ABSTRACT

ROHEDA, SIDDHARTH. Multi-Modal Sensor Fusion: A Principled Approach to Optimality. (Under the direction of Hamid Krim and Tianfu Wu.)

In surveillance settings, one often resorts to fusing data from more than one sensor to successfully carry out target detection or classification. Data fusion has, over the years been recognized to distinguish three levels of fusion, namely data level, feature level, and decision level. Data level fusion generally processes raw data generated by each sensor, and performs the fusion of information according to some criterion before proceeding to inference. Feature level fusion, on the other hand, first gleans features from raw data (e.g., transformed data) from diverse sensors, to subsequently and coherently merge them for inference. In decision level sensor fusion, each sensor reaches an individual decision, prior to a joint analysis and optimal combination of the decisions to yield a more informed and often improved decision, e.g. target classification.

Traditionally, Decision Level Sensor Fusion obtains individual decisions from each sensor, and combines these to make an optimal decision. We explore Decision Level Sensor Fusion under a different light. Each sensor is said to make a decision on occurrence of certain events that it is capable of observing, rather than making a decision on whether a certain target is present. A classification decision is reached by cataloguing sets of events, along with the probabilistic characterization for each sensor, and following a joint probabilistic and coherent evaluation of these events. These events are formalized to each sensor according to its potentially extracted attributes to define targets. What this in effect achieves, is a probability measure assignment to a specific target following its description. The proposed technique also explores the extent of dependence between features/events being observed by the sensors, and hence generates more informed probability distributions over the events. In our case, we will study two different datasets. The first one, combines a Radar sensor with an optical sensor. A radar is used to explore the velocity of an object among other things, thus defining a sample space and a Sigma-Field with an associated probability measure, and is coupled to a telescopic sensor with an analogously associated probability space. This product space thus allows us to define a principled fusion framework with an improved and robust performance. Similarly, the second dataset will involve a seismic sensor, coupled with an acoustic sensor. Provided some additional information about the features of the object, this fusion technique can outperform other existing decision level fusion approaches that may not take into account the relationship between different features. We subsequently develop a data driven approach to multi-modal fusion, where optimal features for each sensor are selected from a hidden latent space between the different modalities. This hidden space is learned via a generative network conditioned on individual sensor modalities. The hidden space, as an intrinsic structure, is then exploited in the detection of damaged sensors, and in subsequently safeguarding the performance of the fused sensor system. The hidden common sub-space is learned using a bank of Conditional Generative Adversarial Networks.
(CGANs), where the Discriminator attempts to correctly identify the modality that the estimate of the hidden space was generated by, while the generators attempt to generate hidden space estimates that are able to confuse the discriminator. We also introduce commutation between the various generators so as to enforce a common eigen-basis for the hidden space estimates. We show that including the commutation term significantly improves the results and leads to closer hidden space estimates. This is significantly different from the standard CGANs as the target distribution for the Generative network is unknown. Experimental results show that such an approach can make the system robust against noisy/damaged sensors, without requiring human intervention to inform the system about the damage.

Finally, in an attempt to reduce the complexity of Convolutional Neural Networks (CNNs), we propose a Volterra filter-inspired Network architecture. This architecture introduces controlled non-linearities in the form of interactions between the delayed input samples of data. We propose a cascaded implementation of Volterra Filtering so as to significantly reduce the number of parameters required to carry out the same classification task as that of a conventional Neural Network. We demonstrate an efficient parallel implementation of this Volterra Neural Network (VNN), along with its remarkable performance while retaining a relatively simpler and potentially more tractable structure. Furthermore, we show a rather sophisticated adaptation of this network to nonlinearly fuse the RGB (spatial) information and the Optical Flow (temporal) information of a video sequence for action recognition. The proposed approach is evaluated on UCF-101 and HMDB-51 datasets for action recognition, and is shown to outperform state of the art CNN approaches. Furthermore, we also evaluate the proposed architecture for target detection, and multi-modal fusion on datasets with multiple modalities collected from sensors like Radar, Telescopic Imaging, Seismic, and Acoustic sensors.
Multi-Modal Sensor Fusion: A Principled Approach to Optimality

by
Siddharth Roheda

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina
2020

APPROVED BY:

__________________________  ____________________________
Brian Hughes                Patrick Louis Combettes

__________________________  ____________________________
Hamid Krim                  Tianfu Wu
Co-chair of Advisory Committee

Co-chair of Advisory Committee
DEDICATION

In loving memory of my grandfather Gyan,
and to my parents, Deepak and Rekha.
Siddharth Roheda received his Bachelor of Technology in Electronics and Communication Engineering from Nirma University, Ahmedabad, Gujarat, India in May 2015. In August 2015, he started pursuing his PhD in the Department of Electrical and Computer Engineering in North Carolina State University at Raleigh and joined the Vision, Information and Statistical Signal Theories and Applications (VISSTA) group in August 2016. His current research interests include Robust Information Fusion, Machine Learning, Deep Learning, Computer Vision, and Signal Processing. More specifically, he is interested in Supervised Learning, developing novel deep learning architectures and robust algorithms for exploiting multi-modal information for target detection and recognition.
ACKNOWLEDGEMENTS

First and foremost I would like to express my sincere gratitude to my advisor Dr. Hamid Krim for the constant support in my PhD study and research. His patience, motivation, enthusiasm, and immense knowledge is commendable. Without his support this would have never been possible. I would also like to thank my co-advisor Dr. Tianfu Wu, who was always available for discussions and provided some very useful insights throughout my PhD. I express my gratitude to the rest of my committee: Dr. Brian Hughes, and Dr. Patrick Lois Combettes for all your suggestions, comments, and questions.

I also acknowledge Dr. Benjamin Riggan for his assistance, expertise, and guidance during my internship at the Army Research Laboratory (ARL). My sincere thanks also goes to Dr. Ashkan Panahi, Dr. Han Wang, and Dr. Anir Adiga, the post-doctoral researchers in the Vision, Information and Statistical Signal Theories and Applications (VISSTA) group for their guidance during my PhD. I would like to thank my colleagues and close friends: Yuming Huang, Shahin Mahdizadehaghdam, Kenneth Tran, Wen Tang, Bo Jiang, Sally Ghanem, Tanmay Asthana, Reza Solemani, Mohsen Ardakani, Divyakumar Badheka, Shalin Rathi, Aditya Rai, Saurabh Bhansali, Varun Thirani, and Srinjoy Sen for the valuable discussions and assistance during my time at NC State. I am grateful to my friends back home: Samyak Jain and Rakesh Ravikumar for being just a phone call away in trying times. Finally, I thank my parents, Deepak Roheda and Rekha Ahuja. I am forever indebted to them for their unconditional love and unparalleled support.
# TABLE OF CONTENTS

LIST OF TABLES .................................................................................................................. vii

LIST OF FIGURES .................................................................................................................. viii

Chapter 1 INTRODUCTION ........................................................................................................ 1
  1.1 Review of State of Art ....................................................................................................... 1
  1.2 Contributions .................................................................................................................. 2

Chapter 2 Event Driven Sensor Fusion ........................................................................................ 4
  2.1 Introduction ..................................................................................................................... 4
  2.2 Related Work .................................................................................................................. 5
    2.2.1 Bayesian Inference ................................................................................................. 6
    2.2.2 Dempster-Shafer Fusion ......................................................................................... 6
    2.2.3 Model Based Fusion ............................................................................................... 7
    2.2.4 Mutual Information ................................................................................................. 8
    2.2.5 Support Vector Machines ....................................................................................... 8
  2.3 Problem Formulation ....................................................................................................... 9
  2.4 Proposed Method ........................................................................................................... 10
    2.4.1 Determining object Probabilities .......................................................................... 11
    2.4.2 Determining the Joint Probability ........................................................................ 11
    2.4.3 Robustness: Addressing damaged sensors ........................................................... 12
  2.5 Experiments and Results ............................................................................................... 17
    2.5.1 Dataset 1: Radar and Telescopic Imaging Sensors ................................................. 17
    2.5.2 Dataset 2: Acoustic and Seismic Sensors ............................................................... 23
    2.5.3 Performance Analysis ......................................................................................... 24
    2.5.4 Robustness Evaluation ......................................................................................... 26
  2.6 Conclusion ....................................................................................................................... 27

Chapter 3 Commuting Conditional GANs for Robust Multi-Modal Fusion .............................. 29
  3.1 Introduction ..................................................................................................................... 29
  3.2 Related Work .................................................................................................................. 29
    3.2.1 Generative Adversarial Networks ........................................................................... 29
    3.2.2 Teacher-Student Distillation Model ....................................................................... 31
  3.3 Problem Formulation ..................................................................................................... 32
  3.4 Proposed Approach ........................................................................................................ 33
    3.4.1 Structure of the Hidden Space/Selection Operators .............................................. 35
    3.4.2 A Special Case of Event Driven Fusion ................................................................. 37
    3.4.3 Sensor Failure Detection ....................................................................................... 38
  3.5 Experiments and Results ............................................................................................... 40
    3.5.1 Implementation Details ....................................................................................... 40
    3.6 Performance Analysis ................................................................................................. 42
      3.6.1 Robustness Analysis ......................................................................................... 45
  3.7 Conclusion ....................................................................................................................... 46

Chapter 4 Volterra Neural Networks (VNNs): A Volterra Filtering Approach for Human Action Recognition, Target Detection, and Multi-Modal Fusion ............................................ 47
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Performance Comparison for the First Dataset</td>
<td>25</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Performance Comparison for the Second Dataset</td>
<td>25</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>Robustness Analysis for the second dataset</td>
<td>28</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Performance Comparison for both the Datasets</td>
<td>43</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Performance Evaluation for one stream networks (RGB only): The proposed algorithm achieves best performance when trained from scratch</td>
<td>65</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Performance Evaluation for two stream networks (RGB &amp; Optical Flow): The proposed algorithm achieves best performance on both datasets.</td>
<td>65</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Norm of $W^u$, where $u \in {\text{RGB,OF,Fusion}}$</td>
<td>66</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Comparison of number of parameters required and processing speed with the state of the art. A video with 60 frames is evaluated</td>
<td>66</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Performance Comparison for the Simulated Dataset</td>
<td>67</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Performance Comparison for Real Data</td>
<td>67</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>Performance Comparison for Imaging, Seismic, and Acoustic Sensor Network for Target Detection.</td>
<td>69</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

| Figure 2.1 | Decision Level Fusion of multi-sensor observations | 5 |
| Figure 2.2 | Possible hyperplanes for separating data belonging to different classes | 9 |
| Figure 2.3 | Block diagram of the Event Driven Approach for fusion | 10 |
| Figure 2.4 | Using a Global Hidden Space for Event Driven Fusion | 13 |
| Figure 2.5 | Using Independent Hidden Spaces for Event Driven Fusion | 15 |
| Figure 2.6 | Hidden Space recovery for a two sensor scenario, \( l \in \{1, 2\} \), hidden space for features defined for damaged sensor, \( l = 1 \), are recovered by observations from sensor, \( l = 2 \) | 17 |
| Figure 2.7 | Example to illustrate advantage of using commuting matrices to determine common hidden subspace | 18 |
| Figure 2.8 | Event Driven Fusion for object detection in Dataset 1 | 18 |
| Figure 2.9 | MATLAB Simulink Block Diagram for generation of simulated radar signals | 19 |
| Figure 2.10 | (a): Original Telescopic Image, (b): Detected Object of interest is marked by a red outline, (c): Detected Object of interest is marked by a red outline (zoomed in) | 21 |
| Figure 2.11 | (a): Sequence of two consecutive images from telescopic sensor, (b): Probability distribution over location of object of interest in \( I_2 \) | 22 |
| Figure 2.12 | Sample (a): seismic sensor observations and (b): acoustic sensor observations for human, vehicular, and no target cases | 23 |
| Figure 2.13 | Event Driven Fusion for object detection - Dataset 2 | 24 |
| Figure 2.14 | ROC Curves for detection (Dataset 1) of (a): Class 1: Object 1, (b): Class 2: Object 2, (c): Class 3: Neither Object 1 nor Object 2 | 25 |
| Figure 2.15 | ROC Curves for detection (second dataset) of (a): Class 1: Human Target, (b): Class 2: Vehicular Target, (c): Class 3: No Target | 26 |
| Figure 2.16 | (a): Using Independent Hidden Spaces to deal with a damaged radar sensor, (b): Comparison of proposed approach with existing techniques, when sensors get damaged | 27 |
| Figure 2.17 | Using Independent Hidden Spaces to deal with damaged Seismic/Acoustic sensors | 28 |
| Figure 3.1 | Conditional Generative Adversarial Networks | 30 |
| Figure 3.2 | Transformation of a uniform distribution on the unit circle to a unit square | 31 |
| Figure 3.3 | Transformation of a uniform distribution on the unit square to a unit circle | 31 |
| Figure 3.4 | Transformation of a uniform distribution on the unit disk to a unit square | 32 |
| Figure 3.5 | Transformation of a uniform distribution on the unit square to a unit disk | 32 |
| Figure 3.6 | Fused joint probability distributions when accounting for the uncertain event. The joint distribution is determined following the rules of Event Driven Fusion | 38 |
| Figure 3.7 | A clustering tree created using Hierarchical Clustering (Agglomerative Clustering) | 40 |
| Figure 3.8 | (a): Sample video frame with a human target, (b): Sample seismic sensor observations for human target, vehicular target, and no target cases, (c): Sample acoustic sensor observations for human target, vehicular target, and no target cases | 41 |
| Figure 3.9 | Block Diagram of Implementation | 42 |
Figure 3.10 (a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when GAN Loss is optimized in tandem with classification loss ........................................ 44
Figure 3.11 (a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when commutation term is included with the GAN and classification term ........................................ 44
Figure 3.12 (a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when all terms in Equation (3.10) are active. ............................................................... 45
Figure 3.13 Comparison of the proposed technique (GAN + Event Driven Fusion) with existing techniques for Dataset-1. ................................................................. 45
Figure 3.14 Comparison of the proposed technique (GAN + Event Driven Fusion) with existing techniques for Dataset-2. ................................................................. 46

Figure 4.1 Adaptive Volterra Filter ........................................ 49
Figure 4.2 Nested Volterra Filter ........................................ 50
Figure 4.3 A Long Short Term Memory Network ...................... 51
Figure 4.4 Volterra Filter must be applied in both space and time for video recognition. 53
Figure 4.5 Block diagram for an Overlapping Volterra Neural Network ............... 54
Figure 4.6 Implementation of a second order Volterra Filter using $Q^{th}$ rank approximation 57
Figure 4.7 Implementation of (a): LSTM (b): O-VNN on a time series, $x = \{x_1, ..., x_t\}$ ... 59
Figure 4.8 (a): Decision Level Fusion, (b): Feature Concatenation, (c): Two-Stream Volterra Filtering ......................................................... 62
Figure 4.9 (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering ........................................ 63
Figure 4.10 (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering ........................................ 64
Figure 4.11 Epochs vs Loss for various number of multipliers for a Cascaded Volterra Filter 64
Figure 4.12 Block diagram for implementation of the Volterra Filtering approach on Radar and Telescopic Imaging data. ................................. 67
Figure 4.13 Data Samples for ISS detection ................................. 68
Figure 4.14 Detection of ISS using VNN learned on Telescopic Imaging. Red box highlights the location of the streak observed as a result of the ISS flyover. ....... 69
Figure 4.15 Detection of human target using VNN learned on the Imaging Sensor ....... 70
In recent years, multi-modal fusion has attracted a lot of research interest, both in academia, and in industry. Multi-modal fusion entails the combination of information from a set of different types of sensors. In particular, target detection and recognition problems have widely used a multi-modal approach in order to boost performance. Different sensors may provide different information about a target, and hence, a fusion system combining information from multiple sensors is expected to yield a better performance than that obtained from an individual sensor. Such a fusion of sensor data, however, requires some principled strategy to ensure that additional data is constructively used, and has a positive impact on performance. Sensor fusion is known to broadly classify fusion techniques into three classes: Data level fusion is used when one wishes to combine raw data, and then proceed with inference. Feature level fusion first extracts information from raw data, and then merges these features to make coherent decisions. Finally, the latest point at which one can perform fusion is known as decision level fusion. Fusion at the decision level allows each sensor to reach its own individual decision (on the target identity), prior to an optimal combination of these decisions.

1.1 Review of State of Art

Decision level fusion has received a lot of attention when managing heterogeneous modalities, as it allows each modality to have an independent feature representation, thus providing additional information. A review of classical fusion approaches can be found in [28, 39, 57], and a more recent alternative data-driven perspective is provided by [83]. Over the years, many techniques like Bayesian Fusion [16], and Dempster-Shafer Fusion [72] have been widely used to combine sensor modalities at the decision level. More recently, approaches for fusion have been proposed [21, 48, 49], including for instance, a network of similar and dissimilar sensors. Similar Sensor Fusion [48, 49] is used when all sensors explore the same characteristics of the target, while Dissimilar Sensor Fusion [21, 48] is used when different and possibly incompatible sensors observe different characteristics of the
target, to hence provide new information about the target.

The viability of many fusion approaches strongly hinges on the functionality of all the sensors, making it a fairly restrictive solution. The severity of this limitation is even more pronounced in unconstrained surveillance settings where the environmental conditions have a direct impact on the sensors, and close manual monitoring is difficult or even impractical. Partial sensor failure (which may occur due to solar loading, jamming, over-heating, low battery, etc.) can hence cause a major drop in performance in a fusion system if timely failure detection is not performed. Furthermore, even if these sensors are successfully detected, the common adopted solution is to ignore the damaged sensor, with a potential negative impact on the overall resulting performance (i.e. relative to when all sensors are functional). We consider exploiting the prior information about the relationship between these sensor modalities, typically available from past observations during training of the fusion system, so that the latter can safeguard a high performance. Recently there has been an interest in such a transfer of knowledge. In [32], the authors introduce hallucination networks to address a missing modality at test time by distilling knowledge into the network during training. Here, a teacher-student network is implemented using an $L^2$ loss term (hallucination loss) to train the student network. This idea has been further extended in [30, 64, 81]. In [64], a Conditional Generative Adversarial Network (CGAN) was used to generate representative features for the missing modality. Furthermore, [59] shows that considering interactions between modalities can often lead to a better feature representation. Recent work in Domain Adaptation [27, 55, 90] addresses differences between source and target domains. In these works, the authors attempt to learn intermediate domains (represented as points on the Grassman manifold in [27, 90] and by dictionaries in [55]) between the source domain and the target domain.

### 1.2 Contributions

In Chapter 2 we look at decision level fusion under a different light, by combining occurrences of events to reach a probability measure for a target identity. Each sensor is said to make a decision on the occurrence of certain events that it observes, rather than making a decision on the target identity. A classification decision is reached by cataloguing sets of events and their associated joint probabilistic and coherent evaluation, together with the probabilistic characterization for each sensor. The events are formalized for each sensor according to its potentially extracted attributes to define targets. What this in effect achieves, is a probability measure assignment to a specific target following its description. This approach also explores the extent of dependence between features being observed by the sensors, and hence generates more informed probability distributions. The major drawback with this approach is the necessity of hand-picking the features which are subsequently used to define the target defining events. This makes classification/fusion particularly challenging when the target high level features (e.g. velocity, weight, etc.) are unknown.

In Chapter 3 we propose a data driven approach to learning the features of interest in order to solve the major drawback of hand-crafting them in Chapter 2. We use a special case of the Event
Driven Fusion technique discussed in Chapter 2, and modify it in order to include reliability of individual sensors. This reliability measure is adaptive, and accounts for the sensor condition during implementation. In addition, we learn a hidden common sub-space between the sensor modalities, and the optimal features for classification are driven by the existence of this hidden space. The hidden common sub-space is learned using a bank of Conditional Generative Adversarial Networks (CGANs), where the Discriminator attempts to correctly identify the modality that the estimate of the hidden space was generated by, while the generators attempt to generate hidden space estimates that are able to confuse the discriminator. We also introduce commutation between the various generators so as to enforce a common eigen-basis for the hidden space estimates. Furthermore, it also provides robustness against damaged sensors. This hidden space is learned via a generative network conditioned on individual sensor modalities. Unlike [64], we do not require a target feature space in order to learn the optimal hidden space estimates. The hidden space is structured so that it can accommodate both *shared and private* features of sensor modalities. Unlike [66], we do not need to assume a-priori knowledge about sensor damage, and can detect defective sensors on the basis of deviations in the generated hidden space.

The emergence of Convolutional Neural Networks (CNNs) [46], along with the availability of large training datasets and computational resources have come a long way to obtaining the various steps required in data classification by a single neural network. While CNNs are able to achieve near perfect classification, the models are extremely heavy to train, and have a tremendous number of parameters. This in addition to determining the necessary degree of non-linearity, makes the analysis difficult to understand, and the tractability elusive. In Chapter 4 we explore the idea of introducing controlled non-linearities through interactions between delayed samples of a time series. We will build on the formulations of the widely known Volterra Series [82] to accomplish this task. We propose a Volterra Filter [82] based network architecture for action recognition in videos. The non-linearities are introduced via the system response functions and hence by controlled interactions between delayed frames of the video. The overall model is updated on the basis of a cross-entropy loss of the labels resulting from a linear classifier of the generated features. An efficiently cascaded implementation of a Volterra Filter is used in order to explore higher order terms while avoiding over-parameterization. The Volterra filter principle is also exploited to combine the RGB and the Optical Flow streams for action recognition, hence yielding a performance driven non-linear fusion of the two streams. We further show that the number of parameters required to realize such a model is significantly lower in comparison to a conventional CNN, hence leading to faster training and significant reduction of the required resources to learn, store, or implement such a model. We also use such a Volterra Neural Network (VNN) for target detection and multi-modal fusion of various sensor modalities. Such an implementation of the Volterra Filter is also shown to be stable and convergent. The network architecture inspired by Volterra Filters achieves performance accuracies which are comparable to the state of the art approaches while using a fraction of the number of parameters used by them.
CHAPTER 2

EVENT DRIVEN SENSOR FUSION

2.1 Introduction

Often times more sensors are required in order to successfully detect and classify targets of interest. Additional sensors may provide supplementary information about a target, which can help the system make a more informed decision about its detection and classification. This data in turn often requires a degree of harnessing and fusion to seek an improved inference. In decision level fusion, each sensor reaches an individual decision, prior to optimal combination of these decisions to yield a more informed inference. The classical approach to decision level fusion is summarized in Figure 2.1. Over the years, we have seen classical techniques like Bayesian Fusion[16] and Dempster-Shafer Fusion[72] used for combining sensors at the decision level. While more recently we have seen model based approaches [21, 48, 49] that take into account the types of sensors that make up the network.

In this Chapter, we present a principled approach to decision level fusion for improved inference performance. A classification decision is reached by cataloging sets of events, along with the probabilistic characterization for each sensor, and following a joint probabilistic and coherent evaluation of these events. These events are formalized to each sensor according to its potentially extracted attributes to define targets. What this in effect achieves, is a probability measure assignment to a specific target following its description. Furthermore, we also address the practical situation where a sensor may be noisy or damaged and is no longer of use for fusion. We show that we can learn a hidden space between the sensors such that the fusion algorithm works around the damaged sensor, while achieving better performance than simply ignoring the damaged sensor, thus ensuring a graceful degradation. We formulate the problem of finding this hidden space by a criterion driven by the classification performance of a Support Vector Machine. In our case, we will study two different datasets. The first one, combines a Radar sensor with an optical sensor. A radar is used to explore the velocity of an object among other things, thus defining a sample space and a Sigma-Field with an
Figure 2.1 Decision Level Fusion of multi-sensor observations

associated probability measure, and is coupled to a telescopic sensor with an analogously associated probability space. This product space thus allows us to define a principled fusion framework with an improved and robust performance. Similarly, the second dataset will involve a seismic sensor, coupled with an acoustic sensor.

2.2 Related Work

As noted earlier, sensor fusion has long been of interest, albeit with limited theoretical success particularly when heterogeneous data are present, hence missing a unified and systematic approach which has remained elusive. An introduction and comprehensive survey to the area of fusion is provided in [28, 39]. As noted earlier, there has been significant research activity, starting from classical techniques like Bayesian Inference [16] and Dempster-Shafer Fusion [72]. Bayesian Fusion has shown success when prior knowledge about sensor reports is available. On the other hand, Dempster-Shafer fusion was proposed to specifically lift such a restriction on the information prior, at a cost of a substantial increase in computational complexity. In [33], a two-stage approach to sensor fusion was proposed, involving knowledge-modeling, which learns from past behavior of classifiers whose results are to be fused, and operation stage, that combines outputs of these classifiers based on knowledge learned in the first stage. More recent work in decision level fusion is based on the sensor network model [48]. Here, the network is modeled as either being made up of similar or dissimilar sensors. Similar Sensor Fusion [21, 48, 49] is used when all the sensors explore the same characteristics/features of the target (for example, a set of 5 radars, looking at the same target), while Dissimilar Sensor Fusion [21, 48] is alternatively used when sensors explore different characteristics/features of the target (for example, a radar and an optical sensor looking at the same target). These assumptions turn out to be too restrictive, in that some sensors, albeit dissimilar, may have some common features while offering additional features to enrich an object/target characterization. Our goal is to explore such a case, and demonstrate that a systematic and principled approach may be designed, and our resulting overall solution is improved on account of this enhancement. The following sub-sections discuss some of these existing techniques for fusion, and
provide the required background for the remainder of the paper.

### 2.2.1 Bayesian Inference

Consider a set of targets/objects to be detected and/or classified, \( O = \{o_1, o_2, ..., o_I\} \). The sensor report for the \( l^{th} \) sensor is defined as, \( D_l = \{P_l(o_i)\} \). The Bayesian Inference method for fusion is dependent on the knowledge of a-priori distributions, \( P_0(o_i) \), and conditional probabilities, \( P(D_l|o_i) \). Bayesian Inference uses Bayes' rule to fuse the reports \( D_1, D_2, ..., D_L \):

\[
P(o_i|D_1, D_2, ..., D_L) = \frac{P_0(o_i)P(D_1|o_i)...P(D_L|o_i)}{\sum_{j=1}^LP_0(o_j)P(D_1|o_j)...P(D_L|o_j)}. \tag{2.1}
\]

The determination of the a-priori probability distributions, \( P_0(o_i) \) is often difficult to obtain, and is one of the major limitations of Bayesian Inference.

### 2.2.2 Dempster-Shafer Fusion

Another approach to combining information from various sources is due to Dempster-Shafer Inference. Dempster-Shafer Inference can assign a probability to any of the \( I \) objects or to a union of these objects. The knowledge of the \( l^{th} \) sensor is summarized in its report, \( D_l = \{P_l(\omega^m_l), \omega^m_l \subset O, m = 1, ..., M_l\} \), where \( P_l(\omega_l) \in [0, 1], \sum_{m=1}^{M_l} P_l(\omega^m_l) = 1 \), and \( \omega^m_l \) denotes the \( m^{th} \) subset of objects seen by the \( l^{th} \) sensor. Due to lack of evidence, the probability 1 may not be completely assigned to any object or unions of objects, bringing an uncertainty in the report. The probability \( P(O) = P(o_1 \lor o_2 \lor ... \lor o_I) \) is therefore referred to as the probability of uncertainty. The sensor reports, \( D_1, D_2, ..., D_L \) are fused to find the final fused report, \( D_{fl} = \{P_{fl}(\omega^{m_{fl}}_{fl}), m_{fl} = 1, ..., M_{fl}\} \),

\[
P_{fl}(\omega^{m_{fl}}_{fl}) = \frac{\mu_{fl}(\omega^{m_{fl}}_{fl})}{1 - \mu_{fl}(\phi)}, \tag{2.2}
\]

where

\[
\mu_{fl}(\omega^{m_{fl}}_{fl}) = \sum_{m_1, ..., m_L: \omega^{m_1} \cap ... \cap \omega^{m_L} = \omega^{m_{fl}}} P_l(\omega^{m_1}_l) ... P_L(\omega^{m_L}_l). \tag{2.3}
\]

Dempster-Shafer rule for fusion suffers from exponentially increasing complexity as \( I \) and \( L \) increase. Some applications of Dempster-Shafer fusion can be found in [67] where LIDAR data is combined with multi-spectral imagery, and in [8] where multi-sensor information like vibration, sound, pressure, and temperature is fused to detect engine faults. Furthermore, [11] provides a detailed comparison between Bayesian Inference and Dempster-Shafer Theory.
2.2.3 Model Based Fusion

2.2.3.1 Similar Sensor Fusion

Similar Sensor Fusion model considers a system of independent similar sensors, to explore a set of common characteristics of a target. The sensors effectively only confirm each other’s reports, and do not provide additional information about a given target. The fusion objective of this model is to find a result which is most consistent with all the sensor reports. Given a set of objects/targets to be classified, \( O = \{o_1, o_2, ..., o_I\} \), a sensor report from the \( l^{th} \) sensor is defined as, \( D_l = \{P_l(o_i)\}_{i=1,...,I} \), \( \sum_{i=1}^{I} P_l(o_i) = 1 \). The goal here, as with any fusion algorithm, is to determine the fused report, \( D_f = \{P_f(o_i)\} \), that best fits the sensor reports. A cost function that measures the discrepancy between the fusion result and each sensor report is used, and a weighted sum of these cost functions is minimized,

\[
D_f = \arg \min_d \sum_{l=1}^{L} w_l \cdot \text{dist}(d, D_l),
\]

where, \( w_l \) is the contribution of the \( l^{th} \) sensor report towards the fused report, and \( \text{dist}(A, B) \) is a distance function measuring the discrepancy between the distributions \( A \) and \( B \). In [48], a Kulback-Liebler distance is used as the distance measure.

2.2.3.2 Dissimilar Sensor Fusion

In the Dissimilar Sensor Fusion model, dissimilar and independent sensors explore different characteristics of a target. Reports from these sensors can reinforce each other to generate increased resolution on target identity. The fusion objective of this model is to find a consensus which best represents an enhanced summary from the sensor reports. The sensor reports and fusion result are defined similarly to those in Similar Sensor Fusion but the cost functional is formulated differently in order to take into account the fact that each sensor report provides new information about the target, as it explores different characteristics of the target. The corresponding optimization problem is formulated as,

\[
P_f = \arg \min_p \sum_{l=1}^{L} w_l \sum_{i=1}^{I} \frac{1}{P_l(o_i)} p(o_i) - \sum_{i=1}^{I} \ln(p(o_i))
\]

subject to: \( \sum_{i=1}^{I} p(o_i) = 1, \ p(o_i) \geq 0 \). \hspace{1cm} (2.5)

As pointed out earlier, these two models should be viewed as "extreme cases" of decision level identity fusion. There are many practical cases in which the sensors are neither completely similar nor completely dissimilar.

A problem with such techniques arises when sensors get damaged during implementation of the system. These models assume that all sensors are operational, and when a sensor is damaged, it is ignored by the model to avoid making erroneous decisions. Doing so however, also amounts
to ignoring the underlying correlation between the sensors which may be exploited by using the available training data of the corresponding sensor.

2.2.4 Mutual Information

Consider two random variables, $X$ and $Y$, with a joint probability mass $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. The Mutual Information, $I(X; Y)$, is the relative entropy between the joint distribution, $p(x, y)$, and the product distribution, $p(x)p(y)$ [13]. The formula for Mutual Information is then given as:

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}.$$  \hspace{1cm} (2.6)

Further, the relationship between mutual information and joint entropy of $X$ and $Y$ is given as [13],

$$H(X, Y) = H(X) + H(Y) - I(X; Y)$$  \hspace{1cm} (2.7)

2.2.5 Support Vector Machines

Support Vector Machines (SVMs) have been widely used for separating data into different classes [79]. For a binary classification problem (i.e. assign a data sample to class ‘+1’ or ‘-1’), SVM computes a score for the test sample, $x_t$, $S(x_t) = w^T x_t + b$, where $w$ is the weight vector and $b$ is the bias term. Based on this score, SVM assigns a label to $x_t$,

$$y_t = \begin{cases} +1, & \text{if } S(x_t) > 0 \\ -1, & \text{if } S(x_t) < 0 \end{cases}$$  \hspace{1cm} (2.8)

The weights and the bias term are learned by optimizing the following cost functional,

$$\min_{w, \xi} \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n$$

subject to:

$$y_n(w^T x_n + b) > 1 - \xi_n,$$

$$\xi_n \geq 0,$$  \hspace{1cm} (2.9)

where, $\xi$ is the vector of slack variables, $y_n$ is the true label for the $n^{th}$ data sample $x_n$, and $C$ is a constant that controls the relative influence of the two competing terms. SVM is known to select the most generalized classifying hyperplane from all the possible options. For example, in Figure 2.2, SVM would select the dotted line over the solid line as it is more generalized, although both lines successfully separate the data.
A formulation for multi-class SVMs, using an all versus one approach was provided by Crammer-Singer in [14],

$$
\min_{w, \xi} \frac{1}{2} \sum_{j=1}^{J} w_j^T w_j + C \sum_{n=1}^{N} \xi_n
$$

subject to:

$$w_{y_n}^T (x_n) - w_t^T (x_n) \geq 1 - \xi_n, \quad t \in \{1, \ldots, J\} \setminus y_n,$$

$$\xi_n \geq 0,$$

where, \(\{w_j\}_{j=1}^{J}\) are the weight vectors that compute a score for each class, \(j \in \{1, \ldots, J\}\), given the \(n^{th}\) data sample \(x_n\), \(y_n \in \{1, \ldots, J\}\) is the corresponding true label for \(x_n\), \(\xi\) is the vector of slack variables, and \(N\) is the total number of data samples available for training. Equation (2.10), focuses on classification without the bias terms, \(\{b_j\}_{j=1}^{J}\). A bias term can be easily modeled by appending an additional constant feature to each \(x_n\).

### 2.3 Problem Formulation

As noted earlier, assume throughout a set of targets/objects, \(O = \{o_1, o_2, \ldots, o_I\}\), whose detection and/or classification are of interest. Let the \(k^{th}\) feature observed by the \(l^{th}\) sensor be \(F_k^l\). Then, a set of mutually exclusive events, \(\Omega_k^l = \{\omega_{k,j}^l\}_{j=1}^{J_i}, \) may be defined for the feature \(F_k^l\). Here, \(\omega_{k,j}^l\) is the \(j^{th}\) event for \(F_k^l\) and is described as, \(\omega_{k,j}^l : F_k^l \in [u_j, v_j], u_j \in \mathbb{R}^+, v_j \in \mathbb{R}^+, \) and \(v_j > u_j\). For example an event may be: The target in view is traveling at velocity < 5 m/s. The probability report for the \(k^{th}\) feature from the \(l^{th}\) sensor is then defined as

$$D_k^l = \{\Omega_k^l, \sigma_B(\Omega_k^l), P_k^l\}. \quad (2.11)$$

Where, \(\sigma_B(\Omega_k^l)\) is the Borel sigma algebra of \(\Omega_k^l\), and can be thought of as the set of all possible events that can be described over the feature. \(P_k^l\) is the set of probabilities over the events in \(\sigma_B(\Omega_k^l)\).
Figure 2.3 Block diagram of the Event Driven Approach for fusion

Let the $n^{th}$ observation made by the $l^{th}$ sensor be denoted by $x_n^l = \{x_{nq}^l\}_{q=1,...,Q}$, where $q$ corresponds to the signal value at time $q$. Furthermore, let $C_k^l(x_n^l) = w_k^l x_n^l$ be a scoring function that gives a detection score, where, $w_k^l$ is the weight vector for a classifier trained to detect the event $a_{kj}^l : F_k^l \in [u_j, v_j]$. The bias term for the classifier can be modeled by appending a constant feature to each $x_n^l$. Then, the probability of occurrence of the corresponding event is determined as

$$P_k(a_{kj}^l) = \frac{\exp(C_k^l(x_n^l))}{\sum_m \exp(C_k^m(x_n^l))},$$  

(2.12)

Since, we can only define objects by a set of characteristic features, it follows that a combination of certain events occurring over different features will be used working in the product space,

$$\Omega = \prod_{k,l} \Omega_k^l = \Omega_1^1 \times \Omega_2^1 \times ... \times \Omega_{K_1}^1 \times \Omega_1^2 \times \Omega_2^2 \times ... \times \Omega_{K_2}^2 \times \Omega_1^L \times \Omega_2^L \times ... \times \Omega_{K_L}^L,$$

(2.13)

where, $K_i$ is the total number of features observed by the $l^{th}$ sensor, and $l = 1, ..., L$. Further, an object will be defined as some combination of events in this product space, $a_i \in \sigma_B(\Omega)$. Given the object definitions and the probability distributions over various features, our goal is to then find the fused probability report over the objects, $D_f = \{O, P_f\}$.

### 2.4 Proposed Method

The sensor reports form a set $\{D_k^l\}_{k=1,...,K_L}^{l=1,...,L}$ which potentially are different sensors providing different features making up events which define targets. Specifically, the definitions of objects are the result of algebraic operations on the event space $\sigma_B(\Omega)$, a Sigma-algebra on the product space $\Omega$, with
associated probability measures as noted in Section 2.3. Thus, we must evaluate the probability distribution on $\sigma_B(\Omega)$.

2.4.1 Determining object Probabilities

Consider the events, $\gamma_1^l \in \sigma_B(\Omega_k^l)$, and the corresponding product space, $\Omega$. Then, for any combination, $\text{Comb}(\gamma_k^l) \in \sigma_B(\Omega)$, the object probability may be determined as, $P_f(o) = g(\text{Comb}(\gamma_k^l))$, where $g$ is a function that uses rules of probability to determine the fused object probability. Considering a 2-D setting, an object may be defined as a combination of events $\gamma_1 \in \sigma_B(\Omega_1)$, and $\gamma_2 \in \sigma_B(\Omega_2)$. The combination defined in the product space, $\Omega = \Omega_1 \times \Omega_2$, may be of the form $o : \{\gamma_1 \land \gamma_2\}$ or $o : \{\gamma_1 \lor \gamma_2\}$. Given the joint probability $P_\Omega$, rules of probability can be used to determine the fused object probability as follows:

- $o : \{\gamma_1 \land \gamma_2\}: P_f(o) = P_\Omega(\gamma_1, \gamma_2)$
- $o : \{\gamma_1 \lor \gamma_2\}: P_f(o) = P_\Omega(\gamma_1) + P_\Omega(\gamma_2) - P_\Omega(\gamma_1, \gamma_2)$

Where, $P_\Omega(\gamma_1)$ and $P_\Omega(\gamma_2)$ are the marginal probabilities for detection of the events $\gamma_1$ and $\gamma_2$ as seen by sensors 1 and 2. This can be easily extended to any number of features and combinations of more than two events.

2.4.2 Determining the Joint Probability

When determining the joint probability in the product space, $\Omega$, it is important to account for the extent of dependence between the features: Completely independent features yield minimal mutual information, and the joint distribution with the minimum mutual information should be selected; a high dependence between features, on the other hand, yields maximal mutual information, and the joint distribution with maximal mutual information should be selected. These are clearly the extreme cases of dependence, and do not address the partial dependence case. To account for partially dependent features, a good approximation to the joint probability would be a convex combination of the joint probabilities maximizing and minimizing the mutual information. For ease of writing, we use $\gamma_{1}^{l_1}, \ldots, \gamma_{k}^{l_k}, \ldots, \gamma_{K}^{l_K}$ to represent $\gamma_{1}^{1}, \gamma_{2}^{1}, \ldots, \gamma_{K}^{1}, \gamma_{1}^{2}, \gamma_{2}^{2}, \ldots, \gamma_{K}^{2}, \ldots, \gamma_{1}^{l_{1}}, \gamma_{2}^{l_{2}}, \ldots, \gamma_{K}^{l_{K}}$. The joint probability of events $\gamma_{k}^{l_{k}} \in \sigma_B(\Omega_k^l)$, can then be determined as,

$$P_\Omega(\gamma_{1}^{l_1}, \ldots, \gamma_{k}^{l_k}, \ldots, \gamma_{K}^{l_K}) = \rho \cdot P_{\Omega_{\text{max}}}(\gamma_{1}^{l_1}, \ldots, \gamma_{k}^{l_k}, \ldots, \gamma_{K}^{l_K}) + (1 - \rho) \cdot P_{\Omega_{\text{min}}}(\gamma_{1}^{l_1}, \ldots, \gamma_{k}^{l_k}, \ldots, \gamma_{K}^{l_K}),$$

(2.14)

where, $\rho \in [0, 1]$ is a pseudo-measure of extent of correlation between the features. $\rho \approx 1$ when features are highly correlated, and $\rho = 0$ when features are independent of each other. $\rho$ can be estimated from the training data by either computing the correlation between the features by using a measure like Pearson's correlation/distance correlation [78] or by optimizing $\rho$ over the training data.

It can be readily seen from Equation (2.6) that mutual information between two random variables is minimized when the joint probability distribution is selected as the product of the marginals, to
yield,
\[ R_{\text{MINMI}}(\gamma_1^l, ..., \gamma_k^l, ..., \gamma_L^K) = \prod_{k,l} P_k^l(\gamma_k^l). \tag{2.15} \]

Maximizing mutual information on the other hand, when given the marginal probabilities requires additional work. Given some random variables \( X \) and \( Y \), and conditioning on the marginal probability distributions of \( X \) and \( Y \) yields constant \( H(X) \) and \( H(Y) \). As may be seen from Equation (2.7), the maximization of Mutual Information between two random variables then becomes equivalent to minimizing their Joint Entropy, which is known to be a concave function.

\[
R_{\text{MMaxMI}} = \min_{P_{\Omega}} \sum_{b_1^1 \in \Omega_1^1, ..., b_l^k \in \Omega_l^k, ..., b_L^K \in \Omega_L^K} -P_{\Omega}(b_1^1, ..., b_l^k, ..., b_L^K) \log P_{\Omega}(b_1^1, ..., b_l^k, ..., b_L^K)
\]
subject to:
\[
\forall l \in \{1, ..., L\}, \forall k \in \{1, ..., K_l\}, 
\sum_{\{b_1^1, ..., b_l^k, ..., b_L^K \} \backslash \{b_l^k\}} P_{\Omega}(b_1^1, ..., b_l^k, ..., b_L^K) = P_l^k(b_l^k), \quad P_{\Omega}(b_1^1, ..., b_l^k, ..., b_L^K) \geq 0 \tag{2.16} \]

A greedy approach for minimizing joint entropy given the marginal probabilities can be constructed [40] and is exploited to find the joint distribution with maximal mutual information. The main idea here is to keep large probability masses intact and not break them down into smaller chunks. The contribution of a probability mass toward the joint entropy only increases if it is divided into smaller chunks. That is, for \( p = a + b \), \(-p \log(p) \leq -a \log(a) - b \log(b)\), when \( 0 < p < 1 \) and \( a, b > 0 \). So, keeping the large probability masses from given marginal probabilities intact ensures that their contribution towards the joint entropy is minimized. As empirically demonstrated in [40], the minimal joint entropies are obtained to within 1 bit of the optimal values. The algorithm for finding the joint distribution with maximal mutual information is described in Algorithm 1. Figure 2.3 summarizes the steps of the proposed fusion approach in a block diagram.

### 2.4.3 Robustness: Addressing damaged sensors

In practice, sensor measurements may often be noisy, missing, or unusable in unconstrained surveillance settings, or just of limited capacity. In this scenario, it is common to ignore such sensors, with a potentially negative impact on optimal performance (i.e. all sensors are available and functional). We consider exploiting prior knowledge about the relationship between the various modalities, so that our system can safeguard a high detection accuracy. This prior knowledge resides in the training data, which is assumed to be available for all the modalities. Such a problem has previously been studied in [64], where Conditional Generative Adversarial Networks (CGAN) were used to replicate features of damaged sensors. This requires that the features for optimal classification be known beforehand, so that the CGAN network can learn to replicate them, whereas, in our case we are searching a hidden space that is shared between sensor modalities, even with the absence of the optimal features. To that end, we propose to find linear operators that transform
Algorithm 1: Joint Entropy Minimization (Mutual Information Maximization)

**Input:** Marginal distributions $p_1, p_2$

1. While $p_1 > 0$ or $p_2 > 0$
2. Find $m_1 = \max_i p_1(i)$, $m_2 = \max_i p_2(i)$
3. Set $a = \arg\max_i p_1(i)$ and $b = \arg\max_i p_2(i)$
4. Assign $M(a, b) = \min\{m_1, m_2\}$
5. Update $p_1(a) \leftarrow p_1(a) - M(a, b)$, and $p_2(b) \leftarrow p_2(b) - M(a, b)$
6. end

**Output:** Joint probability distribution with minimum joint entropy (Maximum Mutual Information), $M$.

---

Figure 2.4 Using a Global Hidden Space for Event Driven Fusion

each sensor modality into a common hidden space\(^1\), so that it represents the shared information between the sensors. It is also important that we formulate the cost functional so that the determined hidden space is discriminative with respect to detection of event occurrences. The first approach finds a global hidden space representing all features ($\{F_k\}$), while the second approach finds an independent hidden space for each feature.

2.4.3.1 Global Hidden Space

We assume here the existence of a global sub-space that can characterize all the features of interest, i.e. the same hidden space may be used to detect all feature events, as shown in Figure 2.4. We seek to find the linear operators, $Z^1, Z^2, \ldots, Z^L$, such that,

$$Z^l_{d \times d_l}X^l_{d_l \times N} = H_d \times N, \forall l \in \{1, \ldots, L\}. \quad (2.17)$$

Where, $X^l = \{x^{l}_{n}\}_{n=1, \ldots, N}$, and $N$ is the number of training samples. The desired dimension of the hidden space is denoted by $d$, while the dimension of the observed signal from the $l^{th}$ sensor, by $d_l$.

\(^1\)In [83], such a space was referred to as an ‘information subspace’
(d < d_l, \forall l).

A key observation to our goal of determining a common subspace for different modalities, is that if a set of linear operators commute, they share common eigenvectors [22, 25]. If these operators are furthermore individually diagonalizable, they will share all their eigenvectors, leading to a common eigenbasis/subspace.

Definition 1. Linear operators \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times n} \) are said to commute if,

\[
[A, B] = AB - BA = 0. \tag{2.18}
\]

Theorem 1. If \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times n} \) are commuting linear operators, they share common eigenvectors.

Proof. Consider an eigenbasis \( \psi = \{v_i\} \), of \( A \) with \( \lambda_i \) the eigenvalue associated to \( v_i \). Then for any \( v_i \),

\[
AB v_i = BA v_i = \lambda_i B v_i, \tag{2.19}
\]

i.e., if \( B v_i \neq 0 \), \( B v_i \) is an eigenvector of \( A \), associated to the same eigenvalue as \( v_i, \lambda_i \).

Theorem 2. If \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times n} \) are commuting operators that are also individually diagonalizable, they share a common eigenbasis.

Proof. If \( A \) and \( B \) are individually diagonalizable, they have \( n \)-distinct eigenvalues, i.e. \( A \) can be diagonalized as, \( A = PDAP^{-1} \), where, \( D_A \) is an \( n \times n \) diagonal matrix with eigenvalues of \( A \) on the diagonal, and \( P \) is an \( n \times n \) matrix with eigenvectors of \( A \) as columns. Since, both \( A \) and \( B \) share common eigenvectors (as seen in Theorem 1), \( B \) can also be diagonalized as \( B = PD_BP^{-1} \). Hence, \( A \) and \( B \) share a common eigenbasis.

As a result, if the operators \( Z^1, ..., Z^L \) commute and are diagonalizable, they will share a common eigenbasis. Furthermore, since the transformation \( Z^l X^l \) lies in the range space of the linear operator, \( Z^l, Z^l X^l \) lie in a common subspace, \( \forall l \in \{1, ..., L\} \), due to the shared basis. This hence yields a common feature representation for the different modalities.

To ensure pairwise commutation between the linear operators, \( \{Z^l\}_{l \in \{1,...,L\}} \), we must make all the operator matrices square, which can in turn be accomplished by using sampled random matrices, \( \{U^l\}_{l \in \{1,...,L\}} \). Since, \( U^l \) is a random projection which will stay constant during the learning process, the information about the transformation, \( (Z^l U^l)X^l \), still lies in the range of \( Z^l \). This results in Equation (2.17) re-expressed as,

\[
Z^l \times d_{d \times d} U^l \times d_{d \times d} X^l \times d_{d \times N} = H_{d \times N}, \forall l \in \{1, ..., L\}. \tag{2.20}
\]

To optimize the event detection on the basis of this hidden feature space we proceed to train a classifier for event detection, and to learn the operators \( Z^l \). Let, \( W^l_k = \{w^l_{kj}\}_{j=1,...,J_k} \), be the weight matrix for classification of events \( \Omega^l_k = \{\omega^l_{kj}\}_{j=1,...,J_k} \) defined over the \( k^{th} \) feature from
the $l^{th}$ sensor. We build on the SVM formulation [14] to uncover the optimal hidden space with sufficient information to successfully detect all events over all features. To that end, the operators, $Z^1, ..., Z^L$ are sought by an optimization of an energy cost functional which includes a penalty term to encourage pairwise commutation of their application on the various sensor data. The objective then becomes,

$$\min_{W_l^k, \xi_k^l, Z^l} J(W_l^k, \xi_k^l, Z^l) = \frac{1}{2} \sum_{l=1}^{L} \sum_{i=1}^{K_l} \|W_l^k\|^2 + C_1 \sum_{l=1}^{L} \sum_{o=1}^{N} \xi_k^l$$

$$+ \frac{1}{2} \sum_{l, m=1}^{L} (C_2 \|Z^l, Z^m\|^2 + C_3 \sum_{o=1}^{N} (Z^l U^l x_n^l - Z^m U^m x_n^m)^2)$$

subject to:

$$w_l^k^T (Z^l U^l x_n^l) - w_l^k^T (Z^l U^l x_n^l) \geq 1 - \xi_k^l, \quad t \in \{1, ..., J_k\} \setminus \{y_n\},$$

$$\xi_k^l \geq 0, \quad \forall l \in \{1, ..., L\}, \forall k \in \{1, ..., K_l\}.$$ (2.21)

The above cost functional does not guarantee the individual diagonalizability of the operators, $Z^1, ..., Z^L$, we, however, empirically observe that resulting operators on convergence are diagonalizable in most cases. The quadratic error constraint in Equation (2.21) encourages corresponding projected samples from different modalities to be close in the common sub-space.

### 2.4.3.2 Independent Hidden Spaces

To reduce the computational complexity due to the number of constraints, we seek to find independent hidden spaces for the features of interest, as illustrated in Figure 2.5. We thus seek

![Figure 2.5 Using Independent Hidden Spaces for Event Driven Fusion](image-url)
linear operators $Z^{11}_{k}, \ldots, Z^{ir}_{k}, \ldots, Z^{LL}_{k}$, such that,

$$Z^{ir}_{k, d \times d} X^{r}_{d \times n} = H^{l}_{k, d \times n}, \quad \forall l, r \in \{1, \ldots, L\}, \text{ and } \forall k \in \{1, \ldots, K_l\}. \quad (2.22)$$

where, $Z^{ir}_{k}$ transforms observed data, $X^{r}$, from the $r^{th}$ sensor to the hidden space, $H^{l}_{k}$. This means, we now have a individual hidden space describing each feature of interest ($F^{l}_{k}$), making the number of cost functions that are independently optimized equivalent to the total number of features of interest $\sum_{k} K_l$. Furthermore, as in Section 2.4.3.1, we again introduce the randomly sampled matrices, $\{U^{lr}_{k}\}_{k \in \{1, \ldots, K_l\}}$, in order to help enforce pairwise commutation between the transformations,

$$Z^{ir}_{k, d \times d} U^{lr}_{k, d \times d} X^{r}_{d \times n} = H^{l}_{k, d \times n}, \quad \forall l, r \in \{1, \ldots, L\}, \text{ and } \forall k \in \{1, \ldots, K_l\}. \quad (2.23)$$

The solution in Equation (2.23), i.e. the hidden space for the $k^{th}$ feature from the $l^{th}$ sensor $H^{l}_{k} = Z^{ir}_{k} U^{lr}_{k} X^{r}$, is obtained by minimizing the following objective,

$$\min_{W^{l}_{k}, \xi^{lr}_{k}} \ J(W^{l}_{k}, \xi^{lr}_{k}, Z^{lr}_{k}) = \frac{1}{2} \|W^{l}_{k}\|^2 + C_{1} \sum_{r=1}^{L} \sum_{n=1}^{N} \xi^{lr}_{k n}$$

$$+ \frac{1}{2} \sum_{r, s=1}^{L} \sum_{r \neq s} (C_{2} \|Z^{lr}_{k}, Z^{ls}_{k}\|^2 + C_{3} \sum_{n=1}^{N} (Z^{lr}_{k} U^{lr}_{k} x^{r}_{n} - Z^{ls}_{k} U^{ls}_{k} x^{s}_{n})^2)$$

subject to:

$$\forall r \in \{1, \ldots, L\},$$

$$w^{l}_{k y_{n}} (Z^{lr}_{k} U^{lr}_{k} x^{r}_{n}) - w^{l}_{k 1} (Z^{lr}_{k} U^{lr}_{k} x^{r}_{n}) \geq 1 - \xi^{lr}_{k n}, \quad t \in \{1, \ldots, J_{K_l}\} \setminus \{y_{n}\},$$

$$\xi^{lr}_{k n} \geq 0. \quad (2.24)$$

The above conditions are satisfied by setting,

$$\xi^{lr}_{k n} = \max_{t \neq y_{n}} 0, 1 - w^{l}_{k y_{n}} (Z^{lr}_{k} U^{lr}_{k} x^{r}_{n}) + w^{l}_{k 1} (Z^{lr}_{k} U^{lr}_{k} x^{r}_{n}), \quad (2.25)$$

whose substitution in Equation (2.24) leads gradient descent to a solution by the way of the algorithm, namely the optimal $W^{l}_{k}$ and $Z^{lr}_{k}, \forall r \in \{1, \ldots, L\},$

$$\forall m \in \{1, \ldots, J_{K_l}\},$$

$$w^{l}_{k m} = w^{l}_{k m} - \mu \frac{d J(W^{l}_{k}, Z^{lr}_{k})}{d w^{l}_{k m}}, \quad (2.26)$$

$$Z^{lr}_{k} = Z^{lr}_{k} - \mu \frac{d J(W^{l}_{k}, Z^{lr}_{k})}{d Z^{lr}_{k}}, \quad (2.27)$$

where, $i$ denotes the iteration number, and $\mu$ is the learning rate. See Appendix-A for the derivations.
If the $m^{th}$ sensor is damaged, the hidden spaces for this sensor are recovered from the available set of sensors, $\Gamma = \{1, ..., L\} \setminus m$,

$$H^m_k = \sum_{r \in \Gamma} Z^{mr} U_{k}^{mr} X^r, \forall k \in \{1, ..., K_m\},$$

(2.28)

where, $|\Gamma|$ is the cardinality of $\Gamma$. Figure 2.6 illustrates a scenario with two sensors, $l \in \{1, 2\}$, and demonstrates the recovery of independent hidden spaces for features defined for a damaged sensor ($l = 1$), using observed data of the available sensor ($l = 2$).

**Example:** To visualize the result of this algorithm at convergence, consider a toy example with two modalities, $X^1 \in \mathbb{R}^{4 \times N}$ and $X^2 \in \mathbb{R}^{3 \times N}$, for binary classification. The random projections $U^1 \in \mathbb{R}^{2 \times 4}$ and $U^2 \in \mathbb{R}^{2 \times 3}$ are first used to project the data onto a 2-dimensional space, as seen in Figure 2.7-(a). Following this, we use the proposed approach to find hidden spaces, and project each modality onto a common hidden space, $H^{2 \times N} \approx Z^{1 \times 4} U^1_{2 \times 4} X^1_{4 \times N} \approx Z^{2 \times 3} U^2_{2 \times 3} X^2_{3 \times N}$. Figures 2.7-(b),(c) show the data transformation into the hidden space for two cases: 1) When no penalty was enforced for non-commuting operators (Figure 2.7-(b)), and 2) When commutation between $Z^1$ and $Z^2$ was enforced (Figure 2.7-(c)). As may be seen from the determined hidden space in both cases, commutation is able to push the determined hidden subspace to be common for both modalities. Furthermore, it can be seen that the common classifier (denoted by the black solid line), which is learned jointly with the linear operators, is a compromise between the optimal classifiers for each modality (i.e. the SVM classifier learned for each modality individually in the transformed space).

### 2.5 Experiments and Results

To substantiate the above proposed approach in the various scenarios, we select two different datasets.

#### 2.5.1 Dataset 1: Radar and Telescopic Imaging Sensors

For the first dataset, we select two sensors, namely a Radar sensor and a telescopic optical sensor, the latter having been measured and collected by Jen-Hung Wang and the TAOS team. Due to technical
difficulty in the field experiment, radar measurements were simulated according to the physical data of the space debris and matched with the optical data. Both sensors are ideally synchronized when observing a given target, which in our case, is a space object as just noted. The radar simulations (obtained through MATLAB Simulink, whose block diagram can be seen in Figure 2.9), together with telescopic image data are used in our first experiment. Each generated radar signal over one second is correlated with two telescopic images. Samples of objects with different velocities, cross-sections, ranges, and aspect-ratios are generated. The radar signals are used to make decisions over velocity, range, and the cross-section, while the telescopic images are used to make decisions over the aspect-ratio, and displacement over time of an object in view.

2.5.1.1 Experiment Design

To proceed with the algorithmic evaluation, we must first generate distributions of features needed for target specification. Figure 2.8 shows a high level block diagram for implementing an
Event Driven Fusion for dataset 1. Let the received radar signal be, \( x(n) \), and its corresponding Fourier transform, \( X(k) = \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N} \) for each object, with associated labels distinguished by the object velocity, range, and cross-section values of that object, \([v, r, c s]\). Using the training data, and the corresponding labels, SVM classifiers are trained over the events of interest defined over \([v, r, c s]\), and used to determine the classification probabilities for the event of interest, \( P_{l_k}(\omega_{l_k}) \), as described in Equation (2.12). For the \( k^{th} \) feature from \( l^{th} \) sensor we train the SVM classifier using the Crammer-Singer formulation for multi-class SVM \([14]\),

\[
\min_{W^l_k, \xi^l_k} \frac{1}{2} ||W^l_k||^2 + C \sum_{n=1}^{N} \xi^l_{k_n},
\]

subject to:

\[
w^l_k (x^l_{n}) - w^l_k (x^l_{n}) \geq 1 - \xi^l_{k_n}, \quad t \in \{1, ..., J_{k_j}\},
\]

\[
\xi^l_{k_n} \geq 0,
\]

where \( y_n \in \{1, ..., J_{k_j}\} \) is the true label of the \( n^{th} \) data sample, and \( N \) is the total number of data samples available for training.

We have two telescopic images associated with 1-sec of radar return for the same object. The object of interest in the telescopic image is first detected using target detection as discussed in [9]. Upon its detection, the probability distribution over the object’s aspect-ratio, and its displacement in the second image relative to its location in the first image is determined using the image flow technique discussed in [74].

### 2.5.1.1.1 Object Detection:

The object of interest in the telescopic imagery is initially detected by using target detection as discussed in [9]. Any pixel in the telescopic image domain is said to follow the probabilistic model, \( I(o) = i_o + n, n \sim G(\mu_n, \sigma^2_n) \), where \( o \) is the object that the pixel belongs to. Points in the image domain are said to belong to one of three sets in regards to the statistics of their neighborhoods[9]:
• **Background Set:** For points in the image domain, the set of all background points is given as,

\[
W = \{ p \mid \forall q \in N(p), f(p) = f(q) = G(\mu_n, \sigma_n^2) \},
\]

where, \( f \) is the probability density function, and \( N(p) \) is the neighborhood of point \( p \).

• **Interior Set:** The set of all interior points (in regards to objects) is defined as,

\[
S = \{ p \mid \forall q \in N(p), f(p) = f(q) \neq G(\mu_n, \sigma_n^2) \},
\]

where, \( f \) is the probability density function, and \( N(p) \) is the neighborhood of point \( p \).

• **Boundary Set:** The set of all boundary points is defined as,

\[
B = \{ p \in B \mid \exists q \in N(p), f(p) \neq f(q), f(p) \neq G(\mu_n, \sigma_n^2) \},
\]

where, \( f \) is the probability density function, and \( N(p) \) is the neighborhood of point \( p \).

Based on the above definitions, a hypothesis test is performed in order to find the interior set of a given image. Let, \( f_n = G(\mu_n, \sigma_n^2) \) be the background distribution, and \( f_p = G(\mu_p, \sigma_p^2) \) be the distribution of object pixels, then,

\[
H_0 : p \sim f_p, q \sim f_p, \forall q \in N(p), \\
H_1 : p \sim f_n \text{ or } q \sim f_q \neq f_p, \forall q \in N(p).
\]

After recognizing the interior points, the next crucial step is to cluster the interior points with respect to objects, for which a proper distance measure is important. The following metric, which reflects both the physical properties, such as, the apparent magnitude and spatial relations is defined in [9],

\[
\forall p_a, p_b \in S, \\
d(p_a, p_b) = d_{\text{Euclidean}}(p_a, p_b) + d_{\text{Intensity}}(p_a, p_b) \\
= \sqrt{(p_a^x - p_b^x)^2 + (p_a^y - p_b^y)^2 + \beta |I(p_a) - I(p_b)|},
\]

where, \( \beta \) balances the contribution of intensity distance and euclidean distance. The corresponding distance matrix is then used as an input to a clustering algorithm. Using single linkage clustering, two sets of pixels, \( A \) and \( B \) are said to belong to the same cluster if,

\[
\min \{ d(p_a, p_b) : p_a \in A, p_b \in B \} < \gamma
\]

The cut-off distance \( \gamma \) can be estimated from the training data, and depends on the expected size of objects. Such a clustering algorithm also identifies night sky stars as objects, which may not necessarily be of interest. In order to identify objects of interest, the ratio of width and height of an object in the image domain is used as a criterion. The spread of pixels representing an object (or the
aspect-ratio of the object) is defined as $R(o) = \frac{W(o)}{H(o)}$, then, $o$ is an object of interest if,

$$|R(o) - c| > \text{median}\{R\}, \quad (2.36)$$

where, $c$ is the parameter measuring the distance between an object of interest and stars, and $\text{median}\{R\}$ is the median of width-height ration for all objects. Figure 2.10 shows the clustering results, with a subsequent detection of object of interest for a sample telescopic image. Although not visible to the naked eye in Figure 2.10-(a), there are stars in the background, which are detected by the clustering algorithm, and can be seen in Figure 2.10-(b),(c).

### 2.5.1.1.2 Displacement Estimation

For two successive images, $I_1$ and $I_2$, captured by the telescopic sensor, a point $P(x, y)$ in $I_1$ moves to $P(x + u, y + v)$ in $I_2$, for which the displacement vector, $(u, v)$ is of interest. A correlation window of size, $\{\max[W(o), H(o)] \times \max[W(o), H(o)]\}$ is defined about the centroid of the target of interest in $I_1$. An error distribution is subsequently computed over a search window ($I_2$) by using sum of squared distances,

$$E(\mathcal{K}, \mathcal{L}) = \sum_{i,j=-N}^{N} [I_1(x + i, y + j) - I_2(\mathcal{K} + i, \mathcal{L} + j)]^2, \quad (2.37)$$

where, $N = \frac{\max[W(o), H(o)]}{2}$, $0 < \mathcal{K} < W(I_2)$, and $0 < \mathcal{L} < H(I_2)$. This error distribution can then be converted into a probability distribution as,

$$P_d(\mathcal{K}, \mathcal{L}) = e^{-E(\mathcal{K}, \mathcal{L})/z}, \quad (2.38)$$

where, $z$ is a scaling factor. Furthermore, given a position $(x, y)$ of the object in image $I_1$, we get the probability distribution over $(u, v)$, $P_d(u, v) = e^{-E(x+u, y+v)/z}$. The probability of an event over
displacement of the object can then be determined as,

\[ p(a < d < b) = \sum_{u,v} \mathcal{I}(a < d < b), P_d(u, v), \]  

(2.39)

where, \( d = \sqrt{u^2 + v^2} \), and \( \mathcal{I} \) is the indicator function. Figure 2.11 shows the estimation of this probability distribution for a sequence of two images.

![Image](image.png)

(a) Image 1: Location of object of interest: \((x, y) = (429, 932)\)

(b) max probability located at: \((x + u, y + v) = (502, 932)\)

**Figure 2.11** (a): Sequence of two consecutive images from telescopic sensor, (b): Probability distribution over location of object of interest in \( I_2 \)

### 2.5.1.2 Object and Event Definitions

For training and testing purposes, we define various events over the feature-sets from both sensors. For the radar, as noted before, we use \([v, r, c_s] \) and the events are defined as,

\[
\begin{align*}
  a^v_1 &: 0 \leq v \leq 10 \text{ mi/s}, \quad a^v_2 &: 15 \text{ mi/s} \leq v \leq 35 \text{ mi/s}, \\
  a^r_1 &: 0 < r \leq 300 \text{ mi}, \quad a^r_2 &: 300 \text{ mi} < r, \\
  a^{c_s}_1 &: 0 < c_s \leq 20 \text{ m}^2, \quad a^{c_s}_2 &: 15 \text{ m}^2 \leq c_s \leq 50 \text{ m}^2.
\end{align*}
\]

(2.40)

From the telescopic imaging sensor, the features, displacement and aspect ratio, \([d, AR] \) define the following events,

\[
\begin{align*}
  a^d_1 &: 0 \leq d \leq 60 \text{ pixels}, \quad a^d_2 &: 90 \text{ pixels} \leq d \leq 210 \text{ pixels}, \\
  a^{AR}_1 &: 0 < AR \leq 1.5, \quad a^{AR}_2 &: 1.5 < AR.
\end{align*}
\]

(2.41)
Furthermore, the objects for classification are defined in terms of these events as,

\[
 o_1 : \{ a_1^r \land [(a_2^v \land a_2^d) \lor (a_2^{cs} \lor a_2^{rr})] \}, \tag{2.42}
\]

\[
 o_2 : \{ a_1^v \land a_1^d \land a_2^r \land a_1^{cs} \land a_1^{ar} \}. \tag{2.43}
\]

Given these events and object definitions, we determine the fused report, \( D_f = \{ P_f(o_1), P_f(o_2), P_f(o_1 \lor o_2) \} \) using our proposed approach. This can be considered a classification problem with 3 classes, Class 1: Object 1, Class 2: Object 2, and Class 3: Neither Object 1 nor Object 2.

### 2.5.2 Dataset 2: Acoustic and Seismic Sensors

![Seismic Sensor](image1.png) ![Acoustic Sensor](image2.png)

**Figure 2.12** Sample (a): seismic sensor observations and (b): acoustic sensor observations for human, vehicular, and no target cases

A second dataset we use in our experimental validation; is pre-collected data from a network of seismic sensors, and acoustic sensors deployed in a field, where people/vehicles were walking/driven around in specified patterns. Details about this sensor setup and experiments can be found in [54]. This dataset has been previously used for target detection in [47, 64], where, the authors focused on detection of human targets. Here, we use this dataset to classify between human targets, vehicular targets, and no targets. Some data samples from the sensors are shown in Figure 2.12

#### 2.5.2.1 Experiment Design

Figure 2.13 shows a high level block diagram for the implementation of Event Driven Fusion for the second dataset. Using the training data, SVM classifiers are trained over the corresponding events of interest, as discussed before for the first dataset in Section 2.5.1.1. The seismic sensor provides decisions over the features, target weight and target speed, \([w, s]\). True labels for target weights
are provided in the dataset, while those for target speeds are obtained from the GPS data of the
target. Similarly, the acoustic sensor provides decisions on the noise-level of the target, and the
target speed, \([n, s]\). The two decisions over the target speed are combined into a single report by
performing a weighted averaging of the decisions of the two sensors. Here, the weights are selected
on the basis of the individual accuracies of the SVMs trained to detect events on target speed.

### 2.5.2.2 Object and Event Definitions

For training and testing purposes, we define various events over the feature-sets from both the
sensors. For the seismic sensor, we use \([w, s]\), while for the acoustic sensor we use \([n, s]\).

\[
\begin{align*}
    a_1^w &: 96.08 \text{ pounds} \leq w \leq 230.61 \text{ pounds}, \\
    a_2^w &: 1311.61 \text{ pounds} \leq w, \\
    a_1^s &: 0.37 \text{ m/s} < s \leq 2.12 \text{ m/s}, \\
    a_2^s &: 1.7 \text{ m/s} \leq s, \\
    a_1^n &: n \leq -30 \text{ db}, \\
    a_2^n &: -10.6658 \text{ db} \leq n \leq 7.84 \text{ db}. \\
\end{align*}
\]  

(2.44)

The range of an event can be determined from the training data. The mean of the feature in question
over the samples of the same class is computed, and a range of twice the standard deviation is taken
on either side of the mean. Furthermore, the targets are defined as,

\[
\begin{align*}
    o_1 (\text{human target}) &: \{a_1^w \land (a_1^w \lor a_1^n)\}, \\
    o_2 (\text{vehicular target}) &: \{a_2^s \land a_2^w \land a_2^n\}. \\
\end{align*}
\]  

(2.45) \hspace{1cm} (2.46)

Given these events and object definitions, we wish to determine the fused report, \(D_f = [P_f(o_1), P_f(o_2)
, P_f(\overline{o_1} \lor \overline{o_2})]\), where, \(\overline{o_1} \lor \overline{o_2}\) represents the no target case.

### 2.5.3 Performance Analysis

Table 2.1, and 2.2 show the classification performance of different techniques (averaged over 10
runs of the technique) when implemented on dataset 1 and 2 respectively.

Classification accuracy is often not the best measure to quantify performance, particularly in
### Table 2.1 Performance Comparison for the First Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar</td>
<td>86.47%</td>
</tr>
<tr>
<td>Telescopic Imaging</td>
<td>81.31%</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>85.93%</td>
</tr>
<tr>
<td>Similar Sensor Fusion</td>
<td>86.07%</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion</td>
<td>88.61%</td>
</tr>
<tr>
<td>Dempster-Shafer Fusion</td>
<td>87.18%</td>
</tr>
<tr>
<td><strong>Event Driven Fusion</strong></td>
<td><strong>90.36%</strong></td>
</tr>
</tbody>
</table>

### Table 2.2 Performance Comparison for the Second Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seismic Sensor</td>
<td>85.41%</td>
</tr>
<tr>
<td>Acoustic Sensor</td>
<td>67.62%</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>81.63%</td>
</tr>
<tr>
<td>Similar Sensor Fusion</td>
<td>86.69%</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion</td>
<td>89.96%</td>
</tr>
<tr>
<td>Dempster-Shafer Fusion</td>
<td>87.93%</td>
</tr>
<tr>
<td><strong>Event Driven Fusion</strong></td>
<td><strong>92.04%</strong></td>
</tr>
</tbody>
</table>

### Figure 2.14 ROC Curves for detection (Dataset 1) of (a): Class 1:Object 1, (b): Class 2:Object 2, (c): Class 3:Neither Object 1 nor Object 2
cases where different classes have different numbers of samples, which is the case here. A better way to compare performance is to look at the Receiver Operating Characteristic (ROC) curves. Fig. 2.14 and 2.15 show the ROC curves for classification for each of datasets 1 and 2 respectively. It can be seen from the ROC curves (for dataset 1) in Fig. 2.14, that other techniques show limited performance in correct classification of objects from class 2 due to the low number of samples for class 2 in comparison to those in class 1 and class 3. This causes the classifier to bias toward selecting class 1 or class 3 in order to achieve high classification accuracy (even when the sample is from class 2). But, our technique trains over occurrence of events rather than the object itself, hence does not face this issue. Improvement in performance is also seen for Dataset 2 (Fig. 2.15). In particular, detection of human targets is significantly improved, by taking ‘or’ between noise level event and weight event, which reduces misclassification due to noise due to winds.

2.5.4 Robustness Evaluation

In order to evaluate the robustness of the proposed algorithm, we consider a damaged sensor scenario that was discussed in Section 2.4.3. For the first dataset, we consider the situation where 3 radar sensors (simulated with different Signal to Noise Ratios) are used along with a telescopic sensor, and a subset of the radar sensors are damaged at a given time (Figure 2.16-(a)). The Global Hidden Space, and Independent Hidden Spaces approaches are evaluated and compared with the common approach of ignoring the damaged sensors in Figure 2.16-(b). The ‘Similar Sensor Fusion + Dissimilar Sensor Fusion’ case in Figure 2.16-(b) refers to the fusion of the radar sensors using Similar Sensor Fusion, followed by fusion with telescopic sensor using Dissimilar Sensor Fusion. As the number of working sensors is reduced, the target detection performance is impacted. We note that the exploitation of Independent Hidden Spaces along with Event Driven Fusion allows for a more graceful degradation. In the second dataset case, we consider the following scenarios, 1) Seismic Sensor is damaged 2) Acoustic Sensor is damaged (Figure 2.17). The results for these two cases are shown in Table 2.3. A similar observation can be made here as in spite of one of the damaged sensors during testing, the prior information from the training phase allows us to learn a
Figure 2.16 (a): Using Independent Hidden Spaces to deal with a damaged radar sensor, (b): Comparison of proposed approach with existing techniques, when sensors get damaged.

transform that helps boost the performance of the working sensor, hence gracefully mitigating the impact.

2.6 Conclusion

In this chapter, we proposed a novel sensor fusion technique that looks at targets as combinations of probabilistic events defined over the feature set used to construct a sigma-field all the while considering the extent of dependence between different features. Experiments on various datasets showed that the proposed technique can outperform existing fusion techniques on the decision level. We also propose a technique to safeguard detection performance of the model when sensors are damaged during the implementation phase by leveraging the prior information available during training.
Figure 2.17: Using Independent Hidden Spaces to deal with damaged Seismic/Acoustic sensors

Table 2.3: Robustness Analysis for the second dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seismic Sensor</td>
<td>85.41%</td>
</tr>
<tr>
<td>Acoustic Sensor</td>
<td>67.62%</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>81.63%</td>
</tr>
<tr>
<td>Similar Sensor Fusion</td>
<td>86.69%</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion</td>
<td>89.96%</td>
</tr>
<tr>
<td>Dempster-Shafer Fusion</td>
<td>87.93%</td>
</tr>
<tr>
<td><strong>Event Driven Fusion</strong></td>
<td><strong>92.04%</strong></td>
</tr>
<tr>
<td>Event Driven Fusion with Global Hidden Space</td>
<td>68.33%</td>
</tr>
<tr>
<td>(Damaged Seismic Sensor)</td>
<td></td>
</tr>
<tr>
<td>Event Driven Fusion with Global Hidden Space</td>
<td>86.13%</td>
</tr>
<tr>
<td>(Damaged Acoustic Sensor)</td>
<td></td>
</tr>
<tr>
<td><strong>Event Driven Fusion with Independent Hidden</strong></td>
<td><strong>71.66%</strong></td>
</tr>
<tr>
<td><strong>Spaces (Damaged Seismic Sensor)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Event Driven Fusion with Independent Hidden</strong></td>
<td><strong>87.36%</strong></td>
</tr>
<tr>
<td><strong>Spaces (Damaged Acoustic Sensor)</strong></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 3

COMMUTING CONDITIONAL GANS FOR ROBUST MULTI-MODAL FUSION

3.1 Introduction

In this Chapter we will address some of the limitations of the approach discussed in Chapter 2. First, we will propose a data-driven approach so that hand crafting of features of interest is no longer required. A measure of sensor reliability, which we refer to as the Degree of Confidence (DoC), will be introduced in order to deal with partially damaged (noisy) sensors. This reliability measure is adaptive, and accounts for sensor condition during the implementation phase as well. Additionally, a hidden common sub-space between the sensor modalities is learned, and optimal features for classification are driven by the existence of such a common subspace. This is similar to the Global Hidden Space which was discussed in Chapter 2, but now the hidden space is structured such that it can accommodate both shared and private features of the sensor modalities. Furthermore, a non-linear approach to finding this hidden sub-space is adopted in contrast to the linear approach adopted in the previous chapter. The hidden space is also used to detect damaged/noisy sensors and adaptively update the DoC so that such sensors can be weighed accordingly without requiring a-priori knowledge about sensor damage.

3.2 Related Work

3.2.1 Generative Adversarial Networks

The Adversarial Network was first introduced by Goodfellow et al. [26] in 2014. In this framework, a generative model is pitted against an adversary: the discriminator. The generator aims to deceive the discriminator by synthesizing realistic samples from some underlying distribution. The discrim-
Figure 3.1 Conditional Generative Adversarial Networks

In this section, we explore and uncover some limitations of Conditional Generative Adversarial Networks, when tasked with generating figures with limited support and otherwise. We take a simple example where the input to the network is uniformly sampled from a unit circle in the 2-D space, and the target is a uniform distribution on the unit square in the 2-D space (Fig. 3.2-(b)). We use a 3-layered CNN to approximate the generator in this case.

It is seen that the generative network struggles with transformation of a sequence of data points sampled from a distribution with a topological dimensionality of 1 (unit circle) into a distribution with a topological dimensionality of 2, and vice versa. It can be seen in Fig. 3.2, that when transforming a unit circle to a unit square, the generated samples lie inside the target distribution, but fail to cover the entire space.

This is, however, solved by using a unit disk \((x^2 + y^2 \in [1-\epsilon, 1+\epsilon])\), which has the same dimensionality as the unit square, as seen in Fig. 3.4. This goes to show that Generative networks are limited in generating a higher dimensional structure, when given an input with a lower dimension. For a reliable generation, dimensionality of the input sequence must be greater than or equal to that of the targeted distribution. Later we see that this phenomenon is also observed in our implementations, as we require the use of a generated hidden space whose dimension is lower than that of the conditional information, in order to achieve optimal performance.
3.2.2 Teacher-Student Distillation Model

This model has been largely used for compressing large neural networks (i.e. teacher) into smaller ones (i.e. students), while achieving a similar performance. In [10], the output from the teacher was used as the target probability distribution for the student. Hence, the student was able to achieve similar performance to the teacher. Further improvements on the technique introduced in [10] are proposed in [30]. In [81], a type of teacher-student model plus additional privileged information is used. This kind of distillation was then extended to training networks to deal with missing modalities in [32], where $L^2$ loss (hallucination loss) is used to train a hallucination network. In [64], hallucination was performed using CGANs, wherein the generator was used to generate representative information from the missing modalities while conditioned on the available modalities. This in effect, distills knowledge from the missing modalities into the model trained for dealing with available modalities.
3.3 Problem Formulation

Consider \( L \) sensors surveilling an area of interest. As in the previous chapter, we wish to detect/recognize targets \( O = \{o_1, o_2, ..., o_I\} \), given the data collected by the sensors. Let the \( n^{th} \) observation from the \( l^{th} \) sensor be denoted by, \( x_{ln} = [x_{lnq}]_{q=1...d_l} \). Let \( X_{dl \times N} = \{x_{ln}\}_{n=1...N} \) be the set of \( N \) observations made by the \( l^{th} \) sensor.

We first seek to discover a hidden common space\(^1\) \( H_{d_H \times N} \) of features present in the \( L \) sensors.

\(^1\)In [83], such a space was referred to as an ‘information subspace’
Since it is unknown a priori, its structure may include shared and non-shared features captured by the sensors. We refer to the non-shared features of specific sensors, as private. Given a hidden space, this thus amounts to being able to select from each sensor the optimal set of features for classification via a selection matrix, $S^l$,
\[
F^l_{d \times N} = S^l_{d \times d_H} H_{d_H \times N}, \forall l \in \{1, ..., L\}. \tag{3.1}
\]
Since the hidden space represents the information shared by the sensors, there must also exist a mapping, $G^l: X^l \rightarrow H$ such that,
\[
\hat{H}^l_{d_H \times N} = G^l(X^l) \approx H_{d_H \times N}, \forall l \in \{1, ..., L\}. \tag{3.2}
\]
From Equations (3.1) and (3.2) we have,
\[
F^l_{d \times N} = S^l_{d \times d_H} [(G^l(X^l))_{d_H \times N}], \forall l \in \{1, ..., L\}. \tag{3.3}
\]
The existence of such a hidden space makes it possible to detect damaged sensors, and safeguard the system performance against them. When the $l^{th}$ sensor is damaged, representative features for that sensor are reconstructed from the hidden space via the selection operator, $F^l = S^l H$. Following the feature extraction, a linear classifier, $c^i_l(F^l) = w^i_l^T F^l + b^i_l$, is used to determine the classification score for the $i^{th}$ object as seen by the $l^{th}$ sensor. The probability of occurrence of this object is then determined as,
\[
P^i_l(o^i_l) = \frac{\exp(w^i_l^T F^l + b^i_l)}{\sum_{m=1}^{L} \exp(w^i_m^T F^l + b^i_m)}. \tag{3.4}
\]
Finally, given these probability reports $R^l = \{P^l(o^i_l)\}$, the objective is to determine the fused probability report $R^f = \{P^f(o^i_l)\}$, which is achieved using a special case of Event Driven Fusion [65, 66].

### 3.4 Proposed Approach

As discussed in the previous section, the hidden space, $H$, can be estimated from the $l^{th}$ sensor observations as, $\hat{H}^l = G^l(X^l)$. The mapping $G^l$ is approximated by a neural network and is realized as the generator of a Conditional Generative Adversarial Network (CGAN), that generates the estimate of the hidden space, $\hat{H}^l$, while conditioned on the observations of the $l^{th}$ sensor. Hence, we will have $L$ generators that generate $L$ estimates of the hidden space.

The desired output of these generators is to generate an estimate $\hat{H}^l$, such that, $\hat{H}^l = G^l(X^l) \approx H$, $\forall l \in \{1, ..., L\}$. On the other hand, the discriminator attempts to correctly identify the modality that the estimate of the hidden space was generated by. That is, it assigns a score $D^i(\hat{h})$ as a probability that the estimate $\hat{h}$, was generated by $G^l$. When updating the parameters for the $l^{th}$ generator,
the hidden space estimates generated by all the other generators are assumed to be the target space, and the $l^{th}$ generator attempts to replicate these spaces, while at the same time attempting to generate an estimate that confuses the discriminator.

The standard formulation for the Generative Adversarial Network [26, 53] is known to have some instability issues [3]. Specifically, if the supports of the estimated hidden spaces are disjoint, which is highly likely when the inputs to the generators are coming from different modalities, a perfect discriminator is easily learned, and gradients for updating the generator may vanish. This issue was addressed in [3], and solved by using a Wasserstein GAN. So, using the sensor observations as the conditional information in the WGAN formulation [3], we have,

$$
\min_{G^l} \max_{D} \sum_{l=1}^{L} V(G^l, D),
$$

$$
V(G^l, D) = \sum_{m=1}^{L} \left[ \mathbb{I}E_{G^m(x^m)|\mathcal{W}}[D^m(G^m(x^m))] - \mathbb{I}E_{x^l|\mathcal{W}}[D^m(G^l(x^l))] \right].
$$

(3.5)

The discriminator $D$, in the above formulation is required to be compact and $K$-Lipschitz. This is done by clamping the weights of the discriminator to a fixed box (e.g. $\theta \in [-0.01, 0.01]$) [3]. Without a random process (i.e. noise), the generator would produce deterministic results. In [85] a Gaussian input noise is provided to the generator. But often, the generator can learn to ignore this noise [34, 52]. In our implementation we use dropout noise as in [34], where dropout is applied to several layers of the generator. The discriminator is updated once after every generator update. That is, for $L$ sensors, the discriminator is updated $L$ times for one update to the $l^{th}$ generator (see Algorithm 1). While this causes the estimates to be close to each other, it does not guarantee that the generated hidden spaces share the same basis. We additionally ensure that the hidden subspace estimates share a common basis. To that end, we exploit an important property of commutation between the operators that are responsible for transforming the data into the common subspace. Let $G^l$ be composed of a Convolutional Neural Network, $M^l$, followed by a fully connected layer, $Z^l$. Note that the fully connected layer does not have any non-linear activation, and acts as a linear transform on $M^l(x^l)$. This thus results in, $\hat{H}^l = Z^l_{d_H \times d_H} [M^l(x^l)]_{d_H \times N}$. Note that this formulation is similar to that discussed in Equation (2.20), but instead of using a random transformation, the network $M^l$ is learned in order to find the appropriate non-linear transform to the hidden space, and the optimization is formulated as a adversarial game between the $L$ generators and the discriminator.

If the operators $Z^1, ..., Z^L$ commute, they will share a common eigenbasis. Furthermore, since the transformation, $Z^l[M^l(x^l)]$ lies in the range space of the operator, $Z^l, \hat{H}^l = Z^l[M^l(x^l)], \forall l$ lie in a common subspace, due to the shared basis. While exact commutation cannot be guaranteed, we include a penalty term in the optimization to encourage the operators to commute, hence leading to operators that are ‘almost commuting’. Note that $Z^l$ must be a square matrix for validly enforcing the commutation cost as per Equation (2.18). By including this with the GAN loss, Equation (3.5)
becomes,

$$\min_{G^l, Z^l} \max_D \sum_{l=1}^{L} \left\{ V(G^l, D) + \gamma_2 \sum_{m=1, m \neq l}^{L} \| [Z^l, Z^m] \|^2 \right\}. \quad (3.6)$$

While the CGAN network is expected to eventually generate samples that lie in the same subspace as the target data, even in the absence of this auxiliary loss, we observe that its inclusion nevertheless improves the generator performance tremendously, in spite of the fact that the target space is ill defined. Experiments show that including the commutation term speeds up convergence, and also yields better hidden space estimates.

3.4.1 Structure of the Hidden Space/Selection Operators

In addition to ensuring that the hidden space estimates generated by different modalities lie in a common subspace, it is also important to structure the hidden space in a way that it not only contains information shared between the sensor modalities, but also contains private information of individual modalities. This is especially important for heterogeneous sensors as they may provide additional information along with some common information. This conforms with the notion that no two sensors are completely similar or dissimilar. This is equivalent to structuring $S^l$ such that it selects features that are relevant to the $l^{th}$ modality, while ignoring others. Such a structure can be achieved by encouraging the columns of the selection matrix to be close to zero. If the $m^{th}$ column of $S^l$ is zeroed out, then the $m^{th}$ feature in the hidden space, $H$, does not contribute toward the $l^{th}$ modality. This allows us to find the latent space, $H$, that naturally separates information shared between different modalities from which is private to each modality. We implement this by minimizing the $L_{\infty,1}$ norm, i.e., $\min_{S^l} \| S^l \|_{\infty,1}$, where,

$$\| S^l \|_{\infty,1} = \sum_i \max_j \{|s^l_{ij}|\}. \quad (3.7)$$

This minimizes the sum of the maximum of each column in $S^l$. Upon adding this term to the generator cost functional, we get,

$$\min_{G^l, S^l, Z^l} \max_D \sum_{l=1}^{L} \left\{ V(G^l, D) + \gamma_1 \| S^l \|_{\infty,1} + \gamma_2 \sum_{m=1, m \neq l}^{L} \| [Z^l, Z^m] \|^2 \right\} \quad (3.8)$$

Notice that the above formulation has a trivial solution of setting $S^l = 0$, and, $Z^l = 0$ for all $l$. In order to avoid this solution, we also optimize the classification based on the selected features, $F^l$, for each modality, via the minimization of the cross-entropy loss. This ensures that the learned features, $F^l$, are optimal for object detection/recognition, which is only possible if $S^l \neq 0$, and, $Z^l \neq 0$. Given the sensor observations, $X^l$, and the classifier $C^l = \{c^l_i\}$, the cross-entropy loss is
\[
C_{\text{LOSS}}^l(F^l) = \sum_{n=1}^N \sum_{i=1}^l -y_{ni} \log \sigma(c_i^l(f_{ni}^l)), \tag{3.9}
\]

where, \(Y_n = \{y_{ni}\}\) is the ground truth for the \(n^{th}\) sample, \(\sigma\) is the soft-max function, and, \(f_{ni}^l = S^l[G^l(x_{ni}^l)]\). Finally, the optimization task is,

\[
\min_{G^l, Z^l, S^l, C^l} \max_{D} \mathcal{L}(G^l, D, Z^l, S^l, C^l)
\]

\[
\mathcal{L}(G^l, D, Z^l, S^l, C^l) = \sum_{l=1}^L \left\{ V(G^l, D) + \gamma_1\|S^l\|_{\infty, 1} + \gamma_2 \sum_{m=1 \atop m \neq l}^L \| [Z^l, Z^m] \|^2 + \gamma_3 C_{\text{LOSS}}^l(F^l), \tag{3.10}\right.
\]

where, \(\gamma_1, \gamma_2, \) and \(\gamma_3\) are hyper-parameters that control the contribution of different terms toward the optimization. The necessary updates to train these networks are summarized in Algorithm 2.

This setup learns the optimal features for classification \(F^l\), while driven by the existence of the hidden space \(H\), such that \(F^l = S^lH\). Due to the inclusion of the selection matrix \(S^l\), followed by the Classification Layer, note that hand-crafting of features as in [65, 66], is no longer required. The

---

**Algorithm 2**: Training the CGAN system

Let \(\{\theta_d^q\}_{q \in \{1, \ldots, Q\}}\) be the parameters for the \(q^{th}\) layer of the \(l^{th}\) generator, and \(y_d\) be the output of that layer.

\(y^Q = Z^l[M^l(X^l)]\) and, \(y^{Q-1} = M^l(X^l)\).

Similarly, Let \(\{\theta_d^r\}_{r \in \{1, \ldots, R\}}\) be the parameters of the \(r^{th}\) layer of the discriminator and \(z^r\) be the output of that layer.

- for \(j\) in 1 : Number of Iterations
  - for \(l\) in 1 : \(L\)
    - Update the discriminator network,
      \[
      \theta_d^{l(j)} \leftarrow \theta_d^{l(j-1)} + \mu_D \left\{ \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dz^l} \frac{dz^R}{dz^{R-1}} \cdots \frac{dz^{l+1}}{d\theta_d^l} \right\} \tag{3.11} \]
    - Update the \(l^{th}\) generator network,
      \[
      C^{l(j)} \leftarrow C^{l(j-1)} - \mu_G \left\{ \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dC^l} \right\} \tag{3.12} \]
      \[
      S^{l(j)} \leftarrow S^{l(j-1)} - \mu_G \left\{ \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dC^l} \frac{dC^l(F^l)}{dC^l} \frac{dF^l}{S^l} \right\} + \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dS^l} \right\} \tag{3.13} \]
      \[
      z^{l(j)} \leftarrow z^{l(j-1)} - \mu_G \left\{ \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dC^l(F^l)} \frac{dC^l(F^l)}{dC^l} \frac{dC^l(F^l)}{dC^l(S^l)} \frac{dC^l(F^l)}{dC^l(C^l)} \right\} \tag{3.14} \]
      \[
      \theta_d^{r(j)} \leftarrow \theta_d^{r(j-1)} - \mu_G \left\{ \frac{d\mathcal{L}(G^l, D, Z^l, S^l, C^l)}{dC^l(F^l)} \frac{dC^l(F^l)}{dC^l} \frac{dC^l(F^l)}{dC^l} \right\} \tag{3.15} \]

---

36
features of interest are learned and automatically selected by the optimization during the training phase.

3.4.2 A Special Case of Event Driven Fusion

In order to fuse the individual decisions from the sensors we use a special case of Event Driven Fusion. Instead of defining feature events for each object as in [65], an event in this case is the occurrence of the i-th object as seen by the l-th sensor, o_i^l. The l-th classifier, C_l = {c_l^j}, provides the corresponding probability given the test sample, x_t^l, \( P_l(o_i^l) = \frac{\exp(c_l^j(f_t^l))}{\sum_{j=1}^{M} \exp(c_l^j(f_t^l))} \), as previously discussed in Section 3.3. Each sensor report is now represented as, \( R_l = \{P_l(o_i^l)\} \), and the fused probability of occurrence of the i-th object as per the rules of Event Driven Fusion [65] is determined as,

\[
P_l(o_i^l) = P_t(o_i^l, o_1^2, ..., o_l^L) = \rho \cdot P_{\text{Max}}(o_1^l, o_2^l, ..., o_l^L) + (1 - \rho) \cdot P_{\text{Min}}(o_1^l, o_2^l, ..., o_l^L),
\]

(3.16)

where \( \rho \) is a pseudo-measure of correlation between the sensor modalities. For making an informed decision in favor of an object, this formulation assumes all the modalities to be equally reliable. This is not always true in practice, as certain sensors may provide more discriminative information than others, and hence more reliable. It is thus important to weigh the various sensor decisions by a Degree of Confidence (DoC). The DoC in the decisions made by the l-th sensor given the test sample, x_t^l, is denoted by, DoC_l = [0,1], where DoC_l = 0 implies that the sensor observations do not provide any useful information about the target identity, and DoC_l = 1 implies that information provided by the sensor is highly discriminative with respect to target classification. The individual sensor reports, \( R_l = \{P_l(o_i^l)\} \), are now redefined as,

\[
R_l = \{P_l(o_1^l), P_l(o_2^l), ..., P_l(o_i^l), P_l(unc^l)\},
\]

(3.17)

where \( P_l(o_i^l) = DoC_l \cdot P_l(o_i^l) \), and \( P_l(unc^l) = 1 - DoC_l \) is the probability that the l-th sensor is uncertain about the target identity. The joint probability distribution for the new sensor reports is now rewritten as,

\[
P_t(a^1, a^2, ..., a^L) = \rho \cdot P_{\text{Max}}(a^1, a^2, ..., a^L) + (1 - \rho) \cdot P_{\text{Min}}(a^1, a^2, ..., a^L),
\]

(3.18)

where, \( a^l \in \{o_1^l, o_2^l, ..., o_i^l, unc^l\} \), and the probability of occurrence of the i-th target is computed as,

\[
P_l(o_i) = P_l(\bigwedge_{m=1}^{L} o_i^m) \bigwedge_{l=1}^{L} (unc_{m_1} \bigwedge_{l \neq m_1}^{L} o_i^l) \bigwedge_{m_1, m_2=1}^{L} (unc_{m_1} \bigwedge_{l \neq m_1, m_2}^{L} o_i^l) \bigwedge_{l=1}^{L-1} (\bigwedge_{j=1}^{m_{l+1}} unc_{m_j} \bigwedge_{l \neq m_j}^{L} o_i^l)),
\]

(3.19)
with a degree of confidence, $D\text{o}C^f_i = 1 - P_t(\bigwedge_{l=1}^i \text{unc}^l)$, in the fused decision.

For a 2D scenario (i.e. two sensors), the probability of occurrence of the $i^{th}$ target is,

$$P^f_i(o_i) = P_t((o_i^1 \land o_i^2) \lor (\text{unc}^1 \land o_i^2) \lor (o_i^1 \land \text{unc}^2)), \quad (3.20)$$

with a degree of confidence, $D\text{o}C^f = 1 - P(\text{unc}^1 \land \text{unc}^2)$, in the fused decision. This formulation takes into account the potential uncertainties in the decisions made by individual sensors. Figure 3.6 shows the joint distribution to be determined in a 2D case.

![Figure 3.6 Fused joint probability distributions when accounting for the uncertain event. The joint distribution is determined following the rules of Event Driven Fusion.](image)

The degree of confidence in the $l^{th}$ sensor for classification of the test sample, $x^l_t$, is determined as,

$$D\text{o}C^l_i = (1 - p_D(\hat{h}_i^l)).\text{Acc}_{\text{train}}^l, \quad (3.21)$$

where, $p_D(\hat{h}_i^l)$ is the probability of damage for the $l^{th}$ sensor given the test sample, $x^l_t$ (discussed in detail in the next subsection), and $\text{Acc}_{\text{train}}^l$ is the training accuracy of the $l^{th}$ sensor. The training accuracy in Equation (3.21) represents prior information about the discrimination power of the sensor, while $(1 - p_D(\hat{h}_i^l))$ represents the sensor condition at the time of operation/testing.

### 3.4.3 Sensor Failure Detection

Since $H$ is designed so that it be common for all the sensors, it may be used to detect a damaged sensor.

#### 3.4.3.1 Cross-Sensor Tracking

The estimate $\hat{H}^m$, based on erroneous observations, will significantly deviate from the estimates from normal observations (i.e. sensors that are not damaged), $\hat{H}^l, l \neq m$. This allows detection of
damage in a sensor during the testing phase. The \( m^{th} \) sensor is said to be damaged if,

\[
\forall j, l \neq m, \sum_{l=1}^{L} \mathcal{I}(||\hat{h}_{l}^{m} - \hat{h}_{l}^{l}||^2 > T) \geq \begin{cases} 
\frac{L-1}{2}, & \text{if } L \text{ is odd}, \\
\frac{L}{2} - 1, & \text{if } L \text{ is even.}
\end{cases}
\]

\[\text{And, } \sum_{j,l=1 \atop j,l \neq m}^{L} \mathcal{I}(||\hat{h}_{l}^{j} - \hat{h}_{l}^{l}||^2 < T) \geq \begin{cases} 
\frac{L-1}{2}, & \text{if } L \text{ is odd}, \\
\frac{L}{2} - 1, & \text{if } L \text{ is even.}
\end{cases} \tag{3.22}
\]

where, \( \mathcal{I}(\cdot) \) is the indicator function, \( T \) is a threshold value determined from the training data, and \( \hat{h}_{l}^{l} \) is the estimated hidden space given the observation \( x_{l} \). The limitation with this approach is that we can only detect up to \( \frac{L-1}{2} / \frac{L}{2} - 1 \) damaged sensors.

### 3.4.3.2 Hierarchical Clustering

A faulty sensor can also be detected by comparing the hidden space generated by the test sample with that generated from training data. One way to proceed is by clustering functional sensor observations (from training data), and verify whether the hidden space estimate generated by the test sample can be associated to any of these clusters. The concatenation of the training data, \( \hat{H}^{C}_{d \times LN} = \{ \hat{H}^{1}, \hat{H}^{2},..., \hat{H}^{L} \} \), is first used to construct a clustering tree based on an Agglomerative approach (see Algorithm 3). The probability of damage of the \( l^{th} \) sensor is then computed as,

\[
p_{D}(\hat{h}_{l}^{l}) = \frac{d_{lev}}{\max_{v} d_{v}}, \tag{3.23}
\]

where, \( d_{v} \) is the cut-off distance at clustering level \( v \), and,

\[
\text{lev} = \arg \min_{v} \sum_{j} \exists j \in \{1,\ldots,J_{lev} \}, \hat{h}_{l}^{l} \in Z_{lev}^{j}, \tag{3.24}
\]

\( v = 1,\ldots,V \) are the clustering levels, \( J_{v} \) is the number of clusters at level \( v \), and \( Z_{v}^{j} \) is the \( j^{th} \) cluster at level \( v \). This is a measure of how quickly the hidden space estimate, \( \hat{h}_{l}^{l} \), can be clustered with the training data. The \( l^{th} \) sensor is said to be damaged if,

\[
p_{D}(\hat{h}_{l}^{l}) > T, \tag{3.25}
\]

where, \( T \), is a threshold value which will depend on the dataset, types of sensors, SNR, etc. In our evaluation, we compute the optimal thresholds at different SNRs for the training data, and these thresholds are later used during the testing phase in order to determine the state of a sensor. This probability measure is also used in Equation (3.21), in order to adapt the DoC based on the sensor condition in a functional mode. If the sensor is damaged, the representative features are generated
Figure 3.7 A clustering tree created using Hierarchical Clustering (Agglomerative Clustering) using the selection matrix as,

\[ \hat{f}_t = \sum_{m \in \Gamma} m \frac{D o C_t m}{D o C_t} \]

where, \( \Gamma \) is the set of working sensors.

**Algorithm 3:** Hierarchical Clustering (Agglomerative Clustering)

- Initialize clusters at \( v = 0 \):
  \[ C_0 = \{ Z_j^0 = \{ h_j \}, j = 1...N \} \]
- while Number of Clusters > 1:
  - \( v = v + 1 \)
  - Among all cluster pairs, \( \{ Z_{v-1}^r, Z_{v-1}^s \} \), find the one, say \( \{ Z_{v-1}^i, Z_{v-1}^j \} \), such that:
  \[ d(Z_{v-1}^i, Z_{v-1}^j) = \min_{r,s} d(Z_{v-1}^r, Z_{v-1}^s) \]
  \[ d(A, B) = \max_{a \in A} \| a - b \|^2 \]

where, \( d(.) \) is a measure of dissimilarity.

- Assign \( Z_{v-1}^q = Z_{v-1}^i \cup Z_{v-1}^j \)
- Get new clustering: \( C_v = \{ C_{v-1} - \{ Z_{v-1}^i, Z_{v-1}^j \} \cup Z_v \} \)

### 3.5 Experiments and Results

We validate our proposed approach by running experiments on two different datasets. The first dataset we use is pre-collected data from a network of seismic, acoustic, and imaging sensors deployed in a field, where people/vehicles were walking/driven around in specified patterns. Details about this sensor setup and experiments can be found in [54]. This dataset has been previously used for target detection in [35, 47, 64, 65]. Here, we use this dataset to classify between human targets, vehicular targets, and no targets. Some data samples from the sensors can be seen in Figure 3.8. This
will be referred to as ‘Dataset-1’ in the following discussions. For our second experiment, we select
two sensors, namely a Radar sensor and a telescopic optical sensor, for one (latter) of which we have acquired real data. For technical reasons, our Radar measurements were never co-measured with the optical data, and so we use MATLAB Simulink in order to simulate the radar responses. Both sensors are ideally synchronized when observing a given target, which in our case, can be any object in outer space, such as satellite or space debris. Each generated radar signal over one second is correlated with two telescopic images. Samples for objects with different velocities, cross-sections, ranges, and aspect-ratios are then generated. Object classes are defined in the same way as in Chapter 2, based on events on the feature values. Note that these events are no longer required for training purposes, and are only used for the purposes of labeling the data. This will be referred to as ‘Dataset-2’ in the following discussions.

---

**Figure 3.8** (a): Sample video frame with a human target, (b): Sample seismic sensor observations for human target, vehicular target, and no target cases, (c): Sample acoustic sensor observations for human target, vehicular target, and no target cases.
3.5.1 Implementation Details

Figure 3.9-(a) shows the detailed block diagram for implementation of the discussed approach on Dataset-1. As noted earlier we no longer require handcrafted features and event definitions as in [65, 66], since we let the generative structure (see Figure 3.9) guide the learning of features. Similarly, the block diagram for implementation of this system on Dataset-2 can be seen in Figure 3.9-(b). The output of the $l^{th}$ generator, $G^l$, for a test sample, $x^l_t$, is a $d_H$-dimensional estimate of the hidden space, $\hat{h}^l_t$. This hidden space is also used to detect potential damages to the sensors deployed. The optimal features $f^l_t$, are subsequently selected by each sensor for making decisions on target identities via the selection matrix $S^l$. The decisions are then fed into the fusion system which synthesizes a decision on the basis of the rules discussed in Section 3.4.2. The generator network uses 1-D convolutions in case of seismic/acoustic and radar modalities, and 2-D convolutions in case of the imaging sensors. We use a 6 layered Neural Network, with 2x2 max pooling layer after the 2$^{nd}$ and 4$^{th}$ layer. ReLU activation is applied after every convolutional layer. The first 4 layers are convolutional, while the last two are fully connected. The first two convolutional layers use a filter size of 5 and the next two use a filter size of 3. The first fully connected layer is used to transform the output of the convolutional layers into a $d_H$ dimensional representation. All the layers preceding the final fully connected layer approximate the mapping $M^l$, while the final layer approximates the operator $Z^l$, and transforms the data into a common subspace.

In determining the dimension of the hidden space, we search over values ranging from $d = 50$ to $d = 5000$. For Dataset-1 we find the best performance at $d = 500$, while for Dataset-2, $d = 700$. In both cases we observe that at best performance, $d << d_l$.

3.6 Performance Analysis

Table 3.1 shows the performance of different techniques as compared with the proposed approach of using a Generative Adversarial Network to learn optimal features, which are then used for target detection. All sensors are assumed to be working normally in Table 3.1. Comparisons are carried
Table 3.1 Performance Comparison for both the Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (Dataset-1)</th>
<th>Accuracy (Dataset-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seismic Sensor</td>
<td>93.62 %</td>
<td>-</td>
</tr>
<tr>
<td>Acoustic Sensor</td>
<td>68.71 %</td>
<td>-</td>
</tr>
<tr>
<td>Imaging Sensor</td>
<td>90.33 %</td>
<td>-</td>
</tr>
<tr>
<td>Radar Sensor 1</td>
<td>-</td>
<td>89.33 %</td>
</tr>
<tr>
<td>Radar Sensor 2</td>
<td>-</td>
<td>86.73 %</td>
</tr>
<tr>
<td>Telescopic Imaging Sensor</td>
<td>-</td>
<td>83.57 %</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>88.13 %</td>
<td>86.45 %</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion</td>
<td>91.61 %</td>
<td>-</td>
</tr>
<tr>
<td>Similar + Dissimilar Sensor Fusion</td>
<td>-</td>
<td>89.63 %</td>
</tr>
<tr>
<td>Dempster-Shafer Fusion</td>
<td>88.77 %</td>
<td>87.30 %</td>
</tr>
<tr>
<td>Event Driven Fusion (Without using GAN structure) [65, 66]</td>
<td>92.04 %</td>
<td>90.36 %</td>
</tr>
<tr>
<td>Canonical Correlation Analysis [35]</td>
<td>86.64 %</td>
<td>85.69 %</td>
</tr>
<tr>
<td>Discriminant Analysis</td>
<td>91.33 %</td>
<td>88.21 %</td>
</tr>
<tr>
<td>Dictionary Learning [6]</td>
<td>94.46 %</td>
<td>93.77 %</td>
</tr>
<tr>
<td>Hidden Space Generated by GAN</td>
<td>95.79 %</td>
<td>91.13 %</td>
</tr>
<tr>
<td>Event Driven Fusion + GAN</td>
<td>97.79 %</td>
<td>95.34 %</td>
</tr>
</tbody>
</table>

out with approaches that perform fusion at the decision level, as well as those that seek a hidden space for fusion. We find that using the individual sensor features (whose learning is driven by the existence of a structured hidden space) for classification and performing decision level fusion on these classifications yields better performance relative to those based solely on the hidden space. For the evaluation of Model Based Fusion approaches (see Section 2.2.3), the dissimilar sensor setting is considered for Dataset-1 as all three sensors are different. On the other hand, for Dataset-2, the two radars are first combined using similar sensor fusion and the result of this is combined with the telescopic sensor using dissimilar sensor fusion.

We also compare the effects of different losses in our objective. First we implement a system which uses an Adversarial setup, along with the classification losses, i.e., $\gamma_1$, and $\gamma_2$ in Equation (3.10) are set to 0. The losses are seen in Figure 3.10-(a). The blue plot represents the negative of the discriminator loss for the GAN network (y-axis on left), while the orange plot corresponds to the commutation loss (y-axis on right). Figure 3.10-(b) shows the sum of pairwise distances between the hidden estimates generated by the three sensors, i.e. Magnitude($k$) = $\sum_{l,m=1}^{L}(H_l^H(k) - H_m^H(k))^2$, where, $k \in \{1, ..., d_H\}$, refers to the feature number in the $d_H$-dimensional hidden space. Figure 3.10-(c) shows the individual classification performance of the sensors. It is observed that the performance of Seismic and Acoustic Sensors improves as the model is updated, but the performance of the imaging sensor is very low. Figure 3.11 shows the optimization losses when the commutation term is
Figure 3.10 (a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when GAN Loss is optimized in tandem with classification loss.

Figure 3.11 (a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when commutation term is included with the GAN and classification term.

added to the objective, i.e. only $\gamma_1$ is set to zero in Equation (3.10). This amounts to minimizing of the pairwise commutation loss, hence forcing the hidden space estimates to lie in a common subspace. It is seen that including the commutation cost leads to closer hidden spaces (Figure 3.11-(b)). The performance of the imaging sensor, however, does not significantly increase. Figure 3.12 shows the optimization loss when all the terms in Equation (3.10) are active. The imaging sensor now starts giving better performance. Since the $L_{\infty,1}$ norm on the selection matrix allows representation of private/shared features in the hidden space, the heterogeneity of the imaging sensor is maintained, and selected features are more optimal for classification based on the imaging sensor. The effects of the $L_{\infty,1}$ norm can also be seen in the differences between the hidden estimates in Figure 3.12-(b), where some features now exhibit more distinguishing diversity compared to others, as they may correspond to private features.
### Figure 3.12
(a): Optimization Losses, (b): Sum of pairwise distances between hidden estimates, (c): Training Accuracies/Epoch when all terms in Equation (3.10) are active.

(a) Gaussian noise is added to Seismic and Acoustic Sensor, while Imaging data is assumed to be clean.
(b) Gaussian noise is added to Imaging and Acoustic Sensor, while Seismic data is assumed to be clean.
(c) Gaussian noise is added to Seismic and Imaging Sensor, while Acoustic data is assumed to be clean.

**Figure 3.13** Comparison of the proposed technique (GAN + Event Driven Fusion) with existing techniques for Dataset-1.

### 3.6.1 Robustness Analysis

The major advantage of learning a hidden space between the modalities is the ability to detect sensor damage in operation, and to generate representative features for that damaged sensor as seen in Equation (3.26). The contribution of the representative features towards the fused decision can also be controlled via the degree of confidence. We also study how the performance of the system varies with different Signal to Noise Ratios. That is noise is selectively added to some sensors while another is normally functioning. Figures 3.13 and 3.14 show the degradation of the fusion performance as the SNR decreases for Dataset-1 and Dataset-2, respectively. The plots with ‘asterisk’ (*) markers represent approaches that search for a common subspace in order to fuse the different modalities, while those with ‘dot’ (.) markers represent approaches that perform fusion at the decision level. Finally, the plots with ‘square’ markers show the performance of our proposed approach. The blue plot uses adaptive DoC $\{Do C_t^l = (1 - p_t(R_t^l)).Acc_{train}^l\}$, while the orange plot only uses the prior information $\{Do C_t^l = Acc_{train}^l\}$, which is also the case for other plots using decision level fusion. It
Gaussian noise is added to the Telescopic Imaging Sensor, while Radar data is assumed to be clean.

Figure 3.14 Comparison of the proposed technique (GAN + Event Driven Fusion) with existing techniques for Dataset-2.

is observed that it is important to update the DoC during the implementation based on the sensor condition, rather than only depending on the prior information about the discriminative abilities of the sensor. Furthermore, the above graphs show that the degradation has severe effects in the case where the seismic and imaging sensors are damaged. This is due to the fact that the discriminative power of the acoustic sensor is low, and in spite of adapting the DoC and generating representative features, the performance is limited by the information contained in the observations of the sensor. A similar trend is also observed for Dataset-2, where the telescopic imagery has lower discriminative ability compared to the radar sensors.

3.7 Conclusion

In this Chapter, we proposed a data driven approach that learns a structured hidden space between sensors by the way of a bank of Generative Adversarial Networks. The hidden space is used to learn the features of interest for target classification. The hidden space serves a dual purpose of detecting noisy/damaged sensors and mitigating sensor losses by ensuring a graceful performance degradation. Experiments on multiple datasets show that the proposed approach secures a great degree of robustness to noisy/damaged sensors, and outperforms existing fusion algorithms.
CHAPTER 4

VOLTERRA NEURAL NETWORKS (VNNS): A VOLTERRA FILTERING APPROACH FOR HUMAN ACTION RECOGNITION, TARGET DETECTION, AND MULTI-MODAL FUSION

4.1 Introduction

Human action recognition is an important research topic in Computer Vision, and can be used towards surveillance, video retrieval, and man-machine interaction to name a few. The survey on Action Recognition approaches [29, 41, 75, 87] provides a good progress overview. Video classification usually involves three stages [37, 51, 56, 76, 84], namely, visual feature extraction (local features like Histograms of Oriented Gradients (HoG) [15], or global features like Hue, Saturation, etc.[63]), feature fusion/concatenation, and lastly classification. In [88], an intrinsic stochastic modeling of human activity on a shape manifold is proposed and an accurate analysis of the non-linear feature space of activity models is provided. The emergence of Convolutional Neural Networks (CNNs) [46], along with the availability of large training datasets and computational resources have come a long way to obtaining the various steps by a single neural network. This approach has led to remarkable progress in action recognition in video sequences, as well as in other vision applications like object detection and segmentation [60, 62, 71], scene labeling [19], image generation [26, 86], image translation [2, 4, 34], information distillation [32, 64], etc. In the Action Recognition domain, datasets like
the UCF-101 [77], Kinetics [38], HMDB-51 [43], and Sports-1M [37] have served as benchmarks for evaluating various solution performances. In action recognition applications the proposed CNN solutions generally align along two themes: 1. One Stream CNN (only use either spatial or temporal information), 2. Two Stream CNN (integrate both spatial and temporal information). Many implementations [12, 18, 20, 73] have shown that integrating both streams leads to a significant boost in recognition performance. In Deep Temporal Linear Encoding [18], the authors propose to use 2D CNNs (pre-trained on ImageNet [17]) to extract features from RGB frames (spatial information) and the associated optical flow (temporal information). The video is first divided into smaller segments for feature extraction via 2D CNNs. The extracted features are subsequently combined into a single feature map via a bilinear model. This approach, when using both streams, is shown to achieve a 95.6 % accuracy on the UCF-101 dataset, while only achieving 86.3 % when only relying on the RGB stream. Carreira et al. [12] adopt the GoogLeNet architecture which was developed for image classification in ImageNet [17], and use 3D convolutions (instead of 2D ones) to classify videos. This implementation is referred to as the Inflated 3D CNN (I3D), and is shown to achieve a performance of 88.8 % on UCF-101 when trained from scratch, while achieving a 98.0 % accuracy when a larger dataset (Kinetics) was used for pre-training the entire network (except for the classification layer).

While these approaches achieve near perfect classification, the models are extremely heavy to train, and have a tremendous number of parameters (25M in I3D, 22.6M in Deep Temporal Linear Encoding). This in addition, makes the analysis, including the necessary degree of non-linearity, difficult to understand, and the tractability elusive. In this Chapter we explore the idea of introducing controlled non-linearities through interactions between delayed samples of a time series. We will build on the formulations of the widely known Volterra Series [82] to accomplish this task. While prior attempts to introduce non-linearity based on the Volterra Filter have been proposed [44, 45, 91], most have limited the development up to a quadratic form on account of the explosive number of parameters required to learn higher order complexity structure. While quadratic non-linearity is sufficient for some applications (eg. system identification), it is highly inadequate to capture all the non-linear information present in videos.

**Contributions:** In this Chapter, we propose a Volterra Filter [82] based architecture where the non-linearities are introduced via the system response functions and hence by controlled interactions between delayed frames of the video. The overall model is updated on the basis of a cross-entropy loss of the labels resulting from a linear classifier of the generated features. An efficiently cascaded implementation of a Volterra Filter is used in order to explore higher order terms while avoiding over-parameterization. The Volterra filter principle is also exploited to combine the RGB and the Optical Flow streams for action recognition, hence yielding a performance driven non-linear fusion of the two streams. We further show that the number of parameters required to realize such a model is significantly lower in comparison to a conventional CNN, hence leading to faster training and significant reduction of the required resources to learn, store, or implement such a model.

Additionally we will also perform experiments for target detection/recognition using the datasets discussed in the previous chapters.
4.2 Background and Related Work

4.2.1 Volterra Series

The physical notion of a system is that of a black box with an input/output relationship $y_t / x_t$. If a non-linear system is time invariant, the relation between the output and input can be expressed in the following form [70, 82],

$$y_t = \sum_{\tau_1=0}^{L-1} W_1^{\tau_1} x_{t-\tau_1} + \sum_{\tau_1, \tau_2=0}^{L-1} W_2^{\tau_1, \tau_2} x_{t-\tau_1} x_{t-\tau_2} + \ldots + \sum_{\tau_1, \tau_2, \ldots, \tau_K=0}^{L-1} W^K_{\tau_1, \tau_2, \ldots, \tau_K} x_{t-\tau_1} x_{t-\tau_2} \ldots x_{t-\tau_K}, \quad (4.1)$$

where $L$ is the number of terms in the filter memory (also referred to as the filter length), $W^K$ are the weights for the $k$th order term, and $W^K_{\tau_1, \tau_2, \ldots, \tau_K} = 0$, for any $\tau_j < 0$, $j = 1, 2, \ldots, k$, $\forall k = 1, 2, \ldots, K$ due to causality. This functional form is due to the mathematician Vito Volterra [82], and is widely referred to as the Volterra Series. The calculation of the kernels is a computationally complex problem, and a $K$th order filter of length $L$, entails solving $L^K$ equations. The corresponding adaptive weights are a result of a target energy functional whose minimization iteratively adapts the filter taps as shown in Figure 4.1.

It is worth observing from Equation (4.1) that the linear term is actually similar to a convolutional layer in CNNs. Non-linearities in CNNs are introduced only via activation functions, and not in the convolutional layer, while in Equation 4.1 we will introduce non-linearities via higher order convolutions.
4.2.2 Nested Volterra Filter

In [58], a nested reformulation of Equation (4.1) was used in order to construct a feedforward implementation of the Volterra Filter,

\[
y_t = \sum_{\tau_1=0}^{L-1} x_{t-\tau_1} W_{\tau_1}^1 + \sum_{\tau_2=0}^{L-1} x_{t-\tau_2} W_{\tau_1,\tau_2}^2 + \sum_{\tau_3=0}^{L-1} x_{t-\tau_3} W_{\tau_1,\tau_2,\tau_3}^3 + \ldots \]. \tag{4.2}
\]

Each factor contained in the brackets can be interpreted as the output of a linear Finite Impulse Response (FIR) filter, thus allowing a layered representation of the Filter. A nested filter implementation with \(L = 2\) and \(K = 2\) is shown in Figure 4.2. The length of the filter is increased by adding modules in parallel, while the order is increased by additional layers. Much like any multi-layer network, the weights of the synthesized filter are updated layer after layer according to a backpropogation scheme. The nested structure of the Volterra Filter, yields much faster learning in comparison to that based on Equation (4.1). It, however, does not reduce the number of parameters to be learned, leading to potential over-parameterization when learning higher order filter approximations. Such a structure was used for a system identification problem in [58]. The mean square error between the desired signal \((d_t)\) and the output of the filter \((y_t)\) was used as the cost functional to be minimized,

\[
E_t = \frac{1}{2} (d_t - y_t)^2, \tag{4.3}
\]

and the weights for the \(k^{th}\) layer are updated per the following equations,

\[
W_{\tau_1,\tau_2,\ldots,\tau_k}^k (t + 1) = W_{\tau_1,\tau_2,\ldots,\tau_k}^k (t) - \eta \frac{\partial E_t}{\partial W_{\tau_1,\tau_2,\ldots,\tau_k}^k}, \tag{4.4}
\]

\[
\frac{\partial E_t}{\partial W_{\tau_1,\tau_2,\ldots,\tau_k}^k} = x_{t-\tau_k} x_{t-\tau_{k-1}} \ldots x_{t-\tau_1} (y_t - d_t). \tag{4.5}
\]
4.2.3 Bilinear Convolution Neural Networks

There has been work on introducing 2\textsuperscript{nd} order non-linearities in the network by using a bi-linear operation on the features extracted by convolutional networks. Bilinear-CNNs (B-CNNs) were introduced in [50] and were used for image classification. In B-CNNs, a bilinear mapping is applied to the final features extracted by linear convolutions leading to 2\textsuperscript{nd} order features which are not localized. As a result a feature extracted from the lower right corner of a frame in the B-CNN case, may interact with a feature from the upper right corner, and these two are not necessarily related (hence introducing erroneous additional characteristics), and this is in contrast to our proposed approach which highly controls such effects. Compact Bilinear Pooling was introduced in [23] where a bilinear approach to reduce feature dimensions was introduced. This was again performed after all the features had been extracted via linear convolutions and was limited to quadratic non-linearities. In our work we will explore non-linearities of much higher orders and also account for continuity of information between video frames for a given time period with the immediately preceding period. This effectively achieves a Recurrent Network-like property which accounts for a temporal relationship.

4.2.4 Long Short Term Memory Networks

Long-Short Term Memory Networks (LSTMs) [31] have been widely used to capture the long-term trajectory information in temporally evolving data applications. The sequential modeling ability of LSTMs makes them particularly appealing for capturing long-range temporal dynamics in videos. An LSTM computes a mapping from an input sequence, \( x = \{x_1, ..., x_T\} \) to an output sequence \( h = \{h_1, ..., h_T\} \). The block diagram for the implementation of a LSTM network is depicted in Figure 4.3. The mapping of the input at time \( t, x_t \) to the output, \( h_t \) entails the following:

1. **Forget Gate:**
   \[
   f_t = \sigma(W^f_f[h_{t-1}, x_t]) = \sigma(W^1_f \cdot h_{t-1} + W^2_f \cdot x_t). \quad (4.6)
   \]
2. Input Gate:

\[
i_t = \sigma(W_i[h_{t-1}, x_t]) = \sigma(W_{i1}^1 h_{t-1} + W_{i2}^2 x_t), \quad (4.7)
\]
\[
\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t]), \quad (4.8)
\]
\[
C_t = f_t C_{t-1} + i_t \tilde{C}_t. \quad (4.9)
\]

3. Output Gate:

\[
o_t = \sigma(W_o[h_{t-1}, x_t]) = \sigma(W_{o1}^1 h_{t-1} + W_{o2}^2 x_t), \quad (4.10)
\]
\[
h_t = o_t \tanh(C_t). \quad (4.11)
\]

Where \(\sigma\) is the sigmoid activation function, \(\tanh\) is the hyperbolic tangent function, and \(W_f, W_i, W_c,\) and \(W_o\) are the weight matrices characterizing the forget gate, input gate, cell state, and output gate respectively.

4.3 Problem Statement

Let a set of actions \(\mathcal{A} = \{a_1, ..., a_I\}\) be of interest following a observed sequence of frames \(X_{T \times H \times W}\), where \(T\) is the total number of frames, and \(H\) and \(W\) are the height and width of a frame. At time \(t\), the features \(F_t = g(X_{[t-L+1:t]})\), will be used for classification of the sequence of frames \(X_{[t-L+1:t]}\) and mapped into one of the actions in \(\mathcal{A}\), where \(L\) is the number of frames in the memory of the system/process. A linear classifier with weights \(W^{cl} = \{w^{cl}_i\}_{i=1,...,I}\) and biases \(b^{cl} = \{b^{cl}_i\}_{i=1,...,I}\) will then be central to determining the classification scores for each action, followed by a soft-max function \((\rho(.)\) to convert the scores into a probability measure. The probability that the sequence of frames are associated to the \(i^{th}\) action class is hence the result of,

\[
P_t(a_i) = \rho(w^{cl^T}_i . F_t + b^{cl}_i) = \frac{\exp(w^{cl^T}_i . F_t + b^{cl}_i)}{\sum_{m=1}^{I} \exp(w^{cl^T}_m . F_t + b^{cl}_m)}, \quad (4.12)
\]

4.4 Proposed Solution

4.4.1 Volterra Filter based Classification

In our approach we propose a Volterra Filter structure to approximate a function \(g(.)\). Given that video data is of interest here, a spatio-temporal Volterra Filter must be applied as shown in Figure 4.4. As a result, this 3D version of the Volterra Filter discussed in Section 2 is used to extract the
Figure 4.4 Volterra Filter must be applied in both space and time for video recognition.

features,

\[
F_{t_1} = g \left( X_{[t-L+1:t]} \right) = \sum_{\tau_1,\sigma_{11},\sigma_{21}} W_{11}^{1} \sigma_{11}^{11} \sigma_{21}^{1} X_{t-L+1:t} - \tau_1 s_1 - \sigma_{11} s_2 + \sum_{\tau_2,\sigma_{12},\sigma_{22}} W_{11}^{2} \sigma_{12}^{1} \sigma_{22}^{1} X_{t-L+1:t} - \tau_2 s_1 - \sigma_{12} s_2 + \sum_{t_1,\sigma_{11},\sigma_{21}} \cdots
\]

(4.13)

where, \( \tau_j \in [0, L-1] \), \( \sigma_{1j} \in [-p_1, p_1] \), and \( \sigma_{2j} \in [-p_2, p_2] \). Following this formulation, and as discussed in Section 3, the linear classifier is used to determine the probability of each action in \( \mathcal{A} \), and updating the filter parameters is pursued by minimizing some measure of discrepancy relative to the ground truth and the probability determined by the model. Our adopted measure herein is the cross-entropy loss computed as,

\[
E = \sum_{t,i} -d_{ti} \log P_t(a_i),
\]

(4.14)

where, \( t \in \{1, L + 1, 2L + 1, ..., T\} \), \( i \in \{1, 2, ..., I\} \), and \( d_{ti} \) is the ground truth label for \( X_{[t-L+1:t]} \) belonging to the \( i^{th} \) action class. In addition to minimizing the error, we also include a weight decay in order to ensure generalizability of the model by penalizing large weights. So, the overall cost functional which serves as a target metric is written as,

\[
\min_g \sum_{t,i} -d_{ti} \log \rho(w^{tir}_i g(X_{[t-L+1:t]}) + b^{tir}_i) + \frac{\lambda}{2} \left[ \sum_{k=1}^{K} \|W^k\|_{2}^2 + \|W^{c}\|_{2}^2 \right],
\]

(4.15)

where \( \rho \) is the soft-max function, and \( K \) is the order of the filter.
Figure 4.5 Block diagram for an Overlapping Volterra Neural Network

4.4.2 Non-Linearity Enhancement: Cascaded Volterra Filters

A major challenge in learning the afore-mentioned architecture arises when higher order non-linearities are sought. The number of required parameters for a $K^{th}$ order filter is,

$$
\sum_{k=1}^{K} (L[2p_1 + 1][2p_2 + 1])^k.
$$

(4.16)

This complexity increases exponentially when the order is increased, thus making a higher order (> 3) Volterra Filter implementation impractical. To alleviate this limitation, we use a cascade of $2^{nd}$ order Volterra Filters, wherein, the second order filter is repeatedly applied until the desired order is attained. This is as a result of the property of Volterra Filters which states that when filters with orders $m$ and $n$ are cascaded together, it leads to a filter of order $m\cdot n$ [69]. If the length of the first filter in the cascade is $L_1$, the input video $X_{[t-L+1]}$ can be viewed as a concatenation of a set of shorter videos,

$$
X_{[t-L+1]} = \left[ X_{[t_1:t_1+L_1]} X_{[t_1+L_1:t_1+2L_1]} \ldots X_{[t_1+(M_1-1)L_1:t_1+M_1L_1]} \right],
$$

(4.17)

where $M_1 = \frac{L}{L_1}$, and $t_1 = t - L + 1$. Now, a $2^{nd}$ order filter $g_1(.)$ applied on each of the sub-videos leads to the features,

$$
F^1_{t_1:M_1} = \left[ g_1(X_{[t_1:t_1+L_1]}) g_1(X_{[t_1+L_1:t_1+2L_1]}) \ldots g_1(X_{[t_1+(M_1-1)L_1:t_1+M_1L_1]}) \right].
$$

(4.18)

A second filter $g_2(.)$ of length $L_2$ is then applied to the output of the first filter,

$$
F^2_{t_1:M_2} = \left[ g_2(F^1_{t_1:L_2}) g_2(F^1_{t_1+L_2:t_2}) \ldots g_2(F^1_{t_1+(M_2-1)L_2:t_1+M_2L_2}) \right],
$$

(4.19)

where, $M_2 = \frac{M_1}{L_2}$. Note that the features in the second layer are generated by taking quadratic interactions between those generated by the first layer, hence, leading to $4^{th}$ order terms. Finally, for a cascade of $2^{x}$ filters, the final set of features is obtained as,

$$
F^{2^x}_{t_1:M_2} = \left[ g_2(F^{2^x-1}_{t_1:L_2}) g_2(F^{2^x-1}_{t_1+L_2:t_2}) \ldots g_2(F^{2^x-1}_{t_1+(M_2-1)L_2:t_1+M_2L_2}) \right],
$$

(4.20)
where, $M_z = \frac{M_{z-1} - L_z + 1}{L_z}$. Note that these filters can also be implemented in an overlapping fashion leading to the following features for the $z^{th}$ layer, $z \in \{1, ..., \mathcal{Z}\}$,

$$F^z_{[1:M_z]} = \begin{bmatrix} g_z(F^{z-1}_{[1:L_z]}), g_z(F^{z-1}_{[2:L_z+1]}), \ldots, g_z(F^{z-1}_{([M_{z-1}-L_z+1]:M_{z-1})}) \end{bmatrix},$$

(4.21)

where $M_z = M_{z-1} - L_z + 1$. The implementation of an Overlapping Volterra Neural Network (O-VNN) to find the corresponding feature maps for an input video is shown in Figure 4.5.

**Proposition 1.** If $\mathcal{Z}$ $2^{nd}$ order filters are cascaded as shown in Figure 4.5, the resulting Volterra Network has an effective order of $K_{\mathcal{Z}} = 2^{2^{\mathcal{Z}-1}}$.

*Proof.* See Appendix-B for proof.

**Proposition 2.** The complexity of a $K_{\mathcal{Z}}$ $\mathcal{Z}^{th}$ order cascaded Volterra filter will consist of,

$$\sum_{z=1}^{\mathcal{Z}} \left[(L_z[2p_{1z}+1][2p_{2z}+1]) + (L_z[2p_{1z}+1][2p_{2z}+1])^2 \right]$$

(4.22)

parameters.

*Proof.* For a $2^{nd}$ order filter ($K = 2$), the number of parameters required is $\left[(L_z[2p_{1z}+1][2p_{2z}+1]) + (L_z[2p_{1z}+1][2p_{2z}+1])^2 \right]$ (Equation (4.16)). Such a filter applied $\mathcal{Z}$ times leads to $\sum_{z=1}^{\mathcal{Z}} \left[(L_z[2p_{1z}+1][2p_{2z}+1]) + (L_z[2p_{1z}+1][2p_{2z}+1])^2 \right]$ parameters to realize a Volterra Filter with the order $K_{\mathcal{Z}} = 2^{2^{\mathcal{Z}-1}}$.

Furthermore, if a multi-channel input/output is considered, the number of parameters is,

$$\sum_{z=1}^{\mathcal{Z}} (n_{z,1}^z n_{z,2}^z \left[(L_z[2p_{1z}+1][2p_{2z}+1]) + (L_z[2p_{1z}+1][2p_{2z}+1])^2 \right],$$

(4.23)

where $n_{z,1}^z$ is the number of channels in the output of the $z^{th}$ layer.

### 4.4.3 System Stability and Convergence

The discussed system can be shown to be stable when the input is bounded, i.e. the system is Bounded Input Bounded Output (BIBO) stable. The following propositions outline the requirements for the system to be BIBO stable and convergent.

**Proposition 3.** An O-VNN with $\mathcal{Z}$ layers is BIBO stable if $\forall z \in \{1, ..., \mathcal{Z}\}$,

$$\sum_{\tau_1, \sigma_{11}^{\mathcal{Z}_1}, \sigma_{21}^{\mathcal{Z}_1}} \left| W_{\mathcal{Z}_1}^{\mathcal{Z}_1} \right| + \sum_{\tau_1, \sigma_{11}^{\mathcal{Z}_2}, \sigma_{21}^{\mathcal{Z}_2}} \left| W_{\mathcal{Z}_2}^{\mathcal{Z}_2} \right| < \infty.$$

(4.24)

*Proof.* See Appendix-B for proof.
Proposition 4. An O-VNN can be shown to be stable and convergent under the condition $|x_i| < \rho < \infty$, where $x_i$ is the input to the filter, and $\rho$ is the radius of convergence for the proposed Volterra filter.

Proof. See Appendix-B for proof.

4.4.4 Synthesis and Implementation of Volterra Kernels

As noted earlier, the linear kernel ($1^{st}$ order) of the Volterra filter is similar to the convolutional layer in the conventional CNNs. As a result, it can be easily implemented using the 3D convolution function in Tensorflow [1]. The second order kernel may be approximated as a product of two 3-dimensional matrices (i.e. a separable operator),

$$W^2 = W^2_{1} \times W^2_{2} \approx \sum_{q=1}^{Q} W^2_{a_q} \times W^2_{b_q},$$

where $P_1 = 2p_1 + 1$, and $P_2 = 2p_2 + 1$. Taking account of Equation (4.13) yields,

$$g \left( X \left[ t-L+1:t \right] \right) = \sum_{\tau_1,\sigma_1,\sigma_2} W^1_{\tau_1,\sigma_1,\sigma_2} X_{[\tau_1,\sigma_1,\sigma_2]} + \sum_{q=1}^{Q} \sum_{\tau_1,\sigma_1,\sigma_2} W^2_{a_q,\tau_1,\sigma_1,\sigma_2} X_{[\tau_1-\tau_1,\sigma_1,\sigma_2]} + \sum_{q=1}^{Q} \sum_{\tau_1,\sigma_1,\sigma_2} W^2_{b_q,\tau_1,\sigma_1,\sigma_2} X_{[\tau_1,\sigma_1,\sigma_2]}$$

A larger $Q$ will provide a better approximation of the $2^{nd}$ order kernel. The advantage of this class of approximation is two-fold. Firstly, the number of parameters can be further reduced, if for the $z^{th}$ layer, $(L_z,[2p_{z_1}+1],[2p_{z_2}+1])^2 > 2Q(L_z,[2p_{z_1}+1],[2p_{z_2}+1])$. The trade-off between performance and available computational resources must be accounted for when opting for such an approximation. Additionally, this makes it easier to implement the higher order kernels in Tensorflow [1] by using the built-in convolutional operator. The approximate quadratic layers in the Cascaded Volterra Filter (see Figure 4.5) yield the following number of parameters,

$$\sum_{z=1}^{Z} (L_z,[2p_{z_1}+1],[2p_{z_2}+1]) + 2Q(L_z,[2p_{z_1}+1],[2p_{z_2}+1]).$$

Proposition 5. The approximation discussed in Equation (4.25) is a $Q^{th}$ rank approximation of the exact quadratic kernel, $W^2$.

Proof. see Appendix-B for proof.
exact method) in Section 4.8. Figure 4.6 illustrates the implementation of a 2\textsuperscript{nd} order filter using a $Q^\text{th}$ rank approximation.

![Figure 4.6 Implementation of a second order Volterra Filter using $Q^\text{th}$ rank approximation](image)

### 4.5 Comparison with Long Short Term Memory Networks

In this section we compare the proposed Volterra Neural Networks with LSTMs, which are frequently used for processing/analyzing temporal data. An LSTM Network cell implemented as discussed in Section 4.2.4, can be shown to be a special case of the Volterra Filter.

**CASE-1: No Activation Functions**

**Proposition 6.** The cell state of an LSTM network at time $t$, $C_t$ is a special case of a 2\textsuperscript{nd} order Volterra Filter, where the 2\textsuperscript{nd} order filter is approximated using $Q = 1$ in Equation (4.25), and weighed by the cell state at time $t-1$, $C_{t-1}$, i.e.

$$C_t = C_{t-1} \left( \sum_{j=1}^{2} W^j_{ai} s_j \right) + \sum_{j, k=1}^{2} W_{ic}^{jk} s_j s_k,$$

where, $s = [h_{t-1}, x_t]$, and $W_{ic} = W_i W_c$.

**Proof.** From Equations (4.6), (4.9), (4.11), At time $t=0$,

$$C_0 = h_0 = W_0 x_0.$$
At time \( t=1 \),
\[
C_i = (W_f^1 h_0 + W_f^2 x_1)(h_0) + (W_i^1 h_0 + W_i^2 x_1)(W_c^1 h_0 + W_c^2 x_1) \tag{4.31}
\]
\[
= W_f^1 h_0^2 + W_f^2 x_1 h_0 + W_i^1 h_0 + W_i^2 x_1 W_c^1 h_0 + W_c^2 x_1.
\]

Define the matrix \( W_{ic} \) such that, \( W_{ic} = W_i.W_c \). This leads to,
\[
C_i = W_f^1 h_{t-1} + W_f^2 x_t C_{t-1} + W_i^1 h_{t-1} C_{t-1} + W_c^1 h_{t-1} + W_i^2 x_t h_{t-1} + W_c^2 x_t^2 \tag{4.32}
\]
\[
= C_{t-1} + \sum_{j=1}^2 W_f^j s_j + \sum_{j,k=1}^2 W_{ic}^{jk} s_j s_k, \tag{4.33}
\]
where, \( s = [h_{t-1}, x_t] \), and \( W_{ic} = W_i.W_c \). This formulation is equivalent to a \( 2^\text{nd} \) order Volterra Filter applied on the input \( s \), with the linear kernel \( W_f \) and quadratic kernel \( W_{ic} \).

**Proposition 7.** The output of the LSTM network at time \( t \), \( h_t \), is a special case of the \( 3^\text{rd} \) order Volterra Filter, where the \( 2^\text{nd} \) and \( 3^\text{rd} \) order filters are approximated using \( Q=1 \) in Equation (4.25), and weighed by the cell state at time \( t-1 \), \( C_{t-1} \), i.e.
\[
h_t = C_{t-1} + \sum_{j,k=1}^2 W_{of}^{jk} s_j s_k + \sum_{j,k,l=1}^2 W_{oic}^{jk} s_j s_k s_l, \tag{4.34}
\]
where, \( s = [h_{t-1}, x_t] \), \( W_{of} = W_o.W_f \), and \( W_{oic} = W_o.W_i.W_c \).

**Proof.** Based on Equation (4.11), the LSTM network output, \( h_t \) can be written as,
\[
h_t = (W_o^1 h_{t-1} + W_o^2 x_t)C_{t-1}. \tag{4.35}
\]
Replacing \( C_t \) with the expression in Equation (4.29), and using \( W_{of} = W_o.W_f \) and \( W_{oic} = W_o.W_i.W_c \),
\[
h_t = W_{oic}^{11} h_{t-1}^2 C_{t-1} + W_{oic}^{12} h_{t-1} x_t C_{t-1} + W_{oic}^{21} x_t h_{t-1} C_{t-1} + W_{oic}^{22} x_t^2 C_{t-1} \tag{4.36}
\]
\[
+ W_{oic}^{111} h_{t-1}^3 + W_{oic}^{112} h_{t-1}^2 x_t + W_{oic}^{121} h_{t-1} x_t^2 + W_{oic}^{122} x_t h_{t-1}^2 + W_{oic}^{211} x_t^3 + W_{oic}^{221} h_{t-1} x_t^2 + W_{oic}^{222} x_t^3. \tag{4.37}
\]
The above expression can be re-written as,

$$ h_t = C_{t-1} \left( \sum_{j,k=1}^{2} W^{jk}_{o,f} s_j s_k \right) + \sum_{j,k,l=1}^{2} W^{jk,l}_{o,i,c} s_j s_k s_l. $$

where, $$ s = [h_{t-1}, x_t], W_{o,f} = W_o \cdot W_f, \text{ and } W_{o,i,c} = W_o \cdot W_i \cdot W_c. $$

On the other hand, for an O-VNN with $Z$ layers,

$$ h_{t}^{Z} = \sum_{j=0}^{L_x} W^{Z1}_{j} h_{t-1}^{Z-1}^{(m_2+L_2)-j} + \sum_{j,k=0}^{L_x} W^{Z2}_{j,k} h_{t-1}^{Z-1}^{(m_2+L_2)-j} h_{t-1}^{Z-1}^{(m_2+L_2)-k}, $$

where, $m_2 \in [1:M_Z], M_Z = M_{Z-1} - L_Z + 1$, and $M_1 = t - L_1 + 1$. In both scenarios, the system uses higher order relations between current and previous samples. In case of an LSTM network the cell state at $t-1$, $C_{t-1}$ is used in order to select features from previous frames that may be relevant to current frames. On the other hand the Volterra filter formulation explicitly selects the interactions between the frames and weighs them accordingly. Figure 4.7 highlights the difference between the implementations of the two approaches.

**CASE-2: With Activation Functions**

The sigmoid activation function can be approximated as a taylor series,

$$ \sigma(x) = \frac{1}{1+e^{-x}} = \frac{1}{2} + \frac{1}{4} x - \frac{1}{48} x^3 + \frac{1}{480} x^5 - ... $$

59
This can be learned by a Volterra Network if required as it is a polynomial expression. Furthermore as long as the condition in Proposition 4 is satisfied, the series is convergent. Similarly, a tanh activation is also approximated by using a Taylor Series:

$$\tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + ...$$  \hspace{1cm} (4.43)

This means there is no longer a need to explicitly define the activation function, as the Volterra Neural Network will learn the required activation function as part of the learning/training process.

### 4.6 Two-Stream Volterra Networks

Most previous studies in action recognition in videos have noted the importance of using both the spatial and the temporal information for an improved recognition accuracy. As a result, we also propose the use of Volterra filtering in combining the two information streams, exploring a potential non-linear relationship between them. In Section 4.8 we will see that this actually boosts the performance, thereby indicating some non-linear relation between the two information streams. Independent Cascaded Volterra Filters are first used in order to extract features from each modality,

$$F^{RGB}_{[1:M^2]} = g_{2}^{RGB}(g_{1}^{RGB}(x_{[i-L+1:i]}))$$  \hspace{1cm} (4.44)

$$F^{OF}_{[1:M^2]} = g_{2}^{OF}(g_{1}^{OF}(x_{[i-L+1:i]})).$$  \hspace{1cm} (4.45)

Upon gleaning features from the two streams, an additional Volterra Filter is solely used for combining the generated feature maps from both modalities,

$$F^{(RGB+OF)}_{1} = \sum_{\tau_{1},\sigma_{11},\sigma_{21},u_{1}} W^{1}_{\tau_{1},\sigma_{11},\sigma_{21},u_{1}} \int_{M_{2}^{\tau_{1}}-1}^{M_{2}^{\tau_{1}}} \int_{M_{3}^{\sigma_{11}}-1}^{M_{3}^{\sigma_{11}}} \int_{M_{4}^{\sigma_{21}}-1}^{M_{4}^{\sigma_{21}}} \cdots \int_{M_{d}^{\sigma_{d1}}-1}^{M_{d}^{\sigma_{d1}}}$$

$$+ \sum_{\tau_{2},\sigma_{12},\sigma_{22},u_{2}} W^{2}_{\tau_{2},\sigma_{12},\sigma_{22},u_{2}} \int_{M_{2}^{\tau_{2}}-1}^{M_{2}^{\tau_{2}}} \int_{M_{3}^{\sigma_{12}}-1}^{M_{3}^{\sigma_{12}}} \int_{M_{4}^{\sigma_{22}}-1}^{M_{4}^{\sigma_{22}}} \cdots \int_{M_{d}^{\sigma_{d2}}-1}^{M_{d}^{\sigma_{d2}}}$$  \hspace{1cm} (4.46)

where \(\tau_{j} \in [0, L_{x}+1], \sigma_{1j} \in [-p_{1}, p_{1}], \sigma_{2j} \in [-p_{2}, p_{2}],\) and \(u_{j} \in [RGB, OF].\) Figure 4.8-(c) shows the block diagram for fusing the two information streams.

### 4.7 Min-Norm Solution of a \(2^{nd}\) Order Filter

Consider the vectorized \(i^{th}\) data sample, \(x_{i,d} = \{x_{i,1}, x_{i,2}, ..., x_{i,d}\}.\) The second order combinations between the features in \(x_{i}\) can be found by \(x_{i}^{T}x_{i}.\) The data sample can hence be rewritten as,

$$\hat{x}_{i,d+2} = [x_{i}, vec(x_{i}^{T}x_{i})].$$  \hspace{1cm} (4.47)

Consider \(N\) such samples, i.e. \(i \in \{1, ..., N\},\) which will be used for training of the system, this leads to the data matrix, \(\hat{X}_{N \times (d + d^2)} = [\hat{x}_{i}^{T}]_{i \in \{1, ..., N\}}.\)

Now consider, \(\bar{W}_{(d + d^2) \times 1} = [W^{1}_{1 \times d}, W^{2}_{1 \times d}]^{T}\) where, \(W^{1}\) and \(W^{2}\) are the Volterra Filter weights.
If \( Y_{N \times 1} \) is the ground truth, we wish to find \( \hat{W} \) such that,
\[
\hat{X}_{N \times (d+d^2)} \hat{W}_{(d+d^2) \times 1} = Y_{N \times 1}. \tag{4.48}
\]
The min-norm solution to the Filter weights in this case turns out to be,
\[
\hat{W} = \hat{X}^T (\hat{X} \hat{X}^T)^{-1} Y, \tag{4.49}
\]
where, \( \hat{X}^T (\hat{X} \hat{X}^T)^{-1} \) is the pseudo-inverse of \( \hat{X} \), and \( \hat{X} \hat{X}^T \) is assumed to have a full rank so that it is invertible.

**Risk Analysis**

The risk of the min-norm estimate, \( \hat{W} \) can be computed as discussed in [7],
\[
R(\hat{W}) = \mathbb{E}_{\hat{x},y} (y - \hat{x}^T \hat{W})^2 - \mathbb{E}(y - \hat{x}^T \hat{W}^\star)^2 \tag{4.50}
\]
\[
= \mathbb{E}_{\hat{x},y} (y - \hat{x}^T \hat{W}^\star + \hat{x}^T (\hat{W}^\star - \hat{W}))^2 - \mathbb{E}(y - \hat{x}^T \hat{W}^\star)^2 \tag{4.51}
\]
\[
= \mathbb{E}_{\hat{x}} (\hat{x}^T (\hat{W}^\star - \hat{W}))^2, \tag{4.52}
\]
where \( \hat{W}^\star \) is the least squares estimate, and \( \mathbb{E} \) is the expectation operator with respect to the observed data and the ground truth. Since \( \hat{W} = \hat{X}^T (\hat{X} \hat{X}^T)^{-1} Y \),
\[
R(\hat{W}) = \mathbb{E}_{\hat{x}} (\hat{x}^T (\hat{W}^\star - \hat{X}^T (\hat{X} \hat{X}^T)^{-1} Y)) \tag{4.53}
\]
Furthermore, considering distortion due to noise, \( \epsilon = Y - \hat{X} \hat{W}^\star \),
\[
R(\hat{W}) = \mathbb{E}_{\hat{x}} (\hat{x}^T (I - \hat{X}^T (\hat{X} \hat{X}^T)^{-1})) \hat{W}^\star - \hat{x}^T \hat{X}^T (\hat{X} \hat{X}^T)^{-1} \epsilon)^2 \tag{4.54}
\]
\[
\leq 2 \mathbb{E}_{\hat{x}} (\hat{x}^T (I - \hat{X}^T (\hat{X} \hat{X}^T)^{-1})) \hat{W}^\star)^2 + 2 \mathbb{E}_{\hat{x}} (\hat{x}^T (\hat{X} \hat{X}^T)^{-1} \epsilon)^2 \tag{4.55}
\]
\[
= 2 \hat{W}^\star^T B \hat{W}^\star + 2 \epsilon^T C \epsilon, \tag{4.56}
\]
where, \( B = (I - \hat{X}^T (\hat{X} \hat{X}^T)^{-1}) \Sigma (I - \hat{X}^T (\hat{X} \hat{X}^T)^{-1}) \), \( \Sigma = \hat{x} \hat{x}^T \)
and \( C = (\hat{X} \hat{X}^T)^{-1} \hat{X} \Sigma \hat{X}^T (\hat{X} \hat{X}^T)^{-1} \).

Equation (4.56) provides an upper bound for the excess risk when using the min-norm estimate \( \hat{W} \), in terms of the observed data, the optimal least-squares estimate \( \hat{W}^\star \) and the distortion due to noise, \( \epsilon \).
Figure 4.8 (a): Decision Level Fusion, (b): Feature Concatenation, (c): Two-Stream Volterra Filtering

4.8 Experiments and Results

4.8.1 Human Activity Recognition

We proceed to evaluate the performance of this approach on two action recognition datasets, namely, UCF-101 [77] and HMDB-51 [43], and the comparison of the results with recent state of the art implementations is given in Tables 4.1 and 4.2. Table 4.1 shows the comparison with techniques that only exploit the RGB stream, while Table 4.2 shows the comparison when both information streams are used. Note that our comparable performance to the state of the art is achieved with a significantly lower number of parameters (see Table 4.4). Furthermore, a significant boost in performance is achieved by allowing non-linear interaction between the two information streams. The Optical Flow is computed using the TV-L1 algorithm [89]. Note that we train the network from scratch on both datasets, and do not use a larger dataset for pre-training, in contrast to some of the previous implementations. The implementations that take advantage of a different dataset for pre-training are indicated by a ‘Y’ in the pre-training column, while those that do not, are indicated by ‘N’. When training from scratch the proposed solution is able to achieve best performance for both scenarios: one stream networks (RGB frames only) and two-stream networks (RGB frames & Optical Flow). To fuse the two information streams (spatial and temporal), we evaluate the following techniques (summarized in Figure 4.8):

1. **Decision Level Fusion**: An independent decision is made using the features from each stream, and these decisions are then combined using either of the following:

   (a) **Weighted Averaging**: The decision probabilities $P_{t}^{RGB}(a_i)$ and $P_{t}^{OF}(a_i)$ are independently computed and are combined to determine the fused probability $P_{t}^{f}(a_i)$ as following,

   $$ P_{t}^{f}(a_i) = \beta^{RGB} P_{t}^{RGB}(a_i) + \beta^{OF} P_{t}^{OF}(a_i), $$

   where $\beta^{RGB} + \beta^{OF} = 1$, which control the importance/contribution of the RGB and Optical Flow streams towards making a final decision.

   (b) **Event-Driven Fusion** [65, 66]: As in 1-(a), the decision probabilities are independently
computed with now the fused decision probability calculated as,

\[ p_f^f(a_i) = \gamma p_{t_{\text{MAX MI}}}^{\text{MAX MI}}(a_i^{RGB}, a_i^{OF}) + (1 - \gamma) p_{t_{\text{MIN MI}}}^{\text{MIN MI}}(a_i^{RGB}, a_i^{OF}), \quad (4.58) \]

where, \( \gamma \) is a pseudo measure of correlation between the two information streams, \( p_{t_{\text{MAX MI}}}^{\text{MAX MI}}(.) \) is the joint distribution with maximal mutual information, and \( p_{t_{\text{MIN MI}}}^{\text{MIN MI}}(.) \) is the joint distribution with minimal mutual information.

2. **Feature Level Fusion:** Features are extracted from each stream independently, and are subsequently merged before making a decision. For this level of fusion we consider the following:

(a) **Feature Concatenation:** Features \( F_{t}^{x_{RGB}} \) and \( F_{t}^{x_{OF}} \) are concatenated together before using a linear classifier to make a decision (see Figure 4.8-(b)).

(b) **Two-Stream Volterra Filtering** (see Section 4.6, Figure 4.8-(c))

![Figure 4.9](image_url) (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering

In our implementation we use an O-VNN with 8 layers on both the RGB stream and the optical stream. Each layer uses \( L_z = 2 \) and \( p_{l_z}, p_{z} \in \{0, 1, 2\} \). The outputs of the two filters are fed into the fusion layer which combines the two streams. The fusion layer uses \( L_{\text{Fuse}} = 2 \) and \( p_{1_{\text{Fuse}}}, p_{2_{\text{Fuse}}} \in \{0, 1, 2\} \). It is clear from Table 4.2 that performing fusion using Volterra Filters significantly boosts the performance of the system. This shows that there does exist a non-linear relationship between the two modalities. This can also be confirmed from the fact that we can see significant values in the weights for the fusion layer (see Table 4.3). Figures 4.9 and 4.10 show one of the feature maps for an archery video and a fencing video. From Figures 4.9, 4.10-(b),(c) it can be seen that when only the RGB stream is used, a lot of the background area has high values, while on the other hand, when both streams are jointly used, the system is able to concentrate on more relevant features. In 4.9-(c), the system is seen to concentrate on the bow and arrow which are central to recognizing the action, while in 4.10-(c) the system is seen to concentrate on the pose of the human which is central to identifying a
Figure 4.10 (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering

Figure 4.11 Epochs vs Loss for various number of multipliers for a Cascaded Volterra Filter

fencing action. Figure 4.11 shows the Epochs vs Loss graph for a Cascaded Volterra Filter when a different number of multipliers ($Q$) are used to approximate the $2^{nd}$ order kernel. The green plot shows the loss when the exact kernel is learned, and it can be seen that the performance comes closer to the exact kernel as $Q$ is increased.

4.8.2 Target Detection/Recognition

We proceed to evaluate the performance of the proposed approach experimenting on both target detection datasets that were previously discussed in 3.5. In addition, we will also evaluate this approach on some real radar and telescopic data collected by the Czech Technical University in Prague.
**Table 4.1** Performance Evaluation for one stream networks (RGB only): The proposed algorithm achieves best performance when trained from scratch

<table>
<thead>
<tr>
<th>Method</th>
<th>Pre-Training</th>
<th>Avg Accuracy UCF-101</th>
<th>Avg Accuracy HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow Fusion [37]</td>
<td>Y (Sports-1M)</td>
<td>64.1 %</td>
<td>-</td>
</tr>
<tr>
<td>Deep Temporal Linear Encoding Networks [18]</td>
<td>Y (Sports-1M)</td>
<td>86.3 %</td>
<td>60.3 %</td>
</tr>
<tr>
<td>Inflated 3D CNN [12]</td>
<td>Y (ImageNet + Kinetics)</td>
<td><strong>95.1 %</strong></td>
<td><strong>74.3 %</strong></td>
</tr>
<tr>
<td>Soomro et al, 2012</td>
<td>N</td>
<td>43.9 %</td>
<td>-</td>
</tr>
<tr>
<td>Single Frame CNN [37, 42]</td>
<td>N</td>
<td>36.9 %</td>
<td>-</td>
</tr>
<tr>
<td>Slow Fusion [5, 36, 37]</td>
<td>N</td>
<td>41.3 %</td>
<td>-</td>
</tr>
<tr>
<td>3D-ConvNet [12, 80]</td>
<td>N</td>
<td>51.6 %</td>
<td>24.3 %</td>
</tr>
<tr>
<td>Volterra Filter</td>
<td>N</td>
<td>38.19 %</td>
<td>18.76 %</td>
</tr>
<tr>
<td>O-VNN (exact)</td>
<td>N</td>
<td><strong>58.73 %</strong></td>
<td><strong>29.33 %</strong></td>
</tr>
<tr>
<td>O-VNN (Q=7)</td>
<td>N</td>
<td>53.77 %</td>
<td>25.76 %</td>
</tr>
</tbody>
</table>

**Table 4.2** Performance Evaluation for two stream networks (RGB & Optical Flow): The proposed algorithm achieves best performance on both datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pre-Training</th>
<th>Avg Accuracy UCF-101</th>
<th>Avg Accuracy HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Stream CNNs [73]</td>
<td>Y (ILSVRC-2012)</td>
<td>88.0 %</td>
<td>72.7 %</td>
</tr>
<tr>
<td>Deep Temporal Linear Encoding Networks [18]</td>
<td>Y (BN-Inception + ImageNet)</td>
<td>95.6 %</td>
<td>71.1 %</td>
</tr>
<tr>
<td>Two Stream Inflated 3D CNN [12]</td>
<td>Y (ImageNet + Kinetics)</td>
<td><strong>98.0 %</strong></td>
<td>80.9 %</td>
</tr>
<tr>
<td>Two-Stream O-VNN (Q=15)</td>
<td>Y (Kinetics)</td>
<td><strong>98.49 %</strong></td>
<td><strong>82.63 %</strong></td>
</tr>
<tr>
<td>Two Stream Inflated 3D CNN [12]</td>
<td>N</td>
<td>88.8 %</td>
<td>62.2 %</td>
</tr>
<tr>
<td>Weighted Averaging: O-VNN (exact)</td>
<td>N</td>
<td>85.79 %</td>
<td>59.13 %</td>
</tr>
<tr>
<td>Weighted Averaging: O-VNN (Q=7)</td>
<td>N</td>
<td>81.53 %</td>
<td>55.67 %</td>
</tr>
<tr>
<td>Event Driven Fusion: O-VNN (exact)</td>
<td>N</td>
<td>85.21 %</td>
<td>60.36 %</td>
</tr>
<tr>
<td>Event Driven Fusion: O-VNN (Q=7)</td>
<td>N</td>
<td>80.37 %</td>
<td>57.89 %</td>
</tr>
<tr>
<td>Feature Concatenation: O-VNN (exact)</td>
<td>N</td>
<td>82.31 %</td>
<td>55.88 %</td>
</tr>
<tr>
<td>Feature Concatenation: O-VNN (Q=7)</td>
<td>N</td>
<td>78.79 %</td>
<td>51.08 %</td>
</tr>
<tr>
<td>Two-Stream O-VNN (exact)</td>
<td>N</td>
<td><strong>90.28 %</strong></td>
<td><strong>65.61 %</strong></td>
</tr>
<tr>
<td>Two-Stream O-VNN (Q=7)</td>
<td>N</td>
<td>86.16 %</td>
<td>62.45 %</td>
</tr>
</tbody>
</table>
Table 4.3 Norm of $W^u$, where $u \in \{\text{RGB, OF, Fusion}\}$.

| $\frac{1}{2}||W^u||_2^2$ | $u = \text{RGB}$ | $u = \text{OF}$ | $u = \text{Fusion}$ |
|--------------------------|------------------|----------------|-------------------|
|                         | 352.15           | 241.2          | 341.3             |

Table 4.4 Comparison of number of parameters required and processing speed with the state of the art. A video with 60 frames is evaluated.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Parameters</th>
<th>Processing Speed (secs/video)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Temporal Linear Encoding</td>
<td>22.6M</td>
<td></td>
</tr>
<tr>
<td>Inflated 3D CNN</td>
<td>25M</td>
<td>3.52</td>
</tr>
<tr>
<td>O-VNN (exact)</td>
<td>4.6M</td>
<td>0.73</td>
</tr>
<tr>
<td>O-VNN ($Q=7$)</td>
<td>3.7M</td>
<td>0.61</td>
</tr>
<tr>
<td>Two-Stream O-VNN (exact)</td>
<td>10.1M</td>
<td>1.88</td>
</tr>
<tr>
<td>Two-Stream O-VNN ($Q=7$)</td>
<td>8.2M</td>
<td>1.54</td>
</tr>
<tr>
<td>Two-Stream O-VNN ($Q=1$)</td>
<td>2.5M</td>
<td>0.34</td>
</tr>
</tbody>
</table>

4.8.2.1 Simulated Radar and Telescopic Imaging Data

We use two sensors, namely a Telescopic imaging and a Radar sensor. We consider two objects of interest which are defined by a combined sets of certain events. The radar observations are generated on the basis of velocity ($v$), cross-section ($c_s$) and range ($r$) of the target,

$$X^R = S^R(v, c_s, r) + N^R,$$  \hspace{1cm}  (4.59)

where $S^R$ is the radar signal characterized by $v, c_s$, and $r$, and $N^R$ is white noise. Similarly, the Telescopic images are based on the Aspect Ratio (AR), and the displacement (d),

$$X^{TI} = S^{TI}(AR, d) + N^{TI}.$$ \hspace{1cm} (4.60)

The same event and target definitions as in the previous chapters, are utilized for labeling the dataset. The Block Diagram for the implementation of this approach can be seen in Figure 4.12.

4.8.2.2 Real Radar and Telescopic Imaging Data

We also implement the VNN approach to detect the International Space Station (ISS) based on the data provided by the Czech Technical University. Figure 4.13 shows sample data for a positive observation of the ISS. The same implementation as shown in Figure 4.12 is used. Although, only one image is associated to a radar observation in this case, and as a result, a 2D VNN is applied on the Telescopic Image. The Telescopic image is divided into patches in order to enhance the training
Figure 4.12 Block diagram for implementation of the Volterra Filtering approach on Radar and Telescopic Imaging data.

Table 4.5 Performance Comparison for the Simulated Dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar Sensor</td>
<td>94.89 %</td>
</tr>
<tr>
<td>Telescopic Imaging Sensor</td>
<td>86.06 %</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>93.69 %</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion [48, 49]</td>
<td>95.72 %</td>
</tr>
<tr>
<td>Event Driven Fusion [65]</td>
<td>90.36 %</td>
</tr>
<tr>
<td>Commuting CGANs</td>
<td>95.34 %</td>
</tr>
<tr>
<td><strong>O-VNN</strong></td>
<td><strong>97.83 %</strong></td>
</tr>
</tbody>
</table>

data. Furthermore, each VNN is individually trained before performing fusion. Table 4.6 shows the performance of the implementation compared to individual sensor performance. The low detection accuracy is a result of smaller number of available data samples for training and testing. Figure 4.14 shows the detection scores as determined by a VNN trained on Telescopic Imaging.

Table 4.6 Performance Comparison for Real Data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar Sensor</td>
<td>80.7 %</td>
</tr>
<tr>
<td>Telescopic Imaging Sensor</td>
<td>73.9 %</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion [48, 49]</td>
<td>75.0 %</td>
</tr>
<tr>
<td><strong>Two-Stream O-VNN</strong></td>
<td><strong>83.33 %</strong></td>
</tr>
</tbody>
</table>

4.8.2.3 Imaging, Seismic, and Acoustic Sensor Network for Target Detection

The final dataset that we use for experiments is pre-collected data from a network of seismic, acoustic, and imaging sensors deployed in a field, where people/vehicles were walking/driven around in specified patterns. Details about this sensor setup and experiments can be found in [54]. Here, we
(a) International Space Station (ISS) detection by Radar

(b) International Space Station (ISS) detection by Telescopic Image sensor

**Figure 4.13** Data Samples for ISS detection
use this dataset to classify between human targets, vehicular targets, and no targets. Table ?? shows the classification performance when compared to previously discussed approaches. It can be clearly observed that the additional non-linearity in form of higher order convolutions is important, and leads to better classification performance. Figure 4.15 shows the detection probability map of a human target as seen by the imaging sensor.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seismic Sensor</td>
<td>95.13 %</td>
</tr>
<tr>
<td>Acoustic Sensor</td>
<td>69.33 %</td>
</tr>
<tr>
<td>Imaging Sensor</td>
<td>94.70 %</td>
</tr>
<tr>
<td>Feature Concatenation</td>
<td>95.69 %</td>
</tr>
<tr>
<td>Similar Sensor Fusion [48, 49]</td>
<td>95.88 %</td>
</tr>
<tr>
<td>Dissimilar Sensor Fusion [48]</td>
<td>97.69 %</td>
</tr>
<tr>
<td>Event Driven Fusion [65]</td>
<td>90.36 %</td>
</tr>
<tr>
<td>Commuting CGANs</td>
<td>97.79 %</td>
</tr>
<tr>
<td><strong>O-VNN</strong></td>
<td><strong>98.48 %</strong></td>
</tr>
</tbody>
</table>

Table 4.7 Performance Comparison for Imaging, Seismic, and Acoustic Sensor Network for Target Detection.

4.9 Conclusion

We proposed a novel network architecture for recognition of actions in videos, where the non-linearities were introduced by the Volterra Series Formulation. We propose a Cascaded Volterra Filter which leads to a significant reduction in parameters while exploring the same complexity of non-linearities in the data. Such a Cascaded Volterra Filter is a BIBO stable and convergent system. In addition, we also proposed the use of the Volterra Filter to fuse the spatial and temporal
Figure 4.15 Detection of human target using VNN learned on the Imaging Sensor

streams, hence leading to a non-linear fusion of the two streams. The network architecture inspired by Volterra Filters achieves performance accuracies which are comparable to the state of the art approaches while using a fraction of the number of parameters used by them. This makes such an architecture very attractive from a training point of view, with an added advantage of reduced storage space to realize the model.
CHAPTER 5

CONCLUSION

In this dissertation, we explored various approaches to fuse multi-modal data for target classification. An event driven approach to sensor fusion was first introduced. Each sensor was said to make a decision on occurrence of certain events that it observes rather than on the target identity. Furthermore, this approach explored the dependence of features on one another, as different sensors may have some correlation, but may not necessarily be entirely similar/dissimilar. Subsequently, a data driven approach to learn the various features and object defining events was introduced. This solved the major drawback of the event driven approach which required hand-crafting of the features describing the target. This was accomplished by finding a common hidden sub-space between the various sensors, and selecting features of interest from this sub-space. A Commuting Conditional GAN system was proposed in order to find this hidden space, and was successfully exploited to detect damaged sensors and deal with them with no human intervention. Finally, in Chapter 4, a novel Neural Network Structure was proposed as a result of an inspiration from Volterra Filters, and referred to as a Volterra Neural Network (VNN). This Network exploited the higher order convolutions in addition to the linear convolution of a standard CNN. It also introduced a non-linear fusion between the various sensors, hence improving fusion performance in comparison to the previously explored techniques which only explored linear convolutions and a linear fusion at the decision level. The VNN was also shown to require a reduced number of parameters to reach a similar performance to that of a conventional CNN. This makes the VNN attractive from a training point of view, particularly in light of the additional smaller storage space. All of these approaches were evaluated on various multi-modal datasets for target detection and shown to significantly improve the fusion performance. The VNN was additionally evaluated on action recognition datasets to highlight its ability to capture temporal and spatial information with a significantly lower number of parameters in comparison to the state-of-art.
BIBLIOGRAPHY


APPENDICES
APPENDIX A

DERIVATIVES FOR INDEPENDENT HIDDEN SPACES

Using $z_{l r}^f$ from Equation (2.25) into Equation (2.24) we get,

$$
\min_{W_l^f, Z_{l r}^f} J(W_l^f, Z_{l r}^f) = \frac{1}{2} \|W_l^f\|^2 + C_1 \sum_{r, n} \max\{0, 1 - w_{k_{y_n}}^r (Z_{l r}^f U_k^l x_n^r) + w_{k_l}^r (Z_{l r}^f U_k^l x_n^r)\}
$$

$$
+ \frac{1}{2} \sum_{r, n} \left( C_2 \|[Z_{l r}^f, Z_{l s}^f] \| + C_3 \sum_{n} (Z_{l r}^f U_k^l x_n^r - Z_{l s}^f U_k^l x_n^s)^2 \right)
$$

(A.1)

In order to update the variables, $W_l^f$ and $Z_{l r}^f$, a derivative of $J(W_l^f, Z_{l r}^f)$ must be computed with respect to each variable. Let $V_i = 1 - w_{k_{y_n}}^r (Z_{l r}^f U_k^l x_n^r) + w_{k_l}^r (Z_{l r}^f U_k^l x_n^r), \forall t \in \{1, ..., J_k\} \setminus y_n$ and $t'_n = \text{argmax}_{t \neq y_n} V_i$. Then we have,

$$
\forall m \in \{1, ..., J_k\},
$$

$$
\frac{d J(W_l^f, Z_{l r}^f)}{d w_{k m}^l} = w_{k m}^l + \mathcal{J}(V_i > 0) C_1 \sum_{r, n} [\mathcal{J}(m = t)Z_{l r}^f U_k^l x_n^r - \mathcal{J}(m = y_n)Z_{l r}^f U_k^l x_n^r]
$$

(A.2)

$$
\forall r \in \{1, ..., L\},
$$

$$
\frac{d J(W_l^f, Z_{l r}^f)}{d Z_{l r}^f} = \mathcal{J}(V_i > 0) C_1 \sum_{n} (- w_{k_{y_n}}^r (U_k^l x_n^r) + w_{k_l}^r (U_k^l x_n^r))
$$

$$
+ \sum_{s} [C_2 ([Z_{l r}^f, Z_{l s}^f]Z_{l s}^f - Z_{l r}^f Z_{l s}^f) + C_3 \sum_n U_k^l x_n^r (Z_{l r}^f U_k^l x_n^r - Z_{l s}^f U_k^l x_n^s)]
$$

(A.3)
where, $\mathcal{I}$ is the indicator function,

$$
\mathcal{I}(a) = \begin{cases} 
1, & \text{if } a \text{ is true} \\
0, & \text{otherwise.}
\end{cases} 
$$  \hfill (A.4)

These derivatives are then used to update the variables at each iteration,

$$
\forall r \in \{1, \ldots, L\},
$$

$$
u_{km}^{l(i+1)} = u_{km}^{l(i)} - \mu \frac{d J(W_{k}^{l}, Z_{k}^{l})}{dw_{km}^{l}}, \hfill (A.5)
$$

$$
z_{k}^{l(i+1)} = z_{k}^{l(i)} - \mu \frac{d J(W_{k}^{l}, Z_{k}^{l})}{dz_{k}^{l}}. \hfill (A.6)
$$
APPENDIX B

PROOFS FOR CHAPTER 4

B.1 Proof for Proposition 1

Proof. Since each layer of the O-VNN is a $2^{nd}$ order Volterra Filter, the order at the $2^{rth}$ layer can be written in terms of the order of the previous layer,

$$K_{2^r} = K_{2^{r-1}}^2, \quad (B.1)$$

where, $K_{2^{r-1}}$ is the order of the $(2^{r-1})^{th}$ layer. Since, the O-VNN only consists of $2^{nd}$ order layers, there exists some $p$ such that,

$$K_{2^r} = 2^p. \quad (B.2)$$

From Equations (B.1) and (B.2),

$$2^p = K_{2^{r-1}}^2 \quad (B.3)$$

Taking log$_2$ on both sides,

$$\log_2 2^p = \log_2 K_{2^{r-1}}^2$$

$$\Rightarrow p = 2 \log_2 K_{2^{r-1}}$$

$$\Rightarrow p = 2 \log_2 K_{2^{r-2}}$$

$$\Rightarrow p = 2^2 \log_2 K_{2^{r-2}}$$

$$\Rightarrow p = 2^{(2^{r-1})} \log_2 K_1$$

(B.4)

Since $K_1 = 2$ and log$_2 2 = 1$,

$$p = 2^{2^{r-1}}. \quad (B.5)$$
Putting this in Equation (B.2), we get,

\[ K_Z = 2^{Z-1}. \]  

(B.6)

\[ \square \]

### B.2 Proof for Proposition 3

**Proof.** Consider the \( z^{th} \) layer in the Cascaded implementation of the Volterra Filter,

\[ F_z^{[z]}_{[1:M_z]} = \left[ g_z(F_z^{[z-1]}_{l_{1,z}}) \ g_z(F_z^{z-1}_{l_{2,z+1}}) \ldots g_z(F_z^{z-1}_{l_{(M_z-1)-L_z+1(M_z-1)}}) \right], \]  

(B.7)

where, \( M_z = \frac{M_z-1}{L_z} \). Then for \( m_z \in \{1,...,M_z\}, \)

\[ \left| F_{m_z}^{z} \right| \leq g_z \left( \sum_{\tau_1,\sigma_{11},\sigma_{21}} W_{\tau_1,\sigma_{11},\sigma_{21}}^{z1} f^{z-1}_{\tau_1-\tau_1,\sigma_{11}-\sigma_{21}} + \sum_{\tau_1,\sigma_{12},\sigma_{22}} W_{\tau_1,\sigma_{12},\sigma_{22}}^{z2} f^{z-1}_{\tau_1-\tau_1,\sigma_{12}-\sigma_{22}} \right) \]  

(B.8)

\[ \leq 2 A \sum_{\tau_1,\sigma_{11},\sigma_{21}} W_{\tau_1,\sigma_{11},\sigma_{21}}^{z1} + A^2 \sum_{\tau_1,\sigma_{12},\sigma_{22}} W_{\tau_1,\sigma_{12},\sigma_{22}}^{z2} \]  

(B.9)

Note that Equation (B.10) states that a bounded input yields \( \sum_{\tau_1,\sigma_{11},\sigma_{21}} f^{z-1}_{\tau_1-\tau_1,\sigma_{11}-\sigma_{21}} \leq A \), for some \( A < \infty \). Hence, the sufficient condition for the system to be BIBO stable is,

\[ \sum_{\tau_1,\sigma_{11},\sigma_{21}} W_{\tau_1,\sigma_{11},\sigma_{21}}^{z1} + \sum_{\tau_1,\sigma_{12},\sigma_{22}} W_{\tau_1,\sigma_{12},\sigma_{22}}^{z2} < \infty. \]  

(B.11)

If the input data (i.e. video frames) is bounded, so is the output of each layer provided that Equation (B.11) is satisfied \( \forall z \in \{1,...,Z\} \), making the entire system BIBO stable. \[ \square \]

### B.3 Proof for Proposition 4

**Proof.** A Volterra Filter can be viewed as a power series,

\[ y_t = \sum_{k=1}^{K} a^k g_k[x_t] = \sum_{k=1}^{K} a^k g_k[x_t], \]  

(B.12)

where \( a \) is an amplification factor and,

\[ g_k[x_t] = \sum_{\tau_1,...,\tau_k} W_{\tau_1,...,\tau_k}^k x_{t-\tau_1} x_{t-\tau_2} \ldots x_{t-\tau_k}. \]  

(B.13)
In general, for a power series $\sum_{k=1}^{\infty} c_k x^k$, converges only for $|x| < \rho$, where $\rho = (\lim_{k \to \infty} \sup |c_k|^{1/k})^{-1}$ [68]. Setting $a = 1$ in Equation (B.12) and replacing the coefficients $c_k$ with the $k^{th}$ order Volterra Kernel $W^k$,

$$\rho = (\lim_{k \to \infty} \sup |W^k|^{1/k})^{-1}.$$  \hspace{1cm} (B.14)

Furthermore, since the system must also satisfy the BIBO stability condition,

$$|x| < (\lim_{k \to \infty} \sup |W^k|^{1/k})^{-1} < \infty.$$ \hspace{1cm} (B.15)

\hfill \square

**B.4 Proof for Proposition 5**

*Proof.* For simplicity, consider a 1-D Volterra Filter with memory, $L$. The quadratic weight matrix, $W^2$ in such a case is of size $L \times L$, and Equation (4.25) becomes, $W^2_{L \times L} = \sum_{q=1}^{Q} W^2_{L \times L} W^2_{L \times L}$. Consider the Singular Value Decomposition of the quadratic weight matrix, $W^2$,

$$W^2 = U \Sigma V^T,$$ \hspace{1cm} (B.16)

where, $U$ and $V$ are $L \times L$ matrices, and $\Sigma$ is a diagonal matrix with singular values on the diagonal. Equation (B.16) can be re-written as,

$$W^2 = \sum_{q=1}^{L} u_q \sigma_q v_q^T,$$ \hspace{1cm} (B.17)

where, $u_q$ and $v_q$ are the $q^{th}$ column of $U$ and $V$ respectively, and $\sigma_q$ is the $q^{th}$ diagonal element of $\Sigma$. A $Q^{th}$ rank approximation is then given as,

$$W^{2(Q)} = \sum_{q=1}^{Q} \hat{u}_q \sigma_q \hat{v}_q^T,$$ \hspace{1cm} (B.18)

where, $\hat{u}_q = u_q. \sigma_q$. If $W^2_{a_q} = \hat{u}_q$ and $W^2_{b_q} = v_q^T$,

$$W^{2(Q)} = \sum_{q=1}^{Q} W^2_{a_q} W^2_{b_q}.$$ \hspace{1cm} (B.19)

Hence, the approximation discussed in Equation (4.25) is a $Q^{th}$ rank approximation of the exact quadratic kernel. \hfill \square