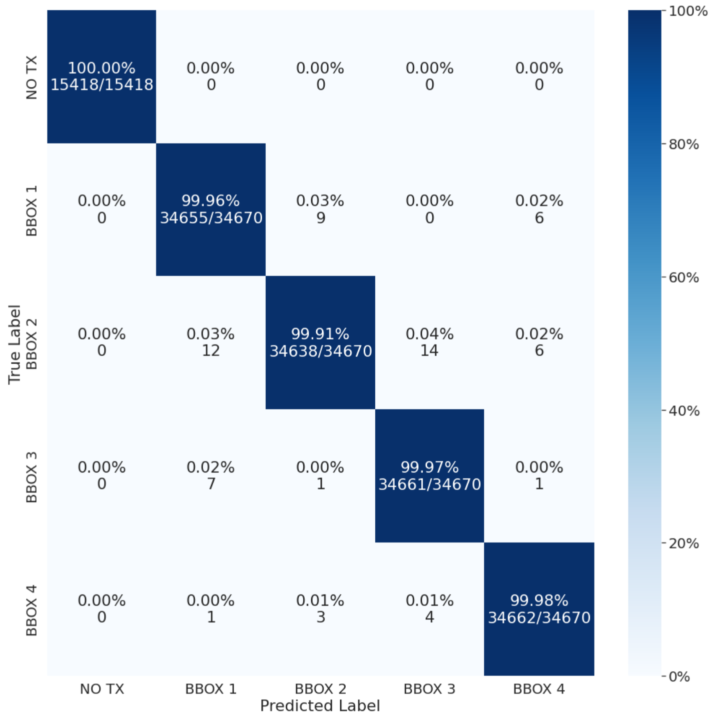


Figura A.36: Confusion Matrix for CIR$_2$ in Scenario 2 for Setup 2 trained with Neural Network Classifier.

A.2.7 CIR$_2$ for Setup 3

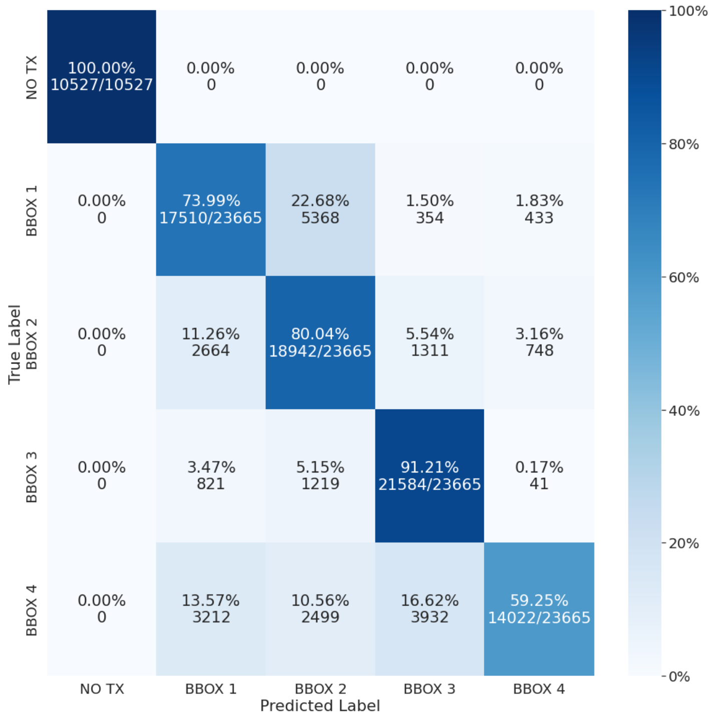


Figura A.37: Confusion Matrix for CIR$_2$ in Scenario 1 for Setup 3 trained with Random Forest Classifier.

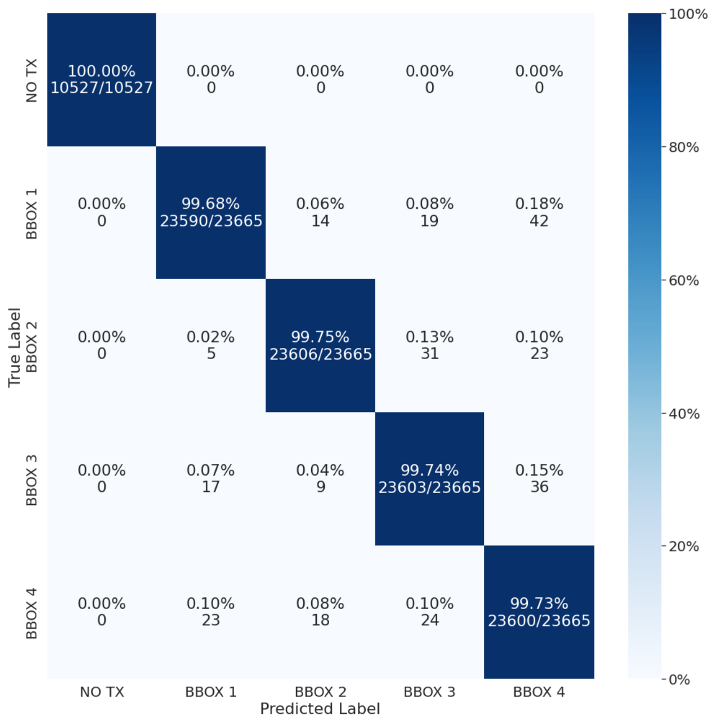


Figura A.38: Confusion Matrix for CIR_2 in Scenario 2 for Setup 3 trained with Neural Network Classifier.

A.2.8 CIR_2 for Setup 4

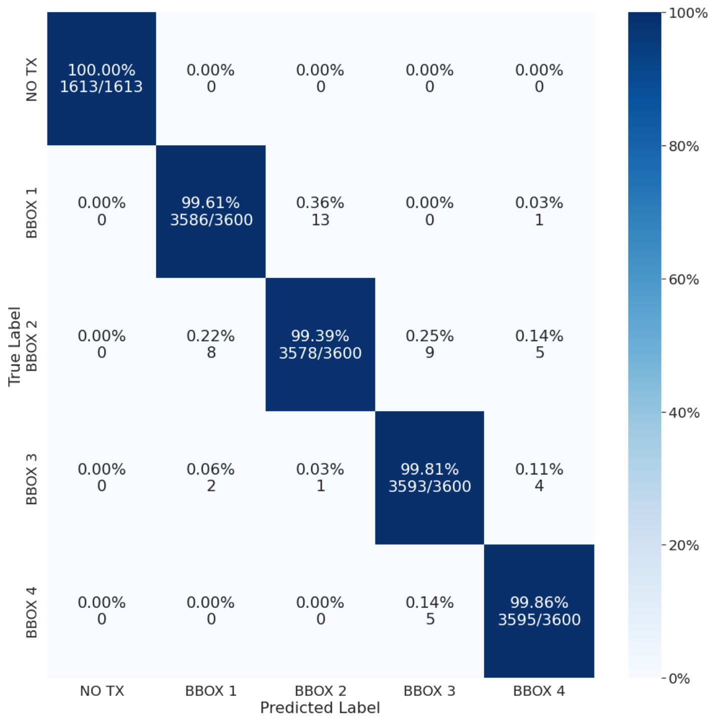


Figura A.39: Confusion Matrix for CIR_2 in Scenario 2 for Setup 4 trained with Random Forest Classifier.

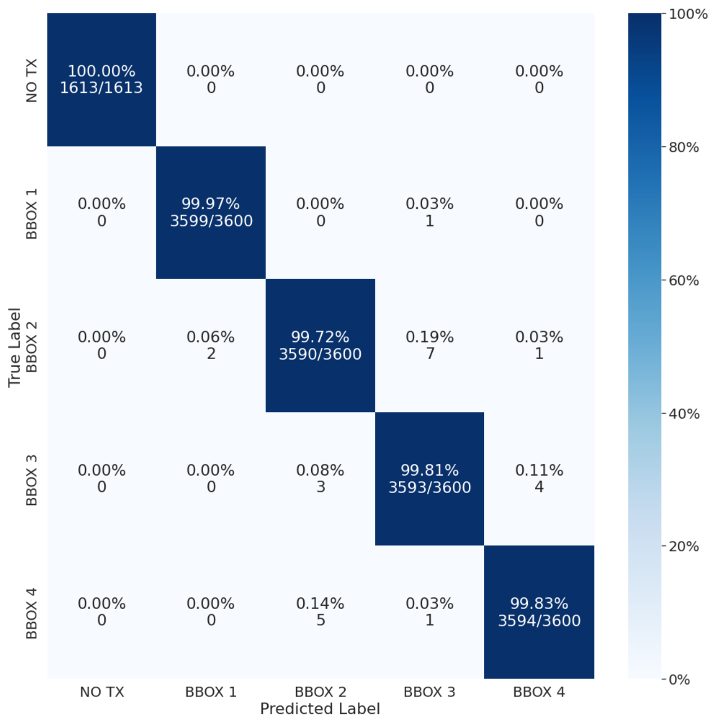


Figura A.40: Confusion Matrix for CIR_2 in Scenario 2 for Setup 4 trained with Neural Network Classifier.

A.2.9 CIR_3 for Setup 1

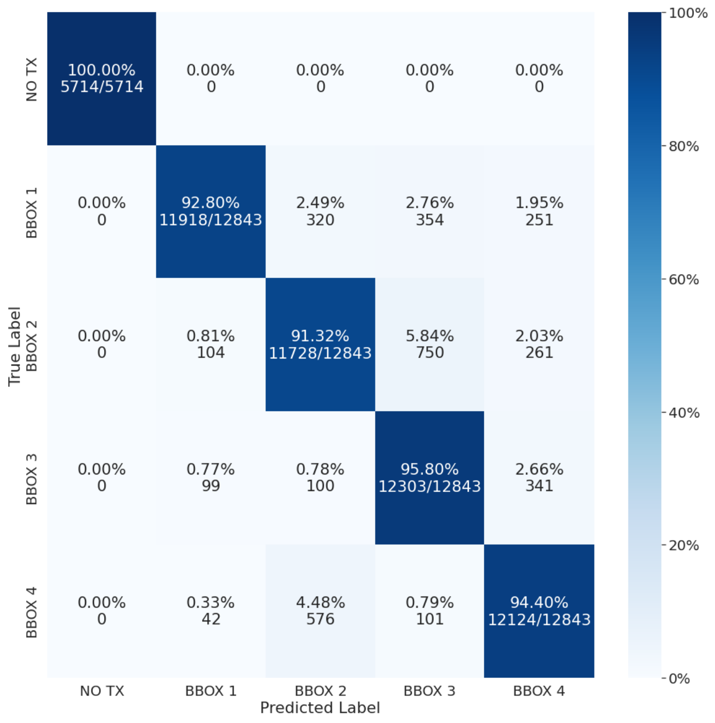
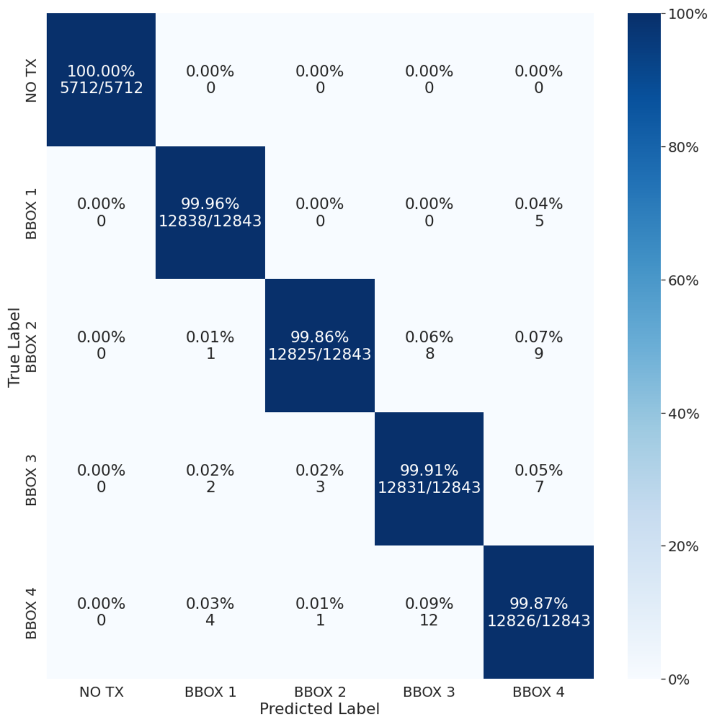


Figura A.41: Confusion Matrix for CIR_3 in Scenario 2 for Setup 1 trained with Random Forest Classifier.



S_1

Figura A.42: Confusion Matrix for CIR$_3$ in Scenario 2 for Setup 1 trained with Neural Network Classifier.

A.2.10 CIR$_3$ for Setup 2

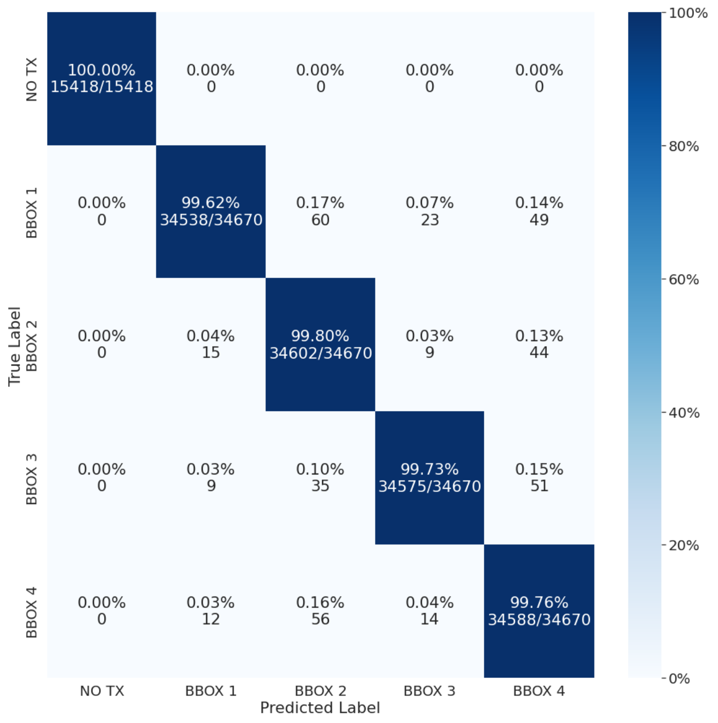


Figura A.43: Confusion Matrix for CIR$_3$ in Scenario 2 for Setup 2 trained with Random Forest Classifier.

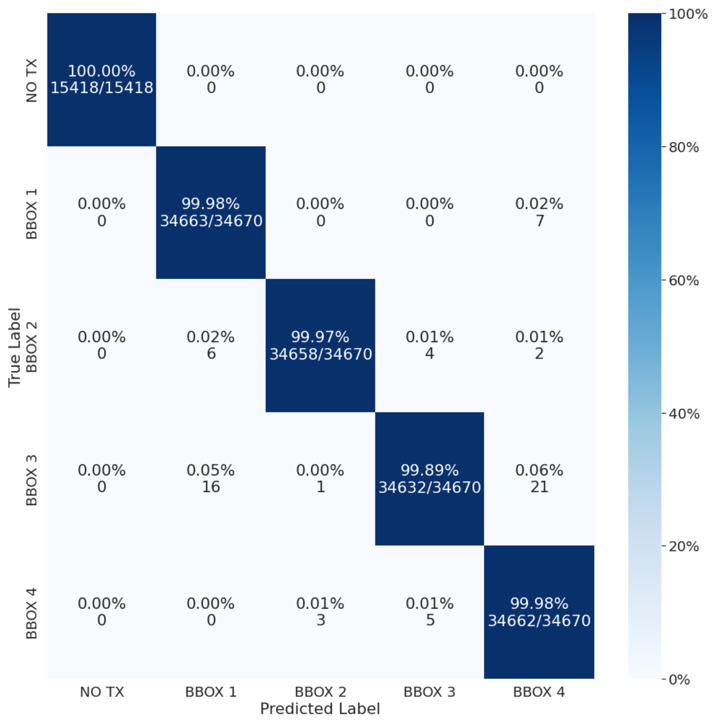


Figura A.44: Confusion Matrix for CIR$_3$ in Scenario 2 for Setup 2 trained with Neural Network Classifier.

A.2.11 CIR$_3$ for Setup 3

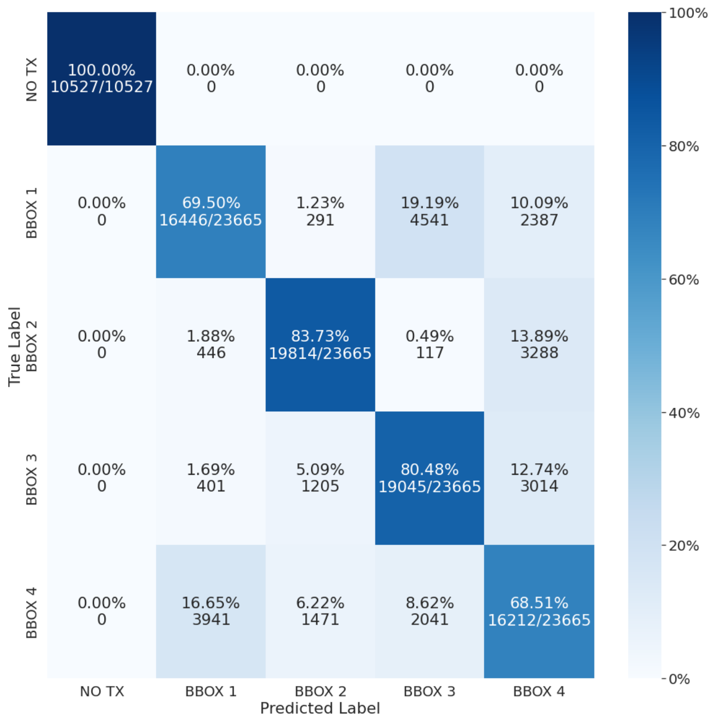


Figura A.45: Confusion Matrix for CIR$_3$ in Scenario 1 for Setup 3 trained with Random Forest Classifier.

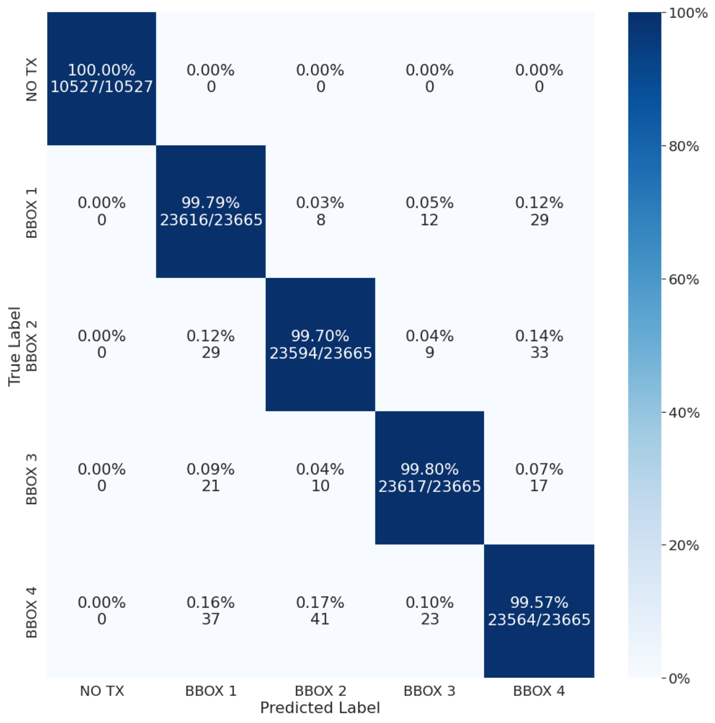


Figura A.46: Confusion Matrix for CIR_3 in Scenario 2 for Setup 3 trained with Neural Network Classifier.

A.2.12 CIR_3 for Setup 4

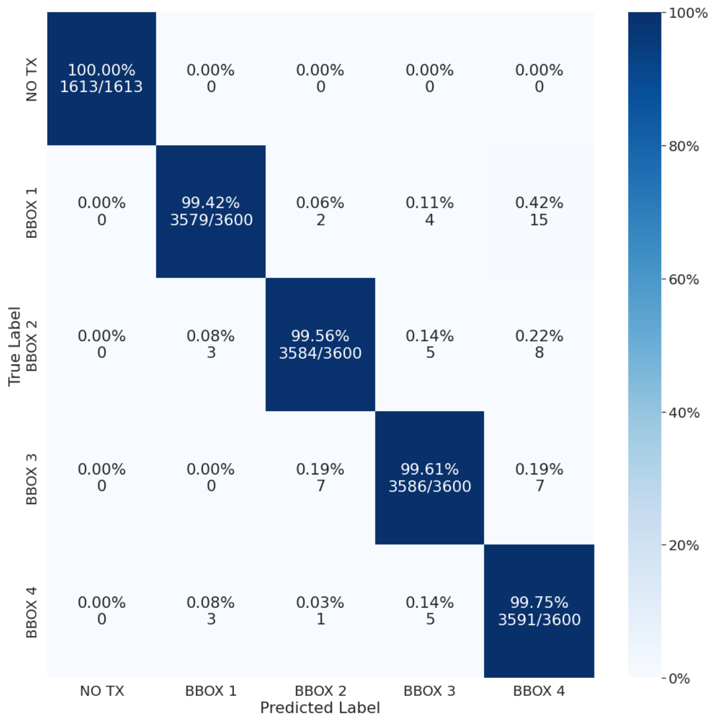


Figura A.47: Confusion Matrix for CIR_3 in Scenario 2 for Setup 4 trained with Random Forest Classifier.

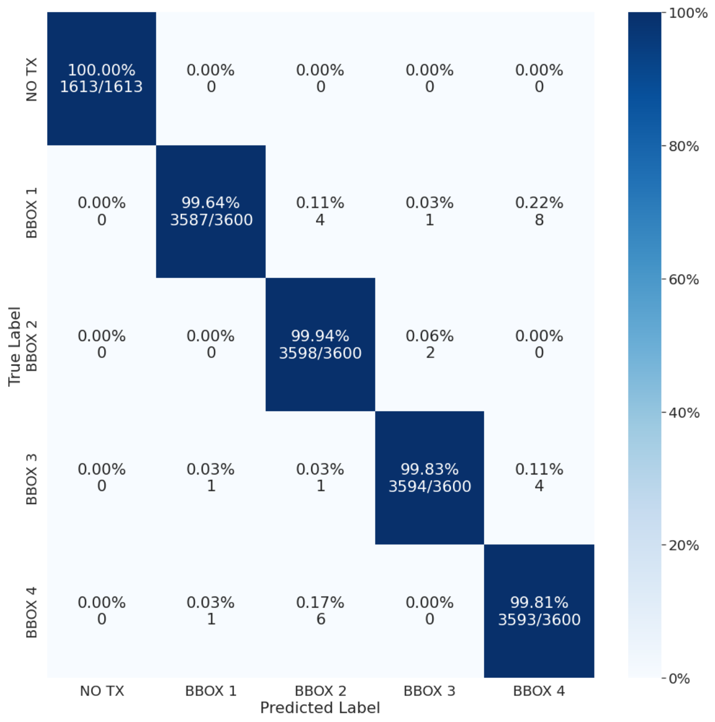


Figura A.48: Confusion Matrix for CIR_3 in Scenario 2 for Setup 4 trained with Neural Network Classifier.

A.3 Scenario 3

A.3.1 CIR_1 for Setup 1

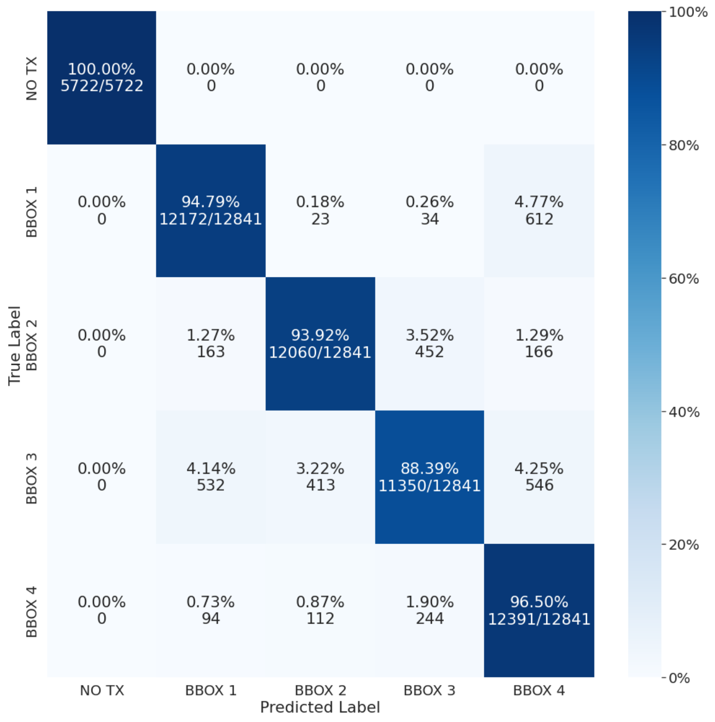
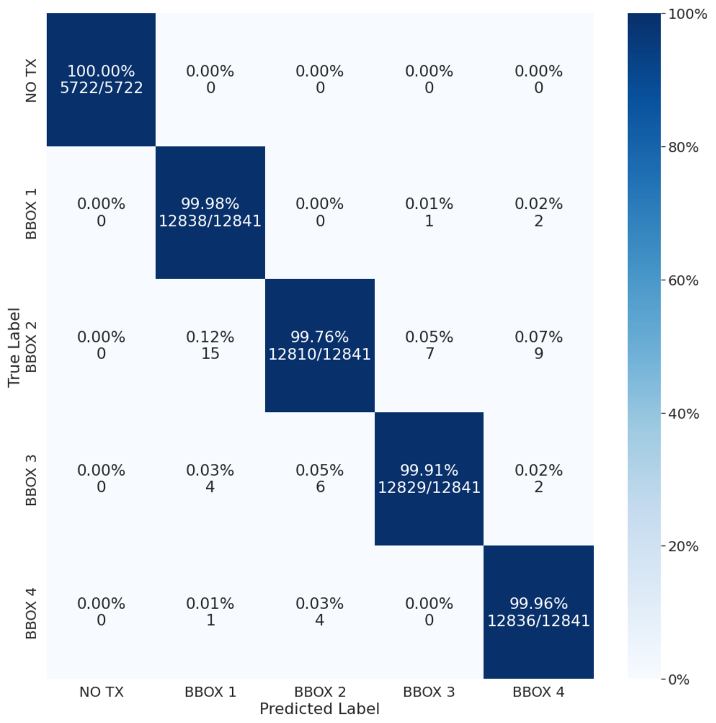


Figura A.49: Confusion Matrix for CIR_1 in Scenario 2 for Setup 1 trained with Random Forest Classifier.



S_1

Figura A.50: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 1 trained with Neural Network Classifier.

A.3.2 CIR$_1$ for Setup 2

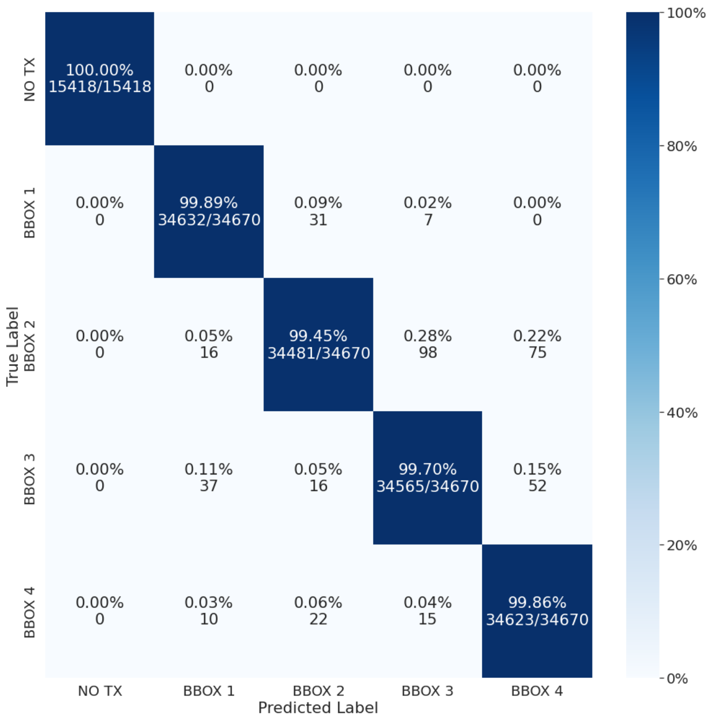


Figura A.51: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 2 trained with Random Forest Classifier.

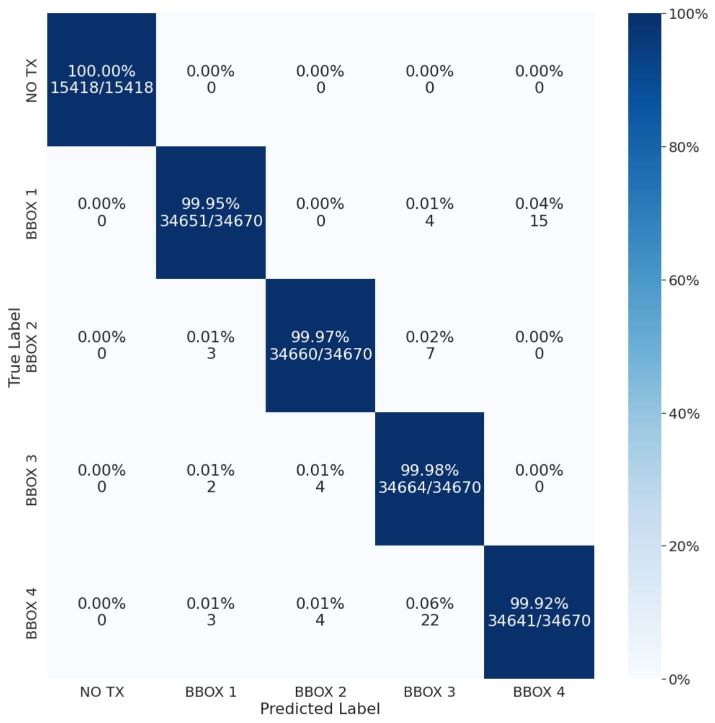


Figura A.52: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 2 trained with Neural Network Classifier.

A.3.3 CIR$_1$ for Setup 3

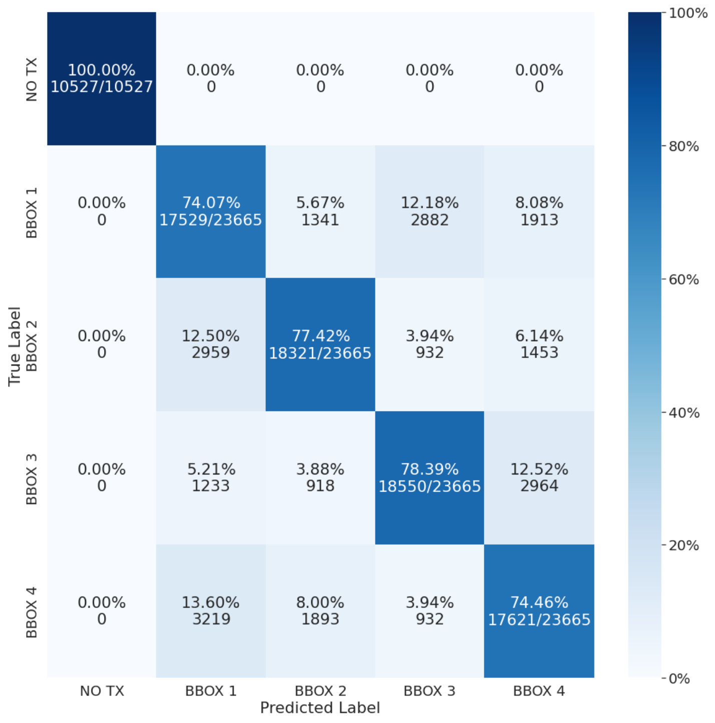


Figura A.53: Confusion Matrix for CIR$_1$ in Scenario 1 for Setup 3 trained with Random Forest Classifier.

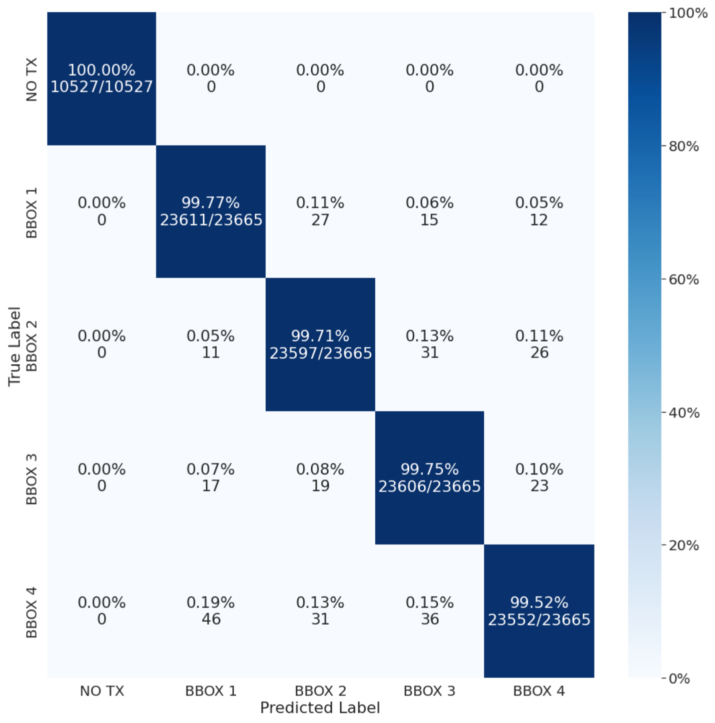


Figura A.54: Confusion Matrix for CIR_1 in Scenario 2 for Setup 3 trained with Neural Network Classifier.

A.3.4 CIR_1 for Setup 4

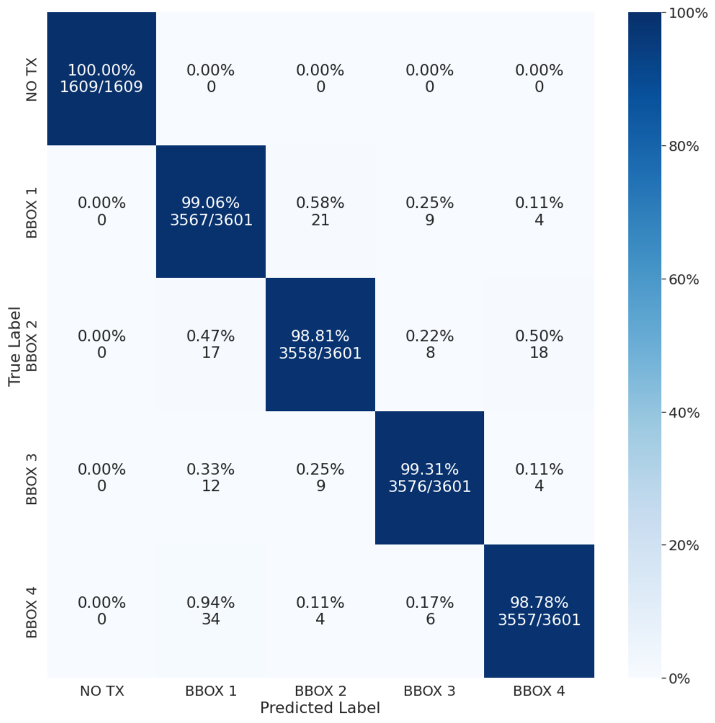


Figura A.55: Confusion Matrix for CIR_1 in Scenario 2 for Setup 4 trained with Random Forest Classifier.

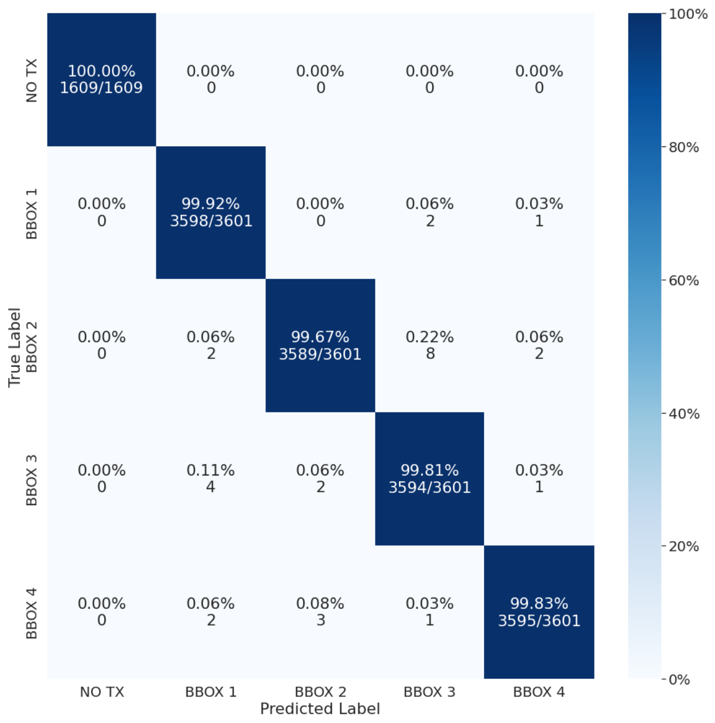


Figura A.56: Confusion Matrix for CIR$_1$ in Scenario 2 for Setup 4 trained with Neural Network Classifier.

A.3.5 CIR$_2$ for Setup 1

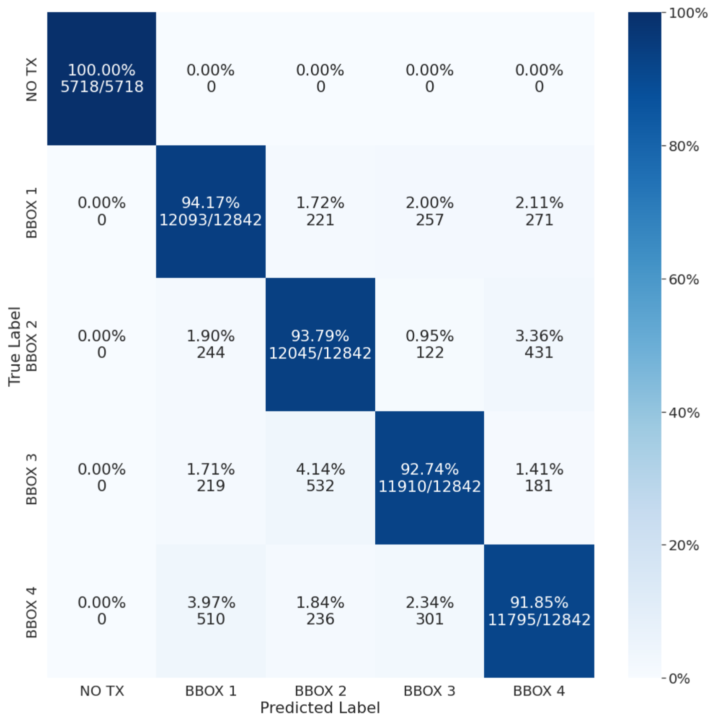
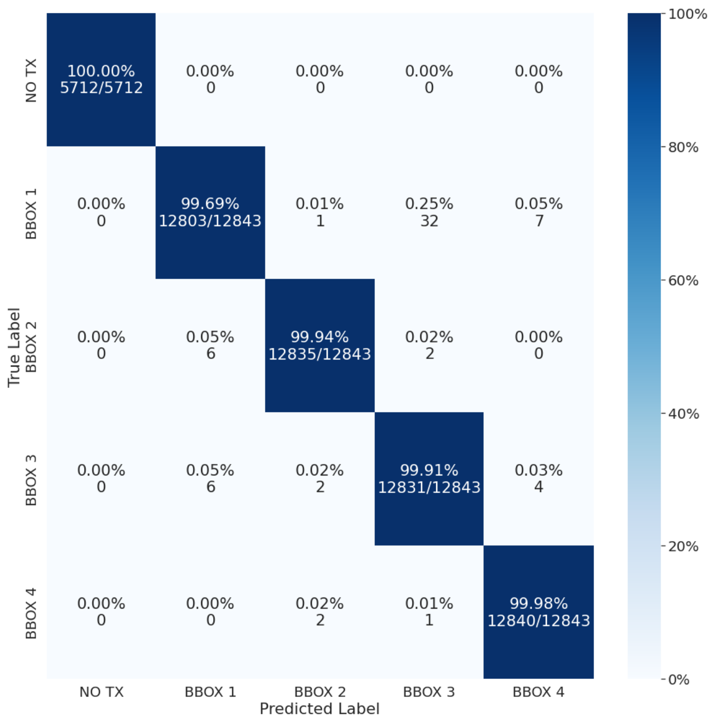


Figura A.57: Confusion Matrix for CIR$_2$ in Scenario 2 for Setup 1 trained with Random Forest Classifier.



S_1

Figura A.58: Confusion Matrix for CIR_2 in Scenario 2 for Setup 1 trained with Neural Network Classifier.

A.3.6 CIR_2 for Setup 2

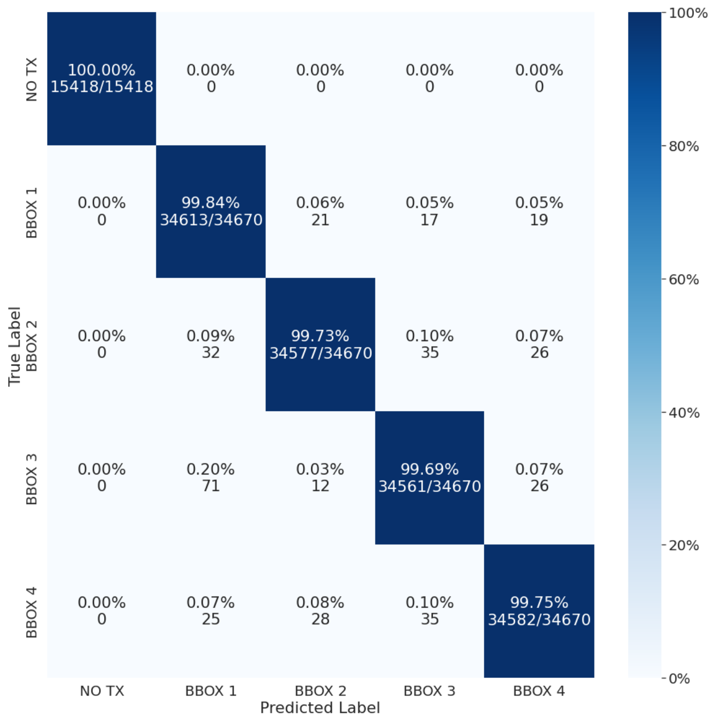


Figura A.59: Confusion Matrix for CIR_2 in Scenario 2 for Setup 2 trained with Random Forest Classifier.

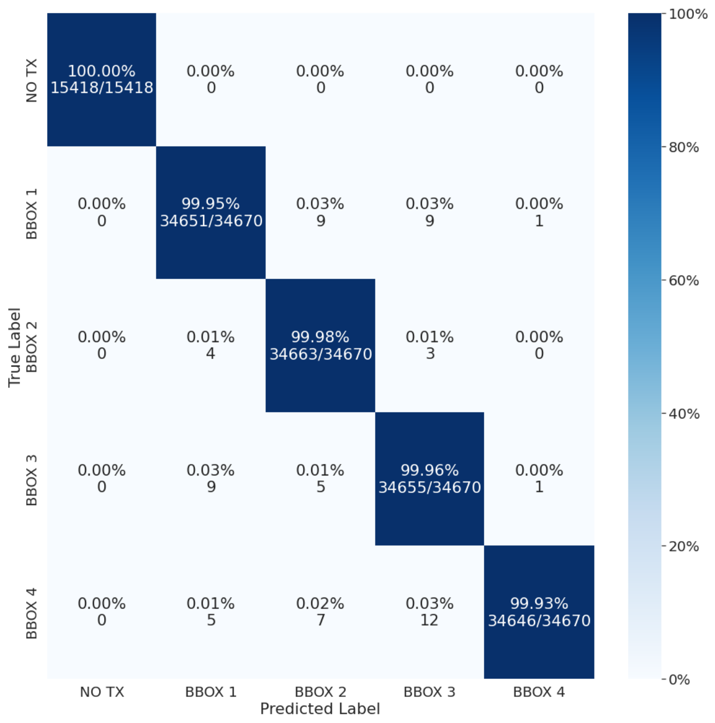


Figura A.60: Confusion Matrix for CIR_2 in Scenario 2 for Setup 2 trained with Neural Network Classifier.

A.3.7 CIR_2 for Setup 3

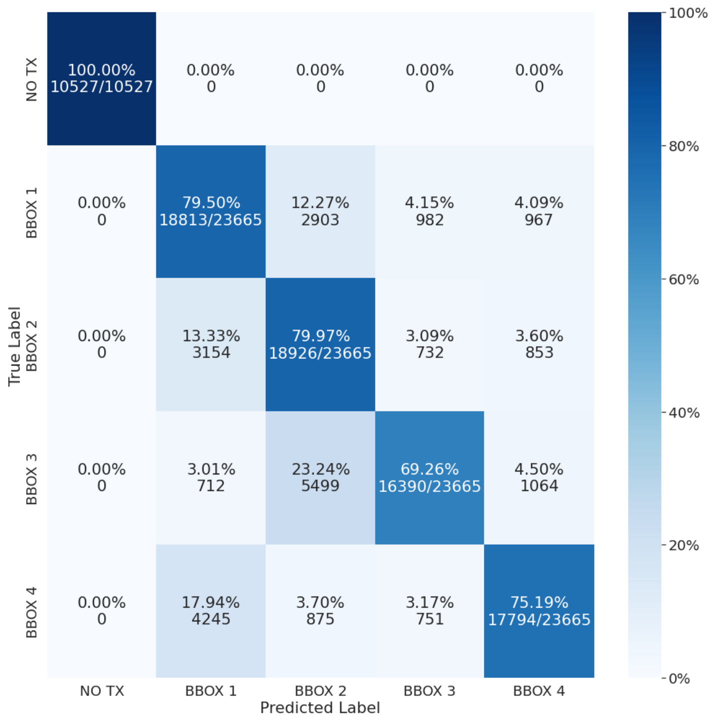


Figura A.61: Confusion Matrix for CIR_2 in Scenario 1 for Setup 3 trained with Random Forest Classifier.

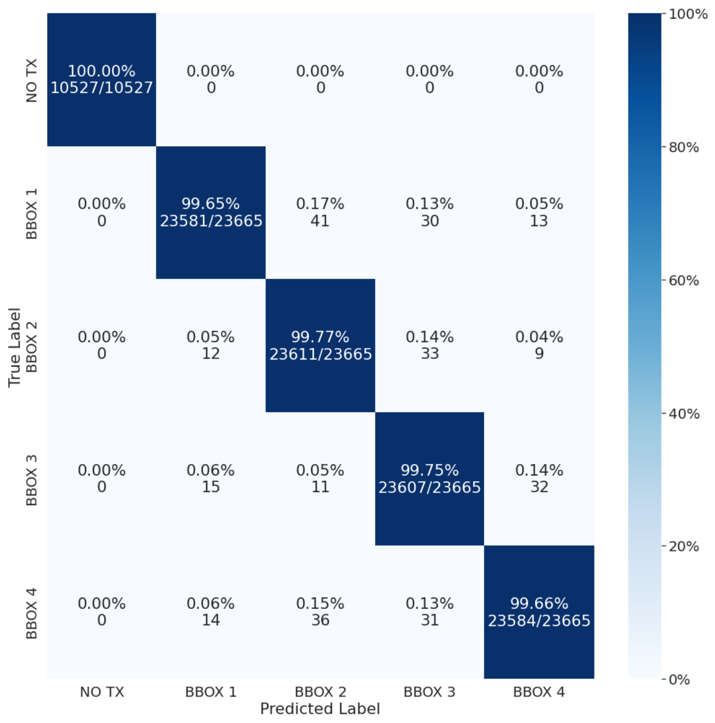


Figura A.62: Confusion Matrix for CIR$_2$ in Scenario 2 for Setup 3 trained with Neural Network Classifier.

A.3.8 CIR$_2$ for Setup 4

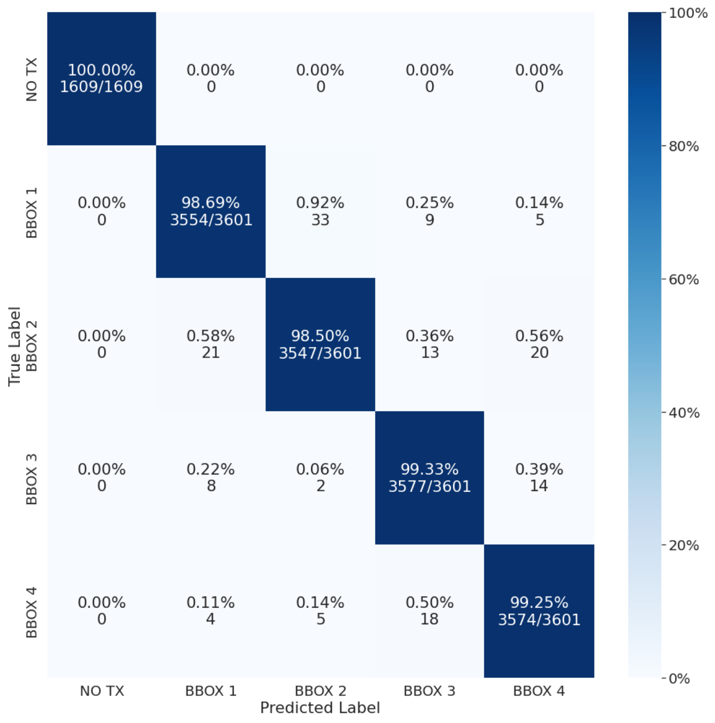


Figura A.63: Confusion Matrix for CIR$_2$ in Scenario 2 for Setup 4 trained with Random Forest Classifier.

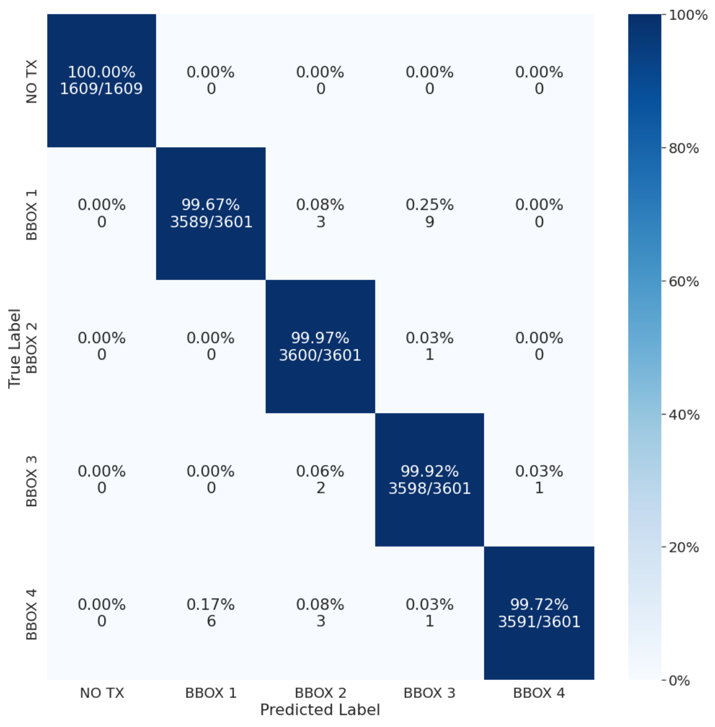


Figura A.64: Confusion Matrix for CIR_2 in Scenario 2 for Setup 4 trained with Neural Network Classifier.