Security and Interpretability in Automotive Systems

by

Shailja Thakur

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Rahul Mangharam Associate Professor, Department of ESE, University of Pennsylvania
Internal-External:	Florian Kerschbaum Associate Professor, School of Computer Science, University of Waterloo
Supervisor(s):	Sebastian Fischmeister Professor, Department of ECE, University of Waterloo
Internal Member:	Mark Crowley Assistant Professor, Department of ECE, University of Waterloo
Internal Member:	Hiren Patel Professor, Department of ECE, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Controller area network (CAN) is the most commonly found bus protocol in automotive systems. The two-wire bus protocol helps accomplish sophisticated vehicle services in real-time through complex interactions between hardware components. However, the lack of any sender authentication mechanism in place makes CAN susceptible to security vulnerabilities and threats. To address the insecure nature of the system, this thesis demonstrates a sender authentication technique that uses power consumption measurements of the electronic control units (ECUs) and a classification model to determine the transmitting states of the ECUs. The method's evaluation in real-world settings shows that the technique applies in a broad range of operating conditions and achieves good accuracy.

A key challenge of machine learning-based security controls is the potential of false positives. A false-positive alert may induce panic in operators, lead to incorrect reactions, and in the long run cause alarm fatigue. For reliable decision-making in such a circumstance, knowing the cause for unusual model behavior is essential. But, the black-box nature of these models makes them uninterpretable. Therefore, another contribution of this thesis explores explanation techniques for inputs of type image and time series that (1) assign weights to individual inputs based on their sensitivity toward the target class, (2) and quantify the variations in the explanation by reconstructing the sensitive regions of the inputs using a generative model.

In summary, this thesis presents methods for addressing the security and interpretability in automotive systems, which can also be applied in other settings where safe, transparent, and reliable decision-making is crucial.

Acknowledgements

I want to thank my supervisor Prof. Sebastian Fischmeister for his dedicated support and guidance throughout my Ph.D. In particular, I am grateful for our regular meetings and conversations that compelled me to think out of the box from different perspectives to form a comprehensive and objective critique. I am also thankful to Carlos, who, despite a tight schedule, ensured the timely availability of the hardware for the experiments. Furthermore, I would like to thank our research team for their collaborative effort during field experiments and data collection. I would also like to thank Ekin, Adan, and Oleg for being around. I am thankful to Saket for his support through good and bad times. Finally, I am incredibly grateful to my family, including Mummy, Ashu, Papa, Rozy Mummy, Papa, and Chetna Didi, for their unconditional support during the intense academic years.

Dedication

This is dedicated to my family and friends.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

CAN is the most commonly found bus protocol in modern automobile systems today. The two-wire bus protocol helps accomplish sophisticated vehicle services in real-time through complex interactions between hardware components. However, one of the fundamental limitations of CAN is the lack of any sender authentication mechanism in place, which makes CAN vulnerable to security threats. For instance, one of the vehicle functionality is adaptive driver-assistance systems (ADAS) that uses a fusion of sensors connected to the CAN network to help regulate the target vehicle's speed. However, a delay introduced in the vehicle adversarially will cause the system to fail to increase or decrease the speed on time. This unexpected behavior will trigger a cascade of incorrect actions, which will result in a collision. It is also equally likely that the unusual behavior of the system is a result of an error in the system or fault in the vehicle's automotive components. And to perform reliable decision-making in such an uncertain situation, it is crucial to identify the root cause of the system's behavior. However, due to the complex nature of the system, the accountability of the system's unexpected behavior is only partially established. In this dissertation, we propose a technique for authenticating the senders of the messages observed on the CAN bus, followed by a solution for reasoning the predictions of the authentication technique.

To address the insecure nature of CAN protocol in automotive systems, we proposed a novel sender-authentication technique that uses power-consumption measurements of the ECUs to authenticate the sender of a message. The method exploits the physical characteristics of the transmitting and non-transmitting states of the ECU, which a machine learning model classifies to determine whether the message originated from the purported sender. We tested our technique across different vehicles' for transferability and robustness of the method. We observe that the evaluation of our approach in a range of real-world

settings such as Sterling Acterra Truck, Heavy-duty vehicles over a long period and under different operating conditions helps attain a false positive rate of 0.01%.

One of the challenges of machine learning-based security controls is the likelihood of false alerts, wrongly denoting the presence of an attack, which can induce panic in the vehicle operator, leads to fatal consequences. Understanding why the system erroneously detects attacks is imperative for rational decision-making in such a context. However, the complex nature of these techniques makes it difficult to determine the cause of the system's behavior. A significant body of prior work attempts to interpret model insights. However, these methods rely on (1) models' parameters and weights, accessible by unwanted tampering with the warranty of the models, and (2) a vast majority of these techniques apply to image-based models and cannot be used to explain time series-based models. However, a solution based on non-intrusive techniques by Petsiuk et al. [77] depends on random input perturbations, which are computationally intensive and lack consistency. Hence, their usage is limited in the safety-critical domain. Therefore, we propose a solution comprising solving three closely related sub-problems that help to explain the outcomes of the time series-based sender authentication technique. The first sub-problem addresses the limitation of the prior work and proposes a non-intrusive, model-agnostic, and computationally fast explanation method. Our work builds upon the previous work by implementing a non-intrusive perturbation-based technique that uses *empirical risk minimization* to optimize a randomly initialized input mask. Because the method works by iteratively retaining and propagating pixels sensitive for mapping to the target class, it converges to an estimate of saliency-map faster than [77].

A limitation with the perturbation-based explanation technique is that it lacks consistency [65]. As a second sub-problem, we address the instability of the explanation and propose a method to generate variations for the salient region of the input for which the model prediction remains unaltered. For generating alternative explanations, we used an image completion technique [108], which reconstructs the pixels in the salient regions of the input by locating encodings in the latent space that are closest to the encoding of the neighboring pixels. Using this approach, we find an exhaustive and contextually similar set of transformations for the pixels in the semantic regions, which are also classified to the target class.

Finally, leveraging the solution for a model-agnostic and generalized explanation, we solve the third sub-problem, that is, to explain the outcome of the time series-based sender authentication technique. To do that, we refine our perturbation-based explanation method for time series-based models and propose a non-intrusive and class-discriminative explanation technique, TiME. This method is a refined version of the perturbation-based approach, tailored to work with time series-based inputs. The method assigns scores to every input

time unit based on their significance toward the target class as an expectation over the weighted random input subsamples. The weights are the model's confidence in the target classes. The sub-samples are chosen such that they are contiguous and windowed segments of the inputs to avoid the introduction of spurious artifacts in the sub-samples. The evaluation of the technique against a wide range of publically available time series datasets and state-of-the-art models shows that the approach learns to focus and discriminate between the relevant inputs for classification to the output classes.

The organization of the thesis is as follows. We provide the introduction in Chapter 1 followed by related background topic descriptions in Chapter 2. This is followed by the sender authentication approach using ECUs power-consumption-measurements in Chapter 3. Chapter 4 describes the motivation, proposed approach, evaluation metric for the model explainability problem. We have also included preliminary results for interpreting a neural network outcome for an example set of images. Following this chapter, we demonstrate the saliency-map generalization technique in Chapter 5. A time series-based model outcome explanation technique follows this chapter in Chapter 6. Finally, we conclude in the last Chapter 7.

Chapter 2

Background

This chapter provides background related to our work. We briefly describe some aspects of the CAN bus that is used in automotive systems, a brief overview of power-based program tracing or monitoring, a description of machine learning-based classifiers, in particular deep learning classification techniques, and the explainable AI techniques.

2.0.1 CAN

CAN uses a broadcast topology where multiple nodes (ECUs) can connect and exchange data [84]. The physical layer of CAN is a twisted-pair cable for serial communication using differential signaling. The operation of the CAN bus at the physical layer is based on "open-collector" or "open-drain" connected devices. Without causing any conflict / short-circuit on the bus, any device can assert a logical 0 on the bus, independently of what any other module is transmitting. Releasing the bus implicitly brings it to a logical 1, provided that no other device asserts a logical 0.

Access to the bus is arbitrated by the devices themselves. To this end, a priority field is used, the *ID* field, as illustrated in Figure 2.1. Our discussion is limited to the 11-bit base frame, which is the most commonly used. However, we emphasize the aspect that our proposed technique operates equally effectively with either 11-bit *ID*s or with the extended frame 29-bit *ID*s. Lower values for this *ID* represent higher priority, and devices read back the state of the bus to detect collisions: if a device transmitting a 1 as part of the *ID* field reads the bus and observes a logical 0, then it concludes that some other module of higher priority is transmitting, so it releases the bus. This is the case since the *ID* is transmitted MSB to LSB. Issues related to the devices' reaction to collisions are not

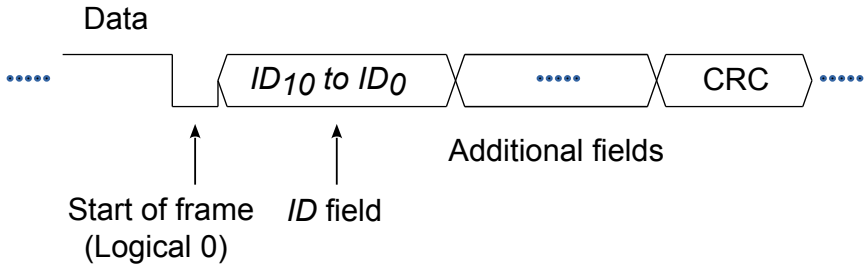


Figure 2.1: Simplified diagram of a CAN frame.

relevant to our work, so we omit any further details. Every CAN frame includes a 15-bit cyclic redundancy check (CRC) field for fault-tolerance purposes — that is, to protect the transmission from unintentional errors due to noise or other artifacts at the hardware level. These may include loose wires, defective, or aged electronics.

2.0.2 Power-based Program Tracing or Monitoring

Based upon the field of side-channel analysis [49], power-based program tracing or monitoring have appeared in the literature in recent years [25, 69, 19, 70, 58]. Among these, the earlier works ([25, 69]) propose the technique as a means to reconstruct a program's execution trace in a deployed (uninstrumented) embedded device, although both works mention other uses. Some of the work [19, 70, 58] focus on observing the power-consumption-measurements of an embedded device to detect security attacks, following the rationale that such attacks would cause the device to deviate from its normal operation. Moreno and Fischmeister argue that this is an effective technique for monitoring safety-critical embedded systems to enforce safety and security properties [67].

The basic idea of using power-based program tracing is to exploit the relationship between a device's power-consumption-measurements and its operation (more precisely, what a processor is executing). By monitoring the power-consumption-measurements, one can detect deviations from normal operation. This can be done either by attempting an explicit reconstruction of the program's execution trace or simply by profiling the power-consumption-measurements patterns during normal operation and detecting deviations from the profiled normal behavior.

In our proposed technique, the use of power-consumption-measurements monitoring has a different yet closely related goal. We are only interested in exploiting the correlation between the power-consumption-measurements of ECUs and one aspect of the ECUs program's execution: whether or not the given ECU is transmitting on the CAN bus. Though

this leads to a more limited accomplishment in terms of enforcing security, it has two important advantages: (1) Our technique can detect attacks executed by a device, which is added to the system by an attacker with physical access (we will discuss this aspect in more detail in Section 3.5.1); and (2) we can achieve a significantly higher accuracy compared to existing power-based monitoring techniques (or similar accuracy at much lower computational/processing power requirements). This is because our system only needs to reconstruct a feature that represents a much lower amount of information than reconstructing the complete execution trace or detecting minor deviations (or deviations during a short amount of time) in the power-consumption-measurements.

2.0.3 Deep Neural Network-based Classification

Classical supervised machine learning approach rely on statistical pattern recognition to perform classification tasks. Given a set of input observations $\{x_1, x_2, \cdots, x_R\} \subseteq \mathcal{X}$ and a set of output class labels $\mathcal{C} = \{C_1, C_2, \cdots, C_S\}$, the goal of any statistical learning method is to learn a mapping $\hat{f}$ from input observations to output class labels, $\hat{f} : \mathcal{X} \rightarrow \mathcal{C}$. This mapping is an estimate of the true but unknown function f that maps each input observation x to the class to which it corresponds. The function $\hat{f}$ approximates f by mapping an input observation x to the most likely class $\hat{C}$ given the observation.

Typical classifier implementations rely on features extracted using mathematical operations applied on the observations $\boldsymbol{x} \in \mathbb{R}^J$ where J is the input dimension. However, for many applications, it is difficult to identify the relevant features because of the complicated nature of input observations. There are many ways to solve this problem. One solution to this problem is to use an unsupervised neural network technique known as autoencoder [8]. Autoencoders learn the most relevant data representation, feed the learned representation as input to a classifier, and perform the classification task. The classifier can be either a neural network classifier or a classical machine learning classifier such as support-vector machine (SVM) or random forest.

The architecture of an autoencoder comprises layers of neurons (the smallest computational unit), with each unit in the previous layer passing the output to all the units in the next layer using a non-linear mapping [53]. Typically, autoencoders are used for learning data representation through an identity function that maps the input to the output. The fundamental components of the network are an encoder and a decoder. The encoder maps a d-dimensional input element represented as $\mathbf{x} \in \mathbb{R}^d$ to a new compressed representation $\mathbf{x}^* \in \mathbb{R}^{d^*}$, where $d^* \leqslant d$, to retain the most useful information from the raw input. This is followed by a decoder that reconstructs the original input from the compressed representation of the input. An improved variant of an autoencoder is a denoising

autoencoder (DAE) [101]. Given a corrupted version of the input, a DAE is trained to reconstruct the original, uncorrupted version of the input. Thus, a DAE leverages the additional constraint imposed on the reconstruction procedure of the model to learn a more robust identity function.

Another solution to the complicated features in high-dimensional space is to use a dimensionality reduction technique such as principal component analysis (PCA) to extract relevant features in lower-dimensional space and classify the principal components using a classical classifier. PCA [38] is a dimensionality reduction technique that seeks to retain maximum variability in the input by projecting the input to linear subspace. Features from the linear subspace can be fed as input to a classifier and perform the classification task.

Given segments of power-consumption-measurements measurement of the ECUs corresponding to a transmission, we aim to identify whether the transmission belongs to an ECU by detecting the state of the ECUs during the transmission. For our proposed technique, we perform binary state classification of the ECUs using a variety of algorithms such as random forest, distance-based clustering, and DAE-based classification. We can use the algorithm to implement a model for every ECU on the bus and use the models for the classification of the segments of power-consumption-measurements measurements to the class of transmission and non-transmission. Finally, we use the predicted states of the ECUs to identify the sender of the transmission as the ECU with the highest value of the transmission. A random forest-based classifier is a discriminative classification algorithm that classifies an input vector to a vector of real-valued numbers that signifies the strength of the model in the output classes. On the other hand, to approach the binary class assignment of a segment of power-consumption-measurements measurement using a clustering algorithm, it is essential to cluster the segments of power-consumption-measurements measurements of the ECUs observed during transmission intervals by ECUs. The idea is that the segments of power measures corresponding to the same ECU will occur in close proximity within the embedded space of learned feature representation. Thus, we use the principal components of the segments corresponding to the ECUs to estimate a similarity measure such as a Mahalanobis distance [61, 52] per ECU, and use the distance estimate as the threshold to assign the PCA components of segments of power-consumption-measurements measurement to the clusters. We also perform binary classification using the encoding of the autoencoder as the features to a classifier. We apply the input reconstruction capability of DAEs described in the above paragraph to develop a classification model that learns to separate the relevant features of the power-consumption-measurements patterns of transmission and non-transmission periods of ECUs. To develop the model we stack multiple DAEs on top of each other to form a SDA [102]. The output from the last DAE

layer is fed to a supervised classification layer which estimates the probability distribution over the two states of the ECU: transmission and non-transmission.

2.0.4 Explainable AI

Explainable mean *to express on understandable terms* [23]. The term *explainability* is used when the explanation is desired from the perspective of the internal working of the model—for instance, when identifying the weights of the neural network and the units within the layers of the network, which are maximally activated upon observing a particular pattern in the input.

With significant success, deep learning approaches can achieve near-real human performance and, in many cases, surpass human-level accuracy in computer vision for object detection and natural language understanding. So, why not just trust the decision of the state-of-the-art model? Why explain the reason for model prediction? The answer depends on the context of usage. The explanation is unnecessary if there are no significant consequences of unusual predictions (such as social media or e-commerce websites). Or if the problem is validated in real-wold such that the end-user trust the decision-making of the system, such as postal code sorting. However, the explanation is necessary if unexpected circumstances impact human lives and the immediate environment. Consider an object recognition algorithm used by an oncologist to detect the presence of a cancerous tumor in a patient. In addition to influencing the oncologist to select the right course of diagnostics, such a decision will also affect the patient physically and psychologically. Therefore, they are well within their rights to ask for the reason for the prediction.

The need for explanation arises because accurate predictions are not sufficient in the real world, and it is equally important to explain the prediction [23]. In addition, they argue that we cannot trust the prediction of machine learning-based classifiers because the response of these models is only partially accurate. The significant reasons for the incompleteness of these models stem from the following,

- **Model biasedness**: The lack of knowledge in the training data results in an under-represented model. We can observe the impact of the under-fitted model in the application domain such as criminal justice and healthcare, where a lack of *fair* and *unbiased* decisions can have an immediate impact on human lives. Some real-world examples where the impact of the biased model directly impacted humans' lives include: (1) The face recognition system developed by amazon is shown to have achieved an accuracy of 90%. But, the software also possesses a high error rate in

marginalized and under-represented groups of people such as black and women. The impact of such a bias led to the arrest of three black men, who were falsely categorized as criminals by the facial recognition system. In another real-world application, (2) the facial recognition system by Amazon, IBM, and Microsoft have shown to have an over-representation of light-skinned subjects as opposed to dark-skinned subjects, as well as an over-representation of men over women. The consequences of this imbalance have resulted in a high error rate in recognizing dark-skinned females as males. When employed in the customs and immigration system, such a system will impact the women travelers psychologically. Hence, they are well within their rights to demand a justification for the reason of the error.

- **Model variance**: Presence of noise in the training data, which results in an over-represented model. In the face of variation, introduces in the training data in the form of Gaussian noise, or, the model must ascertain the properties of *reliability* and *robustness*. An over-represented model can result in learning patterns and features that do not generalize to the real world.

The spirit of the lack of knowledge representation and model variations also extends to other mission-critical applications. Therefore, it is crucial to understand the reason for model predictions. Our research on explaining the model prediction is particularly motivated by the unexpected false attack detection in the automotive systems; that is, the model wrongly detects the presence of an attack. Such an unexpected false-positive generated by a machine learning-based classifier induces panic in operators, requiring justification in the form of model prediction explanation.

Prior interpretability techniques come in varying shapes and sizes. Several criteria can be used to categorize the techniques. Molnar et al. [65] provides a comprehensive overview of the interpretability techniques and provides an overview of the guidelines that we can use to differentiate between these techniques,

- **Intrinsic vs post-hoc** Intrinsically interpretable methods are methods that leverage the inherently interpretable models. For instance, a linear regression model with regression coefficients determines the influence of the corresponding input feature on the target output.

  On the other hand, post-hoc interpretability techniques are methods applied to models that are not inherently interpretable. For instance, deep-learning techniques rely on non-linear interactions between the various layers of its network, and hence the inner-working is obfuscated. In this method of explanation, the technique is applied

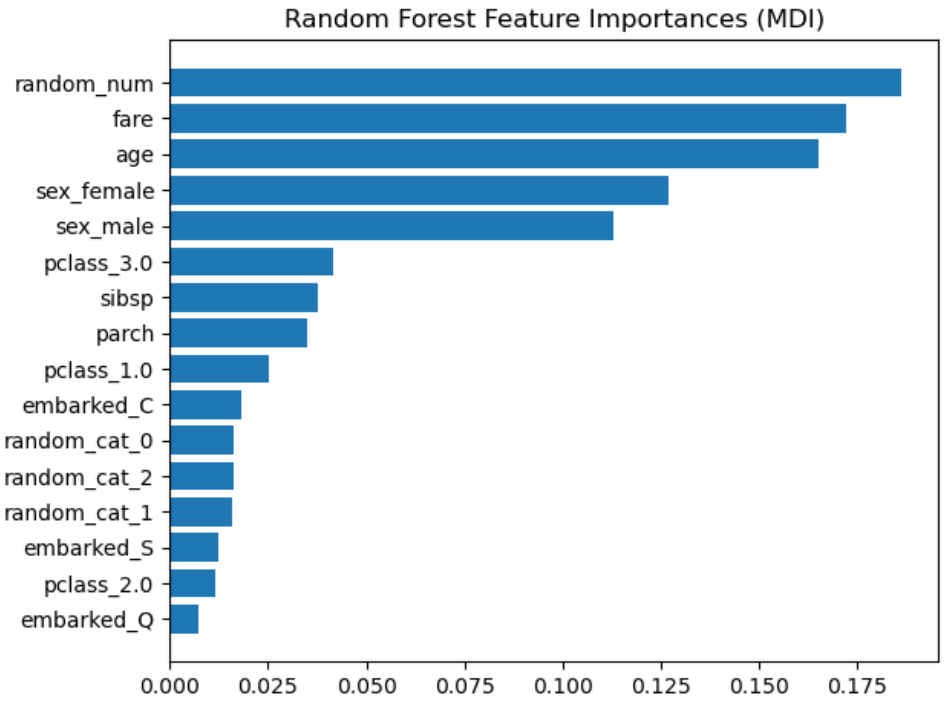


Figure 2.2: *Image source* [60]: Figure shows the feature importance plot for the different attributes of the Titanic dataset [3] obtained from the fitted random forest model.

on top of the deep-learning-based model. SHAP [60] and LIME [83] are the most commonly used post-hoc interpretable methods. SHAP is based on a game-theoretic approach to finding the contribution of different features to the model output for a particular instance by finding the difference in model output, for instance, compared to a base value. On the other hand, LIME generates an explanation for the particular instance by perturbing in a small neighborhood of the input vector space, synthesizing samples from this space, and fitting an interpretable proxy model such as a linear model.

- **Model-specific vs model-agnostic** Model-specific methods achieve explanation by exploiting the capabilities that are specific to the model. For instance, CAM [112] generates a target-specific saliency map by taking the global average pooling of the feature maps at the layer before the fully connected layer. GradCAM [90] is a generalized version of CAM that, in addition to the feature map weights, feeds the class gradient to the fully connected layer to assign importance to each of the input pixels. Both [112, 90] can be applied to limited neural network architectures that have a global average pooling layer. Similarly, attention-based methods for highlighting the important words in a text corpus rely on the attention vectors present in natural language-based neural network architectures such as BERT [22].

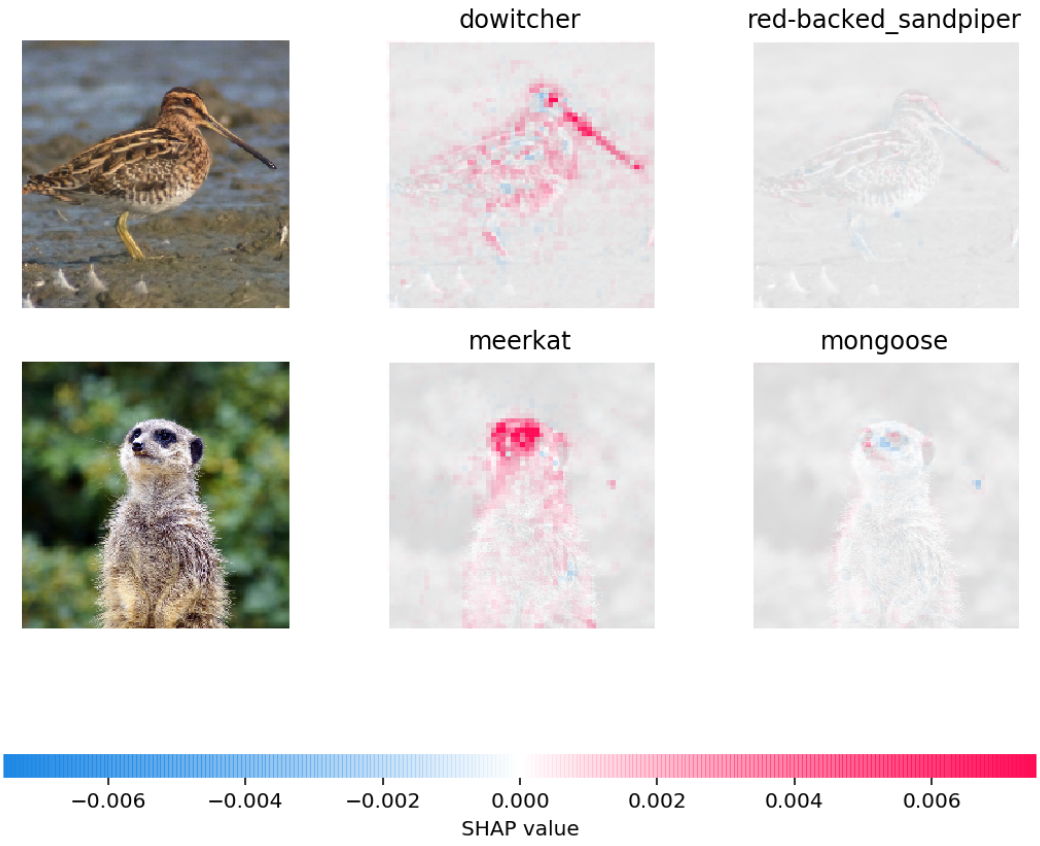


Figure 2.3: *Image source* [60]: Figure shows SHAP explanation for two images *dowitcher* and *meerkat*. Higher (positive) SHAP values are indicated by red pixels, and lower (negative) SHAP values are indicated by blue pixels. Equivalently, positive SHAP values increase the likelihood of the output class, and negative SHAP values reduce the likelihood of the class.

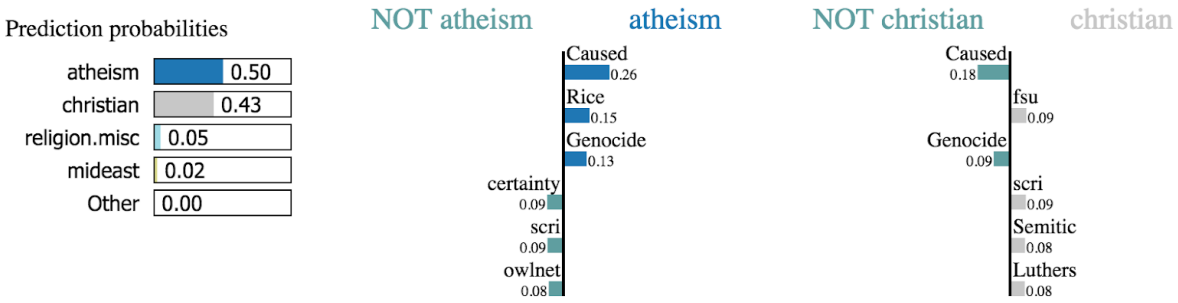


Figure 2.4: *Image source* [60]: The figure shows the locally interpretable model explanation (LIME) interpretation of a text corpus, highlighting certain tokens, which are significant in explaining the predicted class of the input.

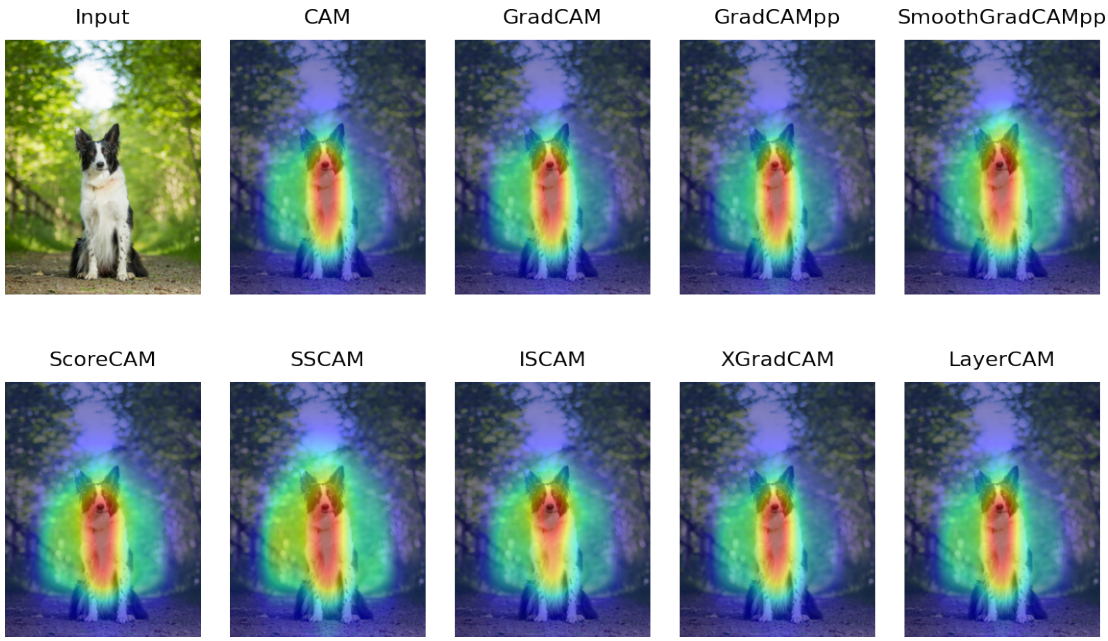


Figure 2.5: *Image source* [28]: Figure shows the class activation map CAM generated for the image of *woopets* on pre-trained ResNet18 [39] across all the variants [112, 90, 12, 74, 104, 103, 72, 30, 45]. Note: CAM and related approaches are post-hoc methods, applied on the pre-trained model, and model-specific; that is, the technique relies on the global average pooling layer prior to the last fully connected layer for the method to work.

In contrast, a model-agnostic method assumes that the model is a black box. That is, the explanation tool will have no access to the model weights and parameters. Consequently, a model-agnostic method of explanation is generic to any machine learning model regardless of its architecture and internal working. The method works by perturbing the input to determine the sensitivity of the input features towards the output target class. For instance, SHAP [60] and RISE [77] are model-agnostic approaches as they apply varying ways of making the inputs and determining the regions of the input that are important for the output class.

- **Local vs global** A locally interpretable method focuses on the patterns specific to the particular instance. For instance, LIME is a method that fits a linear model to a local neighborhood and finds sub-samples within the local neighborhood to understand the significance of the point. Local interpretation is helpful when the end-user decides based on the model prediction on a particular input.

  A globally interpretable method identifies input patterns that generalize to the class of the dataset. For instance, SHAP explainability method assigns scores to input features by averaging the feature contribution over the entire dataset or a subset of the dataset.

Despite the advantages of the explanation methods, they are limited with respect to their applicability. Some of the aspects that can be used to measure the limitations of the interpretability techniques are:

- **Reliability**: Despite being universal, interpretability methods such as RISE [77], are prone to external variations induced in the form of noise. The impact is significantly observed in the explanations of the techniques that rely on input sensitivity for attributions. Work by Petsiuk et al. [77] shows perceivable variations in the generated saliency-map for the input instance across multiple runs. Such variations result in unreliable explanations and cannot be used in a mission-critical domain such as healthcare, where a slight variation in the input can drastically affect the course of action.

- **Scope**: The scope of explanation is limited to the composition of the explanation and the methodology. For instance, techniques such as SHAP can only determine the individual features important for the output class without explaining the relation between features. On the other hand, LIME exploits the sub-samples within a local neighborhood essential for the output class. Another way of measuring the scope of an explanation technique is the ease of applicability given a black-box classifier.

For instance, CAM, grad CAM, and attention-based interpretation all rely on some aspects of the network, limiting the applicability of the interpretation method to a diverse set of machine learning models.

- **Metrices of evaluation**: Measuring the accuracy of explanation depends on factors such as the type of dataset and domain of application. For instance, when applying a technique to explain an input of type image, we can use the metrics from literature such as insertion/deletion [77] that assess the effect of individual pixel or a subset of pixels at a time toward the output. This measure captures the pixels and their variations to which the model is sensitive for the output class. However, the input of type time series requires assessing the impact of the variations of the individual time-points and the corresponding time-interval within which they are significant for the output class. Therefore, in addition to considering metrics for evaluating image-based interpretations, there is a need for metrics that capture the time features.

Our focus through this thesis is on exploiting the limitations of post-hoc and model-agnostic interpretability techniques to deploy the technique in a real-world setting with minimum latency at explaining the model prediction by developing the following sub-techniques, (1) converges to a reliable saliency-map faster than the prior work [77], (2) generalizes the saliency-map for the input by obtaining a set of variations for the detected salient region of the input, and (3) fine-tuned the proposed approach for application with time series based models.

Chapter 3

CANOA: CAN Origin Authentication through Power Side-Channel Monitoring

CAN is a communication protocol widely used in automotive systems for efficient real-time applications. However, its design exhibits significant security limitations. Among the most important of these limitations is the lack of a sender authentication mechanism. Attackers can exploit vulnerabilities, e.g., the connectivity of automobile systems to infiltrate ECUs and inject spoofed messages on the network. Perhaps, the security of CAN protocol may not have been of such paramount importance when the protocol came into existence in 1993. However, in recent decades, security has become a critical aspect in automotive systems, given the increase in complexity and connectivity of modern vehicles [51]. In particular, a study from 2009 suggests that a high-end vehicle contains as many as 70∼100 ECUs with tens of millions of lines of source code running across these devices [2] providing thousands of vehicle functionalities.

Besides CAN, other communication mechanisms and their protocols such as Flex Ray, TTCAN, CANOpen, SafetyBus, CAN-FD etc. are also used widely for real-time communication. However, analogous to CAN, these protocols also lack sender authentication. This inherent security flaw in these protocols makes them vulnerable to remote attacks. These attacks only become more feasible as the connectivity of the systems increases.

Given the insecure nature of the CAN bus, there has been a growing interest among researchers to study the security of in-vehicle communication systems. For instance, Checkoway et al. [13] and Koscher et al. [50] demonstrate the potential vulnerabilities in automo-

tive systems by studying and exploiting various attack vectors, including remote wireless connectivity such as Bluetooth, cellular, GPS, etc. The work by Miller and Valasek [62, 63] highlights the lack of authentication as one of the critical limitations of CAN networks. They exploited the radio connectivity of the infotainment unit to hijack its functionality and send messages to other ECUs. The compromised unit impersonated ECUs involved in the control of critical physical attributes of the vehicle. This allowed them to demonstrate a remote attack that disrupted or hijacked the functionality of systems such as engine, brake, and steering. Moreover, dedicated websites have materialized that provide procedures and guidelines for CAN bus hacking and reverse engineering of vehicles [4].

3.1 Related Work

Several approaches have been proposed for sender authentication in the CAN protocol. The solutions can be broadly categorized into two categories: message authentication using cryptographic techniques and authentication based on fingerprinting of physical characteristics of the transmissions.

Researchers have used traditional cryptographic techniques for message authentication [41, 55, 36] by including secret key as part of the CAN frame to prevent forgery. However, the use of these techniques is restricted due to the limited size (8 bytes) of the CAN frames and strict timing constraints in the trasnmission operation.

Fingerprinting techniques build upon side-channel analysis, or more in general, analysis of physical characteristics of the transmission. One such approach [15] uses the clock skew of periodically transmitted messages for intrusion detection. The method exploits the fact that the crystal clocks of devices are not synchronized with each other resulting in a time deviation that is unique and stable over time and used as the ECU identifier. One of the limitations of the approach is that the technique does not work with aperiodic messages. Furthermore, the work by Sagong et al. [87] demonstrates that the method can be defeated by profiling and reproducing the timing patterns of the target ECU.

Murvay and Groza [71] devised a fingerprinting approach that uses voltage variations to fingerprint ECUs for sender authentication. This approach applies only to the voltage measured on a low-speed CAN bus, while vehicles today operate at varying speed from 10 kbps to 1 Mbps depending on complexity and functionality. To overcome this limitation, Choi et al. [18] proposed an approach that generates ECU fingerprints from voltage measurement using both time and frequency domain features and used a supervised classification algorithm for sender authentication. Although the approach detected transmitter

with improved accuracy, the method has a practical limitation that the measurements are collected at an extremely high sampling rate (2.5 Gsps), and it works with a fixed message format.

Cho et al. [16] developed a model for sender identification by fingerprinting the ECUs using voltage measurements against dominant bits of the transmissions. Kneib and Huth proposed Scission [48], an ECU profiling technique that builds upon the idea of Viden for sender authentication. Scission relies on the use of all the transmission bits instead of just the dominant bits to construct ECU profile. Similar to [15], this technique relies on physical characteristics that conceivably could be profiled and imitated by a different device. Moreover, they could potentially be affected through access to subsystems outside the device implementing the technique, as shown by [88]. We do acknowledge that any such attacks would require temporary physical access to the target CAN, to add a custom device on the network.

3.2 Problem Statement

From the above discussion, it is evident that the inability of CAN bus to authenticate the sender represents one of its most important security shortcomings. This leads us to the problem that motivates our work: given a message containing the purported sender on a CAN bus, determine whether the message originated from the sender. Furthermore, if the message did not originate from the purported sender, then determine the actual sender of the message.

3.3 Contribution

In this chapter, we propose CANOA, a novel technique for sender authentication using power-consumption-measurements of ECUs as the identifying characteristic. power-consumption-measurements leak relevant and critical information about the sequence of operations executed in the ECUs [49].

Specifically, we exploit the correlation between the power-consumption-measurements of each ECU and its state (transmitting or not transmitting). From the power-consumption-measurements for all the ECUs, CANOA determines the actual sender when a transmission is observed on the bus (with a purported sender in its data), which constitutes an effective sender authentication mechanism. One key and unique advantage of CANOA is that

the classification is based on physical characteristics of the transmitting ECU that are guaranteed to be non-clonable. We can see that this is the case, given the strict relationship between ECUs activity (in particular, transmitting vs. not transmitting) and their power-consumption-measurements patterns: if an ECU E is not transmitting, it is *physically impossible* for another ECU to make the power-consumption-measurements of E exhibit the same pattern it does when it is transmitting. We observe that one condition for our technique to be effective is that the power-consumption-measurements patterns when transmitting and not transmitting, and even while receiving, must not only be different: the difference should be large enough for the patterns to be distinguishable. This is one aspect that the results of this study confirm.

The contributions of this paper are as follows:

- We propose and implement CANOA, a technique for sender authentication using power-consumption-measurements characteristics of transmitting and non-transmitting states of the ECUs.
- In addition to authenticating the sender, we also show the capability of CANOA to detect intrusion by determining the presence of compromised and additional illegitimate ECUs on the network.
- We show the applicability of CANOA in practical settings by evaluating our proposed approach for sender authentication and intrusion detection in a lab setup and a real vehicle.
- Lastly, we demonstrate the feasibility and technical viability of CANOA by studying the impact of the variations in bus speed, message format and source code on the accuracy of the CANOA.

3.4 Mathematical Notations

In this Section, we explain the technique to verify the sender of a message given the power-consumption-measurements of the ECUs on the bus.

The mathematical notation describing the problem statement are as follows.

Let $\mathcal{E}$ denote a set of K ECUs connected to the CAN bus.

$$\mathcal{E} = \{E_1, E_2, \ \cdots \ , E_K\}$$

Let E_{P} denoted the purported ECU, and E_{C} denote the compromised ECU.

Let $P_k \in \mathbb{R}^J$ represent a vector of power-consumption-measurements of length J from ECU E_k.

$$\mathcal{P} = \{\boldsymbol{P}_k : \boldsymbol{P}_k \in \mathbb{R}^J, \ 1 \leqslant k \leqslant K\}$$

Let $\mathcal{S} = \{S_1, S_2, \ \cdots \ , S_L\}$ be a set of L source addresses.

A source address is unique to a sender; however, an ECU may generate messages with more than one source address where the source address of a message can be derived from the ID of the message.

Let $\mathcal{T}$ denote a set of N decoded transmissions from the CAN bus.

$$\mathcal{T} = \{(t_n, s_n), \ 1 \leqslant n \leqslant N\}$$

where t represents the start time of the transmission, and S denotes the source address of the transmission.

Let a set of H classification models are denoted by,

$$\mathcal{F} = \{f_1, f_2, \cdots, f_H\}$$

Let g denotes a mapping from a set of source addresses $\mathcal{S} = \{s_1, s_2, \ \cdots \ , s_k\}$ to the set of ECUs $\mathcal{E}$.

$$g : \mathcal{I} \rightarrow \mathcal{E}$$

Thus, given an id corresponding to a transmission, $g(id)$ returns the ECU that most likely triggered the transmission.

Let an estimate of the transmission window be denoted by $\tau \in \mathbb{R}$, corresponding to the amount of time during which a transmission is observed on the bus.

We will use the term *power-trace* to refer to a segment of power-consumption-measurements measurement of length τ, and using notations as, $\boldsymbol{P}[\tau]$. Notice that a power-trace may contain power-consumption-measurements during a time interval where the ECU was not transmitting.

Formally, the problem statement for sender authentication on a vehicular network is defined as:

Given a hypothesis function $f : \mathbb{R} \rightarrow [0, 1]$ for classification task, a transmission $T : (ts)$ containing the purported sender $E_{\mathrm{P}} = g(s)$ on a CAN bus, power-trace $P_{E_{\mathrm{P}}}[\tau]$ corresponding to the window $[t - \tau, t + \tau]$, determine whether the transmission T originated from the purported sender E_{P}. Furthermore, if the message did not originate from the purported sender E_{P}, then determine the actual sender of the transmission from among the rest of the ECUs $E \setminus \{E_{\mathrm{P}}\}$ on the bus.

3.5 Proposed Approach

This section presents the proposed technique CANOA for sender authentication on a CAN network. We first describe the attack model and assumptions, followed by proposed approach implementation details such as data sampling, generation of power-consumption-measurements based ECU identifier, and SDA model implementation for attack detection.

3.5.1 Attack Model and Assumptions

In our attack model, a target ECU is denoted E_{T}, and it represents the ECU that the attacker wants to impersonate. That is, the attacker's goal is to transmit messages with

IDs that correspond to E_{T}. The purpose of such an attack may be to cause some other ECU to operate on false data — data that is logically correct but with contents that are under the attacker's control. For example, E_{T} could be an ECU connected to sensors, and an attacker could transmit false temperature or speed data that could result in physical damage to a car's transmission or engine. This implies that E_{T} does not have any vulnerabilities that the attacker can exploit (for example, an ECU without any connectivity to the Internet or mobile networks, when we consider remote attackers).

An attacker is capable of sending crafted messages using the ID of the E_{T} in the following scenarios:

- **Compromised ECU:** In this scenario, an attacker gains access to an ECU, denoted E_{C}. Such compromised ECU may be, for example, one with connectivity that exposes some vulnerabilities to remote attackers, or exposes services (e.g., open ports) that the designers of the vehicle did not intend to offer (and are unaware that those services are active and available).
- **Added ECU:** Our attack model includes an adversary that may temporarily gain physical access to the vehicle and the target CAN bus. Thus, they can attach an additional module, denoted E_{A}, capable of transmitting CAN messages to the existing network. This additional module can be any arbitrary, custom hardware with firmware entirely created by the attacker, thus capable of listening and transmitting without any restrictions.

Figure 3.1 illustrates the above two attack vectors.

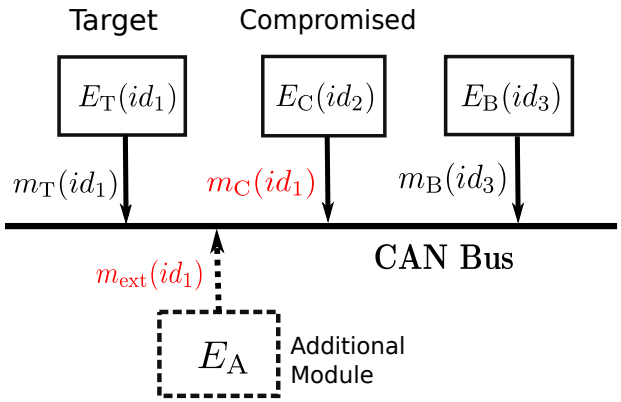


Figure 3.1: Examples of attacks on a CAN network

Our attack model also includes the possibility of an attacker tampering with ongoing transmissions. In particular, the malicious ECU (be it E_C or E_A) can hijack an ongoing transmission from ECU E_1 to change its ID and make it look like the sender is ECU E_2. We observe that this is feasible given the "wired AND" nature of the CAN bus, allowing an attacker to change any 1's to 0's in an ongoing transmission. E_1 will determine that a higher-priority frame is being transmitted, and will withdraw from the bus; from that point, the attacker can complete the transmission. Although this attack does not affect CANOA's ability to determine that the sender is not E_2, it may be beneficial for the attacker to attempt to avoid detection by shifting the blame to E_2.

Assumptions

Our proposed technique operates under the following assumptions and limitations:

- **Secure CANOA implementation:** We assume that our proposed technique is implemented in a secure and tamper-proof manner. In principle, we would expect the module implementing our technique to be physically isolated from any ECUs on the target CAN bus.
- **No DoS:** We assume that an attacker will not perform a "brute" denial-of-service on the CAN bus. In particular, if they can place an arbitrary device, they can certainly disrupt every transmission or even assert a permanent logical 0 on the bus, effectively severing all communications between any ECUs on that bus. Though this sort of "trivial" attack may seem powerful, it may also be trivial to detect; the design of many vehicles already include safety mechanisms that would appropriately deal with situations like this attack [43].
- **Types of Attacks:** Our proposed technique CANOA only detects impersonation attacks. If a transmission from an ECU E_C contains malicious data but legitimate *ID* (i.e., an *ID* that does correspond to E_C), then our system will not flag any anomalies or suspected attacks.

3.5.2 Proposed Approach

In this section, we explain the technique to verify the sender of a message given the power-consumption-measurements measurement of the purported sender.

The entire process of sender authentication is divided into four phases. In the first stage, we decode the transmissions from the captured voltage signal into a series of tuples

comprising of the start times, the end times and the IDs of the transmissions. In the second stage, we fetch and label the instances of the power-traces of the ECUs corresponding to the decoded times of transmissions. The power-traces are labeled as transmitting or not-transmitting based on the state of the ECUs at the time of transmissions. In the third stage, we train individual models for the ECUs with the labeled power-traces. Following the training stage is the operation stage where for every new transmission $T : (u, v, id)$ observed on the bus, CANOA determines the authenticity of the purported sender. To achieve this, we first fetch the purported sender $E_{\mathrm{P}} = g(id)$ based on the id of the transmissions and predict the actual state of E_{P} at time u using the model H_{P}.

power-traces

Sampling

Sampling is the first stage towards CANOA implementation. During sampling, we record voltage signals from the CAN bus and power-consumption-measurements from the ECUs.

For our proof-of-concept implementation in the lab setup, we capture the CAN bus signal at the analog level and power from the ECUs using a simultaneous conversion two-channel Digitizer. This ensures correct alignment between power-traces and transmissions. The CAN bus voltage is measured through an analog differential amplifier, with a configuration that maps recessive bits to a low-voltage output, and dominant bits to a high-voltage output. In the following step, we decode CAN transmissions in the order of their occurrences. As CANOA relies on the times and IDs of transmissions, we decode only the IDs of the transmissions observed on the bus.

For the implementation in a real vehicle, we used a tool which captures the decoded transmissions. However, due to buffer overhead, there is a variable and unpredictable delay between the actual time of transmission and the time at which the CAN controller reports the transmission to CANOA. This is illustrated in Figure 3.2. To address the uncertainty in the start time of transmission, we adjust the captured start time of transmission by an amount of δ equal to half a millisecond.

Estimate transmission window

Assuming that the transmissions windows follow a normal distribution, we calculate τ to be equal to the mean of a set of N transmissions. We refer to the resultant estimate of transmission window τ as *Dynamic Transmission Window*. In contrast to the theoretical method of estimating the transmission window [21], the dynamic method adapts to the

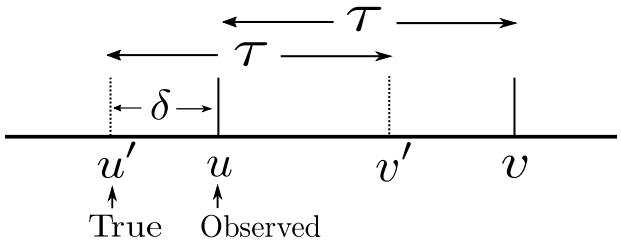


Figure 3.2: Drift in the observed and the true transmission time

network under operation by averaging the transmission windows over multiple occurrences of transmissions.

Notice that the estimation of τ is a one-time task and the same estimate can be used for fetching power-traces during the operation phase of CANOA as well.

Generating features of transmissions

In this section, we describe the method to construct features of transmissions using power-consumption-measurements of the ECUs. As the first step towards the implementation of CANOA, we capture analog voltage signal and power-consumption-measurements of the ECUs. From the captured voltage signal, we decode the times of occurrences and source addresses of the transmissions. We also calculate an estimate of the transmission window (τ), which is equal to the mean of transmission windows corresponding to N decoded transmissions. Using the power-consumption-measurements from the ECUs and the decoded transmissions, we construct features of the decoded transmissions, which are used as training data for the ECU state identification model implementation.

Algorithm 1 summarizes the steps for the construction of the feature vectors from transmissions using power-consumption-measurements of the ECUs. The input to the algorithm is a set of power-consumption-measurements from the ECUs, $\mathcal{P}$, a sequence of N decoded transmissions $\mathcal{T}$. The output generated by the algorithm is a set of pairs $\{(\boldsymbol{X}_1, \boldsymbol{y}_1), (\boldsymbol{X}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{X}_K, \boldsymbol{y}_K)\}$ where $\boldsymbol{X}_k \in \mathbb{R}^{N \times M}$ is a matrix of N feature vectors of length M and $\boldsymbol{y}_k$ is an array of N source addresses corresponding to the transmissions. As a preprocessing step, we first normalize the power-consumption-measurements from the ECUs. This is done to unify the scale of the power-consumption-measurements from different ECUs to a common scale. In the algorithm, in the expression for normalization, $\overline{\boldsymbol{P}}_k$ and s_k are the estimates of mean and the standard deviation of a sample of power signal from the ECU E_k. The normalization of power signals is followed by generating

features from transmissions as follows: For a transmission (t, S) and an ECU, E_k, fetch the power-trace $\boldsymbol{P}_k[\tau]$ corresponding to the ECUs E_k starting with the time t for the length of the transmission window τ. To the extracted segment of the power signal, we apply a windowing called Tukey window [11]. The Tukey window method helps reduce the amplitude of discontinuities at the boundaries of the segment by multiplying it with a finite-length window with an amplitude that varies smoothly and gradually toward zero at the edges. This is followed by applying fast Fourier transform (FFT) to the windowed segment of the power-trace. This helps recognize and eliminate the frequency components that are predominantly noise. This is followed by computing the PCA of the resultant power-trace in the frequency domain to filter out the frequency components which posses most of the variance in the segment, and concatenating the first $M \leq J$ principal components of the PCA, which form the feature vector $\boldsymbol{x}$ for the transmission (t, S) with the decoded source address y as the label.

Algorithm 1 Generate power-traces

Require: $\mathcal{P} \in \mathbb{R}^{K \times J}$, $\mathcal{T}$: a set N of transmissions, $\mathcal{E}$: a set of K ECUs, τ: transmission window

function POWERTRACES($\mathcal{T}, \mathcal{P}, \mathcal{E}$)
  $\mathcal{X} \leftarrow 0$ ▷ Initialize a set of K $N \times M$ matrices
  $\mathcal{Y} \leftarrow 0$ ▷ Initialize a set of K N-dimensional vectors
  $\boldsymbol{P}^* \leftarrow \frac{\boldsymbol{P}_k - \overline{\boldsymbol{P}}_k}{s_k}, 1 \leqslant k \leqslant K$ ▷ Normalize $\mathcal{P}$
  for all $E_k \in \mathcal{E}$ **do**
    for $n \leftarrow 1$ to N **do** ▷ Iterate $\mathcal{T}$
      $\boldsymbol{x_n} \leftarrow \boldsymbol{P}_k^*[t : (t + \tau)]$ ▷ power-trace of length τ
      $\boldsymbol{x_n} \leftarrow$ Apply_Tukey_Window($\boldsymbol{x_n}$)
      $\boldsymbol{x_n} \leftarrow$ Apply_FFT($\boldsymbol{x_n}$)
      $\boldsymbol{x_n} \leftarrow$ Apply_PCA($\boldsymbol{x_n}$)
      $\boldsymbol{x_n} \leftarrow \boldsymbol{x_n}[0 : M]$ ▷ First M principal components is selected as the final feature vector $\boldsymbol{x_n}$
      $y_n \leftarrow S$ ▷ Decoded Source address of T is assigned the label y_n
    end for
  end for
  return $(\boldsymbol{X}, \boldsymbol{y})$
end function

Model implementation

To perform sender authentication,power-consumption-measurements we implement a set of binary (transmission/non-transmission) classification models separately for all the source addresses observed on the CAN bus. As a source address uniquely identifies an ECU on the bus, the model per source address acts as a sender state identifier, and hence, a sender authenticator. Even if multiple source addresses are associated with an ECU, the fact that only one ECU transmits at a time, thus, the non-overlapping features of transmission and non-transmission for the ECUs helps in identifying the correct state of the ECU. The model implementation is subdivided into two stages: training and classification.

- **Training**

  Given a surjective function $g : \{S_1, S_2, \cdot, S_L\} \rightarrow \{E_1, E_2, \cdot, E_K\}$, where an ECU E_k corresponds to atleast one source address S_l. For the pair (E_k, S_l), we train a classifier $f_{kl} : \mathcal{X} \rightarrow \mathbb{R}^{(0,1)}$ that maps a feature vector of transmission, $\boldsymbol{x} \in \boldsymbol{X}$ to a real valued number $y \in \mathbb{R}^{(0,1)}$, which signifies the strength of the classifier in the class of transmission (1). For every pair (E_k, S_l), the training data $(\boldsymbol{X}, \boldsymbol{y})$ comprises of a labelled set of feature vectors of trasnmissions from the ECU E_k. In the set, an example $\boldsymbol{x} \in \boldsymbol{X}$ is labelled as one (or, to the class of transmission for the particular pair) if the trasnmission was observed with source address S_l; otherwise, the example is labelled as zero (or, to the class of non-transmission because the transmission is observed with the source address from the set $\{\mathcal{S} \setminus S_l\}$). Using the prepared training set, we train the model until convergence; that is, until the error between the predicted and the true class approaches a specific predefined threshold, ϵ. At convergence, CANOA generates a trained classifier, f_{kl}, which can be used for predicting the probability of transmission from source address S_l and ECU E_k.

  An ambiguity in classification may arise if more than one source address maps to the same ECU. This is the case because the examples of transmissions used for training the models are generated from the same ECU; and hence, an input transmission may be classified to the same class or different class by the models depending on the similarity of the models. However, as exactly one ECU maps to the source address, the prediction of the source of transmission is unambiguous.

- **Classification**

  During classification, given a decoded transmission (t, S), CANOA determines the transmitting state of each of the pair ECU (E_k, S_l) by feeding the features of transmission extracted from power-consumption-measurements of E_k at time t to the

corresponding model f_{kl}, and calculating the prediction probability of transmission of the pair (E_k, S_l). As the power-consumption-measurements measurement of the ECUs are characteristic of the transmissions, only model $f_{kl} \in \mathcal{F}$ specific to E_k and S_l will report the highest probability of transmission. Thus, the (E_k, S_l) corresponding to the model f_{kl} that reports the highest probability of transmission is marked as the actual source of the transmission. However, as the sum of the probabilities of transmissions from the models may exceed one, we apply an activation function called softmax to the vector of the probability of transmissions from the models so that the output vector sums to one, a property essential for retaining the CAN bus arbitration nature as a result of which only one ECU transmitts at a time. Based on the softmax outcome, a value of 1 is assigned to the source address (E_k, S_l) for which the corresponding model's likelihood of transmission is greater than a predefined threshold δ, and is greater than that of all the other model's predictions. And a value of 0 is assigned for the outcome from models corresponding to all the other source addresses and ECU pairs.

Algorithm 2 Predicting transmitting state of the ECUs

Require: $\mathcal{P} \in \mathbb{R}^{K \times J}$, $\mathcal{T}$: a set N of transmissions, $\mathcal{E}$: a set of K ECUs

1: **function** CLASSIFICATION($\mathcal{T}, \mathcal{P}, \mathcal{E}$)
2: **for all** $E_k \in \mathcal{E}$ **do** ▷ Calculate the features of transmission at time t from E_k
3: **for all** $S_l \in \mathcal{S}$ **do**
4: $\boldsymbol{x}_k \leftarrow powertraces(T, \boldsymbol{P}_k, S_l)$
5: $\hat{y}_{kl} \leftarrow f_{kl}(\boldsymbol{x}_k)$ ▷ $\hat{y}_{kl}$ is the probability of transmission by f_{kl}
6: **end for**
7: **end for**
8: **return** $\hat{\boldsymbol{y}}$ ▷ Proability of transmission by the models $f \in \mathcal{F}$
9: **end function**

3.5.3 Attack Detection

In this section, we describe how to use the model classification result to detect the attack model explained in Section 3.5.1. Given a transmission $T = (t, s)$ with start time of transmission t and s as the source address decoded from the message in transmission, CANOA identifies the purported sender E_{P} of the transmission. And uses the power-trace of E_{P} as input to the classifier to determine whether E_{P} is the source of transmission. Based

on the prediction, CANOA determines whether the transmission constitutes an attempted impersonation attack.

Detection of impersonation attack Consider a transmission from some ECU attempting to impersonate some other ECU. The purported sender E_{P} in that transmission corresponds to the target ECU E_{T} being impersonated. Since E_{P} is not transmitting at that time, the model f_{P} outputs 0, indicating that the message does not originate at E_{P}. This contradiction reveals the presence of an attempted impersonation attack. Upon intrusion detection, the attack vector is revealed as follows:

Compromised ECU detection Upon detection of an intrusion and depending on the attack model, either an ECU on the network is compromised, or an additional illegitimate module is attached to the network. If the transmission originated from a compromised ECU E_{C}, then the model will not predict E_{P} as the source of the transmission. However, there will most certainly exist an ECU on the network for which the corresponding model will output a 1. In other words, the features of transmissions of one of the ECUs at time t will closely match the previously observed features of transmissions from the corresponding ECU.

To detect E_{C}, CANOA constructs the features of transmissions using power-traces for all the ECUs at time t except . Given the power-traces of the ECUs, CANOA iterates over all the E_k and perform model classification to determine whether transmission originated from E_k. The idea is that one of the ECU $\mathcal{E} \setminus \{E_{\mathrm{P}}\}$ will report a probability of transmission greater that δ. The ECU E_k for which the corresponding model f_k output is a 1 is reported as the true source of the transmission T. This reported source ECU is also flagged as compromised.

Additional ECU detection If the impersonating transmission originated from an additional illegitimate device that was added to the network by an attacker, then every model $f_k \in \boldsymbol{f}$ will output 0. This indicates that none of the legitimate ECUs actually transmitted, implying the presence of an additional device on the network which sent the message.

3.5.4 CANOA-aware Attacker

We argue that even if an attacker is aware of the functionality of CANOA, it is still impossible for them to mount a successful impersonation attack. An attacker will attempt to influence the power-consumption-measurements pattern of the compromised ECU, $\boldsymbol{P}_{\mathrm{C}}$,

to match that of the target ECU, $\boldsymbol{P}_{\mathrm{T}}$, or mimic the $\boldsymbol{P}_{\mathrm{T}}$, using an additional device by taking the following measures:

- The attacker will try to use their knowledge about CANOA to drain the battery or heat up the E_{C} in an attempt to influence $\boldsymbol{P}_{\mathrm{C}}$ to match that of $\boldsymbol{P}_{\mathrm{T}}$. However, any such attempt will only show up gradually on $\boldsymbol{P}_{\mathrm{C}}$, and ultimately the changes get nullified upon scaling the signal.
- Having gained full control of E_{C}, the attacker may also try to modify the embedded programs executed on E_{C} to match with that of E_{T} in an attempt to influence the behaviour of the resulting power-consumption-measurements pattern of E_{C}. But in the absence of any knowledge about the exact sequence of instructions executed in E_{T}, the attacker will fail to imitate the program executed on E_{T}.

In the worst case scenario, even if the attacker succeeds in their attempt to imitate the sequence of programs executed on E_{T}, it will be essentially impossible to mimic $\boldsymbol{P}_{\mathrm{T}}$. In addition to the variance due to the source code, the distinction between transmitting and non-transmitting states is most likely determined by the I/O required to transmit. Virtually all ECUs use a hardware-based CAN controller to transmit, and have no physical means to transmit any other way. Thus, the attacker will be unable to do anything to cancel the inevitable power-consumption-measurements profile that the CAN controller exhibits when transmitting.

3.6 Experimental Setup

In this section we give a brief overview of the setup for capturing CAN transmissions and power-consumption-measurements from the ECUs, and the architecture of the neural network for model implementation.

3.6.1 System Description

For the prototype implementation of CAN, we connected four Keil MCB1700 boards to a CAN bus. With each board containing a CAN controller, transceiver and a receiver providing the board with the capability to send and receive CAN messages. We supplied the boards with the same power source to ensure the minimum introduction of noise in

the power-consumption-measurements pattern. Using the setup, we captured a differential voltage signal from the bus with a bus speed of 125 kbps and power-consumption-measurements from the boards using a Digitizer with a sampling rate of 10 Msps.

For the implementation of the technique in a practical setting, we deployed custom hardware in compliance with the vehicle components security and warranty, ensuring minimal modification at the hardware level. The custom hardware is used to capture the CAN signal and power-consumption-measurements of the ECUs. As shown in Figure 3.3, we sample CAN bus voltage by tapping onto the bus, and the power-consumption-measurements measurement of the ECUs by sampling the voltage drop across the shunt resistor. Single source of power and clock distribution for the capturing devices ensure that the mapping between the transmissions (captured via CAN voltage) and their power-consumption-measurements characteristics from the ECUs are aligned with respect to time for accurate ECU state classification. Using the equipment, we captured CAN transmissions on the bus operating at 250 Kbps, and power-consumption-measurements from the ECUs with a sampling rate of 10 Msps.

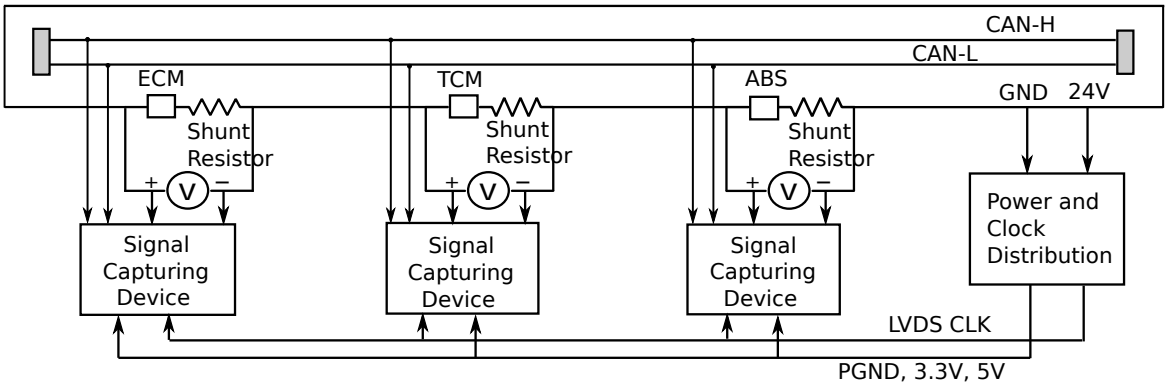


Figure 3.3: Hardware architecture for capturing CAN signal from the bus and power-consumption-measurements of the ECUs.

3.6.2 Model Description

Classical and SOTA machine learning based models

We evaluate CANOA using a wide range of classification algorithms, covering both the machine learning and non-machine learning aspects of model implementation. In particular, our choice of models for assessing sender authentication using power-consumption-

measurements includes Mahalanobis distance-based clustering, Random Forest [38], SVM [38], CNN [53], and ResNet 1D [39].

When using a classifier as a sender authentication model, the transmissions corresponding to the source address of the ECU are treated as the transmission class. And, the transmissions from the rest of the source addresses of the ECU belong to another class (the class of non-transmission). In the case of clustering, the model is trained using the transmissions from the particular source address and ECU pair only, and later the trained model is used to distinguish the true transmissions from the rest of the transmission based on the empirically estimated thresholds of Mahalanobis distance of transmission power characteristics. Finally, based on the results of an evaluation against the set of models, we finalize the model using which CANOA achieved best authentication accuracy with the given constraints of computing resource such as latency, transmission window.

Custom deep learning based model description

We implemented a supervised SDA for sender state classification in Python3.5. As shown in Figure 3.4, the network is composed of a stack of two DAE layers followed by a classification layer.

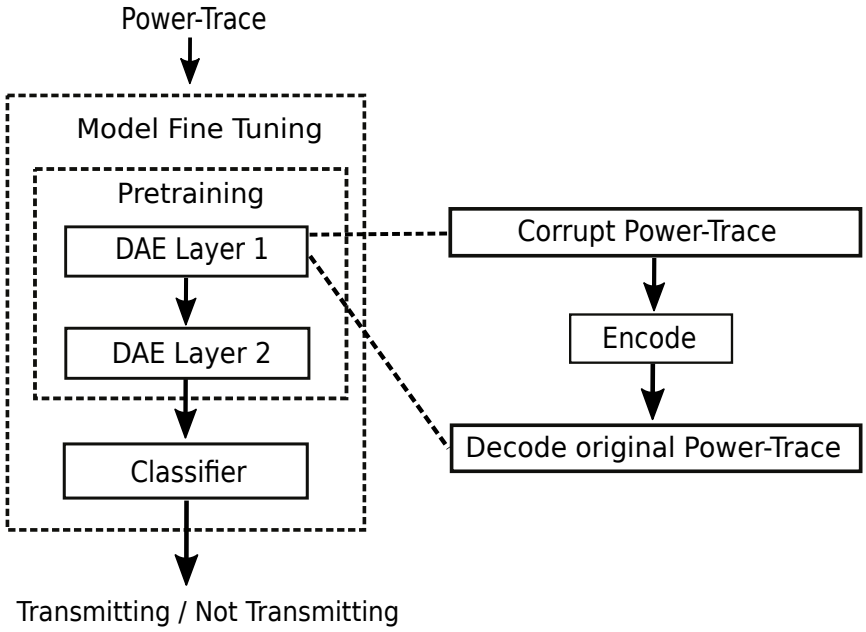


Figure 3.4: Overview of SDA.

As shown in the Figure 3.4, the model training is split into two halves: pre-training or feature-extraction as shown by the first two network layers and fine-tuning over all the network layers. During pre-training, each DAE is greedily trained independently of each other by freezing the parameters of the rest of the network. For a given input power-trace, the first DAE layer is trained to reconstruct the original input power-trace by learning the parameters (weights and biases) of the layer. The output from the previous DAE layer is then passed as input to the second DAE layer which in turn learns the features from the representation learned by the previous layer. Finally, the features from the last DAE layer is input to the classification layer. Once all the DAE layers are pre-trained, the parameters of the entire network (DAEs and Classification layer) are fine-tuned by back-propagating the gradient from the last layer up to the first DAE layer to minimize the prediction error on the classification task.

The values of the model hyperparameters such as the number of epochs, batch size and learning rate are selected as per the recommendation by Bergstra et al. [9]. The models are trained for 500 epochs each with a batch size of 100. To regulate training speed, we choose a learning rate of 0.001.

3.7 Evaluation Metrics

An accurate measure of the empirical evidence of the decision-making algorithm is essential in the automotive system. For instance, the presence of false positives triggers false alarms, inducing panic in the human who controls the system leading to a risk to the safety of life and property in the deployed environment. However, simply using the measure of accuracy to evaluate the binary decision of a classification algorithm can be misleading [80]. Furthermore, as described in the paper [64], there is no perfect numerical measure of the different criterias of classification algorithms. Therefore, we evaluate the performance of CANOA for sender authentication using Receiver Operating Characteristics (ROC) curve [44], Area Under the ROC Curve (AUC), and Confusion Matrix.

ROC curve

To evaluate the model in a real-world setting, we plot the ROC curve, which shows the variation of the true positive rate (denoted as *tpr*) to the false positive rate (denoted as *fpr*). In our evaluation, a *tpr* indicates the total number of transmissions for which the predicted sender is correctly reported as the true sender, and *fpr* indicates the total number of transmissions for which the predicted sender is falsely reported as the true

sender. Furthermore, an ROC curve is advantageous when the dataset is skewed with more negative examples (transmissions belonging to the true sender) than positive examples (transmissions not belonging to the true sender) or vice-versa. In the ROC space, the major diagonal depicts the line of random performance, and the goal is to be in the upper-left corner, which denotes better than random performance. We plot the ROC curve using the estimates of *tpr* and *fpr* on the test set of transmissions. If the curve closely follows the top left corner of the ROC space, away from the line of random performance, this indicates a better model performance.

Area under the ROC curve

In a real-vehicle setting, in addition to the ROC curve, we also estimated the area under the ROC curve. AUC is a metric used to measure the difference between the class distributions. AUC score is the proportion of unit square under the ROC curve, $\int_0^1 tpr\, d\, fpr$, that measures the separability between the binary classes. Analogously, the higher the AUC, the better the model is at separating the transmitting and non-transmitting states of ECUs and vice-versa.

Confusion matrix

For the case of lab prototype where we have power-traces from multiple ECUs, we report the decision of the transmitting state of an ECU as a multi-class classification rule where the different classes are the combination of source addresses and the ECUs. The performance measure of the decision in the case of multiple ECUs is represented as a confusion matrix with the left axis denoting the true sender (true labels), and the top axis denotes the predicted senders (predicted labels). An entry, c_{ij}, in the i^{th} row and j^{th} column denotes the total number of decisions whose true sender corresponds to the i^{th} sender along the left axis, and the predicted sender is the sender corresponding to the j^{th} sender along the top axis. Based on the matrix entries obtained using the test set of transmissions, we estimate the measure of *fpr*. The goal is to attain a *fpr* of 0. Hence, the lower the estimated *fpr* ($\leq t$), the better the classification model where t is the tolerance level to false positives in the context of the application domain.

Precision and recall

Results of sender authentication for a given transmission might show varying results when evaluated on different metrics. Therefore, we assess the performance of the proposed

technique for sender authentication using the metrics of precision, recall, accuracy, and F-measure. Precision rightly captures the false positives of the system by calculating the number of true positives (TP) over the number of true positives (TP) plus the number of false positives (FP). Whereas, recall reports the relevance of the system by calculating the number of true positives (TP) over the number of true positives (TP) plus the number of false negatives (FN). Accuracy is the ratio of correctly predicted sources for the messages to the total number of messages. And, F-measure is used to measure the similarity between the predicted state and true state by calculating the weighted average between precision and recall.

Latency

Another characteristic of the model that is critical is the latency l. Latency is the delay from the time message is received on the bus to the time the message source is authenticated. The latency of the approach in the detection phase is the time to process the message and calculate the probability of the message belonging to the sender denoted by t_{detect}. Low latency authentication system is preferable in a real-time and safety-critical system. This is the case because, outside of a defined time-interval, sender authentication holds no significance. Thus, t_{detect} should be minimal as reported by the proposed approach.

Memory

As the authentication system is an integral part of the system that functions in a resource-constrained environment, the memory requirement of the proposed technique should be within the pre-defined limits.

SWaP-C

We also evaluate the system for the three characteristics of size, weight and power to ensure the applicability in a commercial setting.

- Size: Size of the system should be small to meet the resource-constrained environment criteria. Moreover, the smaller size hardware is preferable for its easy deployability and portability.
- Weight: The system should be light-weight. A light-weight system can be easily integrated with the existing system avoiding any possible re-design, which can be costly.

- Power: The power-consumption-measurements of the system should be low.

3.7.1 Data Description

To evaluate the approach against a variety of scenarios, we captured CAN transmissions and power-consumption-measurements from a lab prototype setting and a real vehicle.

Sterling Acterra dataset

We evaluated our proposed sender authentication approach in a Sterling Acterra Truck. In the truck, we observed messages on the CAN bus with three source addresses: 0, 11, and 15. Of the three source addresses, transmissions with source address 0 and 15 were observed from Engine, and transmissions with source address 11 was observed from anti-lock brake system (ABS). For the evaluation, we captured and decoded a total of 6000 CAN transmissions from the truck. Alongside, we also captured the segments of power-consumption-measurements (feature vectors) of length 5000 from Engine. The only difference observed in the stationary vehicle's power-consumption-measurements from the moving vehicle is the change in the noise floor, which the model learned to ignore over training iterations. Out of the 5000 transmissions, 2000 transmissions, each triggered with source address 0, 11, and 15. Using the power-consumption-measurements of the Engine, we constructed the features of transmissions for the three source address of the size 2000×5000 each. Using the features, we prepared datasets for the following combinations of ECUs and source addresses: (Engine, 0), (Engine, 15), and (Engine, 11) where source address 0 and 15 are mapped to ECM (the Engine), and source address 11 is mapped to ABS. For the mapping (Engine, 0), the class of transmissions comprised of features of transmissions from the Engine with the source address 0 and the class of non-transmission comprised of the features of transmissions from the Engine with source addresses 11 and 15. Similarly, we prepared the dataset for the mapping (Engine, 11) and (Engine, 15). Due to the unavailability of the power-consumption-measurements from ECUs other than the Engine, we show the effectiveness of the approach in authenticating the Engine of the truck. To show that the method is equally effective in authenticating every ECU in the vehicle, CANOArequires the power-consumption-measurements measure from the connected ECU and establish the mapping between the true sender and the message observed on the bus.

Lab prototype dataset

With the prototype setup of the CAN network, we evaluated the accuracy of CANOA for sender authentication. We captured and decoded 5000 CAN transmissions. Unlike Sterling Truck, with access to the power source of more than one ECU, we were able to

capture the segments of power-consumption-measurements measurement from five prototypes ECUs: E_1, E_2, E_3, E_4, and E_5. The captured segments of power-consumption-measurements of the ECUs were segments of length 5000 corresponding to the period of transmission. To authenticate senders of the transmissions, we implemented classification models for the pairs of ECU and source addresses using the feature vectors (M where $M \leq 5000$) prepared from the segments of power-consumption-measurementss corresponding to the pairs, and the ECU and source addresses pairs as the labels. To determine all the pairs of ECUs and source addresses, we used our prior knowledge of the setup and obtained the mapping as follows: (E_1, S_1), (E_2, S_3), (E_3, S_3), (E_4, S_4), and (E_5, S_5) where S_1 maps to E_1, S_2 maps to E_2, and so forth.

We split the feature vectors into balanced chunks (equal no of feature vectors for the class of transmission and non-transmission) of training and cross-validation such that for each of the (ECU, SA) pairs, 70% of the total featured vectors are reserved for training and 30% for cross-validation. Using the training and validation splits, we trained the models for all the pairs of ECU and source address.

For the mapping (E_1, S_1), the training examples comprised feature vectors for the class of transmission and non-transmission. The class of transmissions comprised of features of transmissions from E_1 where the source address was S_1, and the examples of non-transmissions comprised of features of transmissions from E_1 when the source address was not S_1. Similarly, we prepared training and test sets for the rest of the combinations of source addresses and ECUs.

3.8 Results

In this section, we evaluate the proposed technique for sender authentication.

3.8.1 Evaluation in Real-vehicle

Below we describe the evaluation of CANOA in a real-vehicle setting using the sterling truck dataset. Also, we evaluate the impact of different feature lengths (M) on the latency and accuracy of the model.

Transmission features visualization

The segments of power-consumption-measurements of length 5000 observed during the transmission window are difficult to interpret for patterns in the two-dimensional space. Therefore, we visualized the Fourier transform of the segments of power-consumption-measurements for the source address 0, 11, and 15 in the lower-dimensional manifold. To obtain the low-dimensional embeddings of the feature vectors, we use Principal Component Analysis (PCA) [38]. PCA is an algorithm that can be used for dimensionality-reduction by determining the direction of maximum variance in a subspace. The directions of maximum variance are obtained using the eigendecomposition of the covariance of the segments of power-consumption-measurements. We choose the first M eigenvectors in the order of decreasing eigenvalues such that 90% of the variance from the original feature space is retained in the low-dimensional subspace. We also obtain and visualize the non-linear embedding of the segments using tSNE [38].

Figure 4(a) shows the density estimate of the feature vectors in the original feature space. Figure 4(c) shows the pair plot of the first five principal components of the power-consumption-measurements in the frequency domain. Figure 4(d) shows the pair plot of the non-linear embeddings obtained using tSNE. From the pair plots, we observe that the features of transmissions for the source address 11 forms a cluster, distinct from the features of transmissions for the source address 0 and 15. However, the low-dimensional projection of SA (source address) 15 overlap and are within a close neighbourhood with the projections of SA 0 in the subspace. This is the case because the transmissions corresponding to SA 0 and 15 are triggered from the Engine. As a result of the same source of origin of the transmissions of SA 0 and 15, they exhibit similar characteristics in the subspace. This is evident from the t-scores and p-values of the PCA and tSNE projections of the feature vectors shown in Table 3.2 and Table 3.1, where t-score is a measure to determine the difference between the features of transmissions of the various source addresses. The higher the t-score value, the larger the differences. And, the p-value determines the significance of t-scores. A small p-value (typically p-value $\leqslant 0.05$) provides strong evidence for the significance of the t-score. Based on the t-score and p-values of the linear and non-linear embeddings of the feature vectors of transmissions, we observe that the characteristics of transmission for source addresses 0 and 11 are significantly different from each other. And as expected, a p-value > 0.05 for source address 0 and 15, both of which are triggered by the same ECU, indicates that the characteristics of transmissions for the two source address are indistinguishable using non-linear embedding. However, the distinction is captured in the linear subspace using the first five components of the PCA, which is evident from the p-value ≤ 0.05 of all the pairs of source addresses. Thus, the linear projections in the

subspace can be used for the identification of the state of the ECUs during a transmission.

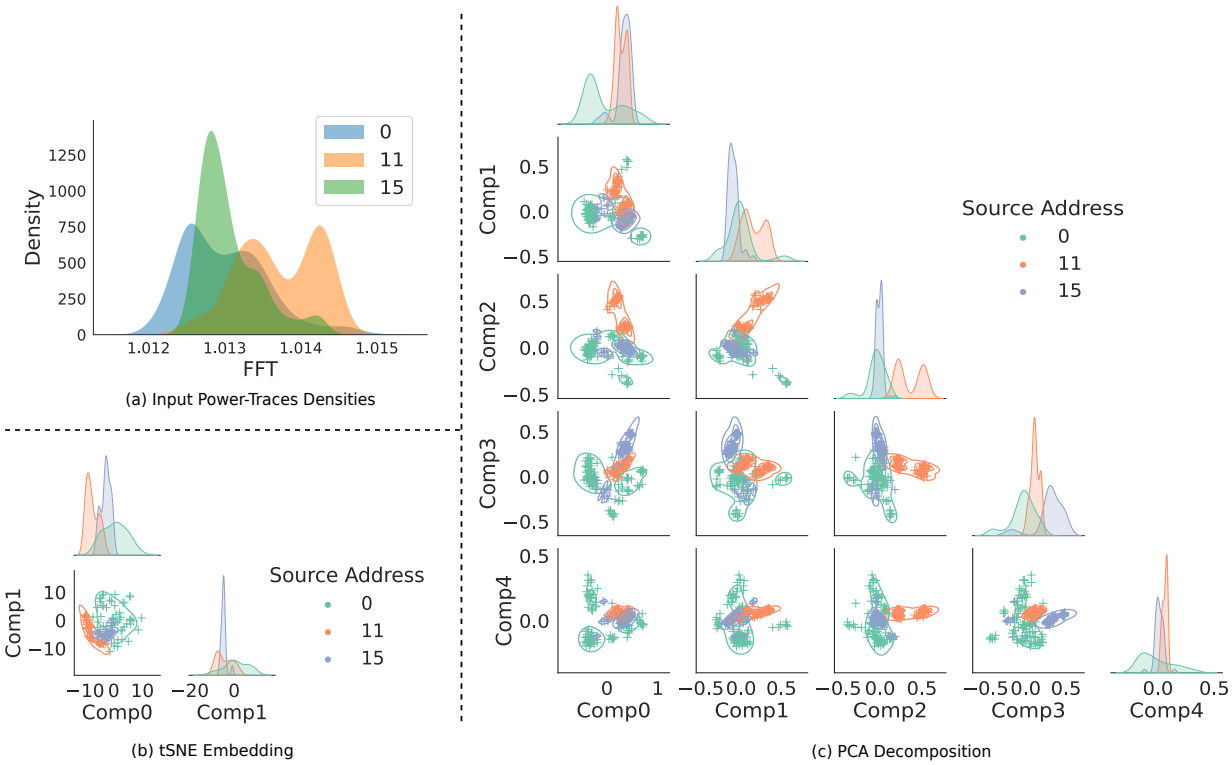


Figure 3.5: Pair plot colored by source address with a density plot of the diagonal.

Table 3.1: The t-score (p-value) between the pairs of source addresses using the power-traces of transmissions

Source address	**ECUs**	**t-score (p-value)**
(0, 11)	Different ECUs	-20.579138 (0.030911)
(0, 15)	Same ECU	-1.773469 (0.3268584)
(11, 15)	Different	21.943604 (0.028992)

Table 3.2: The t-score (p-value) between the source addresses using the PCA components of the power-traces of transmissions

PCA components	Source address	ECUs	t-score (p-value)
Comp0	(0, 11)	Different	-7.747796 (3.712786e-12)
	(0, 15)	Same	-9.245025 (8.525175e-16)
	(11, 15)	Different	3.659489 (3.259903e-04)
Comp1	(0, 11)	Different	-7.449586 (3.640866e-12)
	(0, 15)	Same	4.216789 (4.586968e-05)
	(11, 15)	Different	-16.394153 (1.229590e-35)
Comp2	(0, 11)	Different	-23.039880 (1.620486e-55)
	(0, 15)	Same	-3.164143 (1.933611e-03)
	(11, 15)	Different	-24.859311 (2.892367e-48)
Comp3	(0, 11)	Different	-8.850121 (3.518150e-15)
	(0, 15)	Same	-16.046404 (1.864838e-37)
	(11, 15)	Different	11.858681 (1.851014e-22)
Comp4	(0, 11)	Different	-2.557801 (1.197010e-02)
	(0, 15)	Same	0.058073 (9.537973e-01)
	(11, 15)	Different	-9.563822 (8.133678e-18)

Mahalanobis-distance based sender classification

From the Sterling truck, the set of possible sources of transmissions are

$$\{(Engine, 0), (Engine, 11), (Engine, 15)\} \tag{3.1}$$

. To evaluate our sender authentication approach, we implemented three Mahalanobis distance-based clusters as described in [61], one for every pair of ECU and source address. And later, we used Mahalanobis distance as threshold for the cluster assignment of new transmissions (or equivalently, to determine the state of every pair of ECU and source address during the periods of new transmissions). The model implementation is performed in two stages: training and testing.

In the training phase, to obtain Mahalanobis distance estimate for the clusters, we calculated the covariance matrix estimates of the features of transmissions corresponding to ECUs and source address pairs from the training set. This was followed by calculating

the distance estimates of the transmissions and clustering them by the source addresses and ECUs. In the testing phase, for every new trasnmission, we calculated Mahalanobis distance of the feature vector of the power-traces during the transmission for every pair of ECU and source address using the pairs covariance matrix estimates. Finally, we used the distance estimates of the clusters from the training stage as the threshold to determine the similarity between the new transmission and the clusters of transmissions, and assign the new transmission to the appropriate cluster.

We formulated the binary class assignment of a transmission to the clusters as a one-sample testing problem where a *null* hypothesis is compared against an *alternate* hypothesis. An *alternate* hypothesis is true if the distance of the new transmission is different from the distance estimate of the cluster of transmissions from the training phase for the pair of ECU and source address. On the other hand, *null* hypothesis is true if the distance of the new transmission is similar to that of the distance estimate of the cluster of transmissions from the training phase for the pair of ECU and source address. We determined the similarity of the distance estimates using p-values. Based on the p-values, we determined the transmitting state of every possible source of transmission. If the p-value is greater than 0.05, then the null hypothesis is true, and the new transmission is assigned the cluster of transmissions for the source address (and the class of transmission for the source address is labelled 1). However, if the p-value is less than 0.05, then the alternate hypothesis is true. And hence, the transmission is not assigned to the cluster of trasnmissions for the source address (and the class of non-transmission for the source address is labelled 1). Upon the identification of the state of every possible source of transmission, the source with the lowest p-value (and with the class of transmissions equal 1) is selected as the predicted source of transmission.

To determine the length of the feature vector ($M \leq 5000$) that helps achieve minimum latency without compromising the sender authentication accuracy, we report the average AUC of the classifier for different lengths of the input feature vector. For every feature-length $M \in \{5000, 1000, 500, 100, 50, 25, 10\}$, we prepared the training and test sets where the feature vectors are the first M components of the PCA of the power-trace of transmissions. Based on the retained samples of the feature vectors, we calculated the Mahalanobis distance estimates for the sources. Figure 3.6 shows the distribution of Mahalanobis distance estimates for the transmissions from the training sets across the different feature lengths and source addresses. It is evident from the kernel density estimates of the Mahalanobis distances that when the number of samples of the feature vectors is 50, the different sources are distinguishable with non-overlapping estimates of the distance mean.

Figure 6(a), shows the average AUC scores of sender authentication on the validation set of transmissions. From the figure, we observe that the score gradually increases with

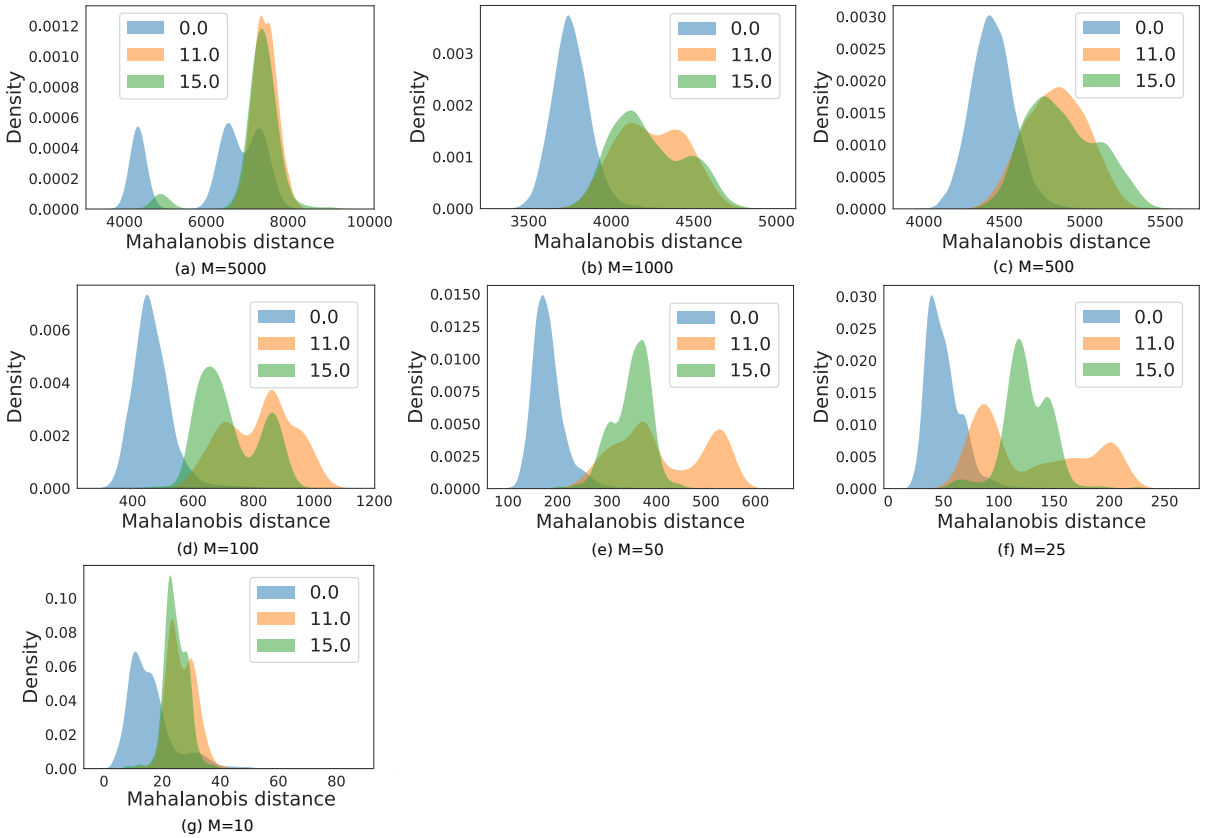


Figure 3.6: Density plot of the Mahalanobis distance estimates of the features in the training set for the source address: 0, 11 and 15.

.

decreasing feature lengths until $M = 50$, after which the score starts to drop. This indicates that the first 50 features contain enough information to reliably determine whether the ECU is transmitting and eliminating the samples any further results in a drop in the accuracy. Figure 6(b) shows the latency of the model for different feature lengths. The decreasing trend in the latency achieves a minimum value of 0.275 ms at $M = 50$, and decreasing M any further has no significant impact on the latency of the model. Figure 6(c) shows the ROC curve of the result of sender authentication on the test set of transmissions. The figure shows that the model achieves a false positive rate of 0.01% with the feature-length $M = 50$.

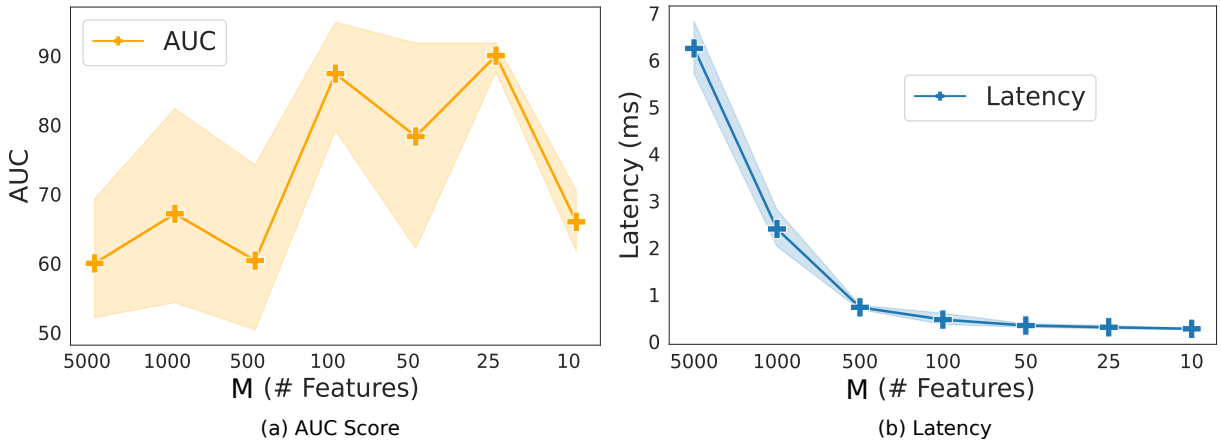


Figure 3.7: (a) Average AUC score, (b) Average latency over the set of different feature lengths.

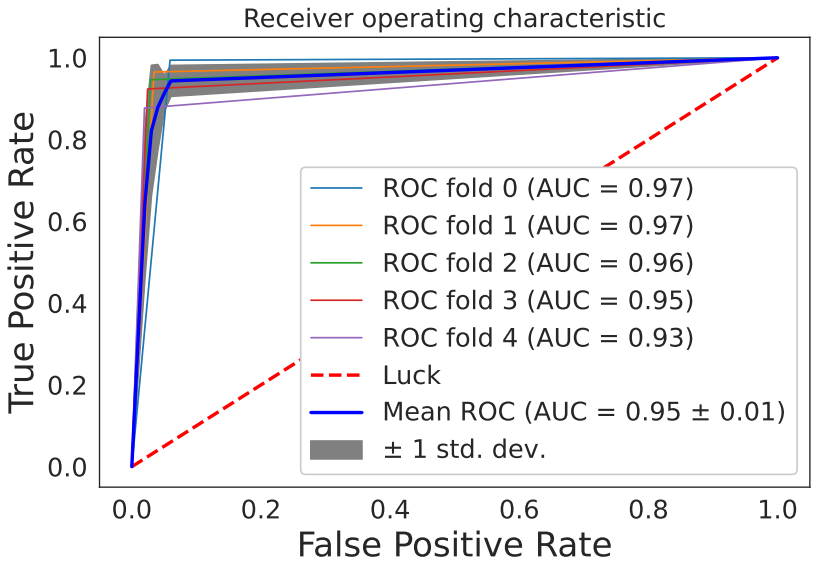


Figure 3.8: ROC curve on a test-set of transmissions with number of PCA components $M = 50$.

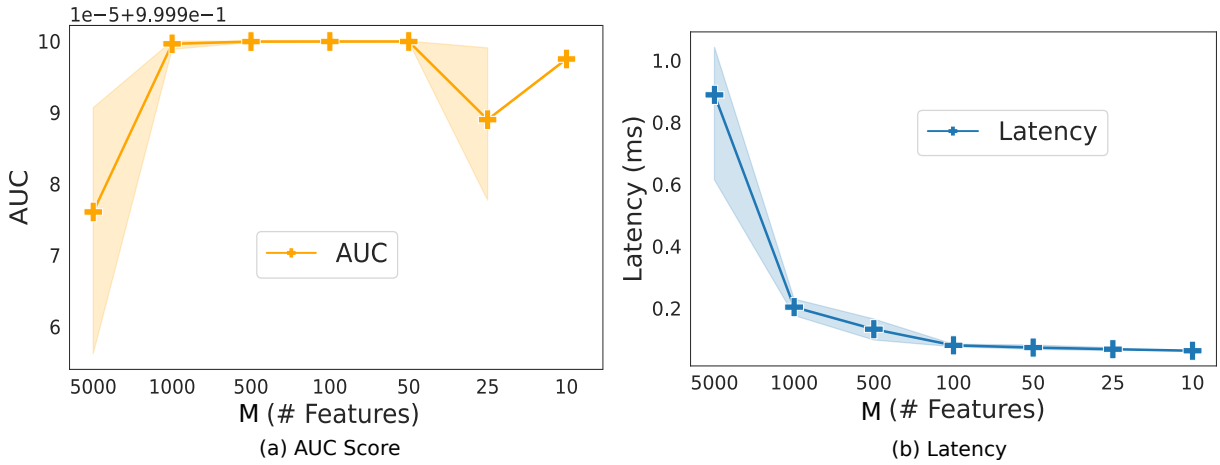


Figure 3.9: (a) Average AUC, (b) Average latency over a set of different feature lengths.

Random forest-based sender classification

We also evaluate CANOA using a random forest-based sender classification algorithm using different feature lengths $M = \{5000, 1000, 500, 100, 50, 25, 10\}$. In the training phase, for each feature length, we prepared feature vectors of transmissions and implemented three binary random forest-based classification models for the three pairs of the source address and ECU. In the testing phase, given a new transmission, the predicted sender is the source of transmission with the highest probability of transmission.

To determine the impact of the different feature lengths on the performance of CANOA, we report the latency and the average AUC score of the classifier in authenticating a sender of a transmission on the cross-validation set. From Figure 7(a), it is evident that the average AUC score of the classifier on the validation set immediately attains the maximum accuracy with a feature length $M = 1000$ until $M = 50$, after which the accuracy fluctuates. The variations in the accuracy immediately after $M = 50$ indicates the first 50 samples of the power-trace of characteristics of transmission are relevant for the sender identification, and eliminating the samples any further leads to a fluctuation (and a drop) in the relevant feature for classification of the input to the sender class. Alongside, we observe that the latency (as shown in Figure 7(b)) of the model gradually drops with the decreasing feature-length and attains a minimum of 0.0367ms at $M = 50$. Therefore, using the feature-length $M = 50$, which is optimal for model accuracy and latency, we observe from the ROC curve (as shown in Figure 7(c)) on the test set of transmissions that the classifier helps achieve a TPR of 99.97% and a false positive rate of 0.00%.

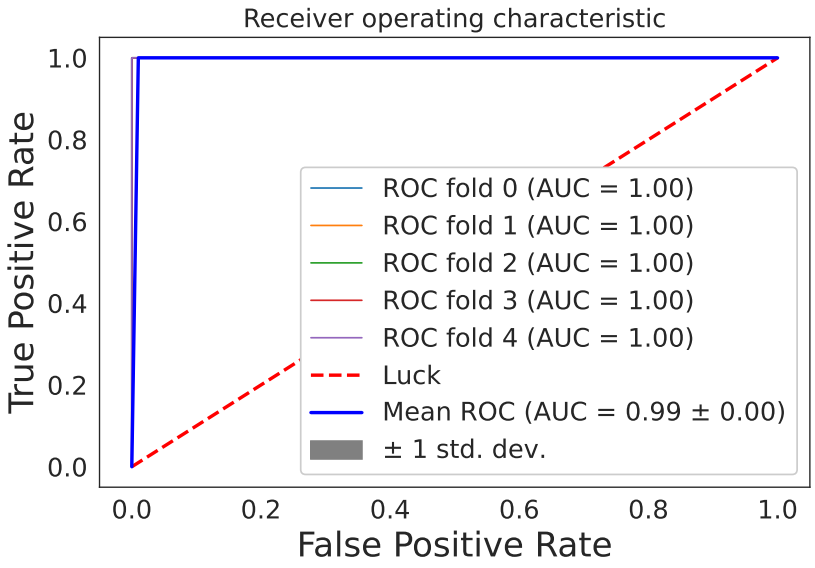


Figure 3.10: ROC curve of the random forest classifier based sender authentication on a test-set of transmissions with number of PCA components $M = 50$.

3.8.2 Evaluation in Lab Prototype

As shown in Table 3.3, we also evaluate the accuracy of the models to authenticate senders using power-traces captured from five ECUs in the lab prototype setup. We implemented a set of twenty-five random forest-based classifiers for all the combinations of five source addresses and ECUs. In the training phase, we trained models using the features of transmissions with feature length $M = 100$. In the testing phase, given a transmission T, we obtained the probability of transmission from all the models $\{(E_k, S_l) : k \in (1, K), l \in (1, L)\}$. The resultant vector of the likelihood of transmissions from all the models constitutes the likelihood of transmission of all the pairs of ECUs and source addresses. As the transmission characteristics are unique to a particular pair of ECU and source address, the vector is such that the value corresponding to the pair has a mixumum probability value. We obtained the AUC scores of predictions on the v partitions of the test set of transmissions. Using the predictions, we obtained the v confusion matrix of the AUC score of the predictions. Using the v confusion matrix, we obtained the resultant confusion matrix where each entry is the mean and standard deviation of the entries from the v confusion matrices as shown in 3.3. In the matrix, the mean AUC score of correct predictions for the true senders (that is, the correct ECU and source address pair) is shown along the diagonal. And, the value following the AUC score is the measure of the standard deviation of the AUC score of transmissions from the test set. The table shows that the model for (E_5, S_5) has the most variance with a minimum achievable accuracy of 99.25% and a maximum accuracy of 99.98%. The large variance in the model accuracy is due to the presence of noise in the extracted features of transmissions. Overall, the subtle difference in the model accuracy across all the combinations of ECUs and source addresses shows that the technique effectively authenticates senders of the transmissions accurately with a false positive rate of 0.0001 and an average AUC of 99.87% with a standard deviation of 0.001.

Table 3.3: Confusion matrix with every entry equal to (mean ± standard deviation) on the test-set of transmissions in a prototype setting

	(E_1, 0)	(E_2, 1)	(E_3, 2)	(E_4, 3)	(E_5, 4)
(E_1, 0)	$\mathbf{1.00 \pm 0.1}$	0.00 ± 0.05	0.00 ± 0.1	0.00 ± 0.2	0.00 ± 0.2
(E_2, 1)	0.00 ± 0.2	$\mathbf{1.00 \pm 0.002}$	0.00 ± 0.02	0.00 ± 0.1	0.00 ± 0.3
(E_3, 2)	0.01 ± 0.2	0.00 ± 0.01	$\mathbf{0.99 \pm 0.003}$	0.00 ± 0.03	0.00 ± 0.5
(E_4, 3)	0.00 ± 0.1	0.00 ± 0.06	0.00 ± 0.09	$\mathbf{1.00 \pm 0.0}$	0.00 ± 0.08
(E_5, 4)	0.00 ± 0.3	0.00 ± 0.2	0.00 ± 0.5	0.00 ± 0.3	$\mathbf{1.00 \pm 0.1}$

3.8.3 Additional Module Detection

To test the model for the detection of an additional device (spoofed transmissions), we injected spoofed messages on the bus with the source address 0 using a Kvaser tool [1] and captured 1000 spoofed transmissions. We evaluated CANOA for sender authentication of the spoofed transmissions along with regular transmissions from the Engine and reporting the confusion matrix in two variables: *attack* and *normal*. A value of one is assigned to the variable *attack* if none of the source address-based model's predicted probability of transmission value is greater than δ, indicating that the transmission is spoofed. And a value of one is assigned to *normal* if at least one of the source address based model's predicted probability of transmission value is greater than δ as well as greater than all the other model's probability value indicating the transmission is not spoofed; otherwise, a zero is assigned. From Table 3.4, it is evident that the technique flags all the spoofed transmissions; thus, indicating the effectiveness of the technique in sender authentication.

Table 3.4: Confusion matrix for attack detection in a Sterling Acterra truck (mean $\pm$ standard deviation)

	Normal	Attack
Normal	$\mathbf{0.99 \pm 0.001}$	0.00 ± 0.003
Attack	0.00 ± 0.01	$\mathbf{1.0 \pm 0.02}$

Table 3.5: Binary state (transmitting/idle) classification results of the ECUs on the bus in the vehicle setting

ECU	Accuracy	Precision	Recall	F1-score
ECU1	1.0	1.0	1.000	0.996
ECU2	0.995	0.983	1.000	0.991
ECU3	0.990	0.970	1.000	0.984

3.8.4 Evaluation in Real-vehicle using SDA Classifier

We evaluated our approach for determining the source of the transmission in a heavy vehicle provided by our industry partner. We monitor the power-consumption-measurements

from three ECUs — Engine, Transmission, and ABS — connected to the bus alongside the observed transmissions.

For the evaluation, we captured a total of 100k CAN transmissions from the vehicle in a stationary position. The only difference observed in the power-consumption-measurements of the stationary vehicle from the moving vehicle is the change in the noise floor, which the model learned to ignore over training iterations. Out of the 100k transmissions, 35k transmissions each triggered from Engine and ABS, and the remaining 30k from Transmission. We constructed 100k instances of power-traces of transmission and non-transmission for all the three ECUs using the value of τ equal to 0.65 ms. For Engine and ABS, of the 100k power-traces, 35k belong to the periods of transmissions which are triggered by Engine itself and the remaining to the periods of non-transmissions which corresponds to the periods of transmissions from other ECUs. Similarly, for Transmission, 30k power-traces belong to the periods of transmissions which are triggered by the ECU itself and the remaining to the periods of non-transmission. We split the power-traces of the ECUs randomly into three chunks: train, cross-validate, and test at a ratio of 6:3:1. Finally, we trained, cross-validated, and evaluated the models corresponding to the three ECUs. We trained the models offline in parallel on a Linux machine with 8 CPU cores. The models finished training on the given number of examples with an average training time of 37 hours. We also observe the response time of the models in an online setting (real-time E_{P} authentication using the stream of transmissions observed on the bus) and is noted to be an average of 0.8 ms which makes the technique feasible for application in real-time settings.

Table 3.5 shows the results of the classification accuracy where the three ECUs: Engine, Transmission and ABS are denoted by ECU1, ECU2 and ECU3. We observe that the precision of ABS and Transmission drops slightly below 100%. This is the case because of the complex nature of the hardware which induces more noise in the measured signal resulting in more randomness and less accurate results. In particular, we found the power-consumption-measurements from ABS more complex than the rest of the ECUs which can be attributed directly to the performance of ABS model. Furthermore, we observe from Table 3.3 and Table 3.5 that the results for the lab setup and the vehicle are consistent with each other with a minor variation in the mean accuracy.

One of the concerns that arise as a result of the intensive computing power requirement for model training is the deployability to different vehicles. However, we argue that the technique can be deployed to new vehicles with similar or different configuration without much overhead by using a machine learning technique called transfer learning. Transfer learning allows using the knowledge of a model called the base model from a similar task to improve the generalizability on a new task. Thus, the models trained on one vehicle are

treated as base models which are then generalized to new settings by training with fewer power-traces examples from the ECUs. By using only a fraction of the examples than the base model to generalize makes the approach feasible for deployment at scale without compromising on the accuracy.

Furthermore, we evaluate the generalizability of the trained models for sender authentication on unseen transmissions by plotting a learning curve. The learning curve shows the accuracy of a model during training over many iterations. A model is generalizable if the learning curve (training and cross-validation curve) increases at first over the iterations and then asymptotically approach an accuracy such that training any further has negligible improvement on the performance of the model. In particular, we evaluate the learning curve of the model for Engine in real-vehicle settings. From the Figure 3.11 it is evident that the model begins to converge after training with 100k example power-traces attaining accuracy of 98%. The generalizability of models trained with power-traces examples corresponding to the transmissions captured over a timeframe of fewer than 30 minutes indicates that the technique can be applied in a real-vehicle for operation in an online setting without losing accuracy.

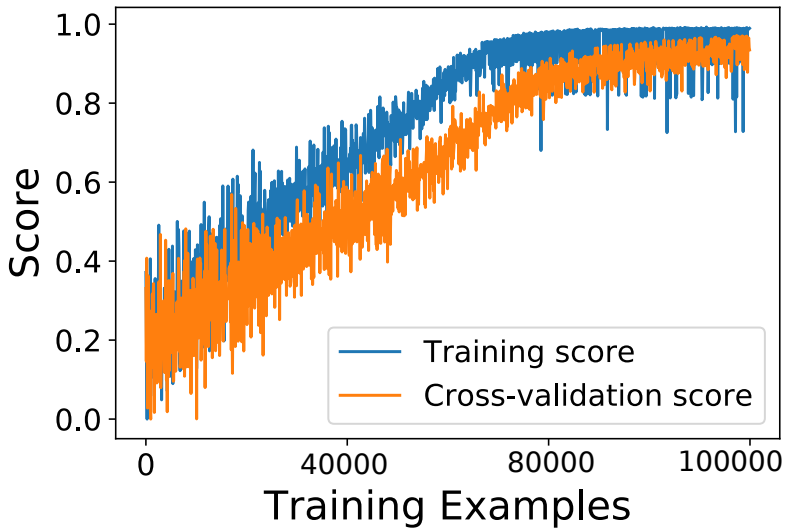


Figure 3.11: Learning curve for Engine model in real-vehicle setting

3.8.5 Summary of Comparison

We perform a qualitative and quantitative comparison of our approach against other sender authentication approaches using a set of attributes. As shown in Table 3.7, for qualitative comparison, the attributes that are satisfied by the techniques are shown by a tick mark (✓), and the attributes that are not satisfied by the techniques are shown by a cross (✕). For quantitative comparison, we report the average accuracy of our approach evaluated on Truck B alongside the sender authentication accuracy reported by [16, 48, 18]. One of the major distinguishing features of CANOA, as compared to the other approaches, is that unlike the existing fingerprinting-based approaches, CANOA is least likely to fall prey to the *profile-and-mimick* attack. *Profile-and-mimick* attack is a term coined in the paper [68], which means an attack where the fingerprint of the ECUs can be mimicked using an additional device to send fake messages unobtrusively with the *ID* of the claimed ECU. For instance, in voltage-based fingerprinting approaches such as Viden and Scission [16, 48], the voltage profile of the claimed ECU E_{P} can be mimicked to inject spoofed messages. Similarly, in the timing-based fingerprinting approach [15], the timing variations of the periodic messages can be learned to flood the CAN bus with fake messages. On the other hand, CANOA relies on the power-consumption-measurements of the ECUs for profiling the transmission and non-transmission state of the ECU, which is difficult to clone. Furthermore, in addition to detecting intrusion from compromised ECU, our approach is capable of detecting intrusion from the external module, which sets our source authentication technique apart from existing approaches [16, 18]. Quantitatively, our approach performs equally well in sender authentication using a purely side-channel-based fingerprinting as opposed to voltage-based fingerprinting, which can be cloned.

3.8.6 Model Selection

To choose the appropriate model for ECU state classification, we compared the performance of a set of candidate classification algorithms against a set of factors, which are the sources of the bottleneck in real-time safety-critical systems. For comparison, we compared the CPU utilization (in percentage) of the models, the number of CPU cores (if CPU only) used during model training and real-time authentication, the GPU (specification of the GPU used for computation), the time to authenticate the source for a given transmission (latency). The suitable model for classification is selected based on the model with optimal resource consumption, where the optimality is achieved when the estimated values of the factors satisfy the thresholds necessary for a smooth function of the model in a resource-constrained environment. From the Table 3.8, it is evident that random forest achieves

Table 3.6: Qualitative comparison of our technique against other intrusion detection systems

		Viden [16]	**Scission** [48]
Objective	IDS	✓	✓
	Source authentication	×	✓
Input source	Voltage-high	✓	×
	Voltage-low	✓	×
	Differential voltage	×	✓
	Clock signal	×	×
	power-consumption-measurements	×	×
Technique	Voltage profile	✓	✓
	Power profile	×	×
	Timing profile	×	×
Attack types	Impersonation attack	✓	✓
Attack vectors	Additional ECU	×	✓
	Compromised ECUs	✓	✓
Properties	Profile-and-mimick	×	×
	Real-time	✓	✓
Accuracy	Accuracy	99.8%	99.85%

a training accuracy comparable to that of ResNet and CNNs with minimal CPU usage (%) and latency of 0.0367 ms at test time. Mahalanobis distance-based classifier performs equivalent to a random forest. Hence, it forms another choice for the sender authentication model; however, we preferred random forest as the suitable choice for classification due to its relatively lower latency, accuracy, ease of implementation and interpretability. Please note that '-' in the CPU and GPU field of the table shows that it is not applicable for the particular model.

3.8.7 Presence of Incomplete Transmissions

We also evaluated the accuracy of the proposed technique using the events of incomplete transmissions. The events of incomplete transmissions are particularly important from the perspective of an attacker. For instance, using the attack model described in Section 3.5.1, an attacker could wait for E_T to start transmitting, then modify the *ID* by replacing one

Table 3.7: Qualitative comparison of our approach against other intrusion detection systems

		Cho et al. [15]	CANOA
Objective	IDS	✓	✓
	Source authentication	×	✓
Input source	Voltage-high	×	×
	Voltage-low	×	×
	Differential voltage	×	×
	Clock signal	✓	×
	power-consumption-measurements	×	✓
Technique	Voltage profile	×	×
	Power profile	×	✓
	Timing profile	✓	×
Attack types	Impersonation attack	✓	✓
Attack vectors	Additional ECU	×	✓
	Compromised ECUs	✓	✓
Properties	Profile-and-mimick	×	✓
	Real-time	✓	✓
Accuracy	Accuracy	96.48%	99.58%

Table 3.8: Evaluation of the various models for ECU state classification

Classifier	AUC score	CPU/GPU	Processor usage (%)	Latency (ms)
MD [52, 61]	95.45	4 CPU cores	**5.55**	**0.00001**
RF [38]	**99.6**	**4 CPU cores**	**5.34**	**0.00002**
SVM [38]	**99.4**	**4 CPU cores**	**5.971**	**0.00002**
CNN [53]	97.3	Tesla V100	11.468	0.0002
ResNet [39]	97.9	Tesla V100	12.55	0.0001

or more 1's to 0's,[1] making E_T abort, and then the attacker completes the transmission.

To evaluate this scenario, we used the signal from the real vehicle. Moreover, as there is an important difficulty in causing incomplete transmissions in a real-vehiclewithout in-

[1] Recall the "wired AND" nature of the CAN bus, as explained in Section 2.0.1

curring a considerable risk of disrupting its functionality, we synthetically constructed the traces corresponding to interrupted transmissions as follows. We captured 4k transmissions with Engine as the source of origin. Out of the 4k transmissions, we randomly selected 2k transmissions and altered the corresponding power-traces features such that they resembled incomplete transmissions. For every power-trace of transmission, we kept the number of samples equal to the first 50% of the ID bits (segment B-C in Figure 8(a)) as it is and replaced the rest of the samples (segment C-D) with the samples of the power-consumption-measurements immediately following the transmission window. As shown in Figure 8(b), the resultant power-trace of aborted transmission leads to the inclusion of artifacts such as a sharp drop in the power-trace at time C. Such events are treated as features by the model, resulting in false positives, which are undesirable in safety-critical systems. To make the model insensitive to such undesirable features, we apply exponential smoothing [42] and Holt's additive model based smoothing [42] with a range of parameter values to the segment $[C-u, C+u]$, where u is the fraction of voltage samples before and after the time point C, as shown by the red shaded region in Figure 8(b). As shown in Figure 8(c) and Figure 8(d), the selected values of the smoothing parameters are such that the generated variants of the segment closely resemble the characteristic of the power-trace about the end of the transmission.

Using the set of complete transmissions and aborted transmissions, we trained and evaluated our proposed authentication model. The results of the evaluation show that the proposed technique treats incomplete transmissions as non-transmission with an accuracy of 100%, a precision of 99.65%, a recall of 100% and an F-score of 99.68%.

3.8.8 Experimental Factors

Power consumed by an ECU is not immune to changes in the bus parameters and other hardware configurations. The variations introduced in the physical properties of the CAN bus may manifest in the measured power-consumption-measurements of the ECUs resulting in power-traces that deviate from the nominal power-traces. The presence of variations implies that the models should be updated upon parameter changes. However, due to the complex structure of the modern automobile system, it is not feasible to update models upon every hardware or firmware update. Therefore, we study the impact of the variations in the factors such as bus speed, message format and embedded programs on the power-consumption-measurements of the ECUs.

We tested the accuracy of the classifier against a set of factors and the corresponding levels. We selected 125 kbps, 250 kbps and 500 kbps as the levels of the bus speed. These

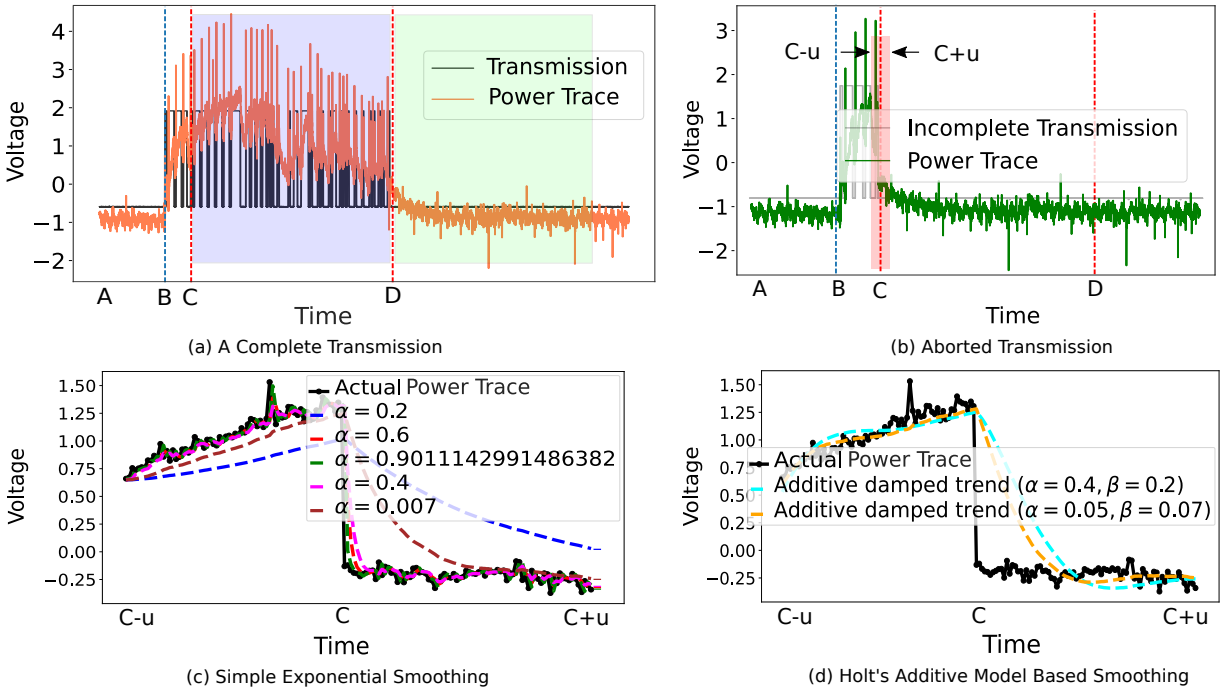


Figure 3.12: Steps in generating incomplete transmissions: (a) shows the voltage signal of the CAN bus and the power-trace of the Engine corresponding to a transmission, which starts at B, ends at D, C is the 50^{th}% of transmission bits, (b) synthesized aborted transmission, (c) exponential smoothed segment $[C+u, C-u]$ with different smoothing levels (α), (d) smoothed segment $[C-u, C+u]$ using Holt's additive damped trend model with different values of smoothing level (α) and smoothing trend (β)

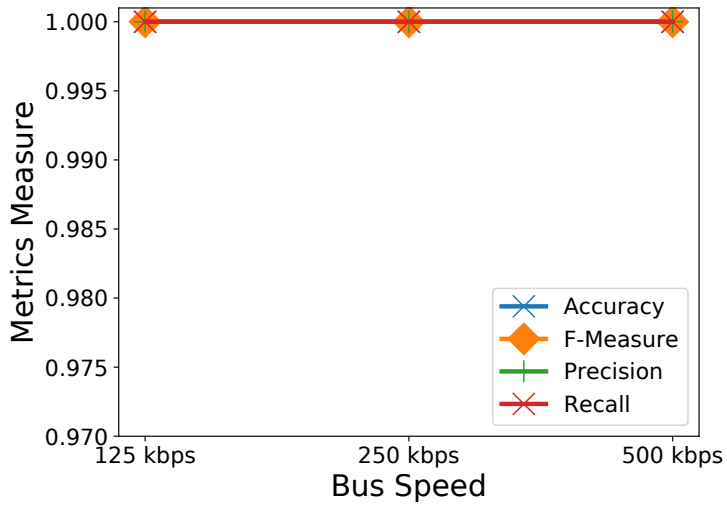
levels of the bus configuration are found in the majority of the modern automobile with 125 kbps used for low-speed CAN communications and 500 kbps for high-speed communications [84]. We selected two levels of message format: standard (11-bit) and extended (29-bit). The two levels are also the only possible format of CAN frames in a CAN protocol. For the source code as the factor, we selected two levels: uniform and heterogeneous. At the uniform level, we executed the same program before and after all the transmissions. Whereas, at the heterogeneous level, we executed a randomly picked source code from amongst a suit of mibench source code [37] before and after the transmissions.

Table 3.9: Experiemntal factors and correponding levels

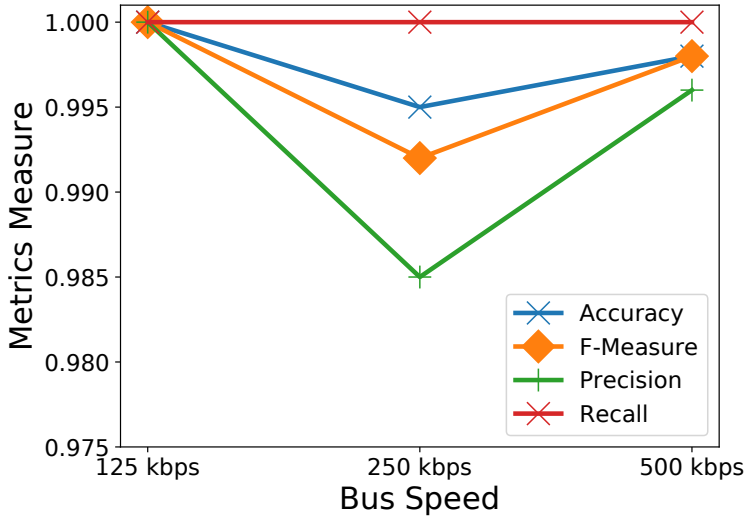
Parameter	**Values**
Bus speed	125 kbps, 250 kbps, 500 kbps
Message format	Standard (11-bit), Extended (29-bit)
Embedded programs	Uniform, Heterogeneous

If we test the impact of the combinations of all the three factors at all the given levels then there will be as many as 25 possible interactions for which we will need to conduct experiments. And the number of experiments will only become approximately intractable as the number of factors and levels grow with the complexity of the system. Therefore, to achieve the task with a limited amount of resource at hand, we designed a factorial screening experiment [66]. Factorial design analysis helps to filter out all the critical interactions between factors and the corresponding levels. Based on the analysis of variance experiment, four of the 25 interactions are reported as most significant.

We conducted the experiments for all the significant interaction in the lab setup on ECU1. For every experiment, we captured and decoded 5k transmissions. Using the decoded transmissions, we calculate the value of τ for the experiments with the bus speed of 125 kbps, 250 kbps and 500 kbps as 0.99 ms, 0.75 ms, 0.50 ms respectively. Using the value of τ corresponding to the bus speed, we constructed power-traces of transmissions and non-transmissions for the ECU across all the experiments. We trained, cross-validate and evaluated the model using the train, cross-validation and test set of power-traces. Figure 3.14 and Figure 3.13 shows the results of classification. Results show that the proposed technique is more accurate with simple bus configuration (125 kbps bus speed, standard format, uniform source code) than with advanced configurations (500 kbps bus speed, extended format, heterogeneous source code). However, the subtle difference between the accuracy of the two extreme network configurations shows that the impact of the varia-

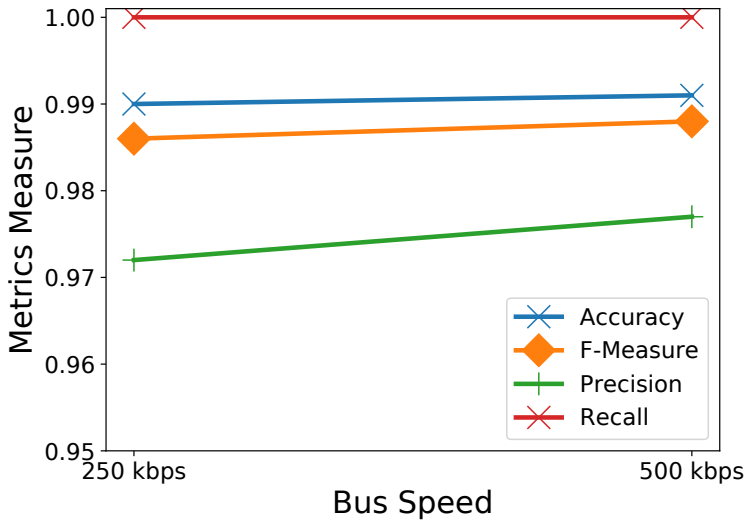


(a) Standard format and uniform source code.

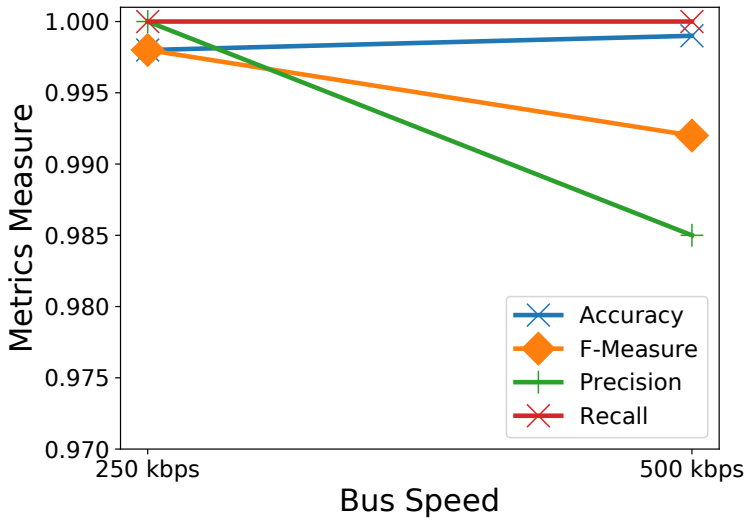


(b) Standard format and heterogeneous source code.

Figure 3.13: Accuracy of CANOA with first two significant interactions between factors and levels.



(a) Extended format and heterogeneous source code.



(b) Extended format and uniform source code.

Figure 3.14: Accuracy of CANOA with another two significant interactions between factors and levels.

Table 3.10: Important interactions between factors and levels

Combination	**Bus speed**	**CAN format**	**Source code**
1	125 kbps	Extended (29-bit)	Uniform
2	125 kbps	Standard (11-bit)	Uniform
3	250 kbps	Standard (11-bit)	Uniform
4	500 kbps	Standard (11-bit)	Uniform
5	125 kbps	Standard (11-bit)	Heterogeneous
6	250 kbps	Standard (11-bit)	Heterogeneous
7	500 kbps	Standard (11-bit)	Heterogeneous
8	250 kbps	Extended (29-bit)	Heterogeneous
9	500 kbps	Extended (29-bit)	Heterogeneous

tions of the factors has a negligible effect on the state classification accuracy of the ECU. And hence, the proposed approach can be applied to practical settings without worrying about the impact of small and frequent changes in the bus properties.

Chapter 4

A Saliency Map-based Interpretation of Model Outcome

Recent advances in the field of deep neural networks have led to widespread applicability of artificially intelligent systems in the field of computer vision for the task of object detection [32], image classification [86], segmentation [59], image captioning [106], visual question-answer [6]. Despite the significant advances in the speed and accuracy of neural networks, the complexity of the models makes the human-level understanding of the model's decision-making a challenging problem. Notably, the highly non-linear interactions between the layers of the network make the outcome unintuitive and unpredictable. As a result of their inexplicable nature, their applicability remains limited in the domain of safety-critical systems (medicine, automotive, robotics, finance, nuclear) where a decision based on the outcome can lead to fatal consequences.

Some of the research in the direction of explainable AI elucidate instances reflecting on the unpredictable nature of complex machine learning systems. For instance, in [83], the author shows how bias manifested in the machine learning algorithm through data leads it to misconstrued the characteristics of the snow for that of *husky*. Another work by Stock et al. [98] demonstrates the ImageNet [86] bias introduced in the ResNet [39] model. As a consequence of the bias, the model prefers the image of a black person with a basketball for the class basketball, and Asians in red dress for the *ping pong* class. Athalye et al. [7] show the sensitivity of the model can lead to misclassification. In the paper, the authors demonstrate that adding imperceptible perturbation to the input causes the model to misclassify the image of the turtle to the class of *rifle*.

All the above examples scenarios show the unpredictable nature of model prediction in

the presence of uncertainty. Consequently, such advanced AI systems cannot be reliably used for decision making in critical systems that demand explanation and verification.

As a consequence of the black-box nature of complex AI systems, many possible solutions for understanding and interpreting the complex machine learning models have been developed in the last couple of years. One such technique is to visualize the activations of the individual layers of the network [109]. However, for a particular image, this method is only able to tell apart what neurons are important for the classification of the input to a certain class. Saliency map-based methods exist [95] that localize the input pixels which are sensitive towards the classification of the input to an output class. However, one of the limitations of these techniques is that they are intrusive; that is, they require access to the network parameters and gradients flowing through the network to localize the important input pixels. Application of such a model to understand the decision making of complex machine learning tasks (navigation, object mapping, and lane detection) in a safety-critical system makes it vulnerable to adversarial access. Thus, there is a need for a non-intrusive explanation technique for target-specific model outcomes.

4.1 Related Work

With the increasing applicability of complex machine learning models, the need for an explainable and verifiable AI is increasing. In an attempt to justify models' outcomes, a variety of techniques have been proposed over the years. The majority of the explainability techniques fall into one of the three categories: (1) *third-party explainer*: a separate model for explaining the outcomes of the base model (2) justify the base model outcome using techniques such as input perturbation and network parameters.

Some of the explainability work that falls in the line of *third-party explainer* include [75, 47]. In particular, [40] uses the class discriminative properties of the objects in the images to provide a textual explanation for the images. [75] is another technique that trains two models for providing textual as well as visual justification for the visual question answering task and activity recognition task. However, these approaches are costly to implement because of the reliance on the availability of large human-annotated ground-truth explanations.

Within the realm of justifiable models, a variety of explainability approaches have been developed. One of the earliest approaches [83] attempts to provide a linear interpretation within the local neighbourhood of the data point. However, the approach is not effective at explaining non-linear models. Some of the approaches [95, 109, 73, 97] attempts to synthesize input images that result in a high activation score for particular neurons. Another

approach CAM [112] generates a target-specific saliency map by taking the global average pooling of the feature maps at the layer before the fully connected layer. GradCAM [90] is a generalized version of CAM that, in addition to the feature map weights, feeds the class gradient to the fully connected layer to assign importance to each of the input pixels. However, [112, 90] can only be applied to limited network architectures with global average pooling. Another work by Zhang et al. [111] proposes to use a backpropagation scheme to generate an attention map by propagating the signal downward through the network hierarchy using a winner-take-it-all strategy. A few techniques examine the relationship between input and output to learn a perturbation mask by backpropagating the error signal [29].

Despite the ability of the techniques [112, 90, 29, 78] to justify the model's decision, the methods mentioned above have limitations. The methods [90] are constrained by the use of network parameters such as gradients flowing through the network and network layer weights. While techniques such as [111, 90] require a specific kind of network architecture, in some cases [109], the method requires access to intermediate layers of computation for visualizing the features at several layers. Furthermore, the techniques can explain only a particular input at a time, without taking into consideration the possible variants (rotation, inversion, deformations) of the image. A work by Kim et al. [47] proposes a technique to provide an explanation that is representative of user-defined concepts, but the manually generated concepts limit the technique. Our work is an extension of the work by Petsuik et al. [77] to localize and generalize the salient pixels of the target class using a saliency map. We obtain a saliency map (using N less than that of [77]) by empirically optimizing the pixels important for target-specific classification. And, we propose an approach to provide a global perspective on the explanation of the outcome using a reconstruction technique, which generates possible variations of the salient pixels of the input.

4.2 Terminology

This section defines the terms that are used in the following sections:

1. **Important input samples**: Given an input vector, a classification model, and an expected outcome of the model given the input vector, an important input sample is a sample from the input vector which when replaced with zero or value outside of the input distribution will lead to a reduction in the confidence of the model in the expected outcome.

4.3 Problem Statement

The limited application of machine learning models in safety-critical systems forms the motivation for our problem which is stated as follows:

Given a classification model, an input image, and a probability distribution over a categorical target variable, identify the individual input samples that are important for classification of the input to the target class.

4.4 Contribution

Stochastic nature of machine learning models makes them less reliable for application in safety-critical systems. To fully exploit their capabilities, it is essential to make the models interpretable. Therefore, we propose a non-intrusive interpretability technique by generating a saliency map based target-specific model outcome explanation.

Inspired by the work of Petsuik et al. [77], we propose a non-intrusive explainability technique by generating a saliency map for a target class in less number of iteration ($N \sim 1000$) than [77] ($N \sim 5000$). We use an *empirical risk minimization* approach with a randomly initialized mask to locate the input pixels sensitive for the classification of the input to the target class. Therefore, if for the masked input, the confidence of the model in the most probable class is given by p, then the optimal set of pixels for the input is empirically located by randomly retaining $p\%$ of the unmasked pixels (with value > 0) and $(1-p)\%$ of the masked pixels (with value zero) followed by weighing the pixels using the class score.

4.5 Mathematical Formulation

Let I denote an input color image of dimension $H \times W$ from the space of images $\mathcal{I} = \{I : \Lambda^{H \times W} \rightarrow \mathbb{R}^3\}$ that maps each pixel coordinate to three color values,

$$\mathcal{I} = \{I | I : \Lambda \in \{1, ..., H\} \times \{1, ..., W\} \rightarrow \mathbb{R}^3\}$$

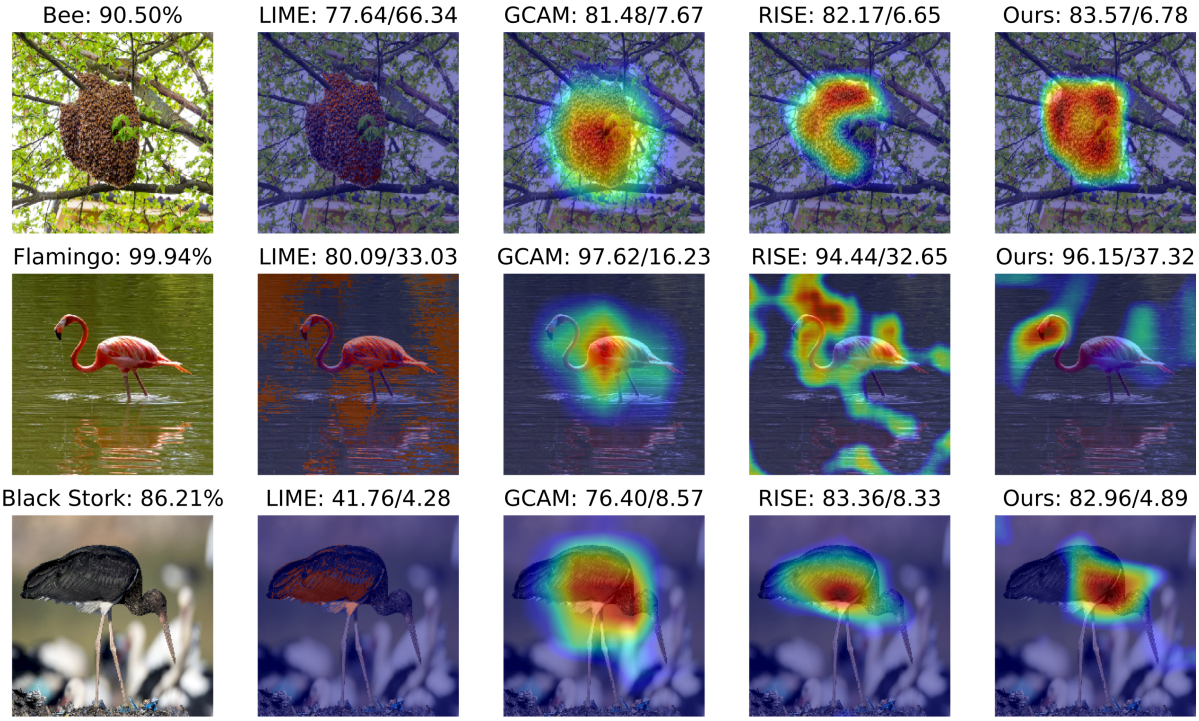


Figure 4.1: Saliency map generated for the target-specific image classification using our approach, RISE [77], GCAM [90], and LIME [83]. The first column shows the input image along with the top predicted class of the model outcome and the accuracy of classification. Second column onwards shows the saliency map overlapped with the input image and the AUC scores (%) of insertion/deletion metrics [77] where a higher value is considered good for insertion, and a lower value is considered good for deletion.

Let a random variable $\mathcal{C}$ represent a categorical variable.

$$\mathcal{C} = \{c_1, c_2, ..., c_x\}$$

Let $c \in \mathcal{C}$ denote a target class.

Let a function f denote the classification task.

$$f : \mathcal{I} \rightarrow \mathbb{R}^{\mathcal{C}}$$

where f maps inputs from the input space, $\mathcal{I}$ to a vector of real numbers signifying the strength of the classifier in the target output classes, $c \in \mathcal{C}$.

Let a random initial mask be denoted as,

$$M_0 = \{\Lambda^{h \times w} \rightarrow [0, 1], h < H, w < W\}$$

M_0 is composed of a set of unmasked, Λ_{on}, and masked pixels, Λ_{off}.

$$\Lambda^{h \times w} = \Lambda_{\text{on}}^{h1 \times w1} \cup \Lambda_{\text{off}}^{h2 \times w2}$$

and,

$$M_0(\Lambda^{h \times w}) = M_0(\Lambda_{\text{on}}^{h1 \times w1}) \cup M_0(\Lambda_{\text{off}}^{h2 \times w2})$$

Thus, given a binary mask M_0, an input pixel $\lambda \in \Lambda_{\text{on}}$ is preserved if $M_0(\lambda) = 1$, and the input pixel $\lambda \in \Lambda_{\text{off}}$ is masked if $M_0(\lambda) = 0$.

Let M be the mask at i^{th} iteration, then masked input $I' = (I \odot M)$ where $\odot$ denotes the element-wise multiplication between the original input I and the mask M.

4.6 Proposed Technique

Our proposed technique for the explanation of model outcome is comprised of two sub-problems. First, to generate a relevance mask highlighting the input samples which are important for the classification of the input to the target class. And second, to identify all the possible variations of the relevant input samples.

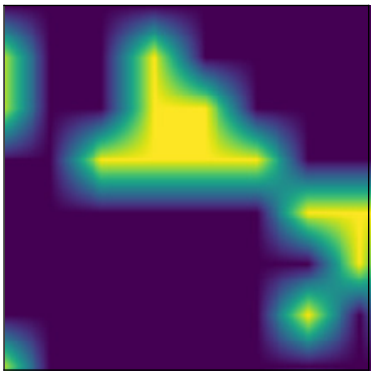


Figure 4.2: A random initial mask with preserved input pixels highlighted in yellow and masked input pixels highlighted in blue

4.6.1 Saliency Map Generation

We use the *empirical risk minimization* approach to find a mask M with a minimal set of unmasked pixels $\Lambda_{\text{on}}^{\text{min}}$. The mask is such that the model's confidence in class c using the masked version of the input departs from the confidence with that of the actual input by no more than a predefined threshold δ.

$$\begin{aligned} &\underset{\Lambda_{\text{on}}}{\text{minimize}} \quad M \\ &\text{subject to} \quad |f(I \odot M) - f(I)| \leq \delta \end{aligned} \tag{4.1}$$

In the equation 4.1, δ is a threshold determined empirically and $\odot$ is the element wise multiplication of the input with the mask. The key to a good estimate of the saliency map, which satisfies the optimization condition in Equation 4.1, is to iteratively update the initial mask M_0 by selectively filtering the masked and unmasked pixels from Λ_{on} and Λ_{off} based on the prediction probability $p = f(I \odot M)$ of the class c, where M is the upsampled version of the mask at the i^{th} iteration. M is upsampled using bilinear interpolation, as shown in Figure 4.3, to the size of the input image as shown in the second column of Figure 5.4. The bilinear interpolation is a common resizing technique in computer vision that helps avoid the inclusion of unwanted artifacts in the mask during empirical optimization by blurring out the edges, as shown in figure 5.4, thus, eliminating misclassifications.

Algorithm 3 shows the saliency map generation method. Based on the prediction probability of the target class c, the algorithm randomly retains $p\%$ of the unmasked pixels, Λ_{on} and $(1-p)\%$ of the masked pixels, Λ_{off}. The subset of the pixels preserved

Algorithm 3 Saliency map generation

Input: Input image $I \in \mathcal{I}$, Target class $c \in \mathcal{C}$
Output: saliency map M

Initialisation : $M_0 \in \mathbb{R}^{h \times w}$
for $i = 1$ to N **do**
    $M \leftarrow$ upsample M_{i-1}
    compute $p_i = f(I \odot M)$
    if i == 0 **then**
        $\Lambda_1 \leftarrow$ randomly select $0.5 \times p_i$ pixels of Λ_{on}
        $\Lambda_2 \leftarrow$ randomly select $0.5 \times (1 - p_i)$ pixels of Λ_{off}
    end if
    $\Lambda_1 \leftarrow$ randomly select $n_1 p_i$ pixels of Λ_{on} $\triangleright$ n_1 is the number of pixels in the set Λ_{on}
    $\Lambda_2 \leftarrow$ randomly select $n_2(1 - p_i)$ pixels of Λ_{off} $\triangleright$ n_2 is the number of pixels in the set Λ_{off}
    $\Lambda_{\text{on}} \leftarrow \Lambda_1 \cup \Lambda_2$
    $\Lambda_{\text{off}} \leftarrow \Lambda \setminus \Lambda_{\text{on}}$
    if $(\lambda \in \Lambda_{\text{on}})$ **then**
        $M_i(\lambda) \leftarrow M_{i-1}(\lambda) \Delta p$
    end if
    if $(\lambda \in \Lambda_{\text{off}})$ **then**
        $M_i(\lambda) \leftarrow 0$
    end if
    $V = \sum_{x,y} \|M_{i-1}^{xy+1} - M_{i-1}^{xy}\|^2 + \sum_{x,y} \|M_{i-1}^{x+1y} - M_{i-1}^{xy}\|^2$
    $M_i \leftarrow M_i + \eta V M_{i-1}$
end forreturn M

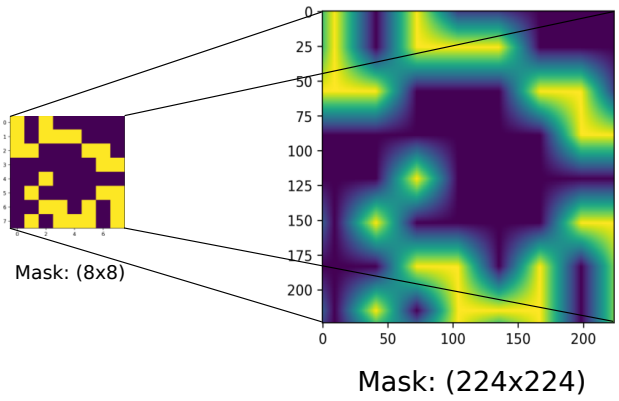


Figure 4.3: Mask upsample

from Λ_{on} and Λ_{off} forms the new set of unmasked pixels Λ_{on}, which are sensitive for the classification of input I to the target class c. However, the mask update is likely to saturate if the change in the prediction probability p of the target class is negligible. To overcome the issue of saturation, we add a regularizer, which penalizes the set of unmasked pixels according to two factors: the change in the prediction probability of the target Δp and the total variation of the mask V where Δ denotes the change in the value from the previous iteration. If the mask is invariant to small changes in the classification accuracy, then the regularizer will heavily penalize unmasked pixel values by adding a large negative penalty. On the other hand, if the change in Δp is significant, the regularizer adds a low penalty to the updated mask. This way, M captures the pixels sensitive towards the classification of I to the target class c. In the update equation for the mask, η is a constant that determines the amount of change in the total variation of the mask to retain.

4.6.2 Relevance Mask Generation

We use an expectation-maximization optimization technique to estimate a relevance mask M_{est} using the input vector I and the target class variable $c \in \mathcal{C}$.

- In the first step, we initialize a random real-valued mask M_0. To generate the mask, we first intialize a binary mask $M_0^{h \times w}$ where $h < H$ and $w < W$, using the technique in [77]. Followed by upsampling the mask to the size of the input image I using bilinear interpolation. Upsampling using interpolation has the effect of blurring the contrasting edges for a smooth transition from the non-masked region to the masked

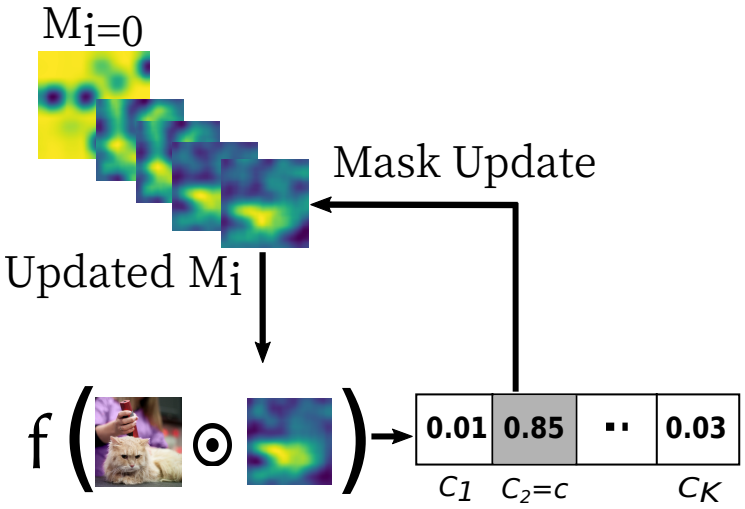


Figure 4.4: Overview of the mask generation approach

region as shown in figure 4.2. which may appear as a feature for the model when performing prediction on the masked input.

- In the expectation step, evaluate the likelihood of the masked input $I' = (I \odot M_i)$ given the target variable $c \in \mathcal{C}$

$$f(I \odot M_i) = \Pr\{c|(M_i \odot I)\} \tag{4.2}$$

- In the maximization step, maximize the expectation to optimize M_{i-1} by evaluating the likelihood at M_i and updating the mask. To update the mask, first calculate, Δp, the change in the prediction probability of the model using the masked input $(M_{i-1} \odot I)$ and $(M_i \odot I)$. The difference in the prediction probability of the outcome c denoted as Δp is computed as follows,

$$\Delta p = f(M_{i-1} \odot I) - f(M_i \odot I) \tag{4.3}$$

This is followed by updating the values of the mask M_i corresponding to the pixels in the revised set of preserved Λ_{on} and masked Λ_{off} pixels, which are obtained using steps 5-8 of Algorithm 3. If a pixel belong to the updated preserved set of pixels, $\lambda \in \Lambda_{\text{on}}$, then retain the value of the mask weighted by Δp, which signifies the degree of change observed in the output confidence score from M_{i-1} to M_i,

$$M_i(\lambda) = \begin{cases} M_{i-1}(\lambda)\Delta p, & \text{if } \lambda \in \Lambda_{\text{on}} \quad (4.4) \\ 0, & \text{if } \lambda \in \Lambda_{\text{off}} \quad (4.5) \end{cases}$$

We add a term to the updated mask called total variation, V, which acts as a regularizer and helps in a smooth navigation over the space of possible set of masks. The variation is calculated on the mask from previous step as the square of difference between every element across rows and columns,

$$V = \sum_{x,y} \|M_{i-1}^{xy+1} - M_{i-1}^{xy}\|^2 + \sum_{x,y} \|M_{i-1}^{x+1y} - M_{i-1}^{xy}\|^2 \quad (4.6)$$

Finally, update the mask using total variation, and the amount of variation to be retained is controlled by η using the following equation,

$$M_i \leftarrow M_i + \eta V M_{i-1} \quad (4.7)$$

- After every k iterations of mask update, check Δp to determine the magnitude of change from previous iterations, if at a point the fluctuations are negligible as compared to the change in previous steps, and the magnitude falls within a range $(\Delta p - \epsilon, \Delta p + \epsilon)$, then stop.
- The process is repeated until N steps are completed. At $i = N$, the estimate of the learned relevance mask is given by $M_{est} = M_N$

As shown in the Figure 4.5, during the optimization, the algorithm navigates an arbitrary space of masks $M \in \mathcal{M}$ to find the best estimate of the relevance mask M_{est}. At the end of each iteration, the algorithm finds an estimate of the mask M_{i+1}, which is at least as good as the current estimate M_i. The total variation value at each iteration determines the degree of variation in the relevance of the preserved set of input pixels from one iteration to another. Over the iterations, the pixels which are rewarded with more positive weights contribute more in deciding the outcome of the input. At the point of convergence, the algorithm returns an estimate of the relevance mask M_{est} with the preserved relevant set of input pixels, which will explain the model's outcome for the classification of the input I to the target c.

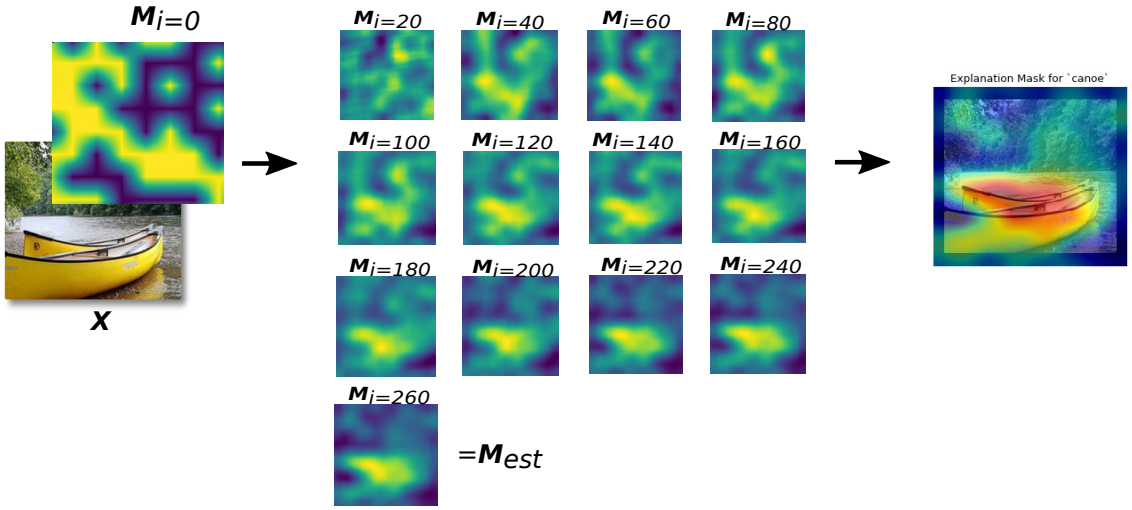


Figure 4.5: Visual representation of the relevance mask during mask optimization

4.7 Experiments

4.7.1 Model and Data Description

:

We evaluate the efficacy of the proposed saliency map based explainability approach using a range of publically available open image datasets such as ImageNet [86] and MS-COCO (Microsoft-Common Object in Context) [57]. ImageNet is a repository of 15 M high-resolution images gathered from more than 20 k categories. MS-COCO, on the other hand, is a significantly smaller database of images but with more number of instances per category. The dataset has 330 k images with more than 80 object categories. The MS-COCO dataset is used for object detection, object segmentation, and image captioning.

We use pre-trained models: VGG16 [96], Inception V3 [100], and ResNet50 [39] as base models for image classification. The pre-trained models are loaded with ImageNet weights and accepts inputs of size 224×224. The models differ in terms of their network structure and number of trainable parameters. VGG16 is a convolutional neural network trained on ImageNet dataset achieving top-5 accuracy of 92.0% on ImageNet. ResNet50 is a 50 layer network achieving 93.29% accuracy on the ImageNet. Inception V3 is another model that achieves an accuracy of 94.4% on the ImageNet. Relying on pre-trained models [96, 100, 39] for image classification helps avoid the common training pitfalls such as model over-fitting,

skewed data distribution, right model selection, and insufficient resources (such as GPUs) for training.

4.7.2 Evaluation Metrics

Motivated by [77, 47], we use the metrics of insertion and deletion to evaluate the efficacy of the mask generation approach for explaining the model decision.

Insertion and deletion metrics

Motivated by [77], we use the metrics of insertion and deletion to evaluate our saliency map approach. In the deletion metric, the deletion of salient pixels from the input causes the model to drop the probability of the target class. And in the insertion metric, the insertion of pixels from the relevant region of the input causes the model to increase the probability of target class. We capture the sensitivity of the model to the removal and insertion of pixels from the relevant region of the input using an average AUC score. Thus, during deletion, as the relevant input pixels are deleted from the masked input, the AUC curve for the model will shrink to a thin area, thus, dropping the average AUC score, indicating the right explanation for the model decision. Similarly, during the insertion, as the pixels from the relevant region of the input are added to the masked input, the AUC curve expands to cover the large area under the probability curve, thus increasing the average AUC score.

Pointing game

The pointing game [111] metric is a method of evaluating the class discriminative nature of saliency map based approaches. Given an annotated segmentation box for an instance of an object and the corresponding saliency map, the method measures the number of pixels on the saliency map of the input that lies on the annotated box of the object instance. To measure the overlap, we calculate the fraction of the area (A_s) of the saliency map that overlap with the annotated segmentation box (A_t) of the image using an IOU,

$$\text{IOU}score(\%) = \frac{\sum(A_s \cap A_t)}{\sum(A_s \cup A_t)} \tag{4.8}$$

The numerator is the sum of pixels values in the union of A_s and A_t, and the denominator is the sum of pixels values in the intersection of A_t and A_s. A high IOU score (typically $\geqslant 50\%$) means that a large fraction of the salient region of the input overlaps with the annotated box, indicating a good explanation.

4.7.3 Results

As a part of the result, we generate relevance mask for a set of images and analyze them for visual understanding of the models outcome.

4.7.4 Evaluation using Insertion/Deletion Metrics

Given a pre-trained classifier, we evaluate the class discriminative capability of saliency map based approaches [83, 90, 77, 97] using the quantitative measures of insertion and deletion metric. For the base models: VGG16 [96], ResNet50 [39], and Inception V3 [100] and the datasets: ImageNet [86] and MS-COCO [57], we report the average AUC score of insertion and deletion on a set of images. The test images are randomly selected from the test set of the datasets. For each technique, Table 4.2 shows the mean AUC score of the insertion and deletion metrics for each technique across all the models and all the datasets. We also show the standard deviation of the AUC of the insertion and deletion metrics for our approach. From the table, it is evident that our approach outperforms other saliency map based approaches in localizing the pixels sensitive for the classification of the input to the target class across both the datasets and all the base models.

Table 4.1: Mean AUC Score(%) using insertion (Ins) and deletion (Del) metrics - I

Model	Dataset	Ours		RISE [77]	
		Ins	Del	Ins	Del
ResNet50 [39]	MS-COCO [57]	**75.53/0.02**	**1.80/0.001**	73.71	4.14
	ImageNet [86]	**63.16/0.004**	**11.48/0.05**	60.32	13.04
VGG16 [96]	MS-COCO [57]	**64.64/0.001**	**4.49/0.003**	62.88	5.00
	ImageNet [86]	58.72/0.03	12.23/0.1	**59.47**	**12.66**
InceptionV3 [100]	MS-COCO [57]	**66.25/0.001**	**5.88/0.005**	65.87	5.00
	ImageNet [86]	**67.43/0.004**	**6.34/0.003**	65.12	10.25

4.7.5 Evaluation using Pointing Game

We evaluate the localization capability of saliency map-based approaches for target-specific objects using the pointing game. For the ImageNet dataset, we report the average IOU score for a set of test images across all the models. The IOU score is calculated for the

Table 4.2: Mean AUC Score(%) using insertion (Ins) and deletion (Del) metrics - II

Model	Dataset	GCAM [90]		LIME [83]		Guided Backprop [97]	
		Ins	Del	Ins	Del	Ins	Del
ResNet50 [39]	MS-COCO [57]	55.08	6.95	46.27	7.02	38.96	4.25
	ImageNet [86]	58.57	18.22	45.89	15.86	49.37	3.54
VGG16 [96]	MS-COCO [57]	40.03	10.07	37.94	7.06	38.96	4.25
	ImageNet [86]	51.59	15.75	45.75	16.23	49.20	3.13
InceptionV3 [100]	MS-COCO [57]	62.43	4.83	53.59	6.41	39.62	6.00
	ImageNet [86]	60.99	11.31	45.76	13.32	49.12	4.27

saliency map of the images using the enclosed area of the bounding box, A_s and the annotated segmentation box, A_t for the target-specific input images. Table 4.3 shows the mean IOU score (%) for the test set of images across all the models and both the datasets. The table also shows the standard deviation of the IOU score for our approach. From the table, it is evident that the performance of our saliency map approach across the models is at least 5% more accurate than [77] and 20% more accurate than [83] at localizing the pixels sensitive towards target-specific classification. Note that we omit the evaluation of the gradient backpropagation technique using the pointing game as the technique highlights only the edges of the target object, which is insufficient for evaluation against this metric.

Table 4.3: Mean IOU score(%) using pointing game metric

Model	Ours	RISE [77]	GCAM [90]	LIME [83]
ResNet50 [39]	**79.01/0.02**	74.9	69.11	57.29
VGG16 [96]	78.0/0.001	**81.12**	62.31	51.17
InceptionV3 [100]	**76.0/0.002**	63.23	57.54	48.21

4.7.6 Convergence

We show that our approach converges to localize pixels sensitive for the target class in less than half the number of iterations compared to [77]. To show this, we calculated the saliency map of a set of randomly selected images from class *honey bee* using our approach and RISE [77] for $N = 5000$. The criteria for choosing a particular image category includes the presence of at least one distractor object, which makes the target class discrimination

difficult. Figure 4.6(a) and 4.6(b) shows the insertion and deletion metric of the saliency map at every iteration. The error bar at each iteration shows the standard deviation of the average AUC. For every 1000^{th} iteration, we report the average AUC of insertion and deletion of the pixels using the saliency map generated using both the approaches. From Figure 4.6(a), we observe that the accuracy increases over iterations and then approaches a point after which updating the saliency mask does not improve the performance of insertion accuracy. This point is the point of convergence. The figure shows that our approach reaches the point of convergence faster than [77]. Furthermore, unlike [77], the monotonically increasing and decreasing curve of the mean AUC of insertion and deletion metric shows that the saliency map over iterations generated using our approach is more reliable for decision-making.

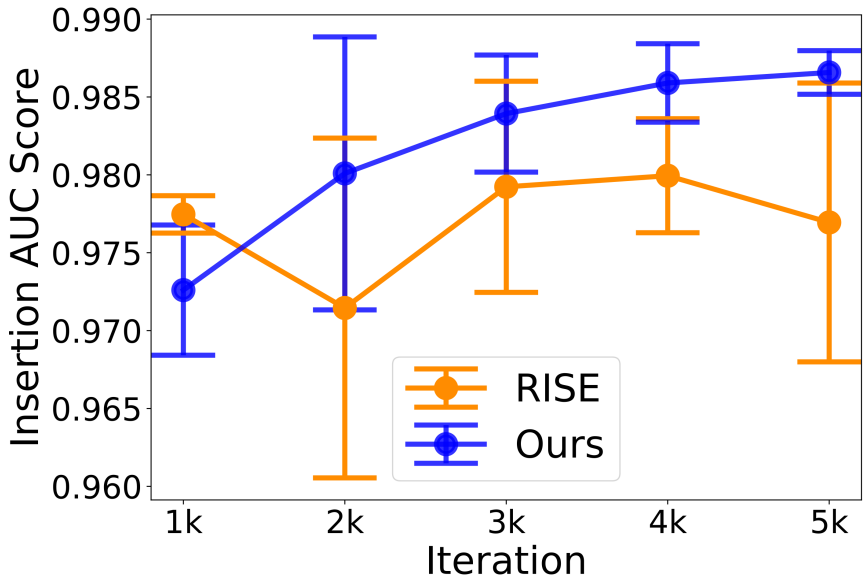
4.7.7 Saliency Map for Examples from ImageNet

We generate relevance mask M_{est} for randomly selected set of ImageNet images using our proposed approach alongside LIME [83], GradCAM [90], and RISE [77]. Figure 4.7 shows the input images in the first column, the corresponding estimate of the relevance mask M_{est} for the target object class, and the mask overlayed with the image for our approach in the last column. The mask overlapping the images across all the approaches shows the parts of the input highlighted with varying degrees of colour intensities showing the sensitivity of the model to the different parts of the input in outcome prediction. The sensitivity of the pixels varies from the most relevant input pixels (shown in red) to the least relevant input pixels shown in blue. The intermediate yellow region is the ambiguous region because the yellow region forms the part of both the relevant and irrelevant input pixels. For example, in Figure 4.7, the saliency map for the image of *banana* shows regions of the input, such as the highlighted yellow region of *grapes*, which also form the explanation for class *banana*. Similarly, for the image of *lynx* (third row in Figure 4.7), a fraction of the non-*cat* portion of the image around the face is assigned more weight than the non-relevant part (highlighted in blue), which also forms the part of the explanation for the *lynx* class. We believe that by penalizing the irrelevant input pixels more than the relevant pixels and by evaluating the approaches using the insertion/deletion metric, we are able to indicate the effectiveness of our approach. The insertion/deletion score reported for the images using our approach achieves a higher insertion score on average than all the other approaches, with significant improvement reported on the image of *banana* (first row). This is the case because our approach detects the smallest region of all the approaches to which the model is sensitive, resulting in an AUC score of insertion that is better than the related approaches. Similarly, the insertion obtained on the image from the fourth row for the target class *banana* achieves

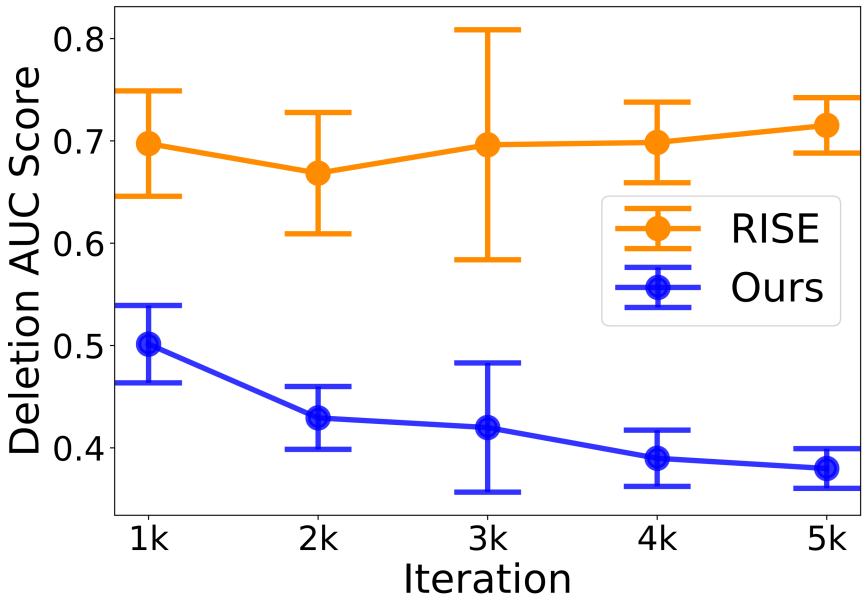
a score higher than all the other approaches by focusing on a significantly small region of the image, including mostly the features of the banana.

4.7.8 Evaluation using Ground-truth Annotations

We also evaluate our approach using pointing game metrics by visualizing the scores across a set of images from the ImageNet dataset. As shown in Figure 4.8, 4.9, 4.10, 4.11, 4.12, for each input image, we report the human-annotated ground-truth for the target class in the first column, the generated saliency map in the second column, and the bounding box for the saliency map in the third column. The ground-truth annotation for the images is available from the ImageNet dataset as the bounding box around the object of interest. We visualize the saliency map for the target class of the image by overlaying the generated relevance mask on top of the input image. To calculate the IOU score for the input image, we also obtain an estimate of the binary bounding box for the corresponding saliency map by assigning a value of 0's and 1's to the corresponding pixels using a threshold. The IOU scores for the images are shown in the third column. We can observe that IOU scores for some of the images are not indicative of the reported saliency map. For instance, the saliency map for the image of *whale* in the first row of Figure 4.8 focuses only on the features of the *whale* as relevant. However, the IOU score of the image of *whale* contradicts with a low score of $\sim$0.42. This is the case because, in addition to the features of the *whale*, the human annotation (shown in the first column) for the *whale* also comprises the background, including the features for the class of *sky* and the *sea*. Such erroneous annotation results in a higher value of the union in the denominator as opposed to the numerator, which is the intersection of the human-annotated bounding box and saliency map bounding box. Similarly, a low score of IOU for the image of *cucumber* can also be understood. The examples in Figure 4.10 reported an IOU score of zero. This is the case because there is negligible overlap between the bounding box for ground-truths and saliency maps for the image of *pomegranate*, *carton*, and *truck*. In particular, the human annotation for the class pomegranate wrongly annotates the part of the image containing apple as the bounding box for the image of *pomegranate*. Similarly, the ground-truth for the class of *carton* wrongly annotates the features of *orange* as the bounding box for *carton*, resulting in a zero IOU score. Therefore, we conclude that relying on one or the other evaluation metric can be unreliable to evaluate a saliency map-based approach. And we must consider an ensemble of the metrics for the evaluation.



(a)



(b)

Figure 4.6: The figure shows the AUC score of insertion 4.6(a) and deletion 4.6(b) for the saliency map of an input image using our approach and RISE [77] over the iterations.

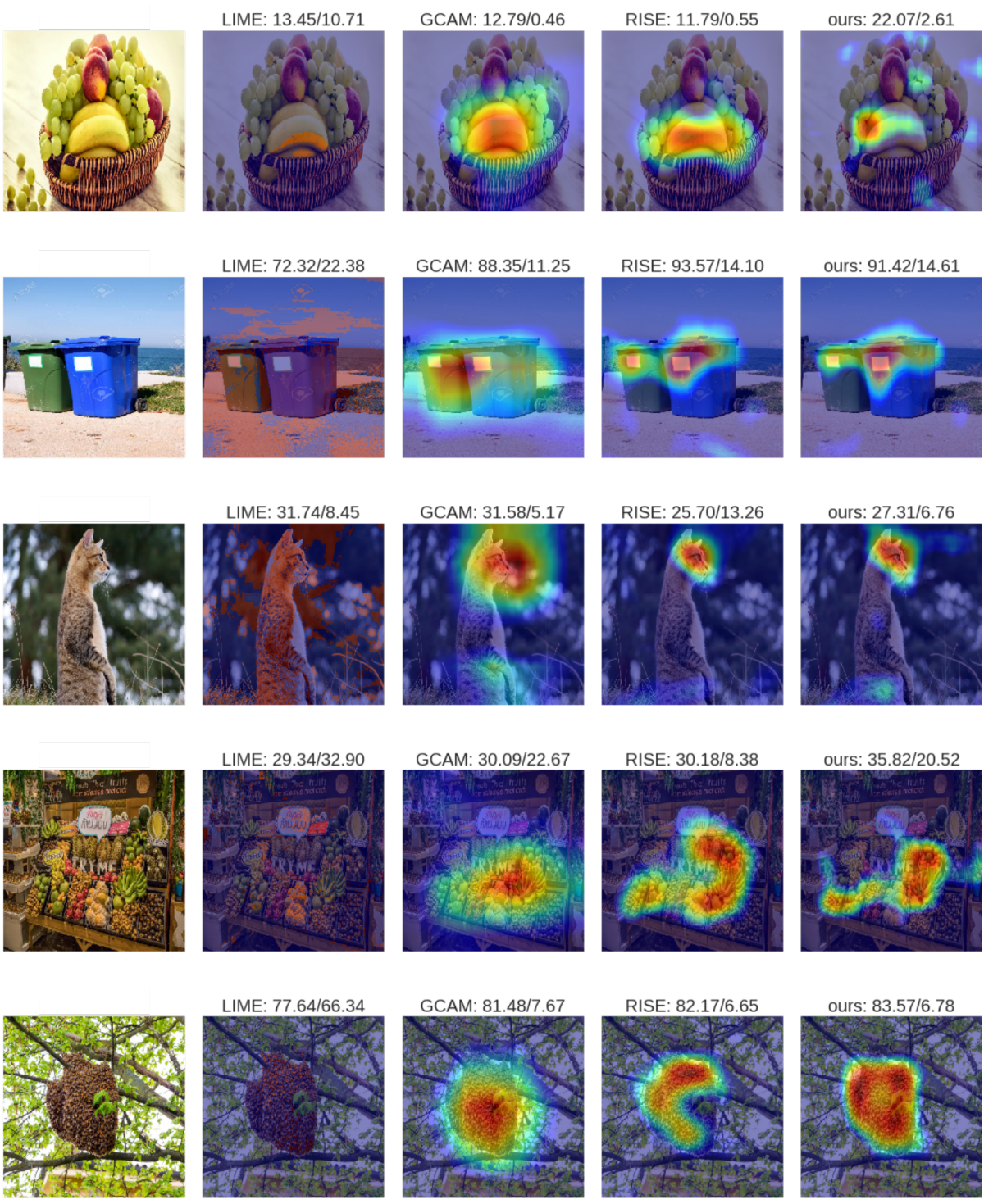


Figure 4.7: Saliency map comparison for the target-specific image classification using our approach, RISE [77], GCAM [90], and LIME [83]. The first column shows the input image along with the top predicted class of the model outcome and the accuracy of classification. Second column onwards shows the saliency map overlapped with the input image and the AUC scores (%) of insertion/deletion metrics [77] where a higher value is considered good for insertion, and a lower value is considered good for deletion.

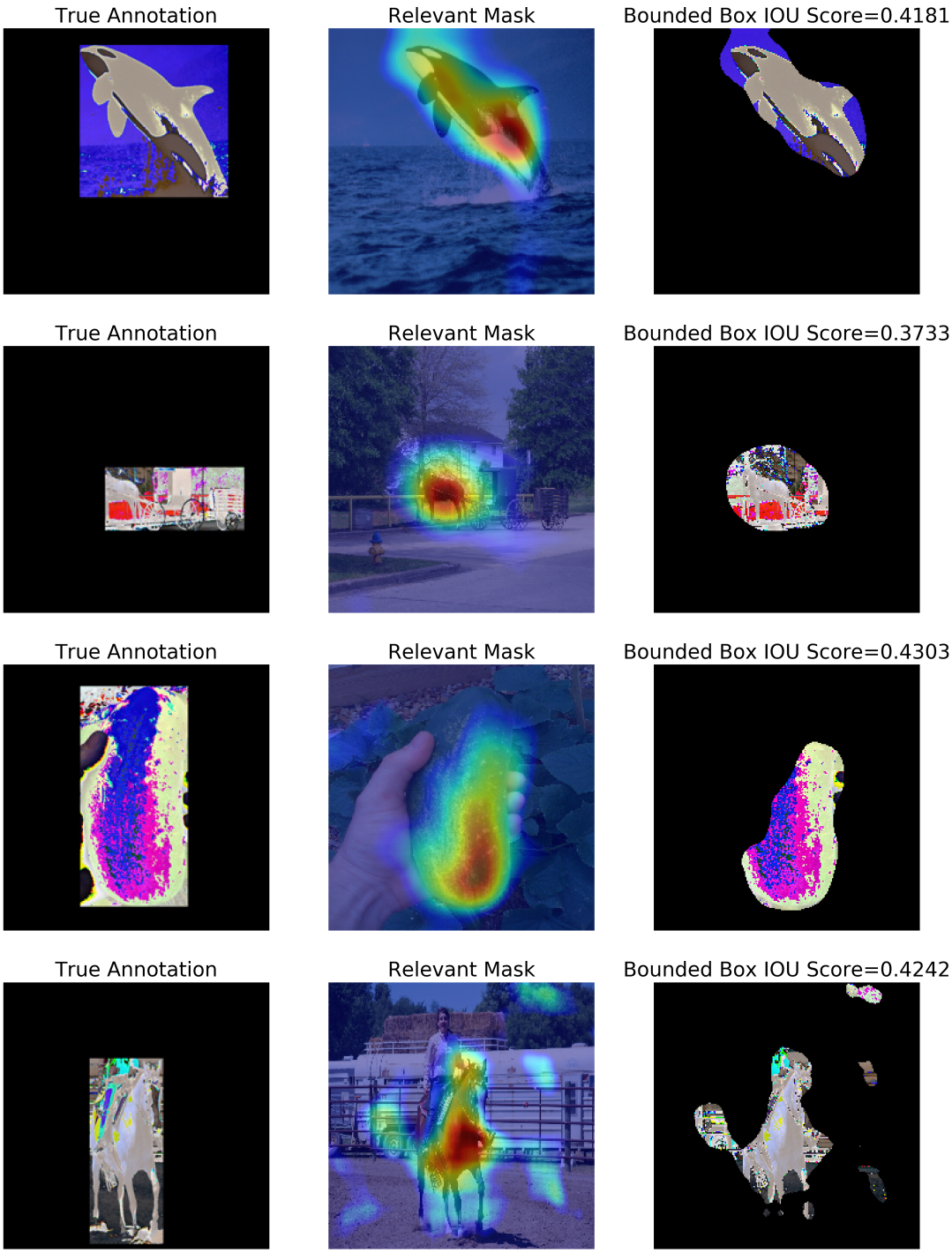


Figure 4.8: Evaluation of our approach on a set of ImageNet datasets using pointing game metric. The figure shows the human-annotated bounding box for the object of interest (target class), followed by the saliency map generated using our approach, followed by the binary bounding box for the corresponding saliency map. The figure also shows the IOU score calculated using the equation 4.8 with the bounding box for ground-truth and the saliency map of the input images shown on top of the figure.

Figure 4.9: Evaluation of our approach on a set of ImageNet datasets using pointing game metric. The figure shows the human-annotated bounding box for the object of interest (target class), followed by the saliency map generated using our approach, followed by the binary bounding box for the corresponding saliency map. The figure also shows the IOU score calculated using the equation 4.8 with the bounding box for ground-truth and the saliency map of the input images shown on top of the figure.

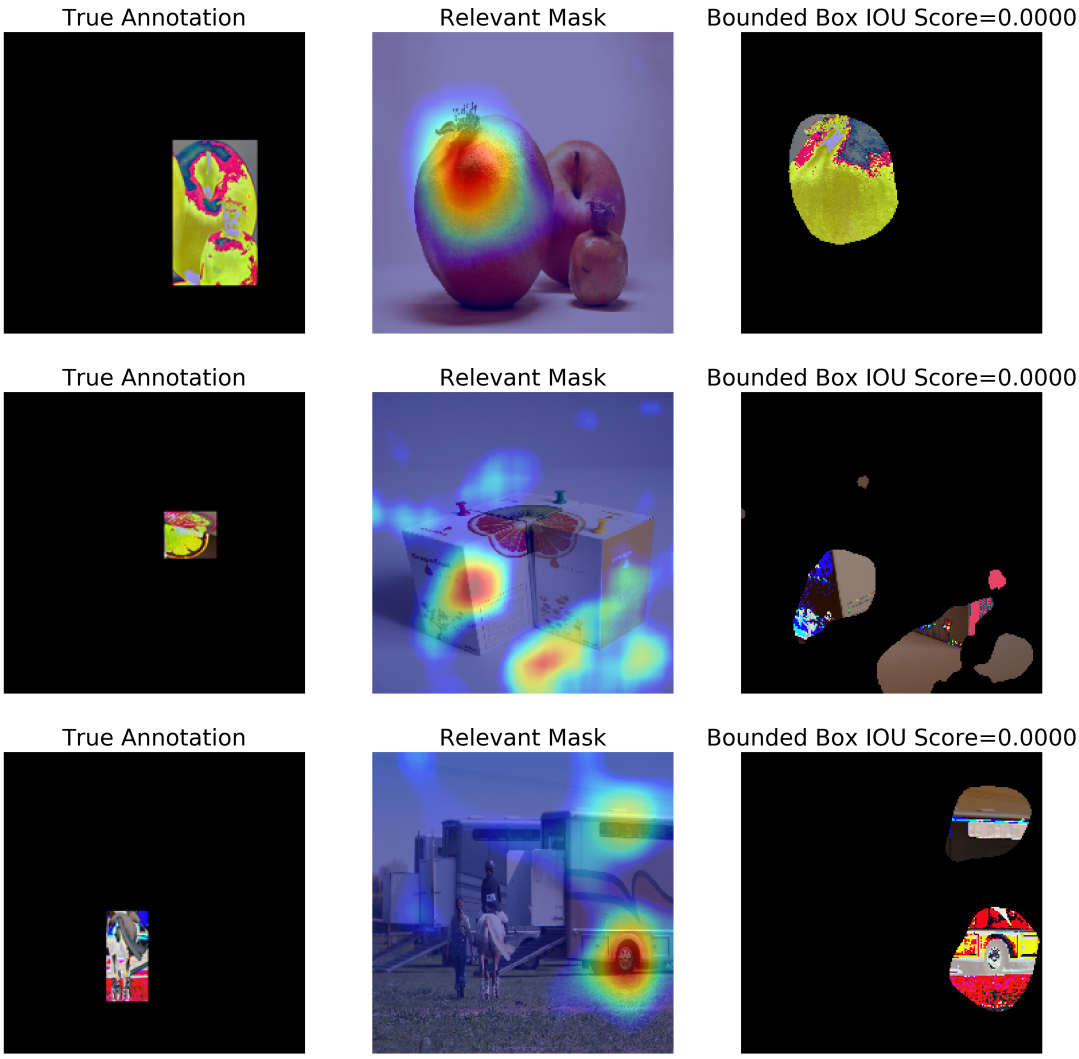


Figure 4.10: Evaluation of our approach on a set of ImageNet datasets using pointing game metric. The figure shows the human-annotated bounding box for the object of interest (target class), followed by the saliency map generated using our approach, followed by the binary bounding box for the corresponding saliency map. The figure also shows the IOU score calculated using the equation 4.8 with the bounding box for ground-truth and the saliency map of the input images shown on top of the figure.

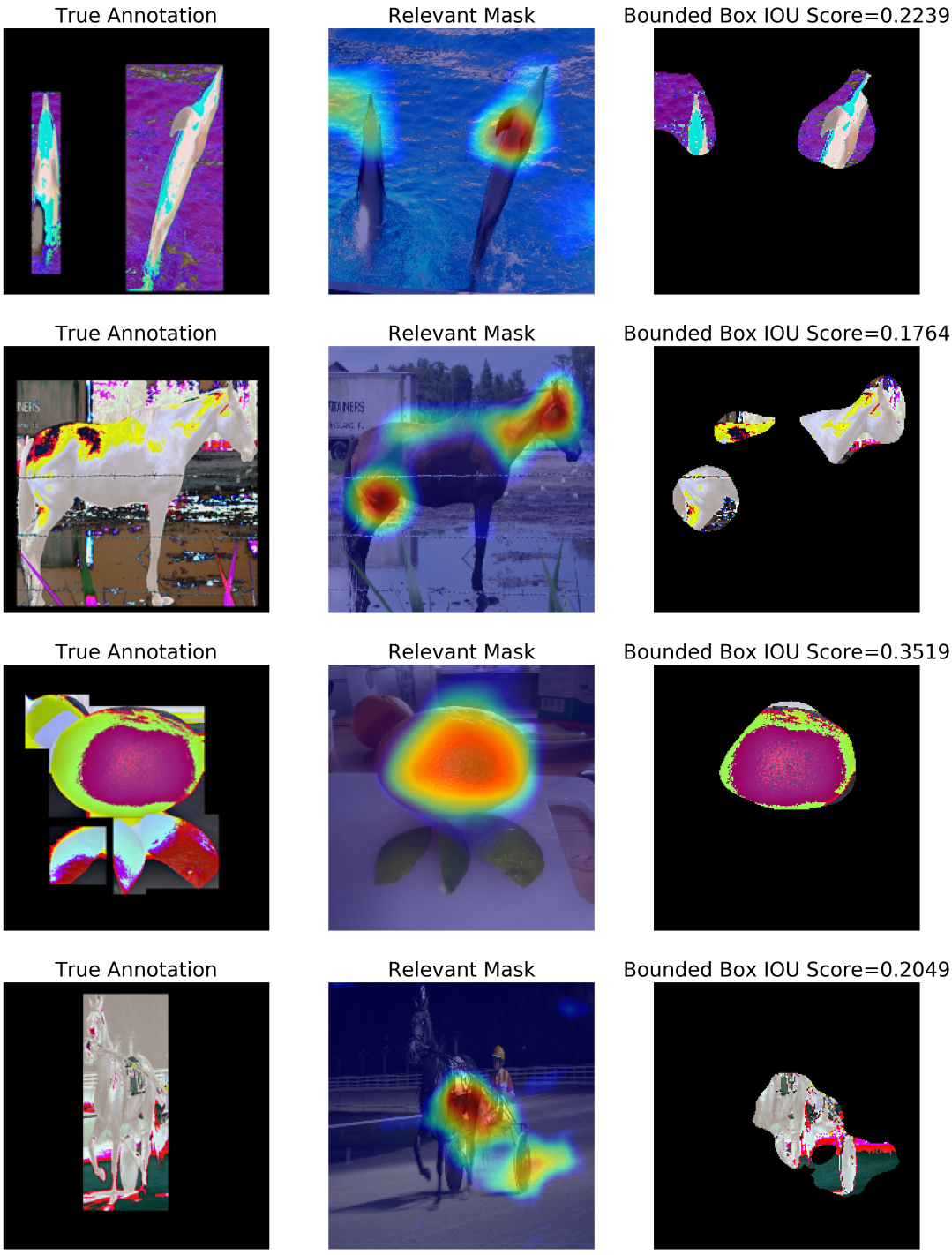


Figure 4.11: Evaluation of our approach on a set of ImageNet datasets using pointing game metric. The figure shows the human-annotated bounding box for the object of interest (target class), followed by the saliency map generated using our approach, followed by the binary bounding box for the corresponding saliency map. The figure also shows the IOU score calculated using the equation 4.8 with the bounding box for ground-truth and the saliency map of the input images shown on top of the figure.

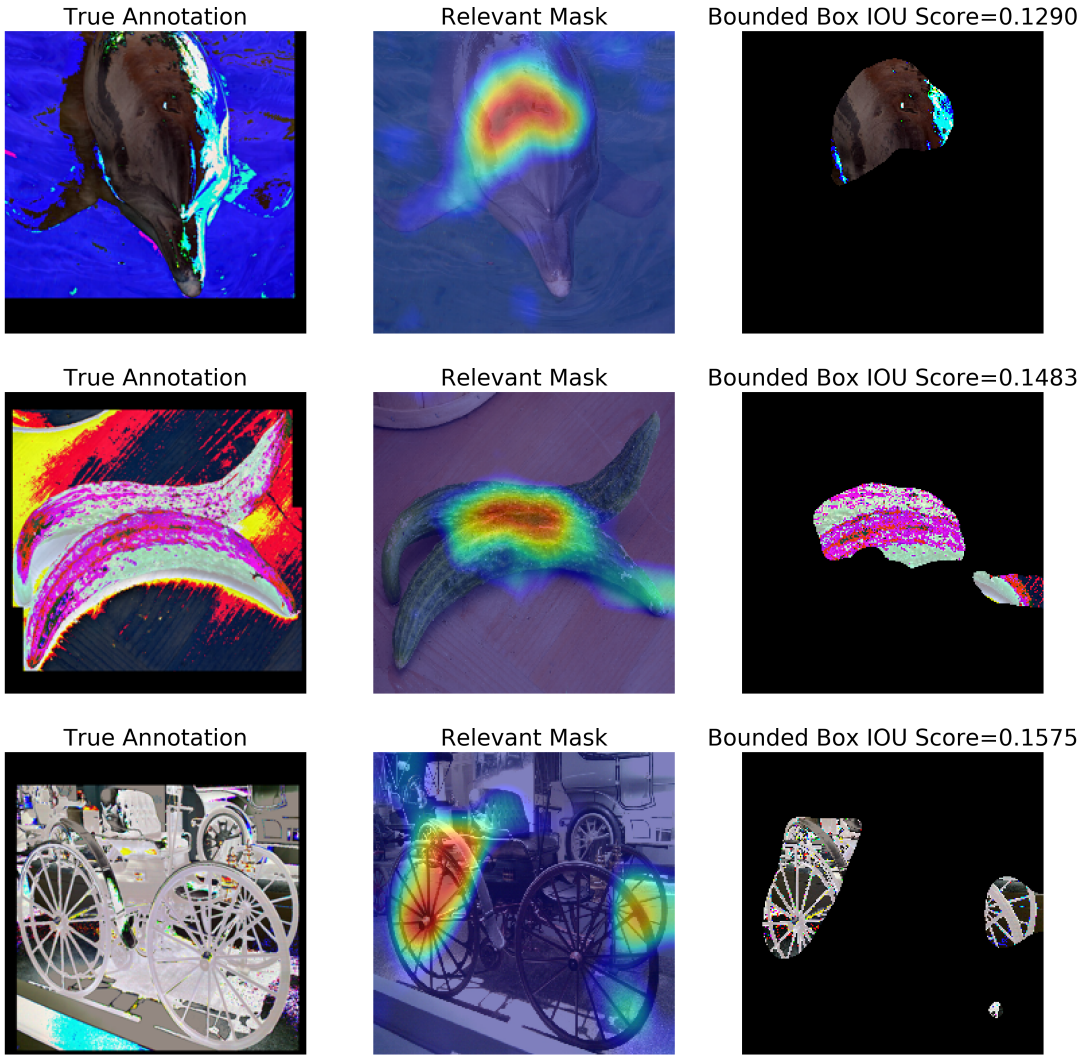


Figure 4.12: Evaluation of our approach on a set of ImageNet datasets using pointing game metric. The figure shows the human-annotated bounding box for the object of interest (target class), followed by the saliency map generated using our approach, followed by the binary bounding box for the corresponding saliency map. The figure also shows the IOU score calculated using the equation 4.8 with the bounding box for ground-truth and the saliency map of the input images shown on top of the figure.

Chapter 5

Generalizability of Saliency Map-based Explanation

The black-box nature of the machine learning models has led to extensive research in the field of model outcome interpretability and transparency. Most of the existing work in this area falls into two categories: First, assessing the sensitivity of the model to the changes in internal network parameters and the gradient flowing through the network in outcome prediction [47, 10, 20, 29, 75, 90]. And second, the identification of learned features in a low-dimensional linear subspace [83, 47]. The former is limited in its capability because it requires access to internal network parameters. Furthermore, the latter can only capture partial information. Therefore, we propose a model-agnostic interpretation method that iteratively revises a mask to converge to focus on the important regions of the image. The method assigns scores to individual pixels signifying their sensitivity toward the output class. A high score means the pixel is significant for classification to the output class and vice-versa.

One of the limitations of the perturbation-based explanation technique is that it lacks consistency. Multiple runs of explanation generated on the same image result in contradictory salient region detection for the target class. As shown in Figure 5.1, the input image of *broccoli* constitutes more several pieces of *broccoli* spread across the area of the image. And the saliency map highlights the salient regions of the input by assigning higher weights to the relevant pixels and lower weights to non-relevant pixels. However, the weights of the relevant pixels across multiple runs are not consistent. The first saliency map assigns more weight to the pixels constituting *broccoli* in the top half of the image than the pixels in the bottom half. However, there is no significant reason for differentiating the *broccoli* from the top half to the bottom half, and vice-versa. Also, it is noticeable that regardless of

the variations in the scores of the pixels, the scores for the region of saliency map remain the same with a tolerable range of variation. A work by Kim et al. [47] addresses the uncertainty of the saliency region by attributing user-defined concepts to the target class. However, the method relies on manually designing user-defined concepts, which is laborious and infeasible for large datasets. To ensure that the technique is reliable in the face of external noise, we propose a method to quantify the variations for the salient region of the input. To this end, we estimate the salient region for the input as the concatenation of the salient regions across multiple runs of the input instance. In turn, the set of variations for which the model outcome remains unaltered also forms a global perspective for the class of interest.

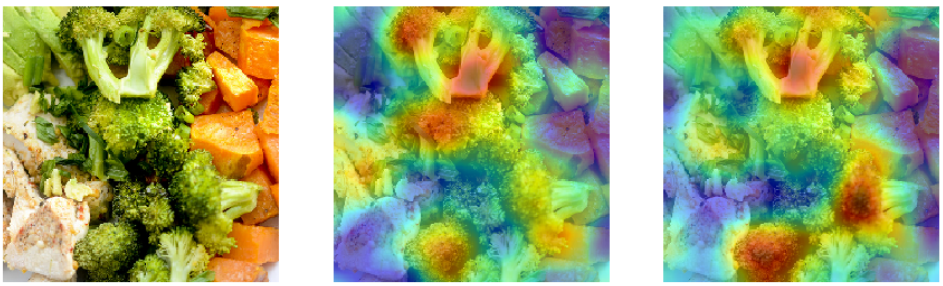


Figure 5.1: Saliency map for the image of *broccoli*. The two saliency map for *broccoli* highlights non-overlapping regions of the image as important for *broccoli* classification.

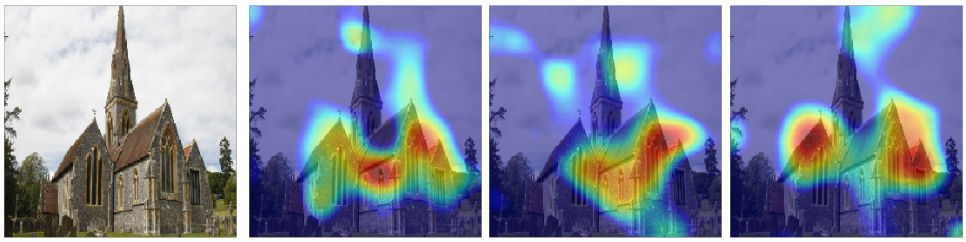


Figure 5.2: Saliency map for the image of a *church*. The three saliency map for *church* highlights a fraction of non-overlapping regions of the image as important for *church* classification.

5.1 Related Work

With the increasing applicability of complex machine learning models, the need for an explainable and verifiable AI is increasing. In an attempt to justify models' outcomes, various techniques have been proposed over the years. The majority of the explainability techniques fall into one of the three categories: (1) *third-party explainer*: a separate model for explaining the outcomes of the base model (2) justify the base model outcome using techniques such as input perturbation and network parameters.

Some of the explainability work that falls in the line of *third-party explainer* include [75, 47]. In particular, [40] uses the class discriminative properties of the objects in the images to provide a textual explanation for the images. [75] is another technique that trains two models for providing textual as well as visual justification for the visual question answering task and activity recognition task. However, these approaches are costly to implement because of the reliance on large human-annotated ground-truth explanations.

Within the realm of justifiable models, there are various explainability approaches. One of the earliest approaches LIME [83] attempts to provide a linear interpretation within the local neighborhood of the data point. However, the approach is not effective at explaining non-linear models. Some of the approaches [95, 109, 73, 97] attempts to synthesize input images that result in a high activation score for particular neurons. Another approach by [112] generates a target-specific saliency map by taking the global average pooling of the feature maps at the layer before the fully connected layer. GradCAM [90] is a generalized version of CAM that, in addition to the feature map weights, feeds the class gradient to the fully connected layer to assign importance to each of the input pixels. However, [112, 90] can only be applied to limited network architectures with global average pooling. Another work by Zhang et al. [111] proposes to use a backpropagation scheme to generate an attention map by propagating the signal downward through the network hierarchy using a winner-take-it-all strategy. A few techniques examine the relationship between input and output to learn a perturbation mask by backpropagating the error signal [29].

Despite the ability of the techniques [112, 90, 29, 78] to justify the model's decision, the methods mentioned above have limitations. The methods [90] are constrained by the use of network parameters such as gradients flowing through the network and network layer weights. While techniques such as [111, 90] require a specific kind of network architecture, in some cases [109], the method requires access to intermediate layers of computation for visualizing the features at several layers. Furthermore, the techniques can explain only a particular input at a time without considering the possible variants (rotation, inversion, deformations) of the image. A work by Kim et al. [47] proposes a technique to provide

an explanation that is representative of user-defined concepts, but the manually generated concepts limit the technique. Our work is an extension of the work by Petsuik et al. [77] to localize and generalize the salient pixels of the target class using a two-step process. First, as proposed in Chapter 4, we obtain a model-agnostic saliency map (using less number of iterations ($\leq N$) than that of [77]) by empirically optimizing the pixels important for target-specific classification. As an extension to our explainability technique, in this chapter, we propose an approach to provide a global perspective on the explanation of the outcome using a reconstruction technique, which generates possible variations of the salient pixels of the input.

5.2 Terminology

This section defines the terms that we use in the following sections:

1. **Important input samples**: Given an input vector, a classification model, and an expected outcome of the model given the input vector, an important input sample is a sample from the input vector, which when replaced with zero or value outside of the input distribution will lead to a reduction in the confidence of the model in the expected outcome.

2. **Salient region**: Given an input image, a saliency map for the target class, a salient region is a region in the input vector that comprises of a set of *important input samples*, which when replaced with zero or value outside of the input distribution, will result in a drop in the confidence score for the target class. Visually, salient regions are the regions that are highlighted in red.

3. **Alternate explanation**: Alternate explanations are variations in the sample values in the *salient region* of the input such that the model's classification of the altered version of the image will also be classified to the target class.

5.3 Problem Statement

The limited application of machine learning models in safety-critical systems forms the motivation for our problem, which is stated as follows:

Given a classification model, an input image, a relevance mask and a generative model, identify the distribution of acceptable variations for the important input samples.

5.4 Contribution

Our contribution is a method for generating *alternate explanations* for the part of the input, which is salient for target-specific classification.

To generalize the explanation for the target class using the saliency map, we propose a technique to identify the variations of the pixels in the salient regions of the input for which the model prediction remains unaltered. The hypothesis for finding variations of the salient region comes from the analogy that the model is invariant to small perturbations in the input. Thereby, the approach helps identify variations (changes in color intensities, object rotation, or inversion) in the salient region of the input space for which the model classification remains unaltered [34]. To generate alternative explanations for the salient regions of the input, we use an image completion technique proposed in [108, 76] that uses the features of the neighboring pixels to reconstruct the pixels in the salient regions of the input. Using this approach, we are able to find an exhaustive and contextually similar set of transformations for the pixels in the semantic regions, which are classified to the same output class as the original input image.

5.5 Proposed Technique

We propose an approach to generate the acceptable variations for the salient region (the highlighted region in the second column of Figure 5.4) of the input image as *alternate explanations* for the model outcome.

The hypothesis for finding relevant salient region variations comes from the analogy that input encoding in the latent space (z) is invariant to small perturbations in the input space. Thereby, the approach aims to identify the variations (changes in color intensities, object rotation, or inversion) in the salient region of the input space for which the model classification remains unaltered. Input with such variations are *alternate explanations* for the model outcome. For instance, Figure 5.3 shows the variants of an instance of *cat* obtained using image translations such as shifting, rotation, inverting the image, and blurring the image.

5.5.1 Notations

In this section, we describe the notations that are used in the proposed method as follows,

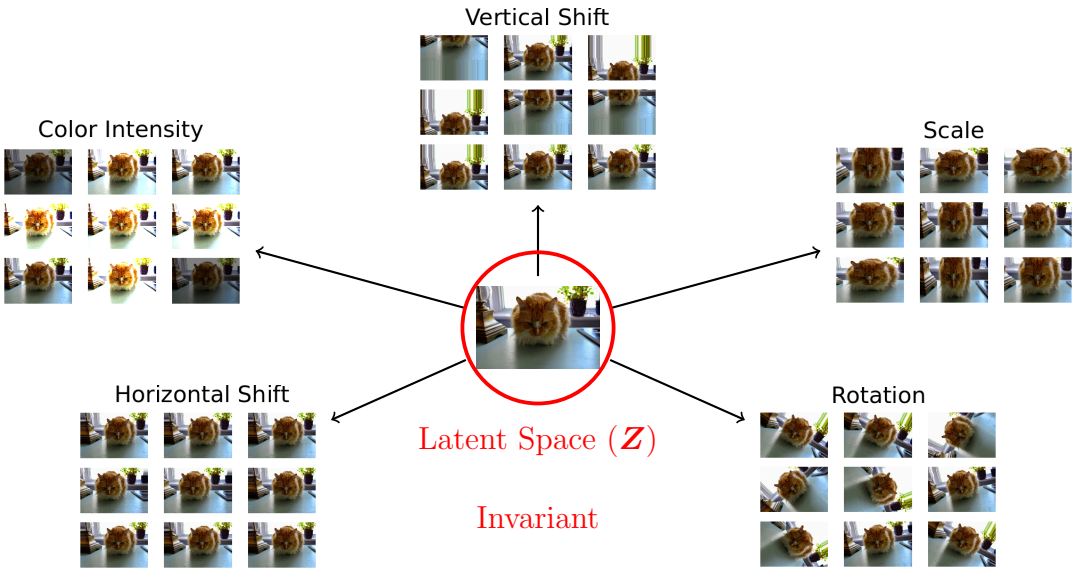


Figure 5.3: Figure showing transformations such as change in color, rotation, shift, and scale applied on the image of *cat*.

Let I be the input image.

Let M_0 be the initial random mask.

Let M is the saliency map obtained for the classification of the input image to the target class $c \in \mathcal{C}$.

Let B be a binary bounding box for the saliency mask M of the input image I, as shown in Figure 5.4 third from left, where pixels inside the box are set to 1's, and the pixels outside are set to 0's using a pre-defined threshold δ.

The reconstruction mask, R, is the refined version of the bounding box B, obtained by inverting B, followed by convolving it with a kernel of size (s, s) such that the weights assigned to pixels are inversely proportional to their distance from the bounding box. As pixels near the box are more important for the reconstruction of missing pixels from the box, the mask R is such that it assigns more importance to the pixels in the vicinity of the box than to the pixels far away.

Let G be a generator that learns an encoding d_z of the input image distribution $d_{\mathcal{I}}$ in the latent space (z).

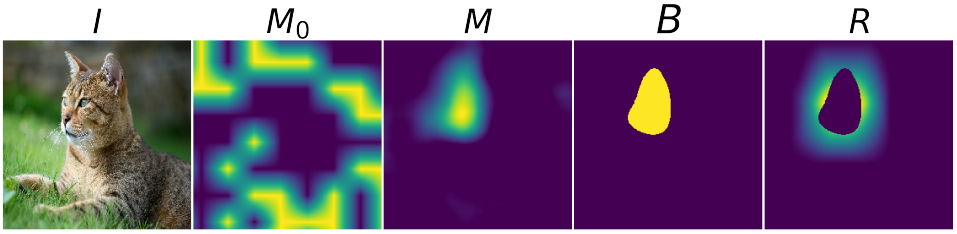


Figure 5.4: From left to right, I: input image, M: saliency map, B: bounding box, R: reconstruction mask.

5.5.2 Variations of the Salient Region of the Input

To generate variations of the salient region of the input, we use an image completion technique proposed by Pathak et al [76], which reconstructs a patch of the input by it-

eratively backpropagating the error in a generator that is trained on the input distribution.

The objective is to reconstruct the corrupt image $((\mathbf{1} - R) \odot I)$ using G. As the corrupt image is not a sample from the input distribution $d_{\mathcal{I}}$; therefore, G will be poor at recognizing the patterns of the missing part of the image. Therefore, we use the image reconstruction technique as described in [108], where the authors use the back-propagation technique with the generator to find an encoding (z') for the missing part of the image that is closest in encoding to the input I while being confined to the learned manifold (z). The objective function for learning the encoding (z') comprises context loss and discriminative loss.

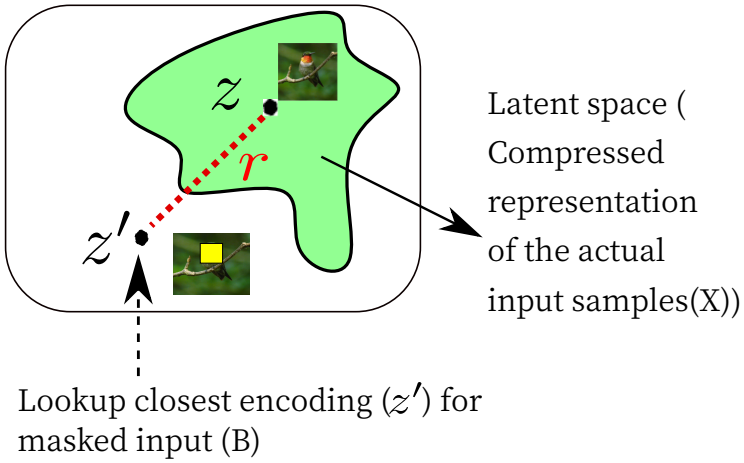


Figure 5.5: Visualizing image reconstruction approach

Context loss is used to reconstruct the missing part of the original image given the corrupt image by measuring a squared error between the corrupt image and the reconstructed image.

$$L_{cxt}(z) = (I \odot (\mathbf{1} - R)) - (G(z) \odot (\mathbf{1} - R)) \tag{5.1}$$

Discriminative loss is used for measuring the authenticity of the generated images by feeding them to the discriminator D, which returns the confidence in $G(z)$ being real.

$$L_{dis}(z) = -D(G(z)) \tag{5.2}$$

The overall loss for learning the encoding for the missing salient region of the input is as follows,

$$L(z') = \underset{z}{\operatorname{argmin}}\{L_{cxt}(\boldsymbol{z}|I, R) + L_{dis}(z)\} \tag{5.3}$$

Using the learned encoding, G(z') generates the image that is approximately close to the missing salient part of the image. Figure 5.5 shows a pictorial view of the technique for reconstructing the salient region of the input image.

5.5.3 Image Reconstruction

To reconstruct the salient region of the input image I given the target class y, we first train a generative adversarial network (GAN) using input distribution.

GAN is a generative model framework for training parametric models. GANs comprises of training two networks: a Generator (G) and a Discriminator (D). The generator aims to generate realistic-looking images by learning a true data distribution $p_{\mathcal{I}}$ from a prior distribution p_z where z is a noise variable. And the discriminator acts as an adversary whose aim is to discriminate fake images generated by the generator G from a real image from the true distribution $p_{\mathcal{I}}$. GANs are trained using a min-max game as shown in equation 5.4 where the generator aims to maximize the error made by discriminator and the discriminator learns to get better at discriminating the true from fake,

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{\mathcal{I}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log(D(G(z)))] \tag{5.4}$$

This is followed by training the generator G with the loss $L(z')$ to learn the encoding z' by back-propagation method until convergence.

The image reconstructed using the generated image is given by,

$$I_{\text{rec}} = (I \odot (\mathbf{1} - B)) + (B \odot G(z')) \tag{5.5}$$

We repeat image reconstruction for k randomly sampled noise vectors z to generate k reconstructed variants of the salient region of the input image I where z is a sample from a gaussian distribution with mean zero and variance one. The generated images will be such that their encoding will lie in the vicinity of the learned manifold in the latent space (z). However, suppose the encoding z' fails to capture the context of the salient region of the input using the evidence from the local neighborhood of pixels. In that case, some of the

reconstructed images will not be similar to the original salient region. The selected set of I_{rec} for which the model prediction remains unaltered is considered *alternative explanations* for the salient regions of the input image. Furthermore, the reconstructed images for which the target class prediction probability falls outside of the confidence interval are considered *outliers*. Such reconstructions are excluded from the explanation for the classification of I to the class y. The outliers are variants of the original image for which the model outcome changes. Such a set of perturbed images can be used to make the model robust against adversarial attacks. Figure 5.6 shows the alternate variations reconstructed for the salient region of the image of *church*.

$I \odot (\mathbf{1} - B)$ $B \odot G(z')$ I_{rec}

Figure 5.6: From left to right, the image of church with occluded relevant-mask (center grey patch B), generated occluded region ($B \odot G(z')$), reconstructed image (I_{rec})

5.6 Evaluation

This section gives an overview of the evaluation metrics, models, and datasets used to describe the results of the evaluation of the proposed approach.

5.6.1 Model and Data Description

We evaluate the efficacy of the proposed saliency map-based explainability approach using a range of publically available open image datasets such as ImageNet [86] and MS-COCO (Microsoft-Common Object in Context) [57]. ImageNet is a repository of 15 M

high-resolution images gathered from more than 20 k categories. On the other hand, MS-COCO is a significantly smaller database of images but with more instances per category. The dataset has 330 k images with more than 80 object categories. The MS-COCO dataset is used for object detection, object segmentation, and image captioning.

We use pre-trained models: VGG16 [96], Inception V3 [100], and ResNet50 [39] as base models for image classification. The pre-trained models are loaded with ImageNet weights and accept inputs of size 224×224. The models differ in their network structure and number of trainable parameters. VGG16 is a convolutional neural network trained on the ImageNet dataset achieving top-5 accuracy of 92.0% on ImageNet. ResNet50 is a 50 layer network achieving 93.29% accuracy on the ImageNet. Inception V3 is another model that achieves an accuracy of 94.4% on the ImageNet. Relying on pre-trained models [96, 100, 39] for image classification helps avoid the common training pitfalls such as model over-fitting, skewed data distribution, right model selection, and insufficient resources (such as GPUs) for training.

5.6.2 Evaluation Metrics

We evaluate the accuracy of the images reconstructed by GAN by comparing the confidence score on the reconstructed image from the confidence score of the original image. Suppose the reconstructed image contains variations close to that of the actual image. In that case, we show that the target object will be identified in the reconstructed image with a confidence score within a tolerable range from the confidence score on the actual image and vice-versa.

5.7 Evaluation of the Variations of the Salient Region

In this section, we evaluate the generalizability of our saliency map approach on an input image of a *lynx*. The image is chosen such that the presence of distractor objects makes the discrimination of the target class against other classes challenging using a saliency map. For instance, the image of *lynx* has a background whose pattern and color match with that of the *lynx*, which contributes to reducing the target accuracy to 62.15%. However, we found that the results of the evaluation followed a similar trend for other images evaluated from the test set of the ImageNet dataset.

5.7.1 Classification Accuracy of Reconstructed Images

Given a saliency map for an input image of *lynx*, we report the classification accuracy of a subset of reconstructed images of *lynx* that are correctly classified as *lynx* by ResNet50. To obtain the reconstructed images, first, we created a reconstruction mask R by convolving the bounding box for the saliency map of *lynx* with a kernel of size $(15, 15)$. We also trained a GAN [35] as the generator (G) using a set of 40 k training images from the ImageNet dataset. The input images were downsampled to a size of (64×64) to speed up the training process of G. The training is followed by feeding G with the input image, the reconstruction mask, and a batch of 64 random noise vector z to generate a batch of 64 images. The generated images are used to reconstruct a set of 64 images using Equation 5.5.

Figure 5.7 shows a subset of the reconstructed images that are correctly classified to the target class *lynx* with an average accuracy of 64%. The figures are pixelated because the reconstructed images are resized from size (64×64) to size (224×224) where (64×64) is the size of generator output and (224×224) is the size of the saliency map. From the figure, we observe that most of the reconstructed parts of the images (face including eyes, nose, mouth, left side of the face) contain a blob, which is black in the vicinity of the mouth and the lower part of the nose. The presence of a blob-like feature across the reconstructed images shows that the area around the nose and the mouth of the image is essential for classifying the image to the class of *lynx*.

Figure 5.8 shows the histograms of the reconstructed salient region of the input and the original salient region of the input. For each of the reconstructions, the figure also shows the accuracy and the t-score for the target class *lynx*. The t-score is a measure to tell apart the difference between the reconstructed and original pixels; thus, the higher the t-score value, the larger the differences. Based on the t-score, it is evident that the reconstructed pixels classified to the target class *lynx* have variations that are significantly different from the original pixels. Thus, these reconstructed pixels of the salient region of the input are the variations, which form alternate explanations for the target class.

5.7.2 Impact of Varying Sizes of Bounding Boxes

We show that the size of the bounding box enclosing the salient region of the input influences the quality of the reconstructed images and hence the alternate explanations. The idea is that as the size of the salient region to reconstruct shrinks, the evidence from the neighborhood to reconstruct missing pixels increases, thereby generating contextually similar images. We reconstructed the input image using the varying sizes of the bounding

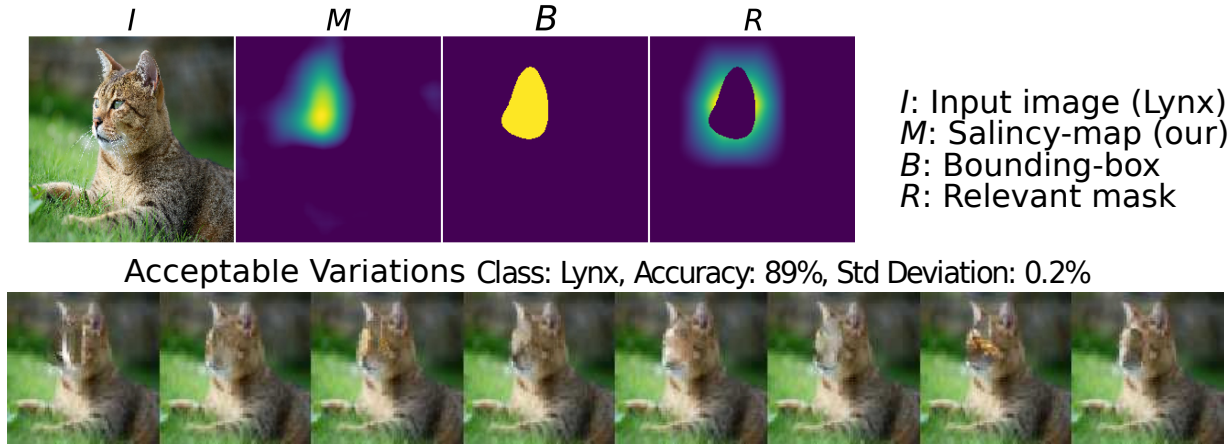


Figure 5.7: Reconstructed images for the image of a *lynx* with saliency map (M) as shown in Figure 5.4

boxes by reducing the bounding box by a factor of $\alpha = 0.1$ until half the original size. From Figure 5.9, it is evident that the number of reconstructed images correctly classified to the target class *lynx* increases as the size of the bounding box decreases. The maximum number of correct classifications is observed with the bounding box half the size of the original salient region. Equivalently, as shown in Figure 5.10, the loss incurred by the generator during the reconstruction of the images decreases as the size of the bounding box decreases. This shows that a carefully chosen value of α helps generate contextually similar images. For the image of *lynx*, a value of 8α, which retains 80% of the salient region in the bounding box, generates reconstructed images with an average classification accuracy of 63.4%.

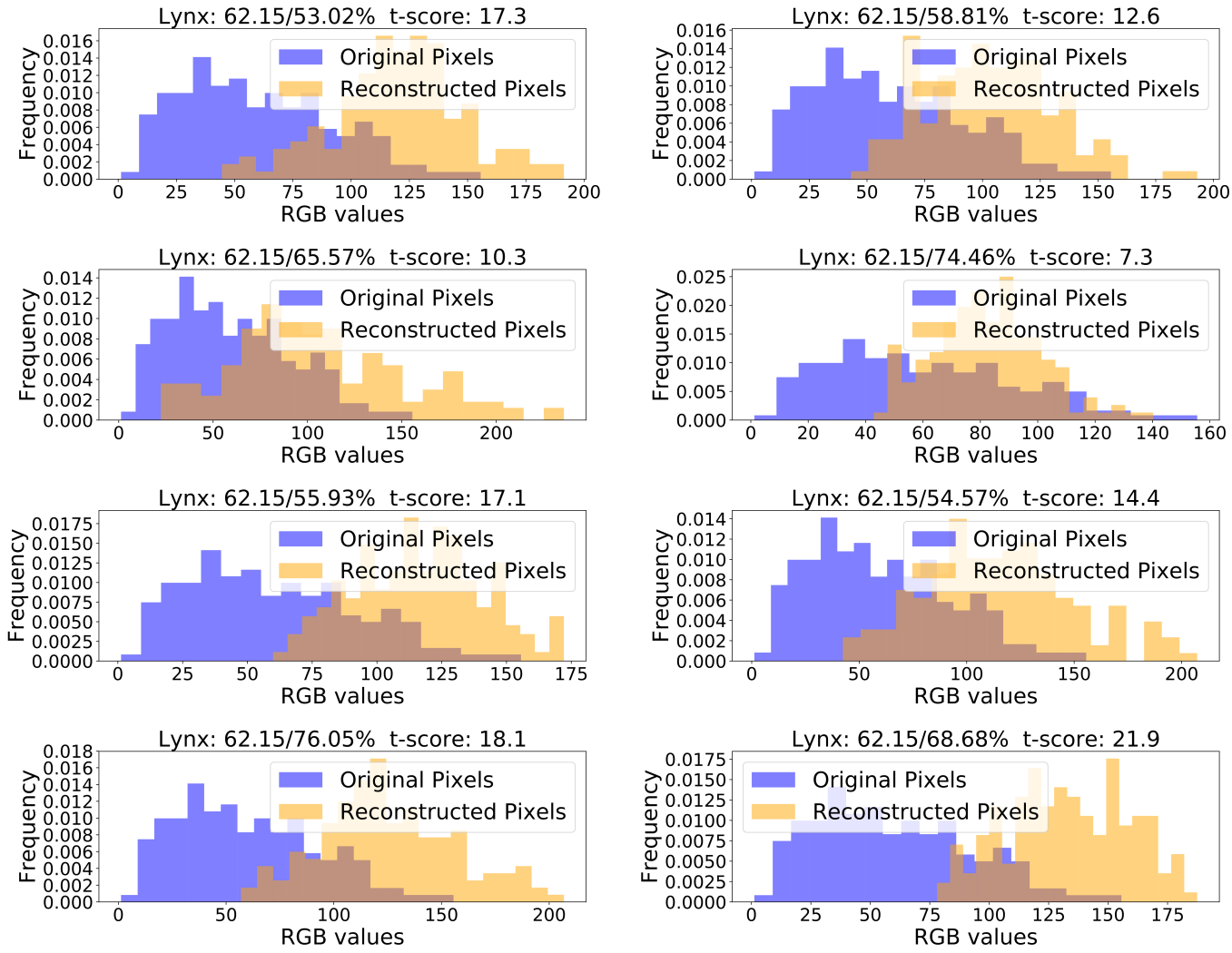


Figure 5.8: Figures showing the histogram of the reconstructed salient pixels and the original salient pixels. The title of the sub-figures show the accuracy of the class *lynx* for input image/reconstructed image.

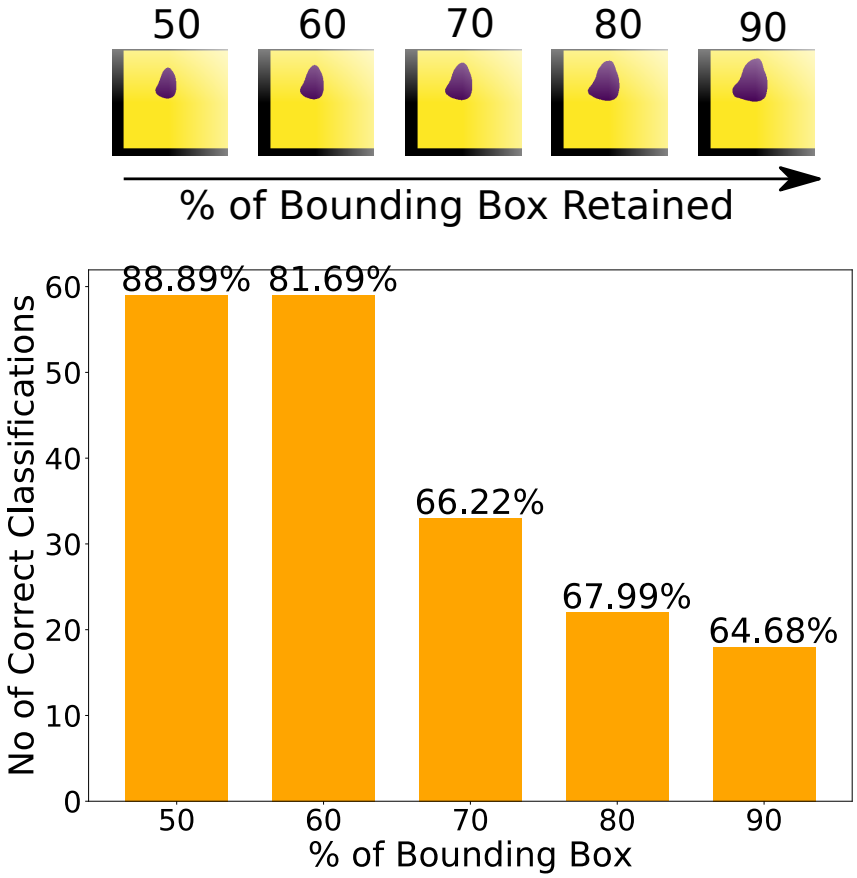


Figure 5.9: Figure shows the number and accuracy of correct classifications using the reconstructed images over different sizes of bounding boxes.

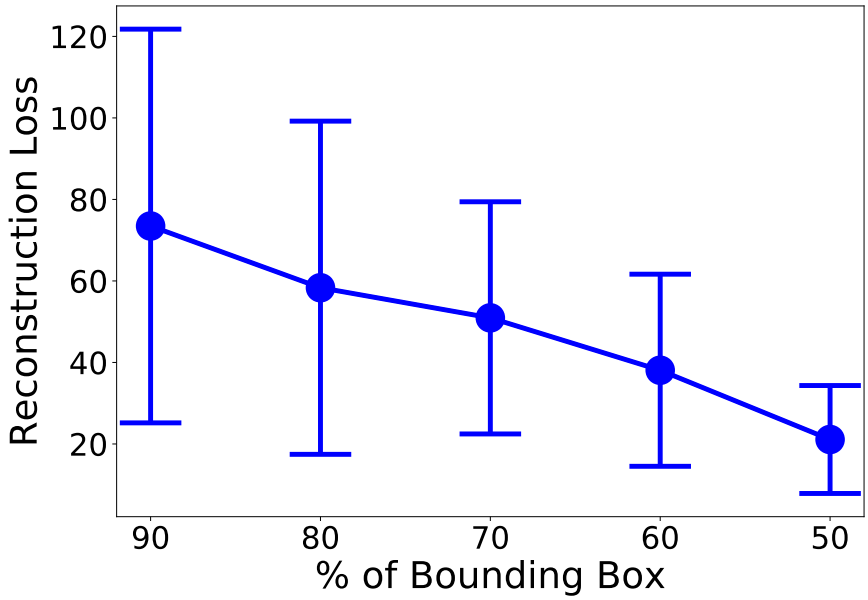


Figure 5.10: Figure shows the reconstruction loss over the different sizes of bounding boxes.

Chapter 6

TiME: Time Series-based Model outcome Explanation

With significant advances in hardware technology, sensors deployed in the real world are capable of generating a continuous stream of high-resolution data. A considerable fraction of these sensors generate data that is variable, which varies in time. In practice, these univariate sequences find space in a wide range of applications for performing tasks such as classification, forecasting, and anomaly detection. In practice, these tasks are realized using deep-learning techniques, which have achieved state-of-the-art performance with time series. For instance, ECG recordings of the patients' heartbeats are used in predictive diagnostics for identifying the presence/absence of a likely ailment and the possible diagnostics. Another example of their real-world application is in the automotive system. The univariate sequences such as GPS logs, acceleration, break events, and CAN logs are used in anomaly detection, driver behavior analytics, and cruise control systems. A vast majority of these tasks are realized using state-of-the-art deep-learning techniques. However, one of the fundamental limitations of deep-learning-based methods is their inherently non-transparent nature. And the models implemented for time series are no exception.

Within the regime of explainable AI, most of the techniques are focused on explaining image-based deep-learning architectures, which cannot be directly applied to inputs of type time series. And, any attempt at tailoring and transcribing these techniques to explain time series-based models is less appealing due to the alignment of evaluation towards qualitative, which cannot be extended to time series. However, unlike images, time series are unintuitive [94] to humans. A human intends to perceive and learn the surrounding objects by consuming the visual representations, and hence, give an image; they are able to form analogy intuitively and instinctively. Unlike images, humans are not trained to represent

sequences that are a function of time, and hence interpreting time series for discrimination can be challenging. For instance, Figure 6.1 shows ECG recordings of a subject with a normal heart beat and recordings of abnormal heart beat (irregular pattern in heart beats) obtained by adding a calibrated amount of noise to the MIT-BIH Dataset [33]. The two ECG recordings have different temporal patterns. However, the spike-like patterns in abnormal heart beat signals during time-interval 0-50, 150-200, and 200-300 are similar to normal heart beat signals in the time-interval 150-200. However, the patterns vary over a larger window. Suppose the model pays undue attention to the specific points in the time series without considering the patterns over a larger window. In that case, the model will fail to discriminate between the two classes accurately. Hence, it is important that the explanation tool not only focus on *what* input variations are important but also *where* in the time series, that is, the time interval in which their presence is essential for discrimination between the classes. This distinction will not only help discriminate between the classes but will help understand questions such as *why not the spike in other time-intervals?*, and hence will aid in answering from the perspective of a counterfactual. Therefore, we propose a perturbation-based model-agnostic technique, TiME, that generates scores for the individual input time-points by querying the model using N input sub-samples. The sub-sampling is such that the retained segments of input time series are windowed, which forces the method to learn patterns and trends over windows. Consequently, the method eliminates the introduction of "spike-like" false pieces of evidence that are not related to the features of the target class and retain patterns important during the time-interval.

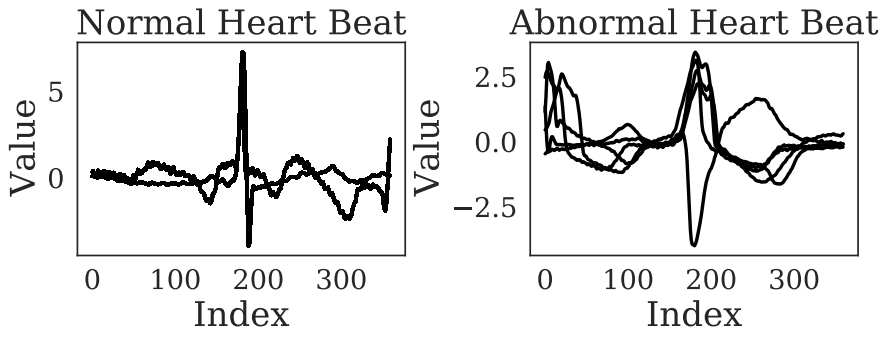


Figure 6.1: ECG time series pattern for a normal heart beat and an abnormal heart beat.

6.1 Related Work

Much before the urge for an interpretable model grew, researchers from the field of psychology, human cognition, sociology studied the importance of explanation [23]. For instance, there exists a significant body of work that attempts at using causality to predict the behavi or of a situation and policymaking under a possible set of conditions such as gender-discrimination in hiring [85], which policymakers have used to unravel the direct and indirect effects of gender on the hiring process. A bulk of this work is observed in the linear regime, which cannot be translated to a non-linear paradigm. However, with the increasing advent of deep-learning-based applications in healthcare and automotive systems, researchers in the field of AI have started exploring the methods for reasoning the model outcome for interpretability and gaining confidence in decision-making. In particular, prior work for explaining time series-based models can be broadly categorized into two depending on the scope and method of explanation as follows: (1) rule-based and data-mining (2) posthoc vs. ante-hoc, and(2) local vs. global.

Some of the approaches that exist in literature explicitly explain time series-based models, for instance, data-mining approaches. Two such data-mining approaches are symbolic aggregate approximation (SAX) [56] and fuzzy logic. SAX transforms time series into high-level abstractions using a set of strings in a two-step process. First, decomposing time series into pieces of fixed-length segments using piece-wise aggregate approximation, and second, assigning symbols to those pieces. An application of the method is observed in work by Senin and Malinchik [91] where they implement an interpretable time series classifier using high-level features for time series. Fuzzy logic is another method that explains in natural language, which is more intuitive. The capability of fuzzy logic has been harnessed by El-Sappagh [26] to develop a fuzzy logic rule-based system that harnesses the semantic interpretation capabilities to predict *diabetes* on a time series and textual features. Another extension of the method is by Wang. et. al [105] who implemented a fuzzy cognition map that relies on the interaction between components for time series forecasting. These approaches only emphasize the discrete subsets of features that are important and limited in their capability and cannot be extended to the state-of-the-art deep-learning-based models' explanation.

Another line of work to explain time series-based models use examples that are maximally representative of the explanation. A similar approach is used to find examples from the space of embedded examples learned by a k-nearest neighbor model [54]. Such examples are maximally representative of the behavior expected in a particular situation, and hence, form example explanations. A similar approach exists for finding explanations using examples for time series is shapelets [107]. They are sub-sequences that maximally repre-

sent a class. Shapelets are obtained by finding the sub-sequences and determining their distances from the candidate shapelets. However, finding shapelets on a high-dimensional time series can be computationally intensive. Several approaches optimize the candidate shapelet estimations using kernel SHAP, Deep SHAP [14], and Max SHAP [60]. All these approaches assume feature independence, which violates the temporal structure of the time series, and hence, are unreliable in pointing at the input features important within a time frame.

With a significant increase in the application of deep-learning-based methods, researchers have also focused on explaining these models using saliency map-based techniques highlighting the inputs important for a target class. Other variants of explanations use captions with keywords that point to the target class. Saliency map-based techniques are further categorized into gradient-based techniques, perturbation-based techniques, and back propagation-based techniques. Gradient (GRAD) [5, 96], Integrated Gradient [99], SmoothGrad [74] are all the variants of the approach where the gradient of the output is taken with respect to the input. DeepLift [92], DeepSHAP [14] computes attribution of the input along a path with respect to a reference point, followed by averaging the attributions per input to get the resultant attributions. CAM [112], is a back-propagation-based technique that relies on neuron activation based on the prediction and averages the channel activations from the last convolutional layer. On the other hand, Feature Occlusion [110], Feature Ablation, FeaturePermutation [93, 96] are approaches that rely on altering the input by masking, adding noise, and computing the attribution with respect to the output. ConTimeNet [46] alters sub-sequences in input time series and computes the attribution as the difference in prediction with respect to the original input. As opposed to post-hoc approaches, natural language-based models are implicitly explainable by design. Recurrent networks are explainable through their attention mechanism that assigns weights to different regions of the time series, signifying their importance towards the output. RETAIN [17] is an explainable two-level neural network that highlights significant clinical variables and important patient visits from the electronic health record dataset. Qin.et.al [81] encoder-decoder based design helps unravel time series variables that influence the model predictions. Guo. et. al[31] analyzed the LSTM for identifying the variables from hidden states that map to the output prediction. However, a limitation with attention mechanism-based approaches is that they either emphasize the discrete-time points that are important or the subsets of time series. However, for explaining time series-based models, the technique must highlight the important time points as well as the time-interval that are maximally discriminative between classes.

6.2 Contribution

Motivated by [77], we implement an approach to explain the outcome of a time series-based classifier. However, unlike [77], our technique is capable of identifying the temporal patterns containing the important features as well as the time-interval of importance for classification to the output class. The key features of our approach are, (1) non-intrusive, that is, the technique assumes the base classifier is a black-box and non of the model's parameters and weights are accessible to the outside world, (2) it is model-agnostic, that is the model relies on the input vector, the confidence score assigned to the output class, and the black-box model to assign scores to the individual points in the input time series, (3) it is class discriminative, that is, the approach assigns scores to the individual points such that the time-intervals and the features within the time-intervals that are essential for the output class will have a higher score than the other time points. The method generates a score vector as an expectation over the random input subsequences where the individual sub-samples are weighted using the model's confidence in the target class. The sub-samples are chosen such that the segments of the retained time points are contiguous and windowed segments of the inputs. This is done to learn important time-interval and to avoid introducing spurious artifacts in the sub-samples. The values within the masked parts of the time intervals are substituted with samples from a gaussian distribution with the mean and standard deviation equal to the input vector's mean and standard deviation.

The contributions of this paper are as follows:

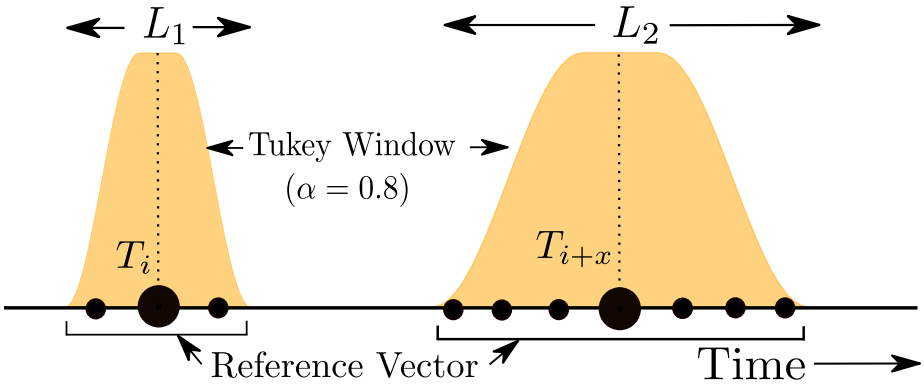


Figure 6.2: Figure showing the key aspects of the mask generations steps.

1. In this chapter, we address the limitations with existing techniques and consider identifying the subset of the features and the time-interval where the presence of the features maximizes the mapping to the output class.

2. We implement an algorithm that assigns a score to individual points in the input time series that determines the sensitivity of the point to the output class.

3. We evaluate the approach against the publicly available time series datasets and against state-of-the-art time series-based classifiers. We show that our approach outperforms the prior approaches in identifying the temporal sub-sequences to discriminate between the classes.

4. We also demonstrate a use-case study, (1) highlighting the important parts of the input power consumption measurements for sender authentication and (2) highlighting the significant portion of the input ECG recordings for categorization to normal/abnormal hear-beat class. We also show that the features and time-interval of importance in the proposed approach align with the domain expertise.

6.3 Notations

A time series $T = \{t_1, ..., t_K\}$ is an ordered set of real values of length K, sampled at regular intervals, where each $t_i \in \mathbb{R}$. A particular point in time i with a value t_i in the sequence is called as a *time point*, or a *point*.

time series based classification function: Given a time series T, a set of output classes $\mathcal{C}$, a classification function is a mapping between the input time series to a vector of logits $P^{|\mathcal{C}|}$ of length $|\mathcal{C}|$, where the values $p \in \mathbb{R}$ signifies the confidence of the classifier in the corresponding output class.

A relevance score is given by $R : \mathbb{R}^d \Rightarrow \mathbb{R}^d$ that maps the input time series to a score vector of the same size as the input.

A mask $M = \{m_1, ..., m_K\}$ is an ordered sequence of the same length as the input.

$$f(T) = \mathbb{R}^{|\mathcal{C}|} \tag{6.1}$$

Note that the predicted class is the index of the vector $\mathbb{R}^{|\mathcal{C}|}$ with the maximum value. And the time series classifier can be any classifier.

6.4 Problem Statement

Given an input time series T, a classifier f, a vector of real values of length $|\mathcal{C}|$, assign a real-valued score to each time-point where the score signifies the sensitivity of the point toward the output class.

6.5 Proposed Method

In this section, we describe the proposed method for explaining time series based model outcome.

Motivated by [77], we use feature occlusion on the input to generate a score vector for mapping the input to the output class. For an input time series T, the score vector is generated by querying the model using a set of the masked version of the input ($T \odot M$), followed by averaging the weighted masks as per the equation 6.2. The weights are the confidence scores observed upon querying the model using the corresponding masked version of the input.

$$S = \lambda \mathbb{E}[f(T \odot M) \odot M] \tag{6.2}$$

where $\lambda = 1/N$ for the N sub-samples.

Figure 6.3 provides an overview of the proposed technique. The idea is that an input vector where a random-subset of features are altered and replaced with zeros or samples from a Gaussian distribution will assign a high confidence score to the output class if the model is sensitive to the unaltered set of input features for the output class. However, unlike [77], we perturb the input such that the sensitivity towards the output is reported with respect to the important features as well as the time-interval where their presence is essential. To generate sub-samples for temporal sequence, we use a mask generation process as described below.

Mask generation: The sub-samples for querying the model is carefully synthesized such that it exposes not only the point-wise input features but also the time intervals of importance for classification to the output class. To retain the important features with respect to their time interval, it is essential that the approach captures the time features such as patterns of the input sequence that are relevant to the output classes. Therefore,

unlike the prior approach, in addition to retaining a random set of input points as on (that is, retain the actual values of these points in the mask), we also apply a windowing on those set of points. The steps toward mask generation are as follows:

- Select a random set of *points* from the initial mask sequence of length L. The number of such points is constrained in the range of $1 < c \leq L/2$. This is done to avoid exposing the model to all the points, in which case, the model's confidence score will be close to the confidence score obtained on the original input. And hence, leave scope for the algorithm to learn from the fidelity from every query.

- To enable the algorithm to focus on the trends and slopes that form a part of the input sequences and that are important for the output class, the methods must pay attention to the locality of the selected *points*. Therefore, as the next step, we assign weights to the time points in the neighborhood of the selected *points* in the order of their distance. That is, the points that are close to the selected *points* are assigned more weights than the points that are far away. This method of peaking into the sequence results in a reference vector (shown in Figure 6.2) for each selected *point*. The reference vector can be defined as the set of points within a distance of d from the selected *points*. Reference vectors corresponding to the selected *points* provides evidence in support of the *points*. The range of a reference vector is decided based on the choice of windowing and empirical study.

- We apply a Tukey window [11] on top of the reference vectors to ensure a smooth transition of the evidence from the retained window of time points. Here, the retained windows are the segments of reference vectors of the selected points, and the masked windows are shown as sub-samples of all 0's segments sandwiched in between the retained windows. A Tukey window is a window obtained by convolving a cosine lobe with a rectangular window. This window function is a preferred choice because of the gradual tapering at the edges, which is helpful in the slow elimination of the evidence at the edges. Based on empirical observations, we allow the length of the window to vary in the range $\frac{1}{10}L < w \leq \frac{1}{20}L$. This choice of window length ensures that no more than slightly greater than half of the input points are exposed using the mask. Furthermore, windows of varying lengths ensure that the method is exposed to all the possible variations in the neighbourhoods, which ensures that the method learns robust patterns that are important in discriminating between classes.

- There can arise a scenario that results in spurious sharp edges in the windowed reference vectors. For instance, a fraction of one window of reference vector overlaps with the other, resulting in a sharp intersection. Another scenario is when the Tukey

window function takes a value of $\alpha < 0.5$. In that case, the retained reference vectors may still introduce spurious evidence due to a sharp drop or increase in evidence at the edges resulting in a spike-like pattern in $(T \odot M)$. And suppose the original input also has a similar spike during the same time-frame important towards an output class. In that case, the model will mislead the presence of this spurious evidence as evidence of interest for that class. It will result in ambiguity and fluctuations in the convergence process. Therefore, in addition to windowing, smoothing aids in diminishing such evidence, which in turn aid in accelerating the convergence process.

Mask substitutes

In this context, a segment of the input sequence is masked if the actual values of the time-points in the segment are replaced with values from a distribution other than the input distribution. Based on observation, we found that replacing the actual values of the selected set of time points with samples from a Gaussian white noise worked across the different datasets and across different doamin.

Using the mask generated using our mask generation approach and for the input time series, we generate relevant scores for the classification of input to output class using the equation 6.2.

Algorithm 4 shows the procedure for calculating relevance score using sub-samples of the input time series,

Also, please note that the method is evaluated against univariate time-sequence unless explicitly mentioned. However, the technique can be extended to work with multi-variate time series as well.

6.6 Evaluation

This section gives an overview of the datasets and the models used to evaluate the approach. This is followed by qualitative and quantitative evaluation. Finally, we discuss the performance analysis of the technique.

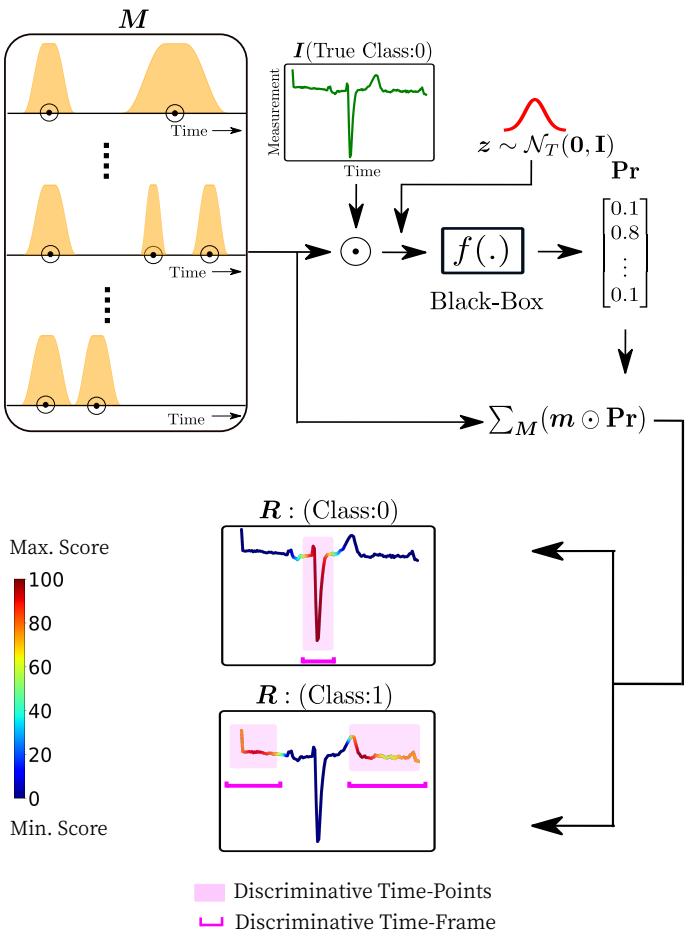


Figure 6.3: TiME overview

Algorithm 4 TiME algorithm

Input: T, f, λ, $\mathcal{M}^{N \times K}$

$T = (T - \mu(T))/\sigma(T)$
$K = |T|$ ▷ Length of input time series T
$\mathcal{S} = \mathbf{0}_{N \times K}$ ▷ Initialize N zero score vectors of length K
for $i = 1$ **to** $N - 1$ **do**
    $\mathcal{Z} = \mathcal{N}(\mathbf{0}, \mathbf{1})$ ▷ Sample N vectors of Gaussian white noise of length K
    $\mathcal{M} = (\mathcal{M} - \mu(\mathcal{M}))/\sigma(\mathcal{M})$ ▷ Scale the masks
    $\mathcal{M} = \mathcal{M} * \mathcal{Z}$ ▷ Add Gaussian noise to the sub-samples
    $\mathcal{P}^{|\mathcal{C}|} = f(T * M)$ ▷ N Logit vectors of length $|\mathcal{C}|$, output classes
    $\mathcal{S}_i = (\mathcal{M} \odot \mathcal{P}^{\mathcal{C}}) * \lambda$
    $\mathcal{S}_i = \mathcal{S}_i / N$
end for
$R = \sum_N \mathcal{S}_{N \times K} / |N|$
$R = (R - \mu(R))/\sigma(R)$
Return R

6.6.1 Data and Model Description

To evaluate the efficacy of the proposed method, we use a range of univariate time series datasets from the UCI repository[24] and PhysioNet repository [33]. UCI repository is a publically available repository of time series and image datasets across a wide range of domains. We use a subset of ten univariate time series datasets, namely PhalangesOutlinesCorrect, CinCECGTorso, ItalyPowerDemand, Trace, GunPoint, GunPointAgeSpan, Strawberry, ECGFiveDays, TwoLeadECG, Chinatown, DistalPhalanxOutlineCorrect. These datasets are used for anomaly detection, classification, and forecasting in real-world settings. For instance, TwoLeadECG andECGFiveDays are used for detecting the different categories of ECG traces. GunPoint and GunPointAgeSpan are used for gesture recognition. We also evaluate TiME against the MIT-BIH ECG dataset from the PhysioNet repository. The dataset comprises of ECG recording of normal and abnormal heart heat patterns. The abnormality is introduced in the ECG recording using a calibrated amount of noise in the MIT-BIH dataset. Manual annotations of the QRS patterns [33] from the ECG recordings are also available from the cardiologists. We aim to leverage this knowledge to determine the alignment of the attributions reported by our proposed method against the domain expertise and validate the proposed method. This form of evaluation also helps build the users' confidence who wish to apply the technique in a safety-critical system for decision making.

We evaluate our method against the state-of-the-art models ResNet1D, Inception1D, CNN, LSTM [53]. Based on a comparative analysis of the time series-based classification algorithm on UCR time series datasets by [24], ResNet1D outperforms all the other techniques and is only second to the FCN network. Another study by the same author shows that InceptionTime that uses AlexNet architecture for time series classification achieves comparable accuracy on UCR time series datasets. Evaluating the technique against all the high-performance models helps avoid the common pitfalls that arise due to poor model selection, such as model complexity, model generalizability, and model bias.

6.6.2 Evaluation Metrics

We evaluate our approach against a set of evaluation metrics, where different metrics evaluate different aspects of the approach. For instance, insertion/deletion metrics evaluate the effectiveness of the individual time points. And swap relevant time points, and mean time points determine the quality of the highlighted patterns.

Insertion (Ins) and deletion Metrics (Del)

Motivated by [77], we use the metrics of insertion and deletion to evaluate our relevant mask approach. In the deletion metric, the deletion of salient time points from the input causes the model to drop the probability of the target class. And in the insertion metric, the insertion of time points from the relevant region of the input causes the model to increase the probability of the target class. We capture the sensitivity of the model to the removal and insertion of pixels from the relevant region of the input using an average AUC (Area Under the Curve) score. Thus, as the relevant input pixels are deleted from the masked input, the AUC curve for the model will shrink to a thin area, thus dropping the average AUC score, indicating the right explanation for the model decision. Similarly, during the insertion, as the pixels from the relevant region of the input are added to the masked input, the AUC curve expands to cover the large area under the probability curve, thus increasing the average AUC score.

Complexity (Cx)

Another criterion for measuring the goodness of an attribution method is its complexity. The technique should report the minimum number of relevant time points sufficient to

achieve an output confidence score within a tolerable range from the actual input's confidence score. We use the approach discussed in [79] to measure the complexity of the method. We report the L1 norm of the relevant time-points located obtained from the relevance score using a threshold e. That is, every time-point whose relevance score is greater than e is considered a relevant time-point and added to the set of relevant time-points, rel $\Rightarrow (R(I_{t_i}) > e)$, followed by obtaining the L1 norm of the set. Qualitatively, we can measure the complexity as the highlighted (red time-points)—the small the number of relevant time-points, the lower the complexity.

Swap relevant time-points (STP)

Deletion and insertion metrics rely on independent queries to the model to measure the quality of relevance. Consequently, they fail to capture the interdependence of the relevant time points. They also fail to capture the impact of the fidelity in the relevant trends and patterns on the confidence score of the target class. To capture the behavior of the attribution method on the relevant time frame, we apply the metrics approach from [89]. The idea is that any perturbation within the sequence of relevant time points will result in a sharp drop in the target class confidence score, indicating the right time frame of relevance. That is, if for an input time series $I = \{t_1, t_2, \cdots, t_L)\}$ the relevance score is given by $R = \{r_1, r_2, \cdots, r_L\}$. The sub-sequence of the relevant time-points, that is, the time-points in the input whose relevance score is greater than a threshold e, can be obtained in the order of their occurrence in the sequence as $t_{sub} = \{t_i, t_{i+1}, \cdots, t_{i+l_s}\}$ with length l_s. The sub-sequence gets flipped to $t_{sub} = \{t_{i+l_s}, \cdots, t_{i+1}, t_i\}$, and inserted back into the original input sequence. This is followed by querying the model with the modified input time series to measure the change in the confidence score for the target class—the lower the measured confidence score, the accurate the relevant time-points.

Mean relevant time-points (MTP)

Similar to STP, another way of determining whether the attribution captures the relevant time series features is by replacing the whole sequence of relevant time-points with the mean of the relevant time-points [89]. That is, if the input time series is $I = \{t_1, t_2, \cdots, t_L\}$ and the relevance score is $R = \{r_1, r_2, \cdots, r_L\}$. Find the sub-sequence of relevant time-points using a threshold e and identify all the time-points in the input $r_i > e$. This is followed by replacing the values in the sub-sequence with the mean of the sub-sequence, $t_{sub} = \{\mu_{t_{sub}}, \mu_{t_{sub}}, \cdots, \mu_{t_{sub}}\}$, and inserting the subsequence back into the original sequence. Use the modified input sequence to measure the change in the confidence score for the target

class. A good measure of relevant time-frame is indicated by a lower confidence score on the modified time series.

Combined score

In addition to evaluating the individual metrics, we also evaluate the approach using an ensemble of metrices [79], that considers evaluating the method by addressing the desirable properties in explanation such as the coherency, complexity, and average drop in the output confidence score in the presence of perturbations and additional noise. The mathematical expression of the combined metric of evaluation is given by:

$$C = 5(Ins + Del + STP + MTP + Cx)^{-1}$$

6.6.3 Qualitative Evaluation

Like the saliency map for images, the saliency map for time series univariate sequences can also be created using the generated relevance score. We can achieve this by creating a heat-map of the relevance score overlapping the input time series. This form of visualizing the relevance score enriches the line plot of the input by assigning more weight to the time-point corresponding to a higher relevance score and vice versa. We can use this plot in addition to domain knowledge from experts to uncover useful insights such as the significance of the highlighted interval and pattern with respect to the output category.

Figure 6.4 shows the heat-map of the relevance score for the two classes of the Strawberry dataset from the UCI repository. The features and the time intervals of relevance for the two classes are shown as red regions of the contiguous segment of time points. For class 0, one of the two segments, from time-interval 190 to 235, is assigned a relevance score in the range lower than the first segment and has fluctuations across the segment. This is also reflected from a high SSE score of variance of the regression line fitted to the window of relevance scores and hence, is likely a less relevant interval. We confirmed this analogy from an experiment where we observed an insignificant drop in the confidence score of the model, which was queried using the same set of examples from class 0 but with the original values of time-points in the second most relevant window replaced with samples from a Gaussian distribution. Therefore, this region with a low relevance score can be safely ignored from the set of relevant segments for class 0. Furthermore, we also visualize a linear regression fitted to the relevant segments of the input time series alongside the set of examples using the learned coefficients. This is done to determine a template of relevant patterns, which can generalize across the class of time series. From the visualization, we

observe that the fitted coefficients of regression achieve a low SSE score with minimum variance across the set of examples from the class. This set of templates forms the relevant patterns for the class of time series. However, in the lack of domain knowledge, the identified patterns of relevance may be insignificant, making this evaluation method less meaningful to an end-user. Therefore, we also evaluate the attributions using quantitative metrics.

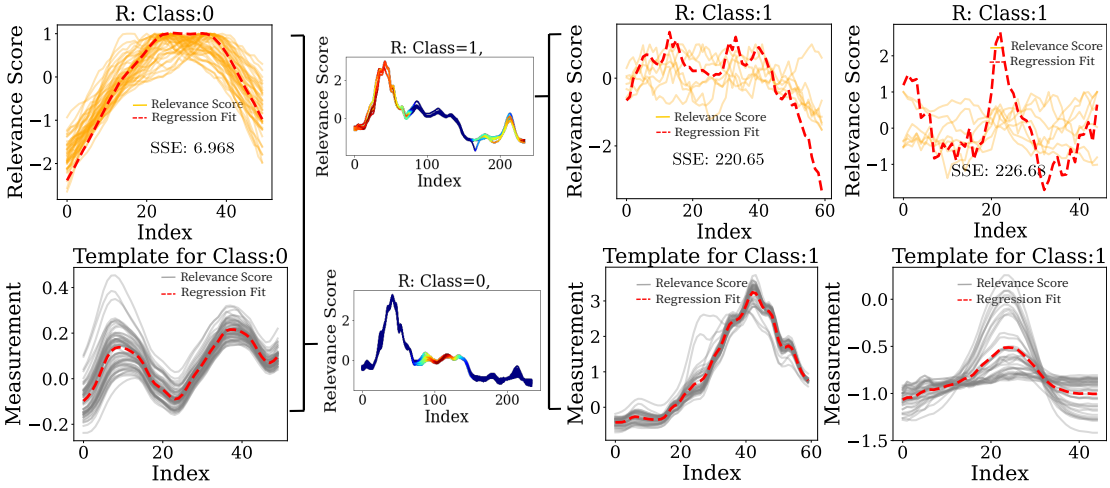


Figure 6.4: Middle, relevance map for an instance of class 0 and class 1 from Strawberry dataset. Left and Right, a global perspective of the reported relevance map of the input examples.

6.6.4 Quantitative Evaluation

We also evaluate the attributions of the time series-based explainability techniques [90, 83, 10, 14] using a range of quantitative metrics, each capturing an aspect essential for evaluating the attribution and a combined metric (C). The metrics of insertion score and deletion score determine the correctness of the attribution with respect to the learned hypothesis. Complexity score determines the number of time points relevant for the class of interest. Swap, mean, and inverse time-points metrics are related to assessing the quality of the pattern in the captured time-frames towards the output class. And combined (Cs) score determines the overall significance of the attribution by combining the separate evaluation metrics into one weighted score. We report the metric scores across all the techniques for the base models: ResNet1D, InceptionTime, CNN1D, and LSTM, and datasets from the

UCI repository and MIT-BIH. We report the metric score for each technique and model by averaging over a randomly selected set of ten time series datasets from the UCI repository and the ECG binary classification dataset from the MIT-BIH dataset. Table 6.3 shows that our technique achieves the highest combined (Cs) score for the base models ResNet1D and LSTM against all the other approaches. Perhaps, the inability of the related techniques to capture the significance of attribution with respect to both the time-points of relevance and the time-frame of relevance is the cause for the low score of attributions.

Furthermore, to evaluate the effectiveness of our approach against the individual metrics, we report the insertion score, deletion score, complexity, inverse time-points, mean time-points, and swap time-points scores for a randomly selected set of ten time series UCI repository and MIT-BIH ECG dataset. Table 6.4 shows the mean and standard deviation of the metric scores averaged over the test set of the datasets. We observe that as desired, the attribution for Strawberry and ItalyPowerDemand datasets achieves a high insertion and deletion score, indicating correct time points of relevance. Furthermore, the strawberry dataset also achieves a near-zero complexity. We can verify this result from the heat-map 6.4, which shows a tiny time window of the input time series highlighted as relevant for the output classes with a clean separation from the non-relevant time-frame. Furthermore, a value of zero for the MPT, STP, and ITP measures of evaluation supports our claim that the reported time-frame of relevance is also essential for the time-points to classify them together as the class of interest (0 or 1).

Table 6.1: The table contains the baseline accuracy averaged over 10 runs of each implemented model on the UCR/UEA archive, with the standard deviation.

Dataset Name	**Repository**	**Accuracy** [27]	**Model**
CinCECGTorso	UCI [24]	82.6 ± 2.4	ResNet1D [27]
DistalPhalanxOutlineCorrect	UCI [24]	77.1 ± 1.0	ResNet1D [27]
ECGFiveDays	UCI [24]	97.5 ± 1.9	CNN [27]
GunPoint	UCI [24]	99.1 ± 0.7	ResNet1D [27]
GunPointAgeSpan	UCI [24]	98.0 ± 1.0	ResNet1D [27]
ItalyPowerDemand	UCI [24]	96.3 ± 0.4	ResNet1D [27]
PhalangesOutlinesCorrect	UCI [24]	83.9 ± 1.2	ResNet1D [27]
Strawberry	UCI [24]	98.1 ± 0.4	ResNet1D [27]
Trace	UCI [24]	100 ± 0.0	CNN [27]
TwoLeadECG	UCI [24]	100 ± 0.0	CNN [27]
MIT-BIH	PhysioNet [33]	94.5 ± 2.3	ResNet1D [27]

Table 6.2: TiME evaluation with base model ResNet and UCI repository time series datasets

Ins score	Del Score	Cx	MTP	STP	ITP	C
0.73/0.23	0.73/0.23	0.027/0.01	0.46/0.43	0.05/0.43	0.27/0.40	0.09/0.07
0.72/0.37	0.72/0.37	0.17/0.01	0.34/0.45	0.03/0.45	0.24/0.41	0.11/0.21
0.81/0.17	0.81/0.16	0.13/0.09	0.37/0.41	0.04/0.41	0.18/0.34	0.32/0.26
0.71/0.29	0.70/0.29	0.11/0.02	0.86/0.22	0.07/0.27	0.50/0.5	0.36/0.09
0.90/0.06	0.90/0.06	0.32/0.11	0.76/0.26	0.07/0.25	0.27/0.31	0.71/0.15
0.27/0.49	0.27/0.35	0.15/0.35	0.23/0.03	0.02/0.41	0.20/0.40	0.12/0.20
0.91/0.04	0.91/0.04	0.02/0.02	0.00/0.00	0.08/0.00	0.00/0.00	0.00/0.00
0.81/0.23	0.81/0.23	0.09/0.01	0.68/0.45	0.06/0.44	0.02/0.14	0.22/0.15
0.61/0.35	0.61/0.35	0.21/0.14	0.17/0.23	0.04/0.43	0.48/0.50	0.42/0.29
0.61/0.35	0.61/0.35	0.21/0.14	0.17/0.23	0.04/0.43	0.48/0.50	0.42/0.29

Table 6.3: TiME evaluation using combined (C) metric score and a quantitative comparison against related approaches.

Dataset	Ours	DeepSHAP [14]	Grad CAM [90]	LRP [10]	LIME [83]
CNN [53]	**0.73**/0.23	0.73/0.23	0.76/0.26	0.46/0.43	0.27/0.40
Inception [100]	**0.81**/0.17	0.72/0.37	0.17/**0.01**	0.34/0.45	0.24/0.41
ResNet [39]	**0.90/0.06**	0.81/0.16	0.68/0.45	0.37/0.41	0.18/0.34
LSTM [53]	**0.91/0.04**	0.70/0.29	0.61/0.32	0.86/0.22	0.50/0.5

6.6.5 Class Discrimination Evaluation

We evaluate the class discrimination ability of our approach qualitatively and using quantitative metrics. Visually, a key to reliable discrimination is highlighting the minimum number of input features within time-interval maximally representative of the output classes. Furthermore, we described in the introduction that due to the high signal-to-noise ratio in the real-world time series datasets, the features of different classes could all be observed in the input time series. To show the discriminating nature of TiME generated relevant scores, we experiment and elaborate upon the interpretations for the datasets from UCI and the ECG dataset from PhysioNet.

Trace dataset

The trace dataset is a subset of the dataset from the Transient Classification Benchmark. It is an artificial dataset designed to capture four classes of instrumentation failures in nuclear power plants. Primarily, it is used in [82] for classification tasks. For each input instance, regardless of their true class, we ask TiME to generate a relevance score for all the output classes $\{c_1, c_2, c_3, c_4\}$. This is repeated for a set of the first 50 instances from each class of the test set. Finally, we visualize the heat-map of the relevance score generated against all the output classes for each input instance. Depending on the presence of class-specific evidence, some of the time intervals in the input time series, whose relevant patterns align with those for that class, are assigned a higher score than others. This distinction is observed in the heat-map, where time-points are assigned colours in the order of decreasing scores. Therefore, certain segments of the line plot of the input time series are highlighted in red, signifying their importance for the output class, and the remaining segments are suppressed (time-points in the shades of blue), denoting non-essential intervals for the output class.

Figure 6.5 shows a matrix of heat-map visualization of the relevance scores generated across all the inputs and all the classes. That is, across a row, the output classes vary, and the inputs remain the same, whereas across a column, the inputs vary, and the output class remains the same. In particular, a row highlights the different segments of the input time intervals that are important for the different output classes. For instance, for a set of input time series $\{I_1, I_2, \cdots, I_{10}\}$ labelled class 0, the figures in the first row shows the relevant time-intervals for the output classes $\{c_0, c_1, c_2, c_3\}$. The pattern in the time interval (70,90) is highlighted as relevant for class 0 and is distinct from the relevant pattern for class 1, which lies in the time interval (100,130), and so forth. As class 0 is also the true class of $\{I_1, I_2, ..., I_{10}\}$; therefore, the features highlighted in time-interval (70,90) forms the template for class 0.

On the other hand, a column highlights the relevant pattern for the particular class of interest across a set of input time series. The time interval contains patterns that closely resemble the features of the template for the class. For instance, the first column highlights time-interval containing patterns that match the pattern of class 0. This is also evident from the similarity in the patterns of the highlighted time intervals across the inputs, which is observed to have a steep drop in magnitude. Similarly, the relevant time-interval for class 1 highlighted shows a consistency in the pattern across the inputs where the feature magnitude is in the range (0,-2) and is observed to have a logarithmic growth.

This observation aligns with a robust models objective, which aims to maximize the distance between the features of different classes and minimize the distance between the

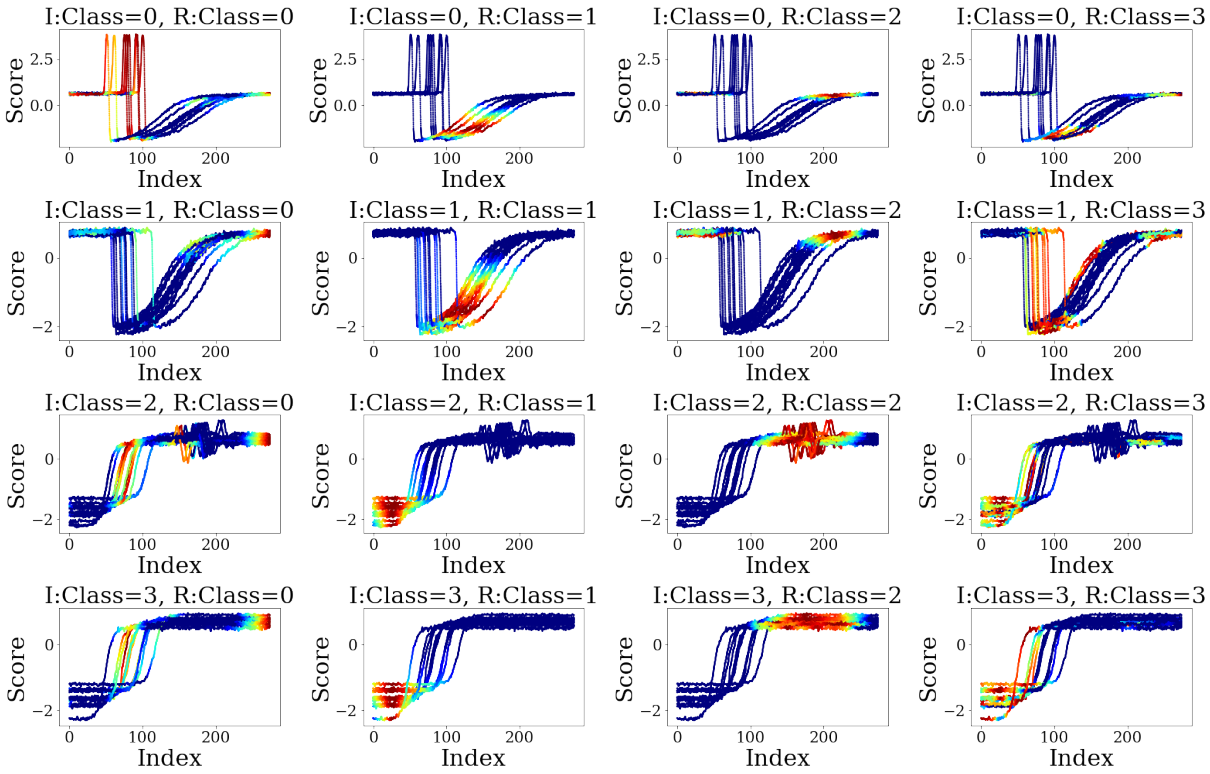


Figure 6.5: Relevance map for instances of Trace dataset across all the output classes.

Table 6.4: Trace UCI evaluation

$I_{c=0}, R_{c=0}$	$I_{c=0}, R_{c=1}$	$I_{c=0}, R_{c=2}$	$I_{c=0}, R_{c=3}$
Insertion: 0.99	**Insertion**: 0.99	**Insertion**: 0.99	**Insertion**: 0.99
Deletion: 0.99	**Insertion**: 0.99	**Insertion**: 0.99	**Insertion**: 0.99
Complexity: 0.10	**Complexity**: 0.10	**Complexity**: 0.10	**Complexity**: 0.10
MTP: 4.8e-06	**MTP**: 4.8e-06	**MTP**: 4.8e-06	**MTP**: 4.8e-06
STP: 0.99	**STP**: 0.99	**STP**: 0.99	**STP**: 0.99
ITP: 0.94	**ITP**: 0.94	**ITP**: 0.94	**ITP**: 0.94
C Score:0.53	**C Score**:0.53	**C Score**:0.53	**C Score**:0.53

same class features. The discriminatory behavior of TiME also helps interpret the cause of ambiguity in the model's output. For instance, a majority of the time series inputs from class 3 are miss-classified as class 2. This is the case because, as shown in cell 32 (third row and second column), the highlighted relevant time-interval (110, 210) for class 2 is assigned a higher score as compared to the relevant time-interval (0,90) for class 3 (shown in cell 33), which is the true class of the input. Consequently, the instances of class 3 are classified as class 2. Such ambiguity in the output can be explained using TiME, which can discriminate between the relevant time intervals for different classes.

6.6.6 Use-case: ECG Recording Classification using MIT-BIH ECG Dataset

MIT-BIH dataset is a dataset from the PhysioNet repository for arrhythmia detection. The dataset comprises ECG recordings of 47 subjects. Each of the recordings is categorized into normal and abnormal heart-beat. The ECG recordings for the class of abnormal heart-beat is obtained by adding a calibrated amount of noise to a subset of ECG recordings from subjects with normal heat. The annotations for the dataset are obtained by marking the QRS region of the recording (shown in the highlighted red region of cell 00). We leverage the annotations to evaluate TiME generated importance map. Figure 6.6 shows a matrix of the heat-maps of relevance score obtained on the input ECG recordings, and the ground-truth annotations for the recordings belonging to the two classes are shown in the boxed regions. It is observed that the relevant samples identified by TiME align with the expert ground-truth annotations. The highlighted time-interval around the QRS complex region overlaps with the boxed annotations for the class of normal heart-beat, and the highlighted time-interval for the class of abnormal heart-beat overlaps with the boxed annotation for the class of abnormal heart. Furthermore, the relevant features for the two classes are distinct. This distinction is essential for discrimination between the classes. For instance, the relevant features of recording belonging to the normal heartbeat class are concentrated around the QRS complex. In contrast, the relevant parts for abnormal heart-beat comprise the noisy features in the QRS region and the spikes at the beginning of the time series. Hence, these regions form the template for distinguishing the recordings belonging to the two classes and comparing the model's output in noisy and ambiguous situations.

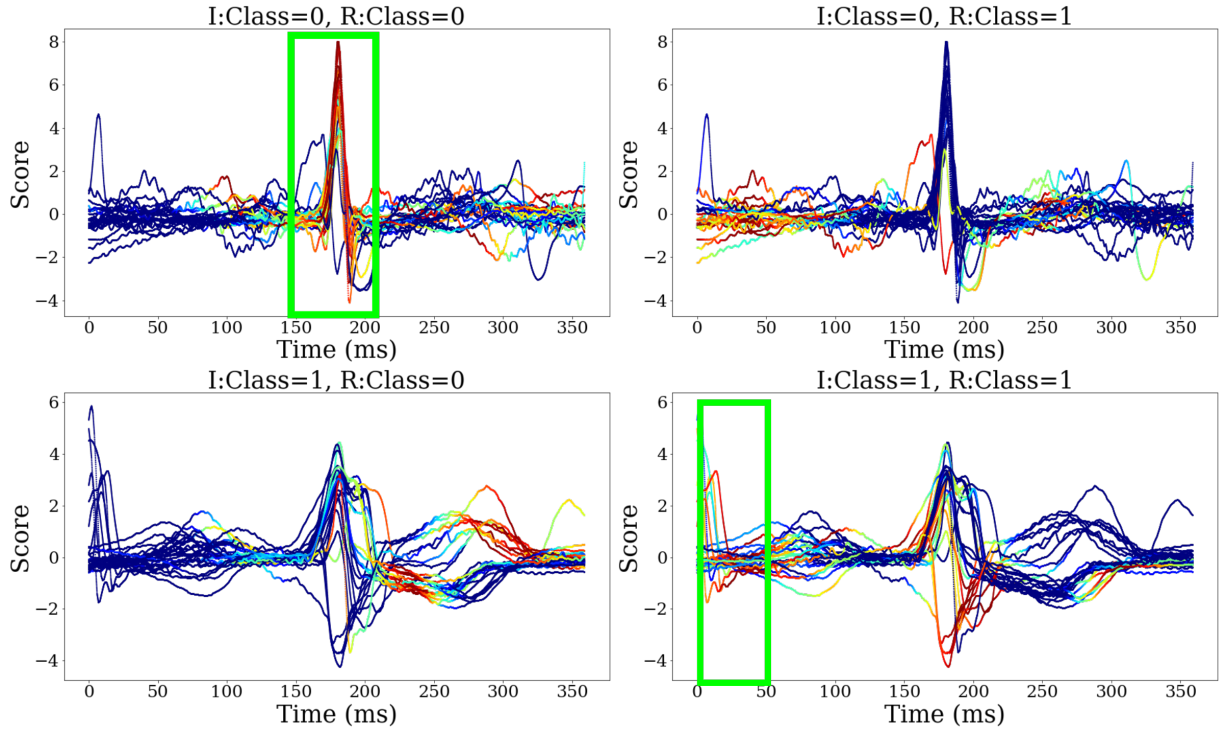


Figure 6.6: Relevance map for instances of MIT-BIH ECG dataset across all the output classes.

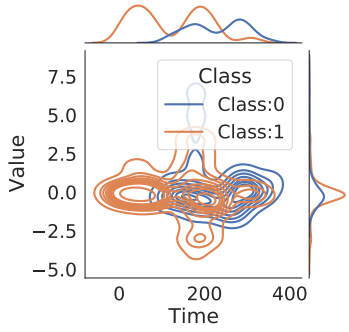


Figure 6.7: Feature of Trace dataset across all the output classes.

6.6.7 Use-case: Sender Authentication in CAN protocol

We used side-channel information – power consumption measurements of the ECUs to classify the transmissions observed on the CAN bus to the state of transmission and non-transmission. We also evaluate the outcome of the classifier using TiME to determine what part of the input power consumption measurements are essential for the classification of the input to the class of transmission and non-transmission. This information can be used when misclassification is observed to compare the deviations from baseline (correct classification) for further investigation.

Correct sender authentication Figure 6.8 and 6.9 show the heat-map of the relevance scores obtained for the input belonging to the class of transmission and non-transmission. The figure also shows the ground truth as the boxed regions of the inputs obtained from the judgment of human experts. We can observe that the power consumption measurement pattern for the two classes strongly resembles each other. However, the key features of distinction for the two classes have a slight nudge in the intervals that is not present in the same interval for either of the classes. These features are important for distinguishing between the two states, hence forming the feature template for the two classes.

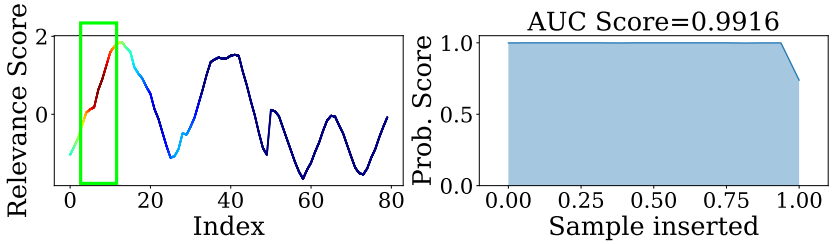


Figure 6.8: Correct classification of the input power consumption measurement to the state of transmission. Green box shows the input region with features for the state of transmission.

Incorrect sender authentication

Figure 6.10 shows the relevant region for an input belonging to the class of transmission and misclassified to the class of non-transmission. The reported region of relevance is not aligned with the template for the correct state. The relevant region shown in the red box partially overlaps with the template for its correct state. This is the case because the key feature of distinction, the nudge-like feature, is missing from the input, causing the model to behave randomly. This means that the model will fail to correctly determine the state

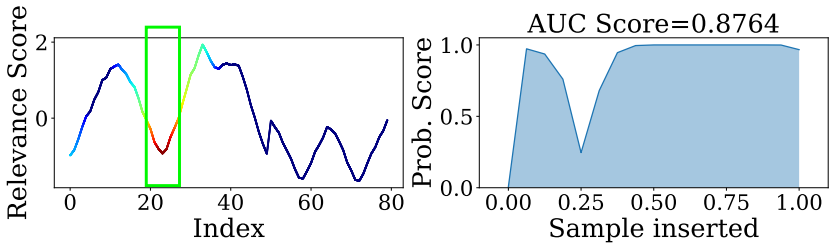


Figure 6.9: Correct classification of the input power consumption measurement to the state of non-transmission. Green box shows the input region with features for the state of non-transmission.

of the input transmission when the input pattern has any discrepancies from the training set. As a precautionary measure, such input instances should be filtered out for deeper evaluation of the system function and correction steps.

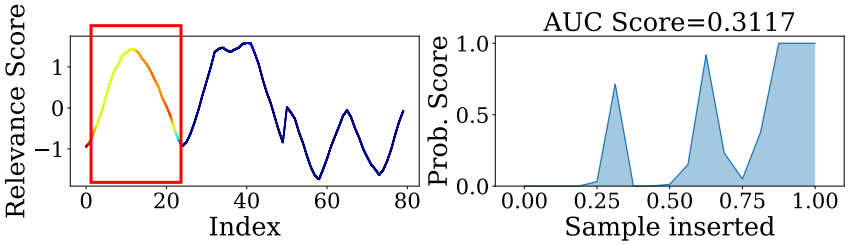


Figure 6.10: Incorrect classification of the input power consumption measurement to the state of transmission. Red box shows the input region with features for the state of transmission and non-transmission, and the source of confusion for the misclassification.

6.6.8 Performance

We measure the performance of TiME by reporting the latency across a set of pre-defined input feature-length and a set of different numbers of queries for generating relevant scores. The two factors that can induce delay are the number of iterations for which the model is queried to generate a relevance score and the length of the input time series. Figure 6.11 shows the latency of the model. The figure on the left shows the latency, that is, the time taken to generate a relevance score when the number of queries to the model is varied from 10 to 5000. With fewer queries to the model, the delay at the output is insignificant, but the relevance score is inaccurate. On the other hand, with 5000 queries to the model, a

large delay is observed at generating the relevance score; however, the estimate of relevance score is reliable with high insertion, deletion, and ITP scores. Therefore, we chose to query the model with 2500 iterations, keeping the delay short while achieving a good relevance score. Similarly, the plot of the latency of the technique against input length shows that the delay in generating relevance score increases as the length of the input time series increase.

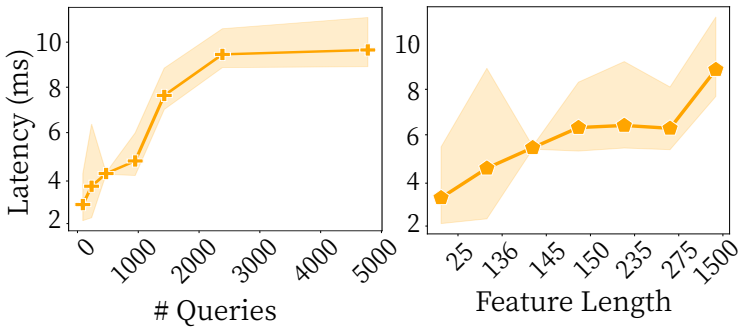


Figure 6.11: The figure shows the latency of the technique against the number of queries to the black-box model, and against the length of input time series.

6.6.9 Relevance Score on UCI Dataset

Similar to the Trace dataset and MIT-BIH ECG dataset, we also visualize the heat-map of the relevance score for the test set of CinCECGTorso, DistalPhalanxOutlineCorrect, ECGFiveDays, GunPOint, ItalyPowerDemand, PhalangesOutlinesCorrect, Strawberry, TwoLeadECG datasets from the UCI repository. Figure 6.12 shows a matrix of four columns and four rows for the four classes. Each column highlights the time-interval-specific features significant for the particular class. For instance, the third column shows the time-interval of interest for class 2 across the inputs, which has a common bump in the second half. However, an overlap is observed in the time interval of interest for the different classes on the same input. That is, the first row shows that the features in time-interval [600-700] is important across all the classes with a variation such that the strength of the features captured in this interval across the classes are different. Similarly, Figure 6.13, 6.14, 6.15, 6.16, 6.16, 6.17, 6.18, 6.19 shows the heat-map of the relevance score generated for the inputs across the output classes. And a quick glimpse of the heat-map is able to tell apart the features in the distinct time-intervals that are significant for the output class on the given input time series.

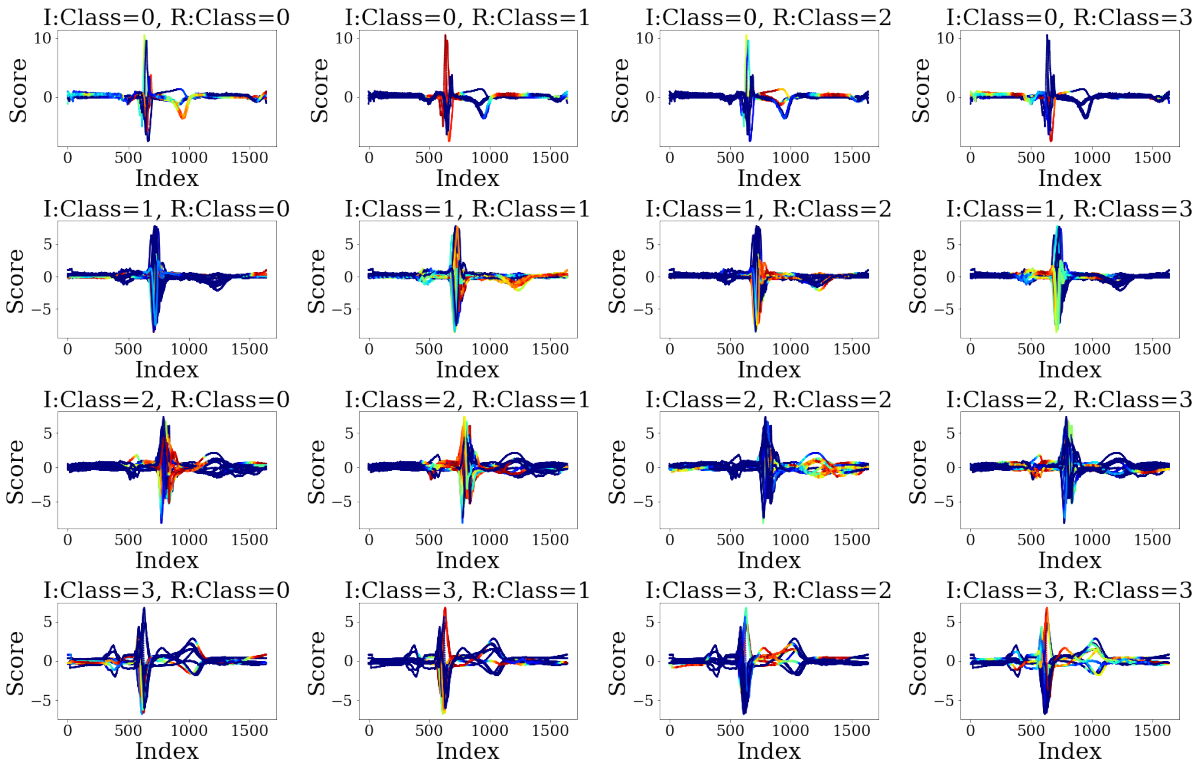


Figure 6.12: Relevance map for CineCECGTorso dataset across all the output classes.

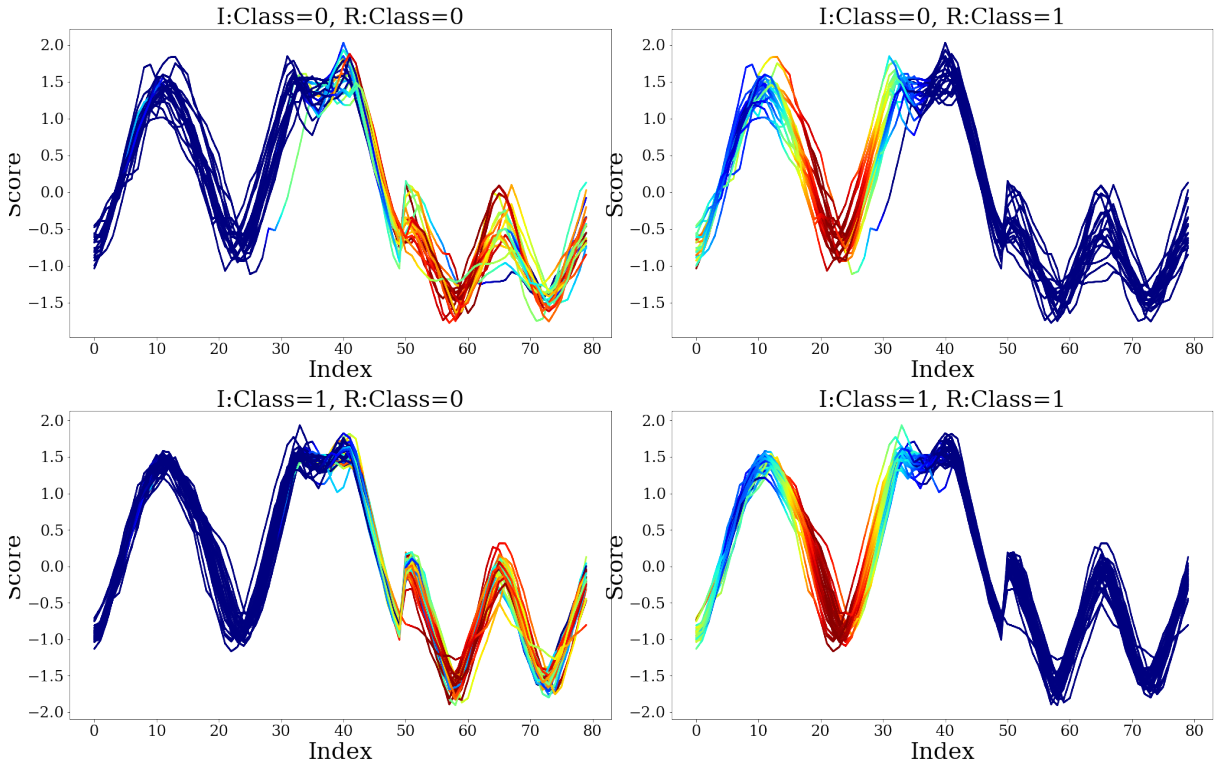


Figure 6.13: Relevance map for instances of DistalPhalanxOutlineCorrect dataset across all the output classes.

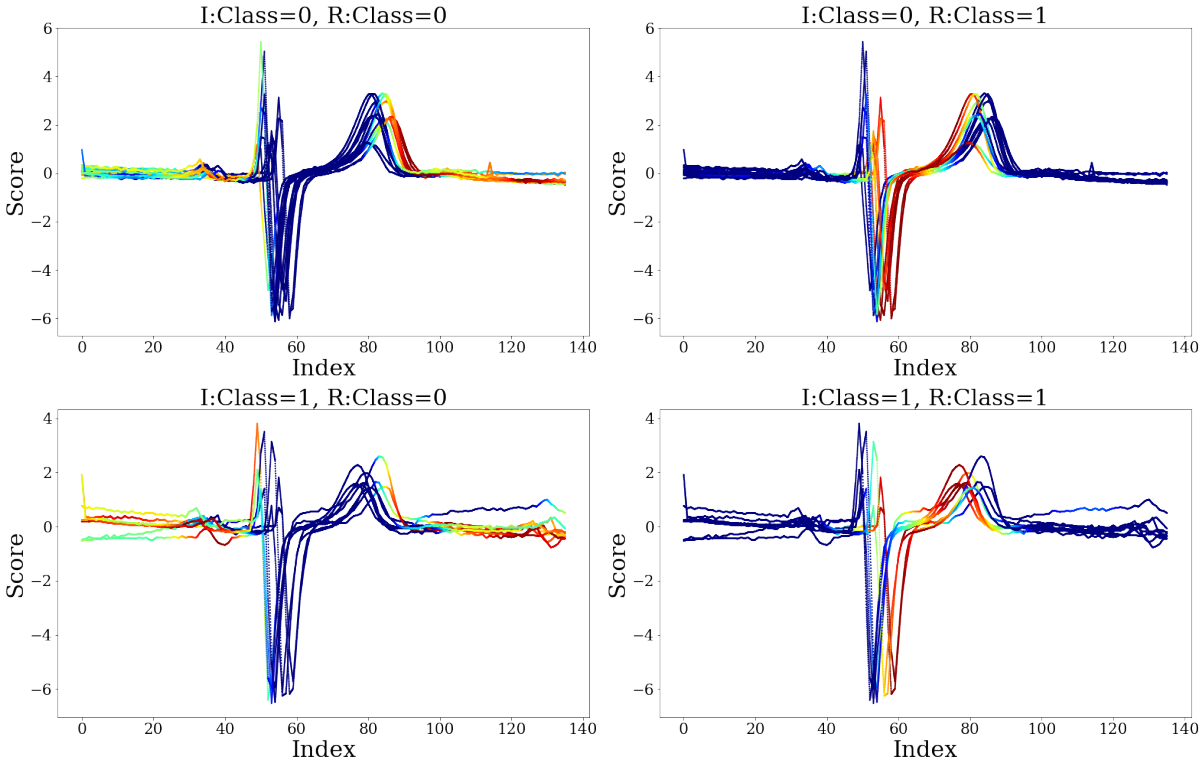


Figure 6.14: Relevance map for instances of ECGFiveDays dataset across all the output classes.

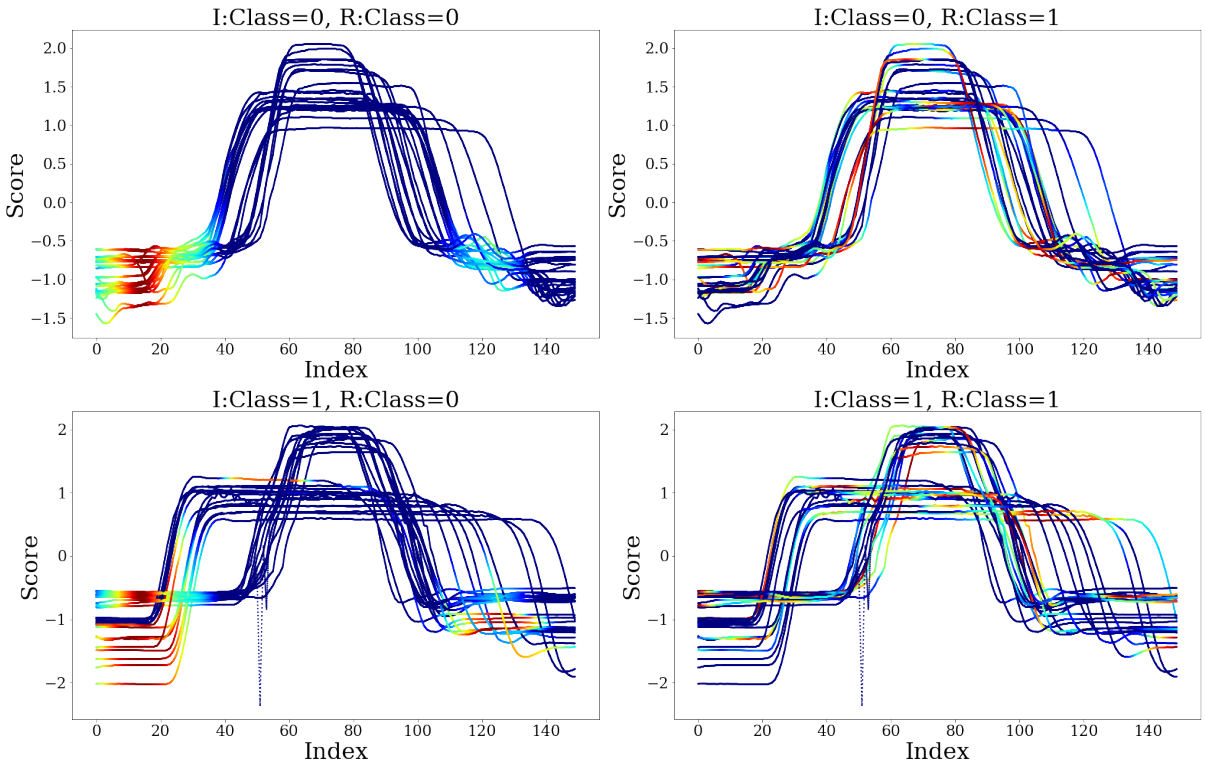


Figure 6.15: Relevance map for instances of GunPoint dataset across all the output classes.

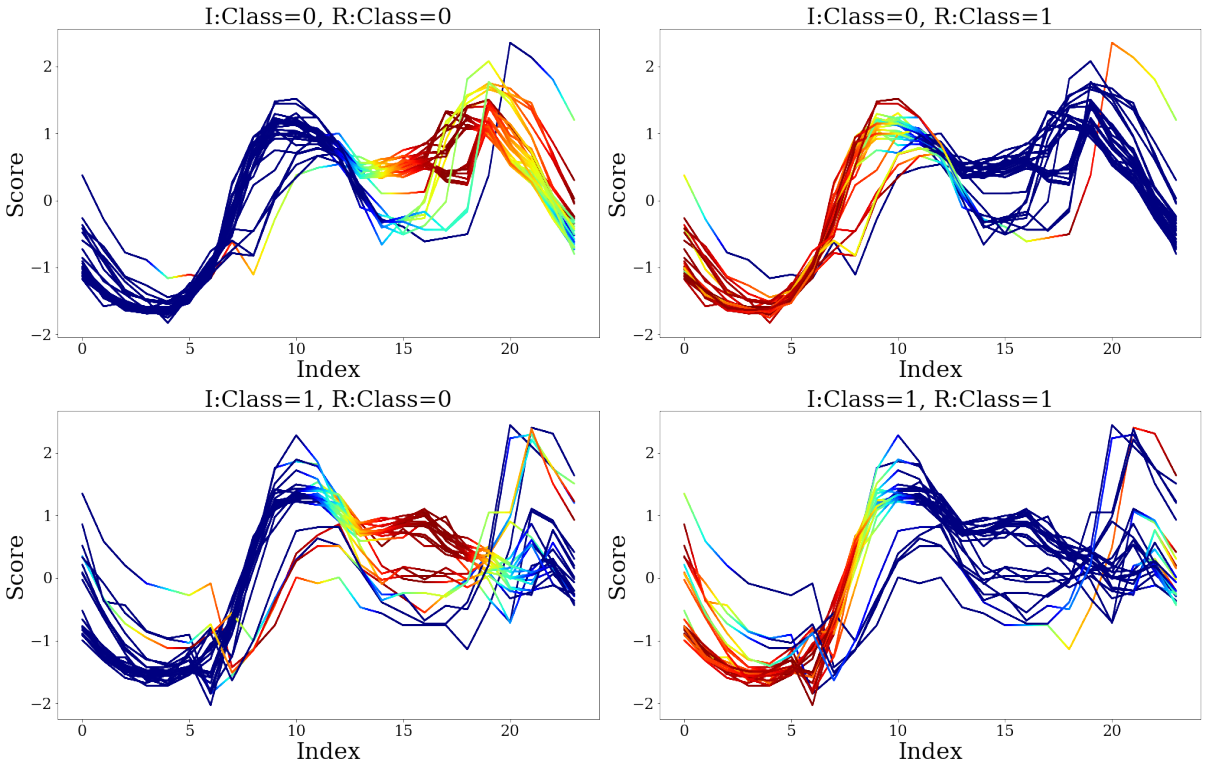


Figure 6.16: Relevance map for instances of ItalyPowerDemand dataset across all the output classes.

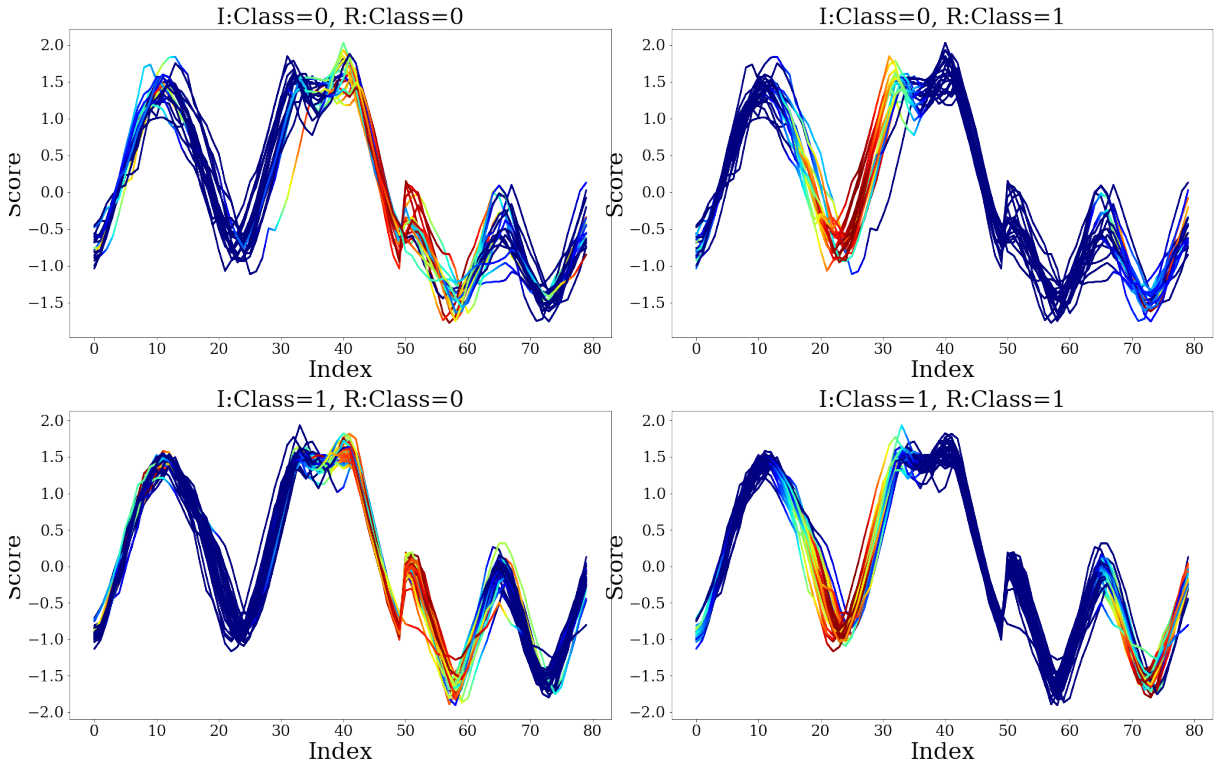


Figure 6.17: Relevance map for instances of PhalangesOutlinesCorrect dataset across all the output classes.

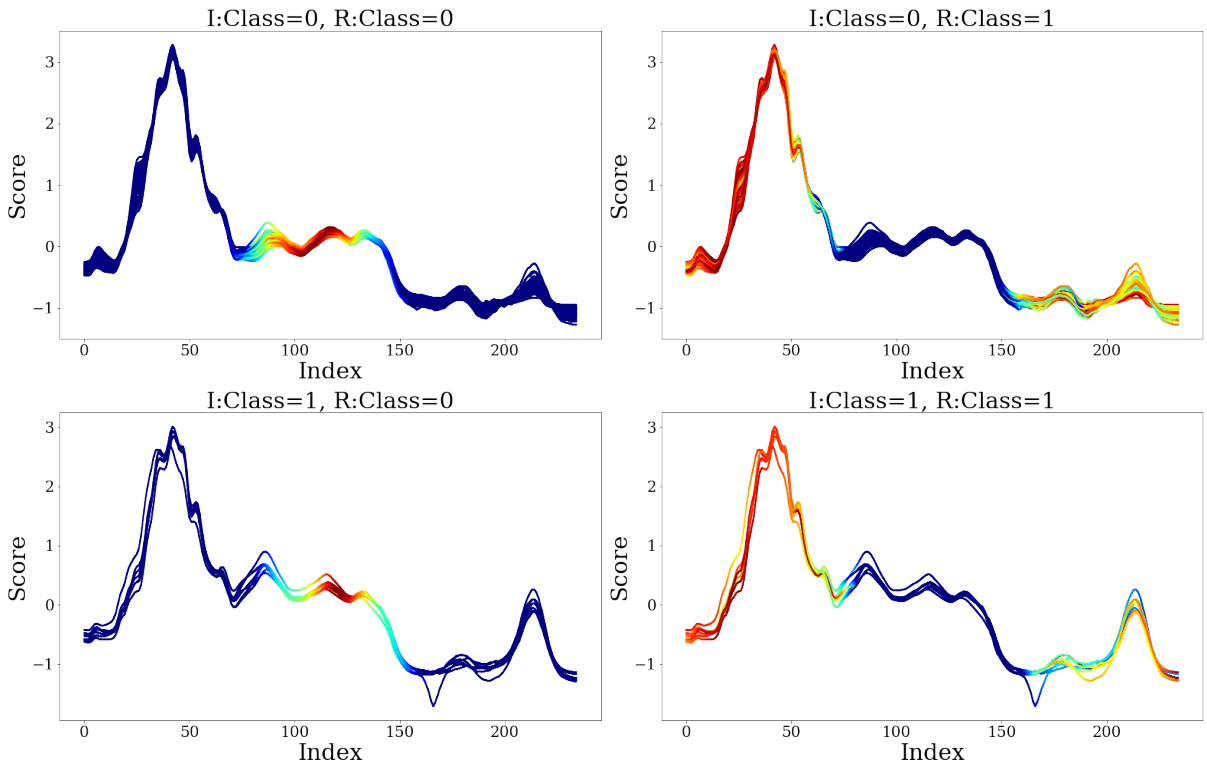


Figure 6.18: Relevance map for instances of Strawberry dataset across all the output classes.

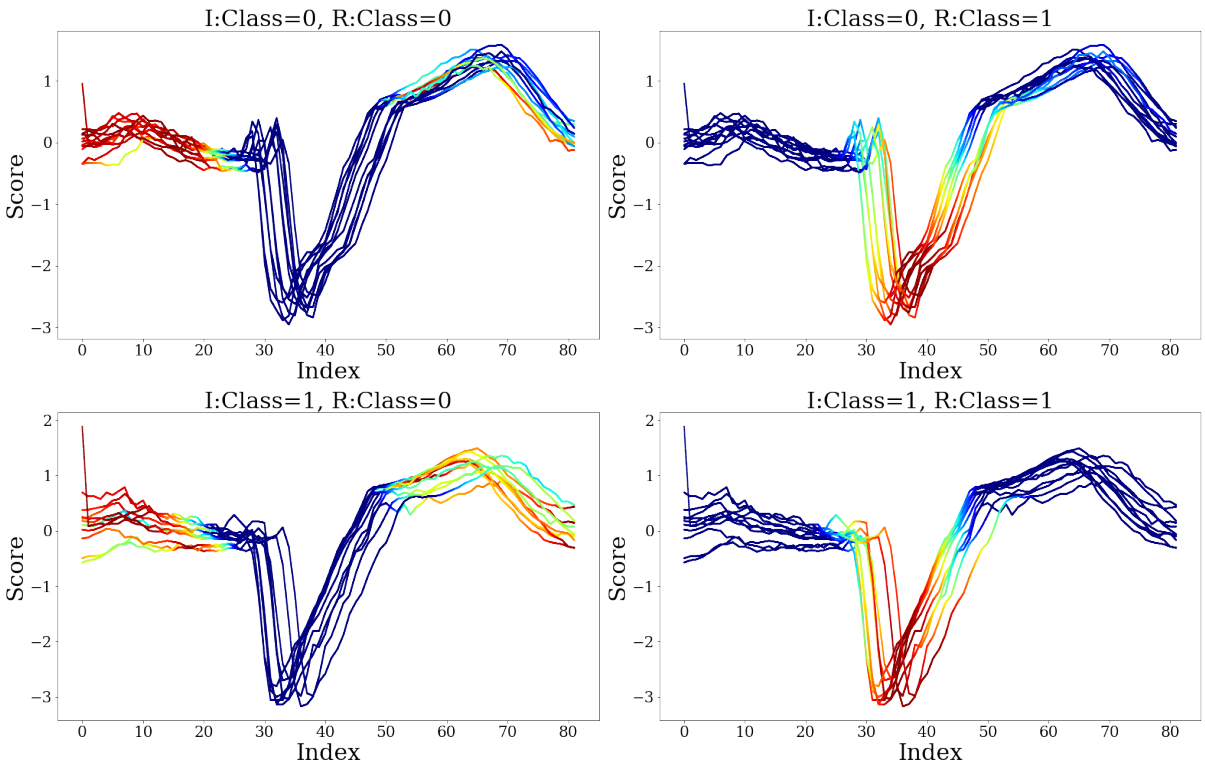


Figure 6.19: Relevance map for instances of TwoLeadECG dataset across all the output classes.

Chapter 7

Conclusion

In this thesis, we propose a novel approach for securing the automotive systems and interpreting the decision-making of the machine learning based classifier used in automotive systems and other safety-critical systems.

In the first chapter, we show the capability of the power consumption measurement-based method to distinguish the transmission from the idle state of the ECUs resulting in accurate sender identification. We show that the approach can be used to detect the presence of compromised and additional devices on the network. Preliminary results of the approach against a lab setup and a practical setting show that the technique is highly effective with a false positive rate of 0.004%. We also show that the approach applies to different network settings without compromising the accuracy and without completely retraining the model.

In the second and third chapters, we propose explanation techniques using perturbation-based method for explaining models decisions on different types of inputs such as images and time series. In addition, we propose an approach to generalize the explanation by find a range of acceptable variations for the sensitive parts of the input by generating various transformations of the original input. The transformations are such that the model classifies the input to the same target output class. The visualized interpretation of the relevance masks for a set of example inputs shows that the technique can detect the most sensitive features of the object under classification.

In the third chapter, we proposed a time series-based model outcome explanation approach. We show that the approach can locate the features and the time interval of importance for classifying the output class. And we observe that the explanations align

with expert ground truth knowledge, pointing towards users' confidence and reliability in using the technique for decision making.

We can apply the techniques proposed in this thesis to other avenues where the security and interpretability of the decision-making system are crucial. The method will be secure against impersonation attacks as the technique relies on non-clonable power consumptions to fingerprint the ECUs. The sender authentication followed by a model outcome explainability technique alleviates the gap between the decision-making process and the rationale behind the model's decisions.

As the future work, a direct extension of the time series-based explanation approach is to explain the multi-variate time series (MTS) that are widely applied across domains for decision-making. The method for explaining the time series-based inputs and images can be extended to answer questions such as why the model is biased and the robust features for discriminating between the classes?

References

[1] Kvaser Tool. https://www.kvaser.com/.

[2] This Car Runs on Code. https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code.

[3] Titanic dataset. http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.html.

[4] Vehicle Reverse Engineering Wiki, 2019. http://vehicle-reverse-engineering.wikia.com.

[5] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Zhiwei Steven Wu, and Himabindu Lakkaraju. Towards the Unification and Robustness of Perturbation and Gradient Based Explanations. *arXiv:2102.10618 [cs]*, June 2021. arXiv: 2102.10618.

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. 2015.

[7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing Robust Adversarial Examples. 2018.

[8] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives. *CoRR*, 2012.

[9] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*. 2011.

[10] Alexander Binder, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers. 2016.

[11] Peter Bloomfield. Fourier Analysis of Time Series: An Introduction. *New York: Wiley-Interscience*, 2000.

[12] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-CAM++: Generalized Gradient-based Visual Explanations for Deep Convolutional Networks. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar 2018.

[13] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[14] Hugh Chen, Scott Lundberg, and Su-In Lee. Explaining Models by Propagating Shapley Values of Local Components, 2019.

[15] Kyong-Tak Cho and Kang G. Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 911–927, Austin, TX, 2016. USENIX Association.

[16] Kyong-Tak Cho and Kang G. Shin. Viden: Attacker Identification on In-Vehicle Networks. *CoRR*, 2017.

[17] Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism, 2017.

[18] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. Identifying ECUs using Inimitable Characteristics of Signals in Controller Area Networks. *CoRR*, 2016.

[19] Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices. In *Presented as part of the 2013 USENIX Workshop on Health Information Technologies*, 2013.

[20] Piotr Dabkowski and Yarin Gal. Real Time Image Saliency for Black Box Classifiers. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 2017.

[21] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real Time Syst.*, 2007.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.

[23] Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning, 2017.

[24] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017.

[25] Eisenbarth, Thomas, Paar, Christof, Weghenkel, and Bj orn. Building a Side Channel Based Disassembler. Springer Berlin Heidelberg, 2010.

[26] Shaker El-Sappagh, José M. Alonso, Farman Ali, Amjad Ali, Jun-Hyeog Jang, and Kyung-Sup Kwak. An Ontology-Based Interpretable Fuzzy Decision Support System for Diabetes Diagnosis. *IEEE Access*, 6:37371–37394, 2018.

[27] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep Learning for Time-Series Classification: A Review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019.

[28] François-Guillaume Fernandez. TorchCAM: Class Activation Explorer, March 2020.

[29] Ruth Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. 2017.

[30] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based Grad-CAM: Towards Accurate Visualization and Explanation of CNNs, 2020.

[31] Wendong Ge, Jin-Won Huh, Yu Rang Park, Jae Ho Lee, Young-Hak Kim, and Alexander Turchin. An Interpretable ICU Mortality Prediction Model Based on Logistic Regression and Recurrent Neural Networks with LSTM units. 2018.

[32] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2013.

[33] Ary Goldberger, Luis A Nunes Amaral, J.M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 2000 (June 13).

[34] Ian Goodfellow, Honglak Lee, Quoc V. Le, Andrew Saxe, and Andrew Y. Ng. Measuring Invariances in Deep Networks. 2009.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. 2014.

[36] Bogdan Groza and Pal-Stefan Murvay. Efficient Protocols for Secure Broadcast in Controller Area Networks. *Industrial Informatics, IEEE Transactions on*, 9:2034–2042, 11 2013.

[37] Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, 2001.

[38] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. The Elements of Statistical Learning, 2001.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. 2015.

[40] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating Visual Explanations. 2016.

[41] Anthony Van Herrewege, Singelée, Dave Singelee, and Ingrid Verbauwhede. CANAuth – A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. page 7, 01 2011.

[42] Rob Hyndman, Anne Koehler, Keith Ord, and Ralph Snyder. Forecasting with Exponential Smoothing, 2008.

[43] International Organization for Standardization. International Standard ISO-26262 – Road Vehicles Functional Safety, 2018.

[44] Hanley JA and McNeil BJ. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. 1982.

[45] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. LayerCAM: Exploring Hierarchical Class Activation Maps for Localization. *IEEE Transactions on Image Processing*, 30:5875–5888, 2021.

[46] Kathan Kashiparekh, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time-Series Classification, 2019.

[47] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). 2018.

[48] Marcel Kneib and Christopher Huth. Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 787–800, New York, NY, USA, 2018. ACM.

[49] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, Berlin, Heidelberg, 1999. Springer-Verlag.

[50] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak N. Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 447–462, Washington, DC, USA, 2010. IEEE Computer Society.

[51] Karl Koscher, Tadayoshi Kohno, and David Molnar. SURROGATES: Enabling Near-Real-Time Dynamic Analyses of Embedded Systems. In *9th USENIX Workshop on Offensive Technologies, WOOT '15*, 2015.

[52] Raghavan Krishnan and Sarangapani Jagannathan. Hierarchical Mahalanobis Distance Clustering Based Technique for Prognostics in Applications Generating Big Data. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 516–521, Cape Town, South Africa, December 2015. IEEE.

[53] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 2015.

[54] Yen-Hsien Lee, Chih-Ping Wei, Tsang-Hsiang Cheng, and Ching-Ting Yang. Nearest-Neighbor-Based Approach to Time-Series Classification. *Decision Support Systems*, 53(1):207–217, 2012.

[55] Chung-Wei Lin and Alberto L. Sangiovanni-Vincentelli. Cyber-Security for the Controller Area Network (CAN) Communication Protocol. In *Proceedings of the 2012 International Conference on Cyber Security*, CYBERSECURITY '12, pages 1–7, Washington, DC, USA, 2012. IEEE Computer Society.

[56] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A Symbolic Representation of Time-Series, with Implications for Streaming Algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. Association for Computing Machinery.

[57] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference*, 2014.

[58] Liu et al. On Code Execution Tracking via Power Side-Channel. In *ACM Conference on Computer and Communications Security*. ACM, 2016.

[59] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. 2014.

[60] Scott M. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. page 10.

[61] Igor Melnykov and Volodymyr Melnykov. On K-means Algorithm with the use of Mahalanobis Distances. *Statistics & Probability Letters*, 84:88–95, January 2014.

[62] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle, 2015.

[63] Charlie Miller and Chris Valasek. Advanced CAN Injection Techniques for Vehicle Networks, 2016.

[64] Glenn W. Milligan. Construction and Assessment of Classification Rules. 1997.

[65] Christoph Molnar. Interpretable Machine Learning, 2019.

[66] D.C. Montgomery. Design and Analysis of Experiments, 2008.

[67] Carlos Moreno and Sebastian Fischmeister. On the Security of Safety-Critical Embedded Systems: Who Watches the Watchers? Who Reprograms the Watchers? In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, 2017.

[68] Carlos Moreno and Sebastian Fischmeister. Sender Authentication for Automotive In-Vehicle Networks through Dual Analog Measurements to Determine the Location of the Transmitter. In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP)*, 2019.

[69] Carlos Moreno, Sebastian Fischmeister, and M. Anwar Hasan. Non-intrusive Program Tracing and Debugging of Deployed Embedded Systems Through Side-Channel Analysis. 2013.

[70] Mehari Msgna, Konstantinos Markantonakis, and Keith Mayes. The B-side of Side Channel Leakage: Control Flow Security in Embedded Systems. In *Security and Privacy in Communication Networks - 9th International ICST Conference, SecureComm 2013*, 2013.

[71] Pal-Stefan Murvay and Bogdan Groza. Source Identification Using Signal Characteristics in Controller Area Networks. *Signal Processing Letters, IEEE*, 21:395–399, 04 2014.

[72] Rakshit Naidu, Ankita Ghosh, Yash Maurya, Shamanth R Nayak K, and Soumya Snigdha Kundu. IS-CAM: Integrated Score-CAM for Axiomatic-based Explanations, 2020.

[73] Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks. 2016.

[74] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models, 2019.

[75] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal Explanations: Justifying Decisions and Pointing to the Evidence. 2018.

[76] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. 2016.

[77] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *British Machine Vision Conference 2018, BMVC 2018*, 2018.

[78] Dabkowski Piotr and Gal Yarin. Real Time Image Saliency for Black Box Classifiers. 2017.

[79] Samuele Poppi, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Revisiting the Evaluation of Class Activation Mapping for Explainability: A Novel Metric and Experimental Analysis. page 6.

[80] Foster J. Provost, Tom Fawcett, and Ron Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms. ICML '98, 1998.

[81] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction, 2017.

[82] Chotirat Ratanamahatana and Eamonn Keogh. Making Time-Series Classification More Accurate Using Learned Constraints. 04 2004.

[83] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. 2016.

[84] Robert Bosch GmbH. CAN Specification, Version 2.0, 1991.

[85] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable Artificial Intelligence (XAI) on Time-Series Data: A Survey. *arXiv:2104.00950 [cs]*, April 2021. arXiv: 2104.00950.

[86] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. 2015.

[87] Sang Uk Sagong, Xuhang Ying, Andrew Clark, Linda Bushnell, and Radha Poovendran. Cloaking the Clock: Emulating Clock Skew in Controller Area Networks. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 32–42, 2018.

[88] Sang Uk Sagong, Xuhang Ying, Radha Poovendran, and Linda Bushnell. Exploring Attack Surfaces of Voltage-Based Intrusion Detection Systems in Controller Area Networks. In *Proceedings of the 16th ESCAR Europe (ESCAR'18)*, 2018.

[89] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A. Keim. Towards a Rigorous Evaluation of XAI Methods on Time-Series. 2019.

[90] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. 2017.

[91] Pavel Senin and Sergey Malinchik. SAX-VSM: Interpretable Time-Series Classification Using SAX and Vector Space Model. In *2013 IEEE 13th International Conference on Data Mining*, pages 1175–1180, 2013.

[92] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. *arXiv:1704.02685 [cs]*, October 2019. arXiv: 1704.02685.

[93] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. *arXiv:1605.01713 [cs]*, April 2017. arXiv: 1605.01713.

[94] Shoaib Ahmed Siddiqui, Dominique Mercier, Mohsin Munir, Andreas Dengel, and Sheraz Ahmed. TSViz: Demystification of Deep Learning Models for Time-Series Analysis. *IEEE Access*, 7:67027–67040, 2019.

[95] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014.

[96] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015.

[97] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. 2015.

[98] Pierre Stock and Moustapha Cissé. ConvNets and ImageNet Beyond Accuracy: Explanations, Bias Detection, Adversarial Examples and Model Criticism. 2017.

[99] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. 2017.

[100] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. 2015.

[101] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, New York, NY, USA, 2008.

[102] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.*, 2010.

[103] Haofan Wang, Rakshit Naidu, Joy Michael, and Soumya Snigdha Kundu. SS-CAM: Smoothed Score-CAM for Sharper Visual Feature Localization, 2020.

[104] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks, 2020.

[105] Jingyuan Wang, Zhen Peng, Xiaoda Wang, Chao Li, and Junjie Wu. Deep Fuzzy Cognitive Maps for Interpretable Multivariate Time-Series Prediction. *IEEE Transactions on Fuzzy Systems*, 29(9):2647–2660, 2021.

[106] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemela, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. 2015.

[107] Lexiang Ye and Eamonn Keogh. Time-Series Shapelets: A New Primitive for Data Mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 947–956, New York, NY, USA, 2009. Association for Computing Machinery.

[108] Raymond A. Yeh, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. Semantic Image Inpainting with Perceptual and Contextual Losses. 2016.

[109] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding Neural Networks Through Deep Visualization. 2015.

[110] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks, 2013.

[111] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down Neural Attention by Excitation Backprop. 2016.

[112] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.